PRICE NEGOTIATING E-COMMERCE CHAT-BOT SYSTEM USING ARTIFICIAL INTELLIGENCE AND DEEP LEARNING

Submitted by

PRADEEP KUMAR RAMESH (211517104123)

HEMA KUMAR A V (211517104058)

MANOJ KUMAR S (211517104097)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR INSTITUTE OF TECHNOLOGY, CHENNAI – 600 123

ANNA UNIVERSITY :: CHENNAI -600 025

JULY 2021

BONAFIDE CERTIFICATE

Certified that this project report "**PRICE NEGOTIATING E-COMMERCE CHAT-BOT SYSTEM USING ARTIFICIAL INTELLIGENCE AND DEEP LEARNING**" is the bonafide work of "**PRADEEP KUMAR RAMESH (211517104123)**, **HEMA KUMAR A V (211517104058)** and **MANOJ KUMAR S (211517104097)**" who carried out the project work under my supervision.

SIGNATURE

Dr. V. SUBEDHA, M.Tech, Ph.D
Professor and Head,
Department of CSE,
Panimalar Institute of Technology,
Poonamallee, Chennai – 600 123.

SIGNATURE

Mrs. D.Lakshmi , M.E, (Ph.D)
Associate Professor,
Department of CSE,
Panimalar Institute of Technology,
Poonamallee, Chennai – 600 123.

Certified that the above candidates were examined in the university project work viva voce examination held on 29-07-2021 at Panimalar Institute of Technology, Chennai – 600 123.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

We would like to express our deep gratitude to **Our Beloved Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D**., for his kind words and enthusiastic motivation which inspired us a lot in completing this project.

We also express our sincere thanks to **Our Dynamic Directors Mrs. C. VIJAYA RAJESWARI, Mr. C. SAKTHI KUMAR, M.E.,** and **Mrs. SARANYASREE SAKTHIKUMAR, B.E.,** for providing us with the necessary facilities for completion of this project.

We also express our gratefulness to **Our Principal Dr. T. JAYANTHI, M.E., Ph.D.,** who very much helped us in the completion of the Project.

We wish to convey our thanks and gratitude to our **Head of the Department, Dr. V.SUBEDHA, M.Tech., Ph.D.,** Department of Computer Science & Engineering, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project Guide, **Mrs. D.Lakshmi, M.E., (Ph.D).,** Associate Professor, Department of Computer Science & Engineering for his guidance throughout the Course of our project.

TABLE OF CONTENTS

Chapter 1 : SYSTEM OVERVIEW

Introduction

Agent based automated negotiation can give a flexible instead of a fixed price in e-commerce, and can maximize the payoffs of bath buyer and seller. It can be seen as an ideal and efficient mechanism for business. After discuss the negotiation in e-commerce and intelligent agent technology, state of the art overviews is given of agent based automated negotiation, specifically three main approaches for automated negotiation: the decision theory, the game theory and the negotiation analysis. Then, give a view of some famous models. Finally, we point out that the automated negotiation is still in its infant stage, because there are still some difficulties in this field. The first is the ontology issue, the second is agents' strategies and third is Communication protocol. Electronic negotiations are becoming an important research subject in the area of electronic commerce. Decision analysis and especially multi attributive utility theory play an important role for the support of electronic negotiations. The preferences are usually represented as a utility function on the set of alternatives such that the user prefers an alternative exactly when it has higher utility. Successful experience of the human traditional negotiation is the valuable learning resources of automatic negotiation. Automated negotiation model can learn from experience in negotiation, reason, and give a reasonable choice of negotiations on a new strategy.

Modules and their Description

The system comprises of 2 major modules with their sub-modules as follows:

1. **Admin**
    - **Login:** Admin need to login by providing the login credentials to access the below given admin modules.
    - **Add / View Product:** Admin can enter details about new products details and view its details.
    - **View Order:** Admin can view details about the order placed by the user.
    - **View Users Details:** Admin can view all the registered user's details.

2. **User**
    - **Login/Registration:** User can register on the system and get his online account on site.
    - **Chat Bot:** Automated negotiation model can learn from experience in negotiation, reason, and give a reasonable choice of negotiations on a new strategy. User can chat with the bot for price negotiation for a particular product.
    - **Add to Cart:** Users can add multiple products to cart.
    - **Pay using Card:** After total bill is calculated user can pay via credit card online.
    - **View Order:** User can view details about the order placed.

Existing System & Proposed System

❖ **Problem with current scenario**

- In existing system, shopping has done in a manual way, the customer has to go for shopping, and then he is having the possibility to choose the products whatever customer wants.
- Product's in the store are normally arranged by their types and price.

- Customers chooses from the collection of Cloths, where the items are labeled by their price and occasionally, the discounts offered on the particular item. Sales staff are always there in case the customers want some assistance.
- The customer takes the items he chooses by their requirements or interests and takes the items to the billing section.
- The bill is collected in the form of cash or credit card and a memo is prepared for the sold items which contains the information about the product such as price and quantity.
- It is a time consuming process.
- Thus, the system has to be automated.

LIMITATION IN EXISTING SYSTEM

- In Existing System, the Customer is completely depending on the manual process for buying the product.
- Manual process is a time consuming factor and when customer approaches for a manual shopping directly, actually he/she does not have an idea about things like, price range, etc….
- While online payment using card, the card details are stored as it is into the database without any encryption.
- There is a possibility of card details getting hacked.
- The time which has been spent by the customer in manual shopping can equates to multiple number of shopping. As customer can sit at home and browse in a fraction of seconds.
- The system is limited to a particular area as the store generally caters the need of people living in a particular territory.
- Customers have to take pain to go to the shop in case of heat, cold, rain etc.
- No common platform and easy facility normally available where many dealers can interact with one as many stores have products of just one particular company or dealer.
- Thus, we need to change to a system like “Online Shopping”.

PROPOSED SYSTEM

- Considering the anomalies in the existing system computerization of the whole activity is being suggested after initial analysis.
- The web application is developed using Visual Studio with Asp .Net with C# as a programming language.
- Proposed system is accessed by two entities namely, Admin and User.
- Admin need to login with their valid login credentials first in order to access the android application.
- After successful login, admin can access all the modules and perform/manage each task accurately.
- Admin can perform task such as adding new product with its details and viewing added product details, viewing order details and user details.
- User need to register using basic registration details and need to create valid login id and password.
- User can login using valid login credentials in order to access the system.
- User can view products and chat with the AI bot for price negotiation for a particular product.
- After negotiation, user can buy the product from the website and pay.

Chapter 2 : SYSTEM ANALYSIS

Gantt Chart

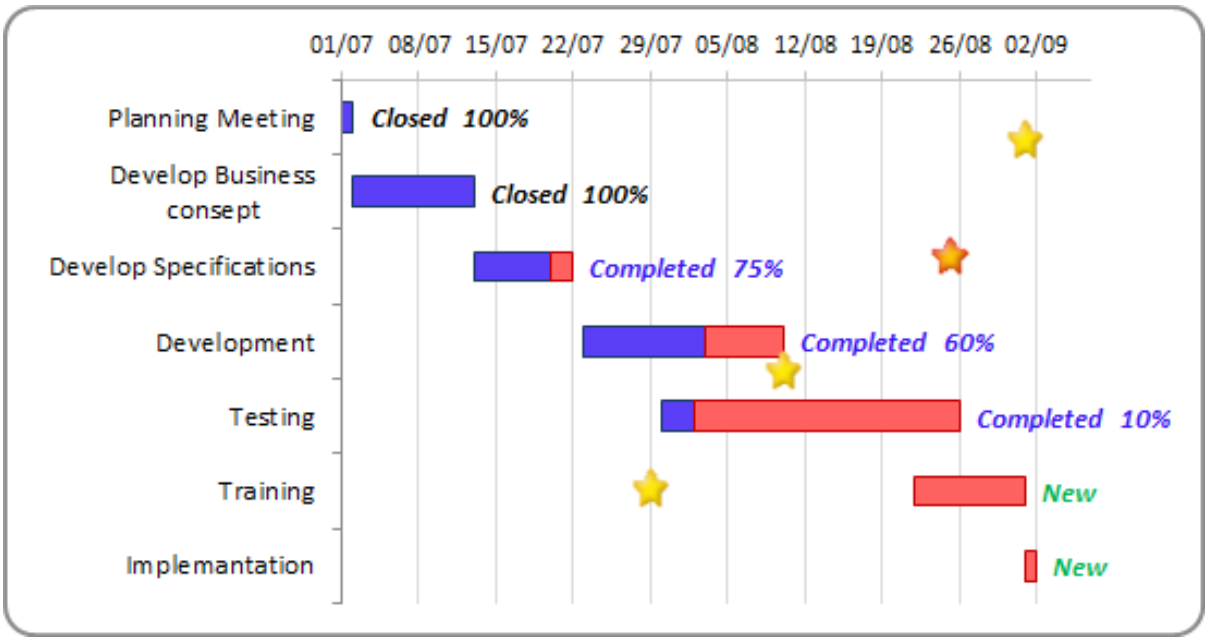


Fig. 2.1 Gantt Chart Analysis

Chapter 3: SYSTEM LIFE CYCLE

Waterfall Model

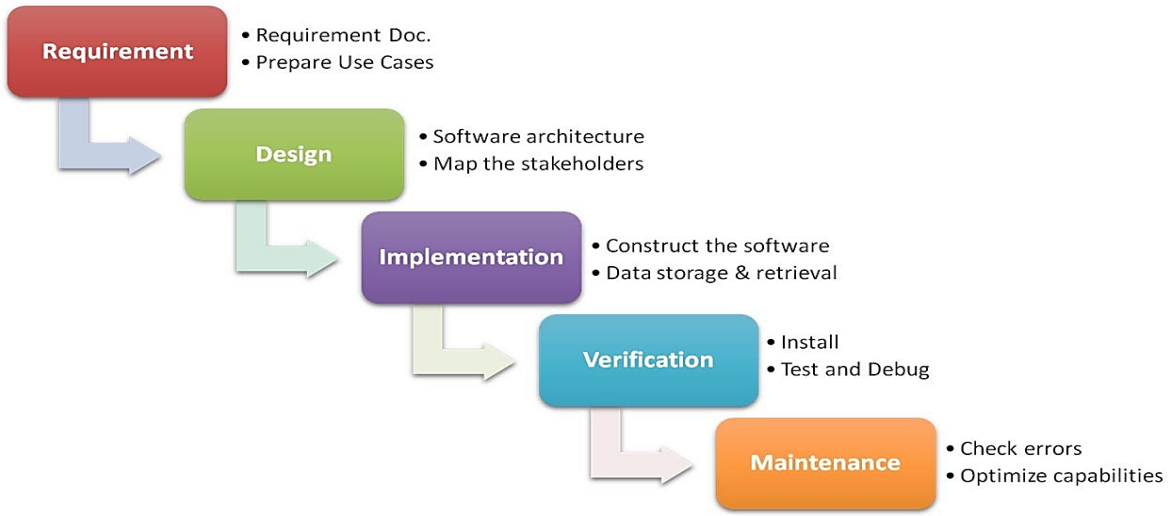


Fig. 3.1 System Waterfall Model

Description

The waterfall Model is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach that was used for software development.

Chapter 4: SYSTEM DESIGN

E-R Diagram

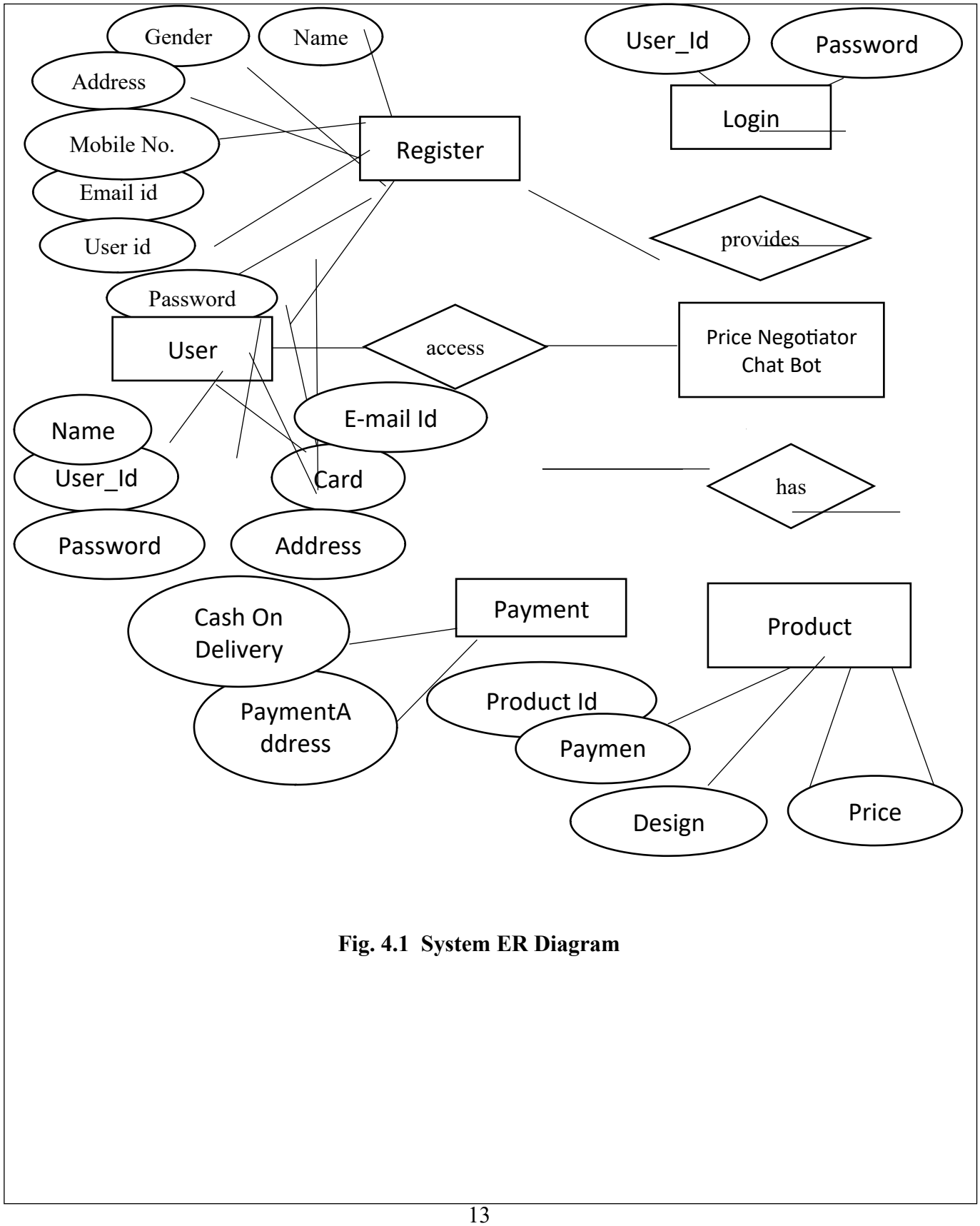


Fig. 4.1 System ER Diagram

Use Case Diagram

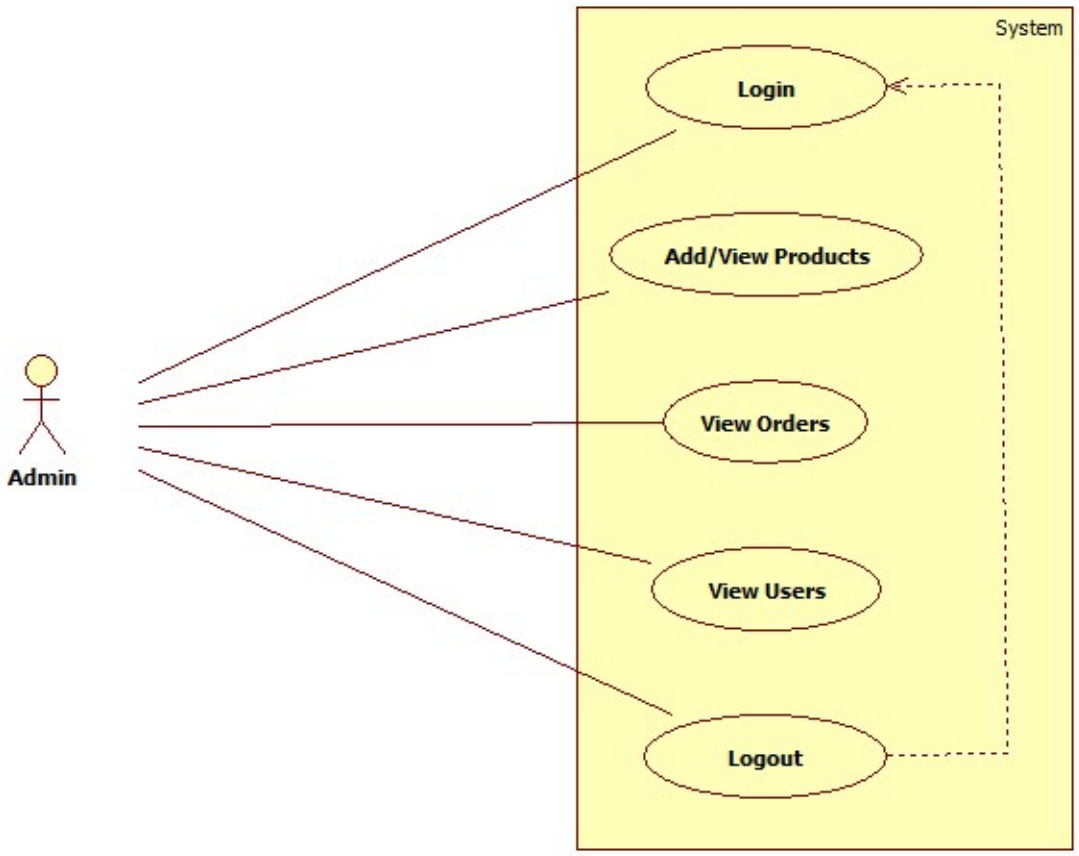


Fig.4.2 Use Case Diagram of Admin

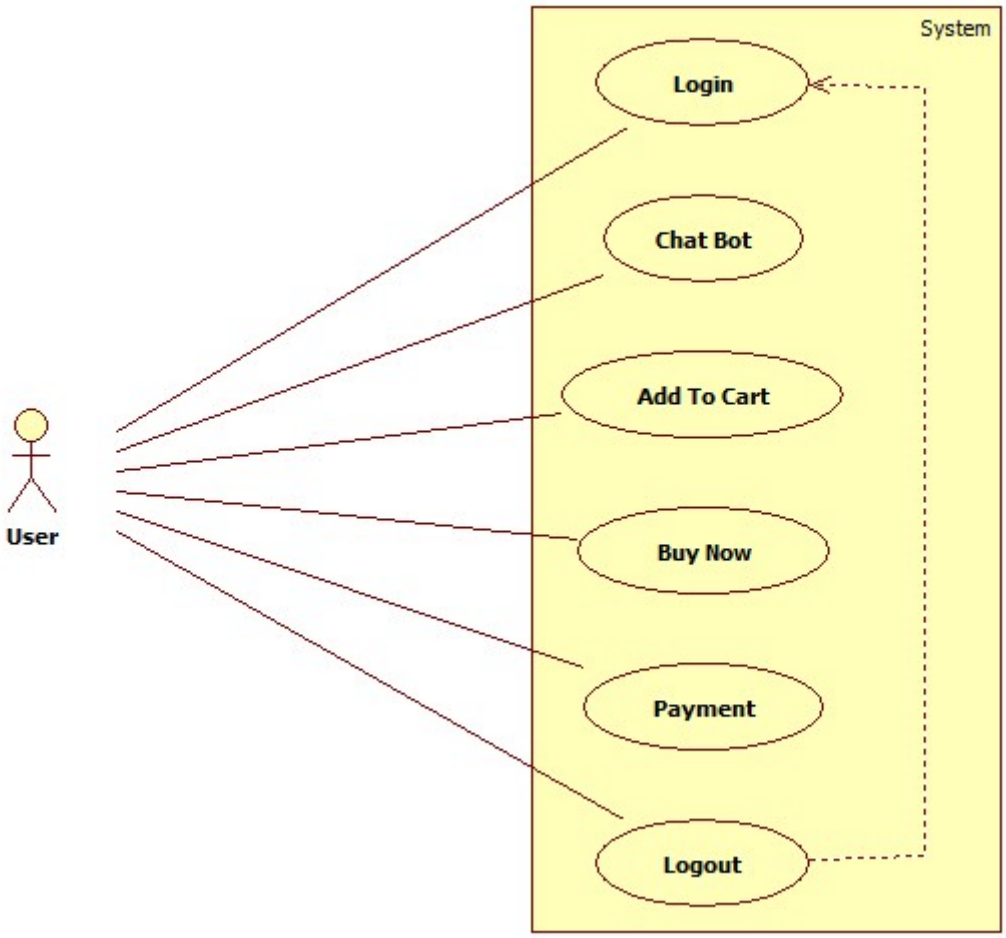


Fig.4.3 Use Case Diagram of User

Sequence Diagram

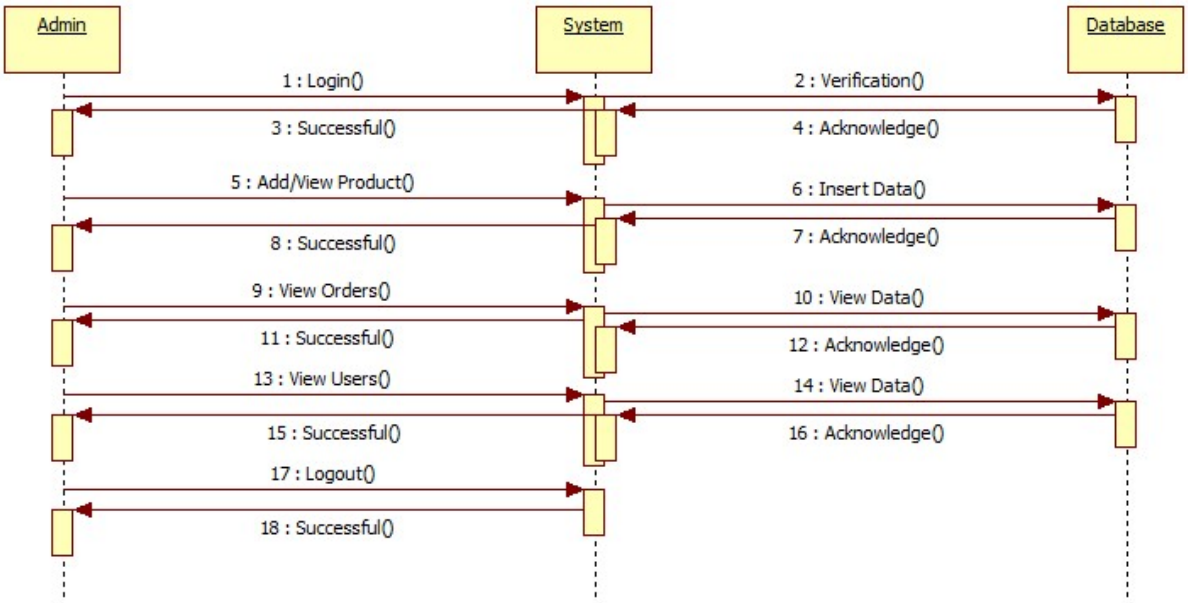


Fig.4.4 Sequence Diagram of Admin

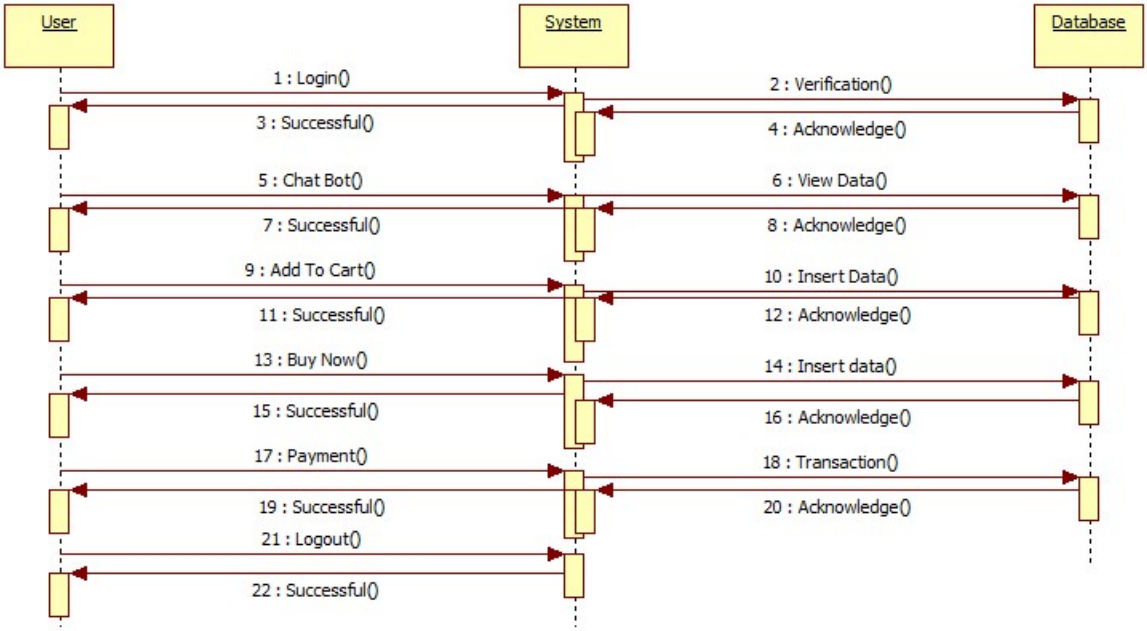


Fig. 4.5 Sequence Diagram of User

Activity Diagram

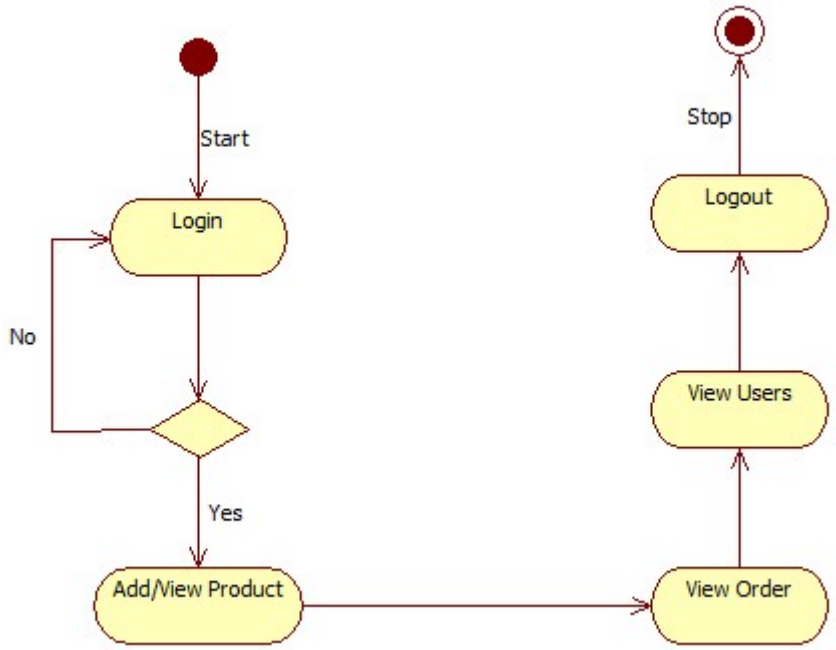


Fig.4.6 Activity Diagram of Admin

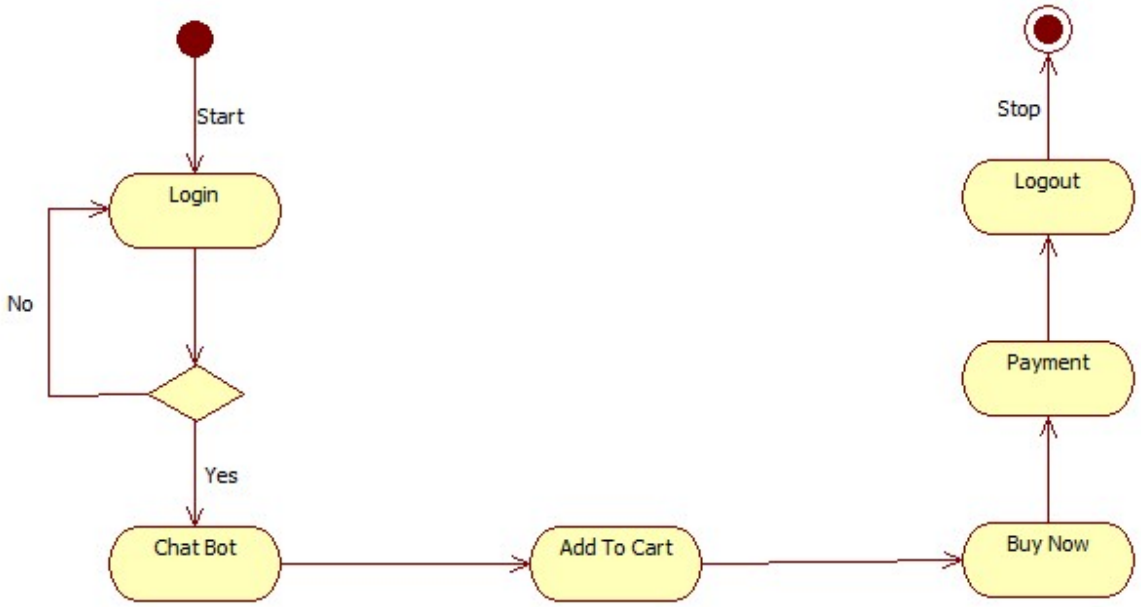


Fig.4.7 Activity Diagram of User

Class Diagram

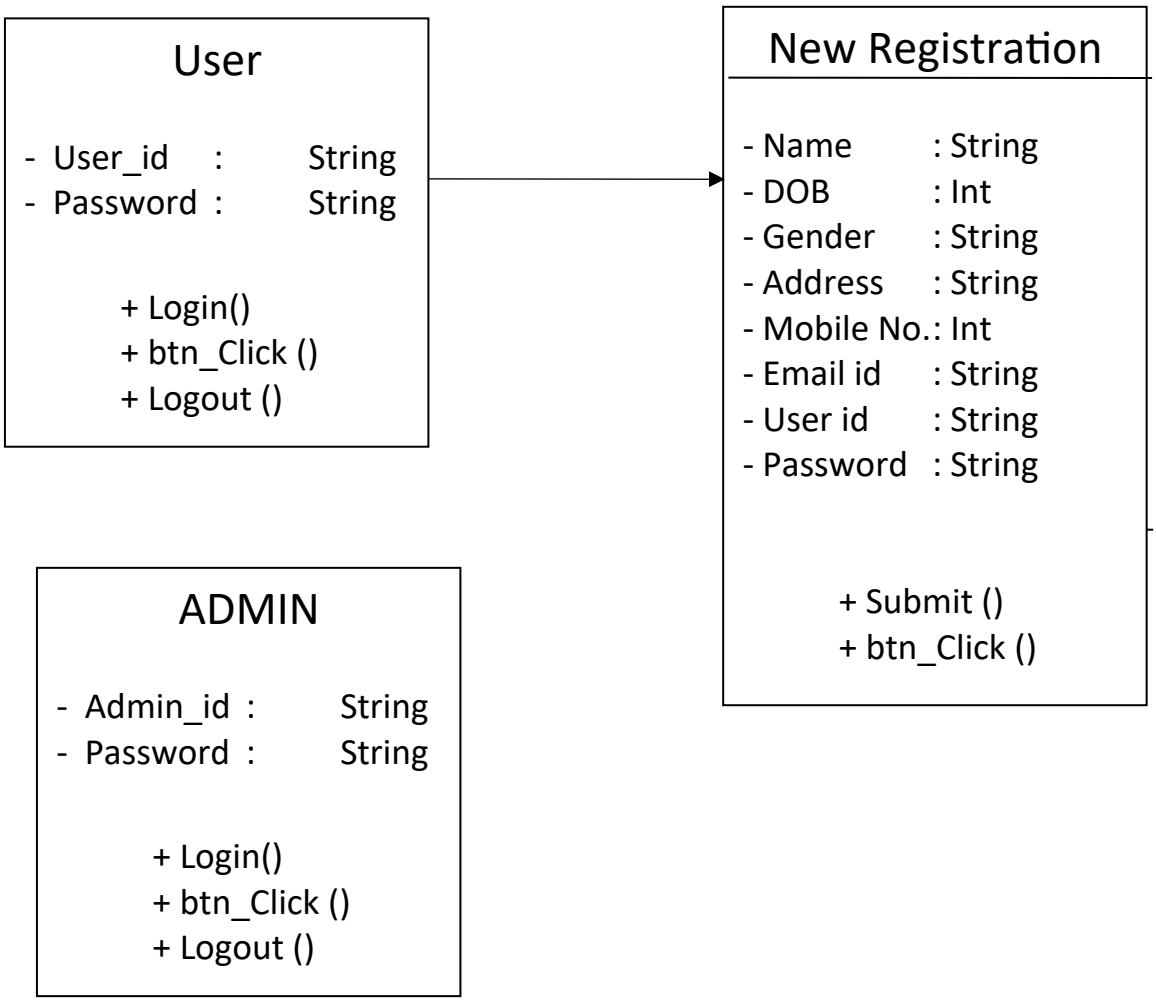


Fig. 4.8 Class Diagram

Data Flow Diagram

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD's is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

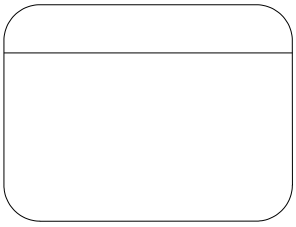
A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

DFD SYMBOLS:

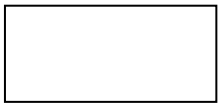
In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows

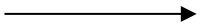
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data



Process that transforms data flow.



Source or Destination of data



Data flow

Data Store

CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD's:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

   A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

SAILENT FEATURES OF DFD's

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the data flows take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

CURRENT PHYSICAL:

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly, data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

CURRENT LOGICAL:

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

NEW LOGICAL:

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current

logical model while having additional functions, absolute function removal and inefficient flows recognized.

NEW PHYSICAL:

The new physical represents only the physical implementation of the new system.

RULES GOVERNING THE DFD'S

PROCESS

1) No process can have only outputs.
2) No process can have only inputs. If an object has only inputs than it must be a sink.
3) A process has a verb phrase label.

DATA STORE

1) Data cannot move directly from one data store to another data store, a process must move data.
2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3) A data store has a noun phrase label.

SOURCE OR SINK

The origin and /or destination of data.

1) Data cannot move direly from a source to sink it must be moved by a process
2) A source and /or sink has a noun phrase land

DATA FLOW

1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later it usually indicated however by two separate arrows since these happen at different type.

2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3) A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4) A Data flow to a data store means update (delete or change).
5) A data Flow from a data store means retrieve or use.

Data Flow Diagrams

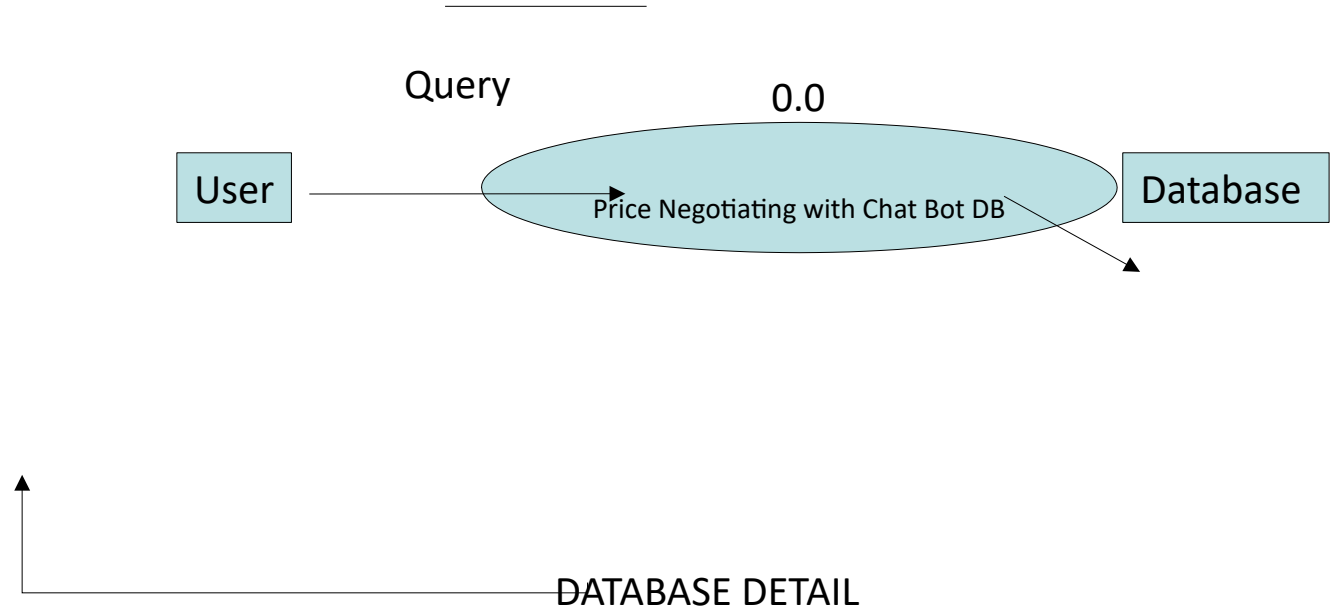


Fig. 4.9 Data Flow Diagram

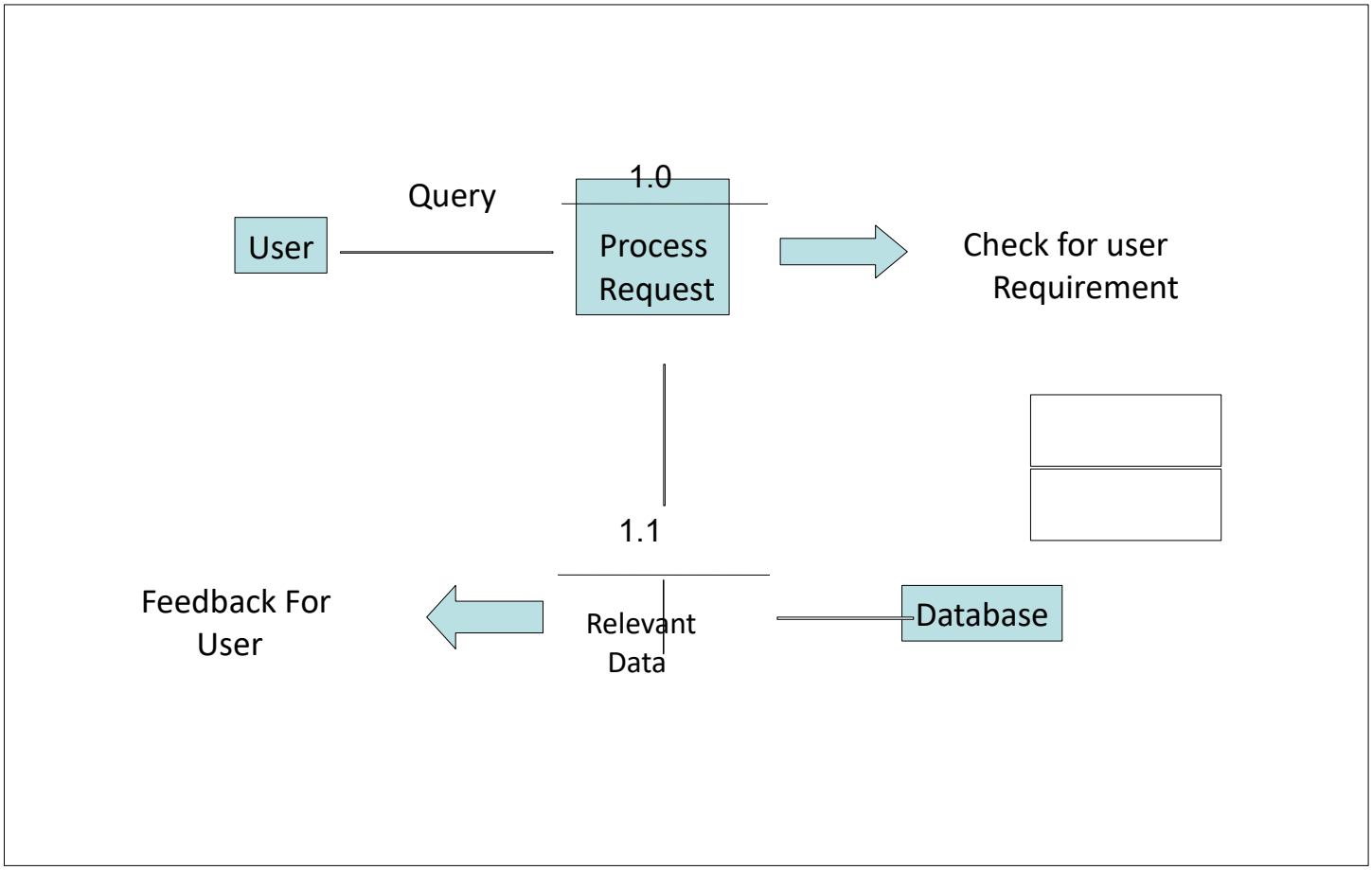


Fig.4.10 Data Flow Diagram

LEVEL 2 DFD: PREDICTION

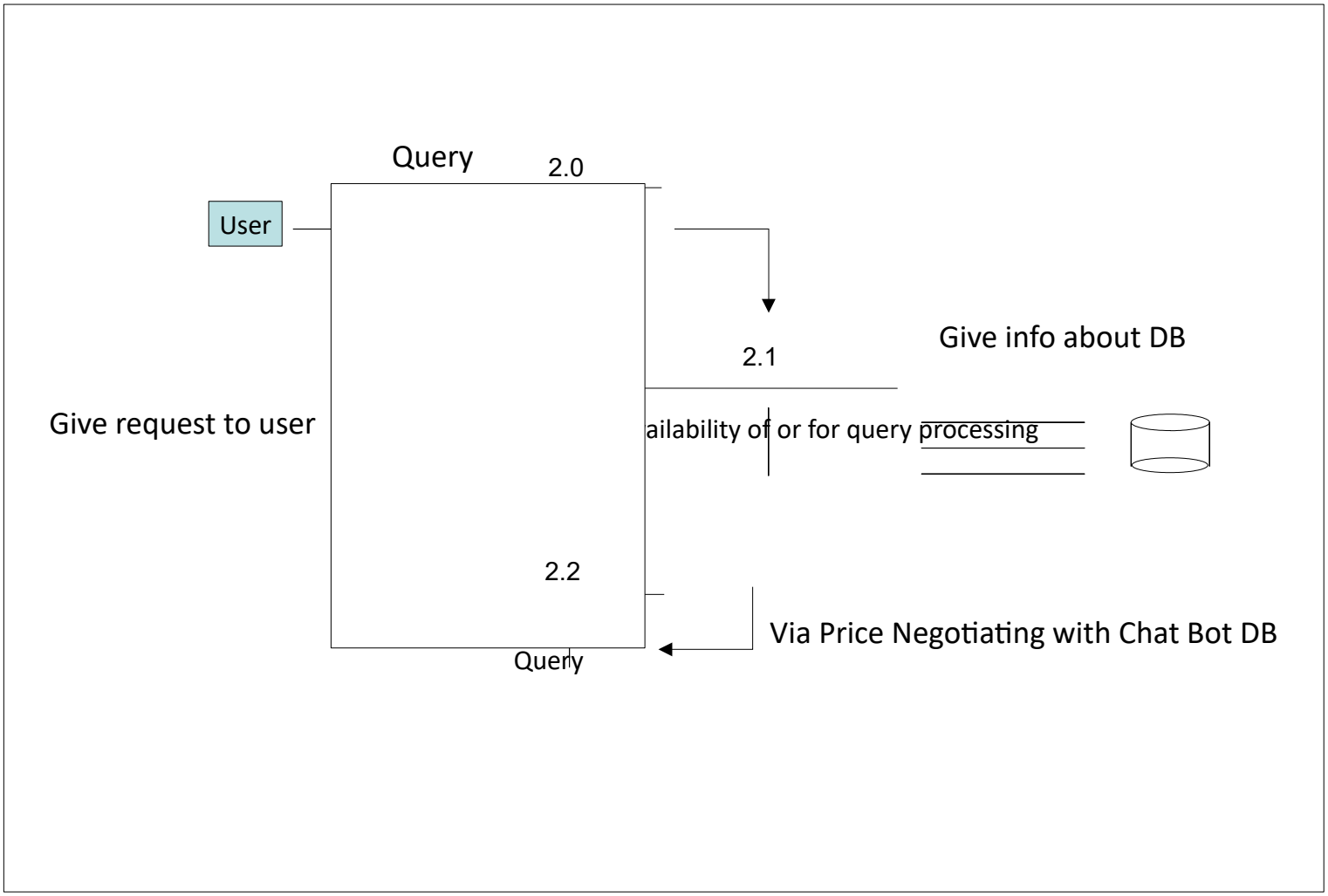


Fig.4.11 DFD Prediction

System Architecture

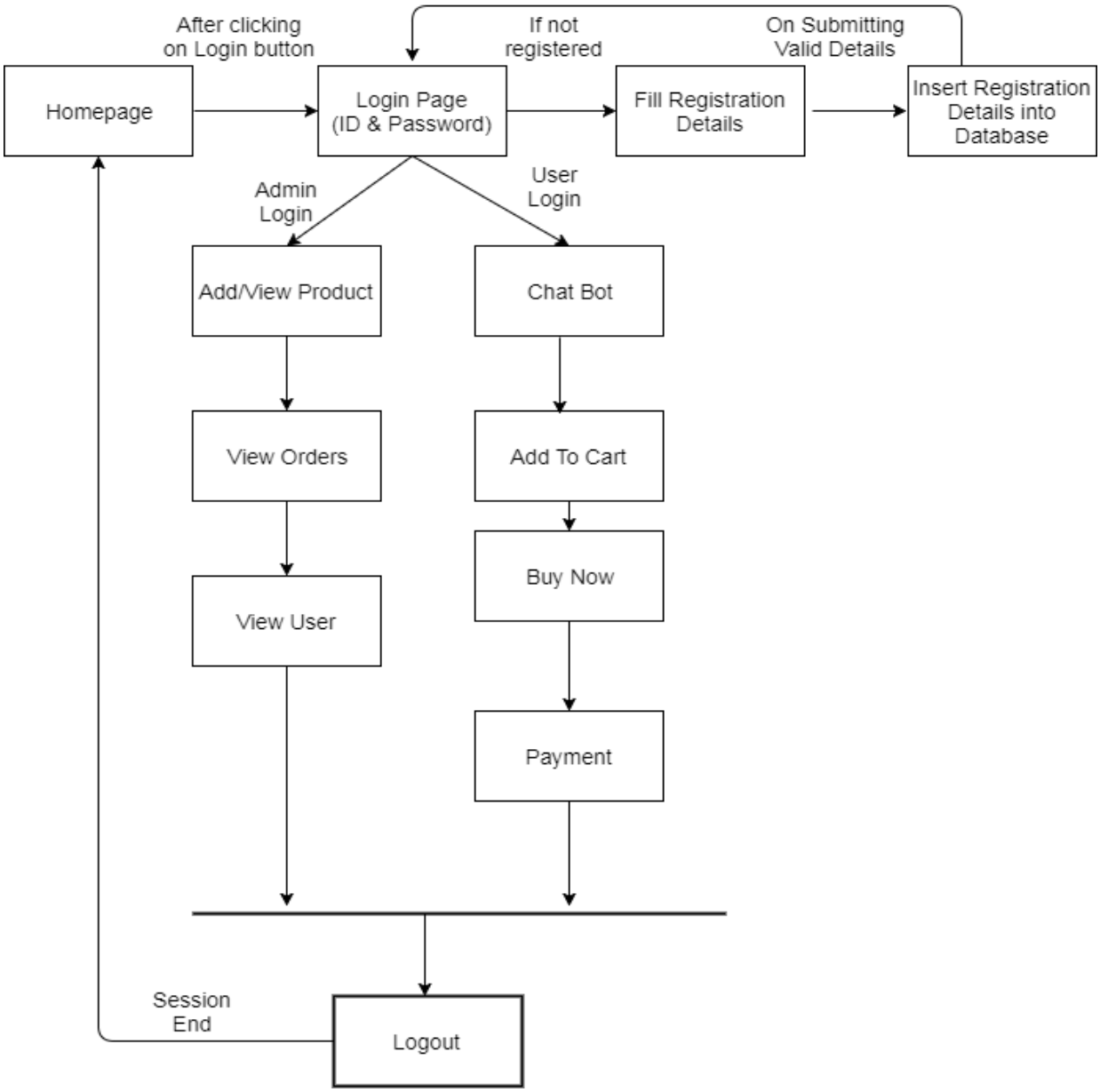


Fig. 4.12 System Architecture

Chapter 5: Snapshots

User interaction:

Fig.5.1 Main Page

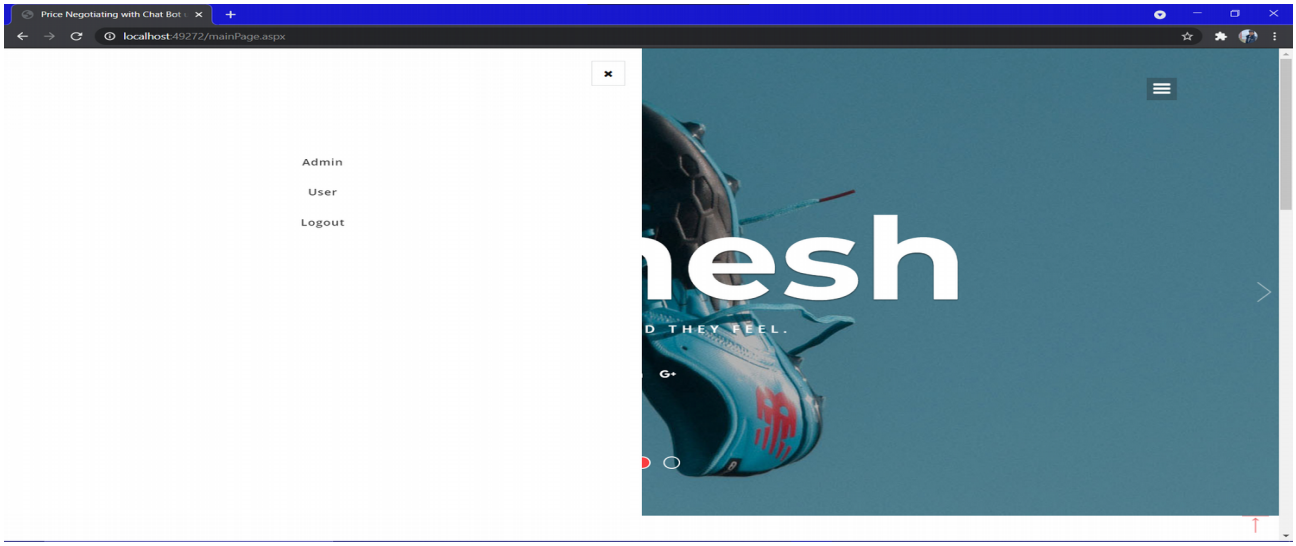


Fig. 5.2 Sidebar

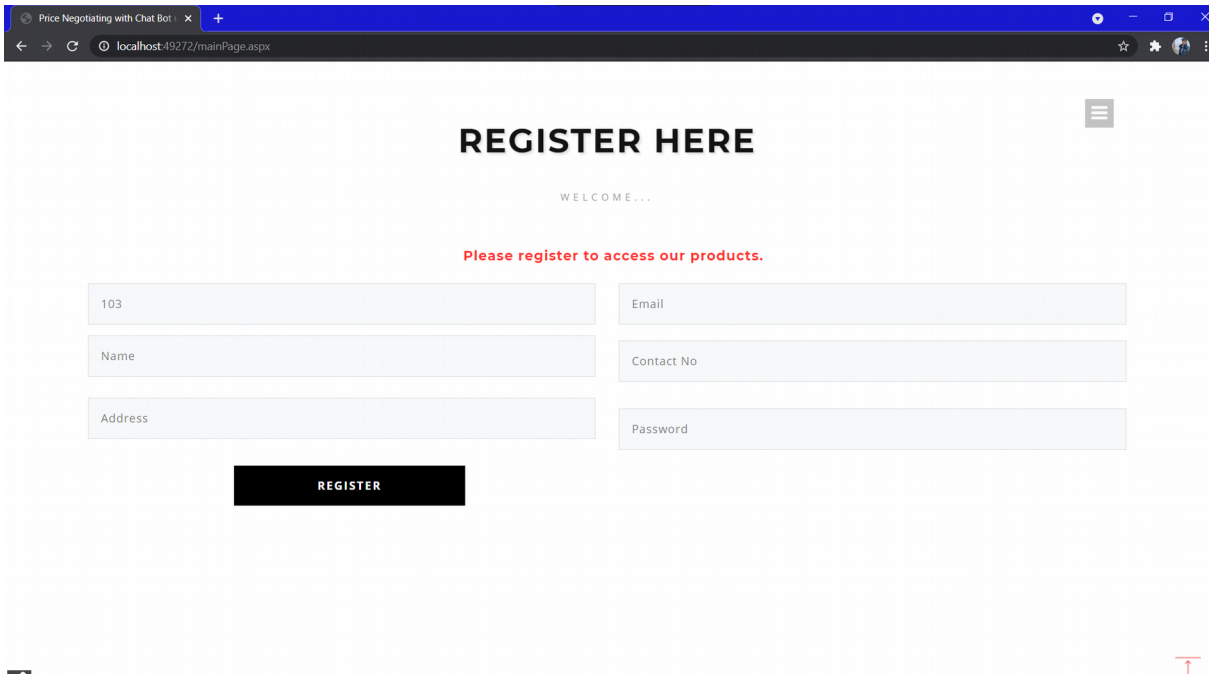


Fig.5.3 Registration Page

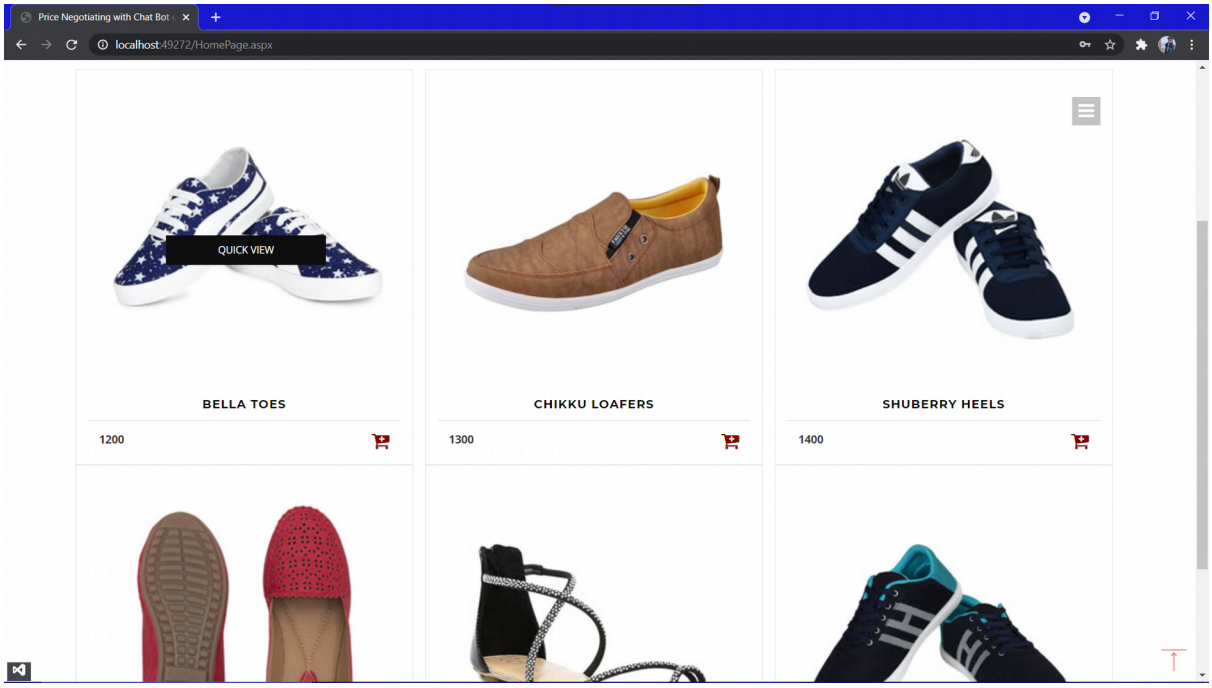


Fig.5.4 Catalog Page

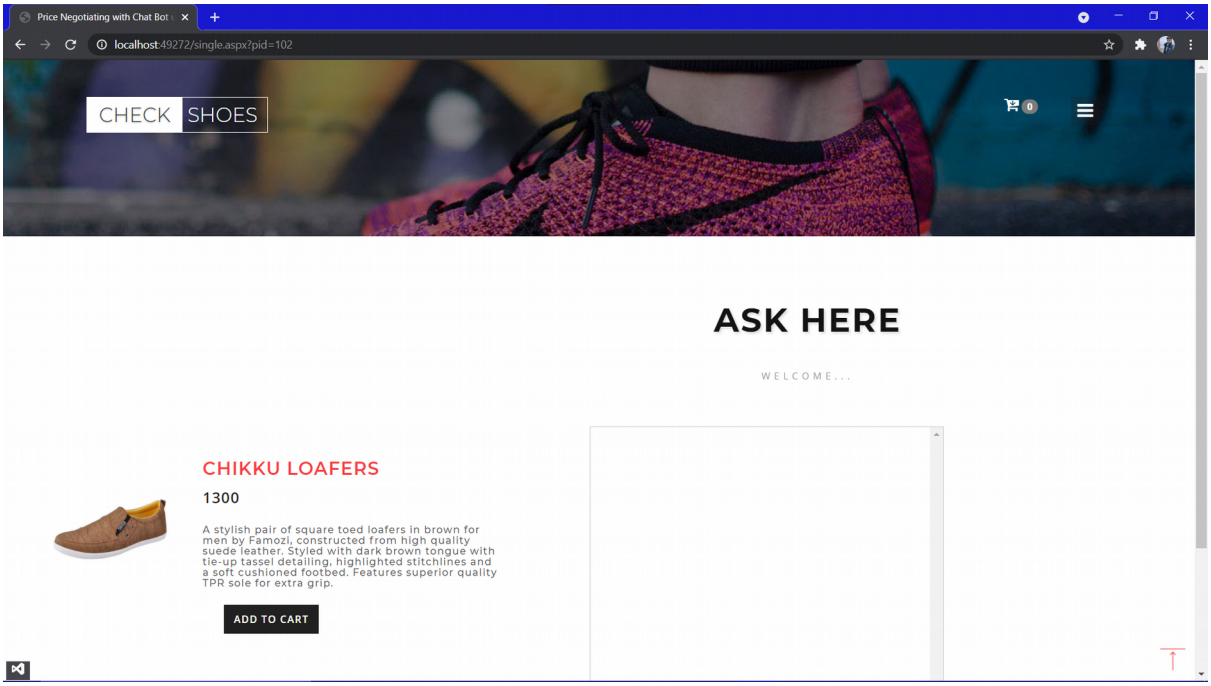


Fig.5.5 Item Page

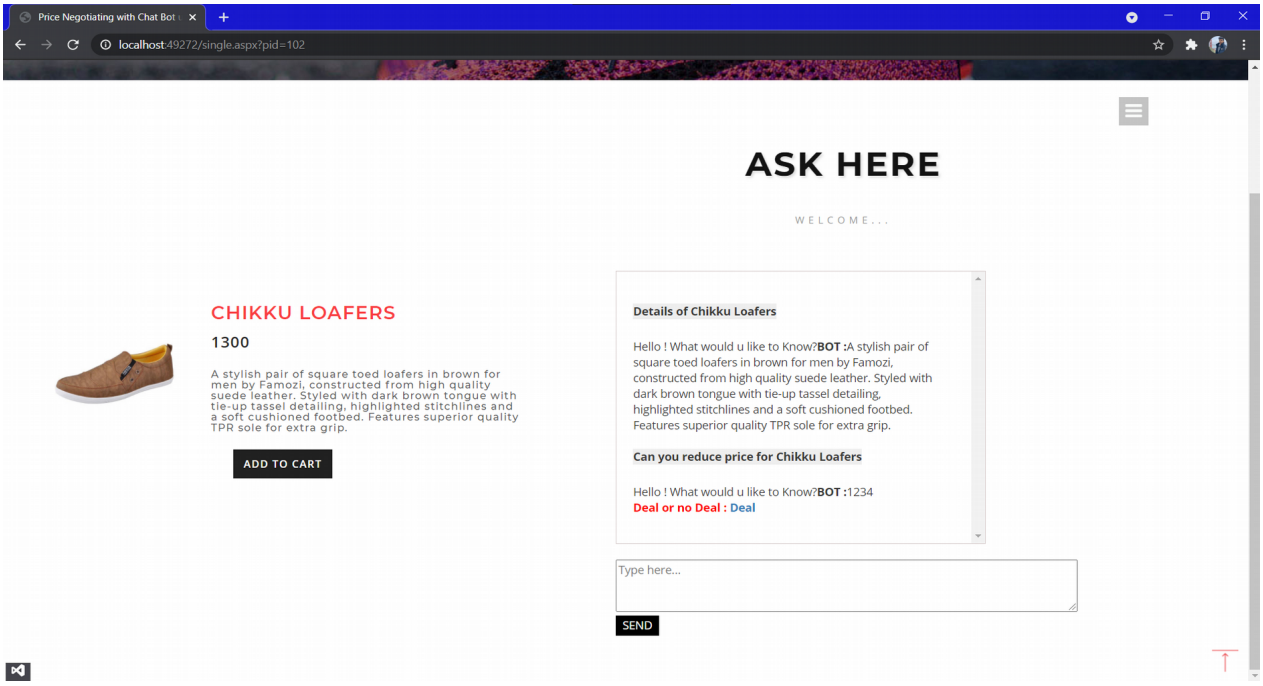


Fig.5.6 Chat bot Reply

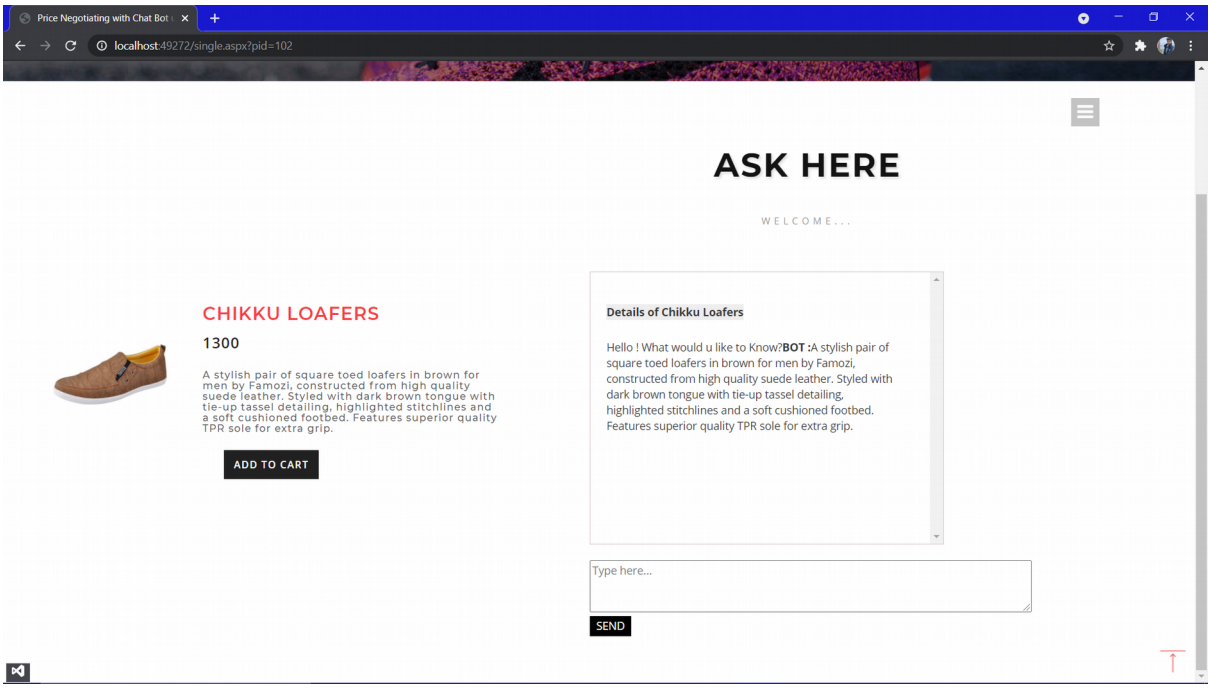


Fig.5.7 Chat bot Reply

Fig.5.8 Payment Page

Admin Interaction:

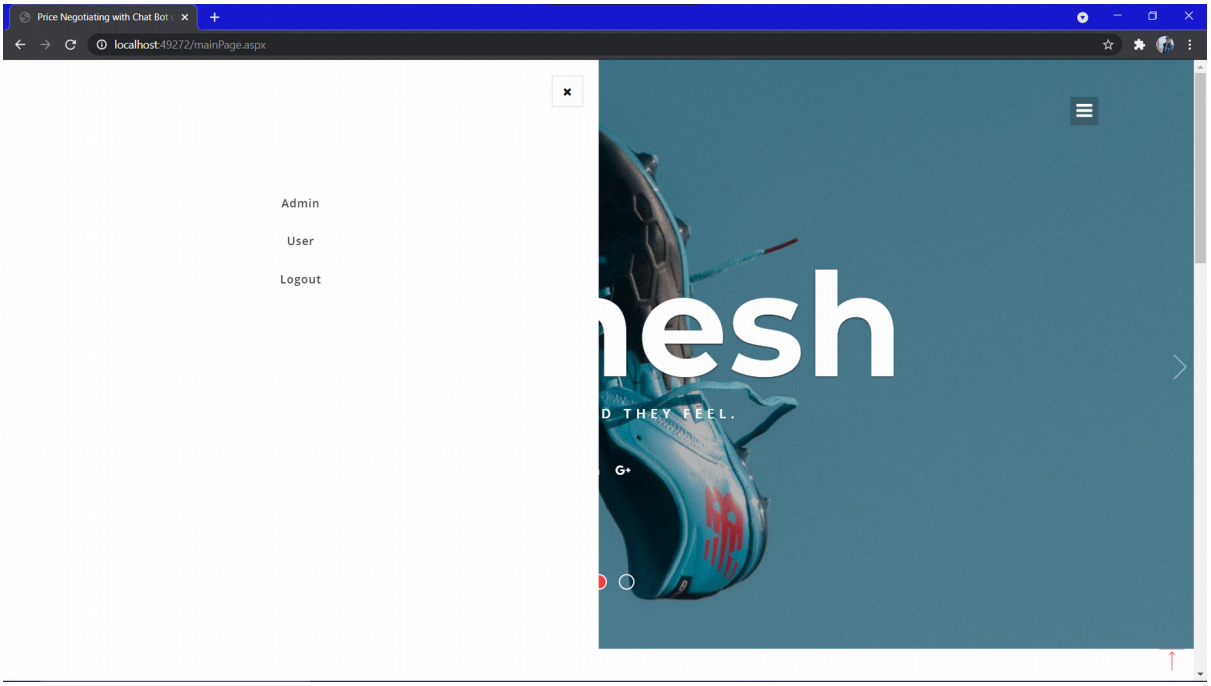


Fig.5.9 Admin Interaction

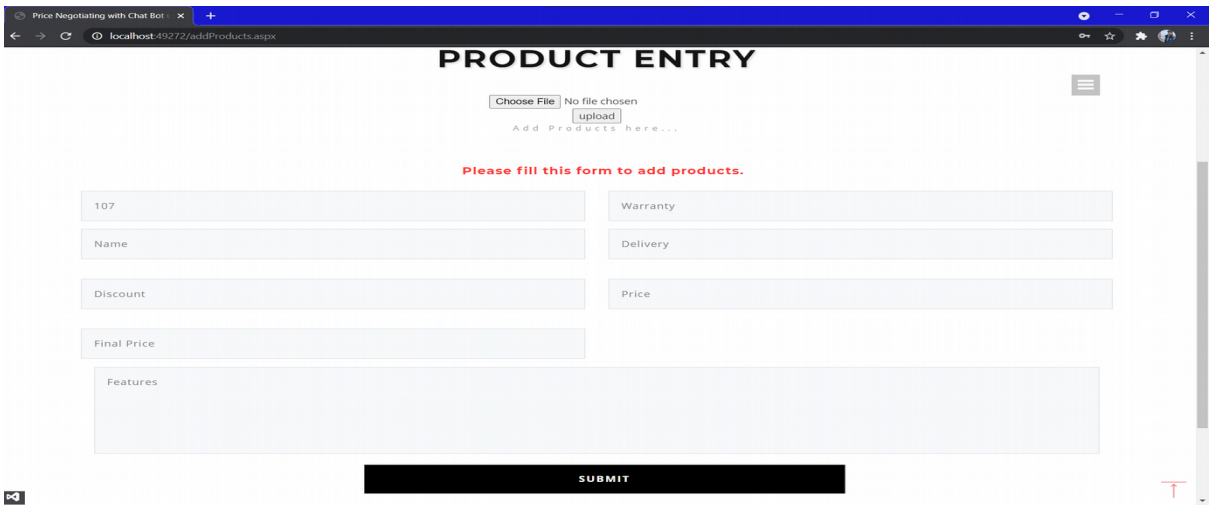


Fig.5.10 Product Entry

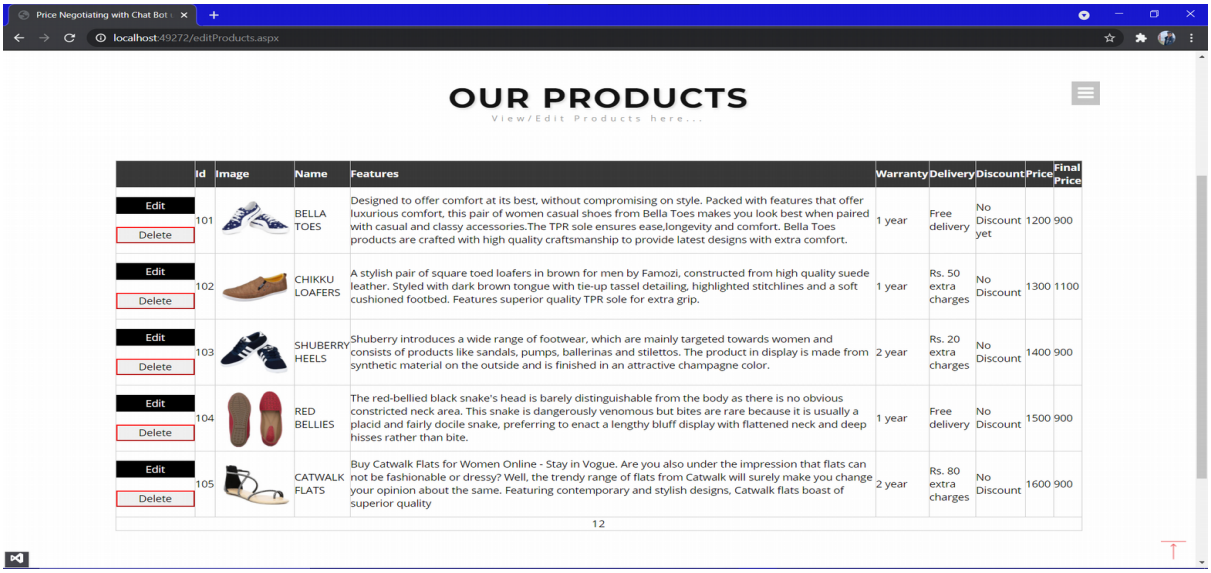


Fig.5.11 Product Entry

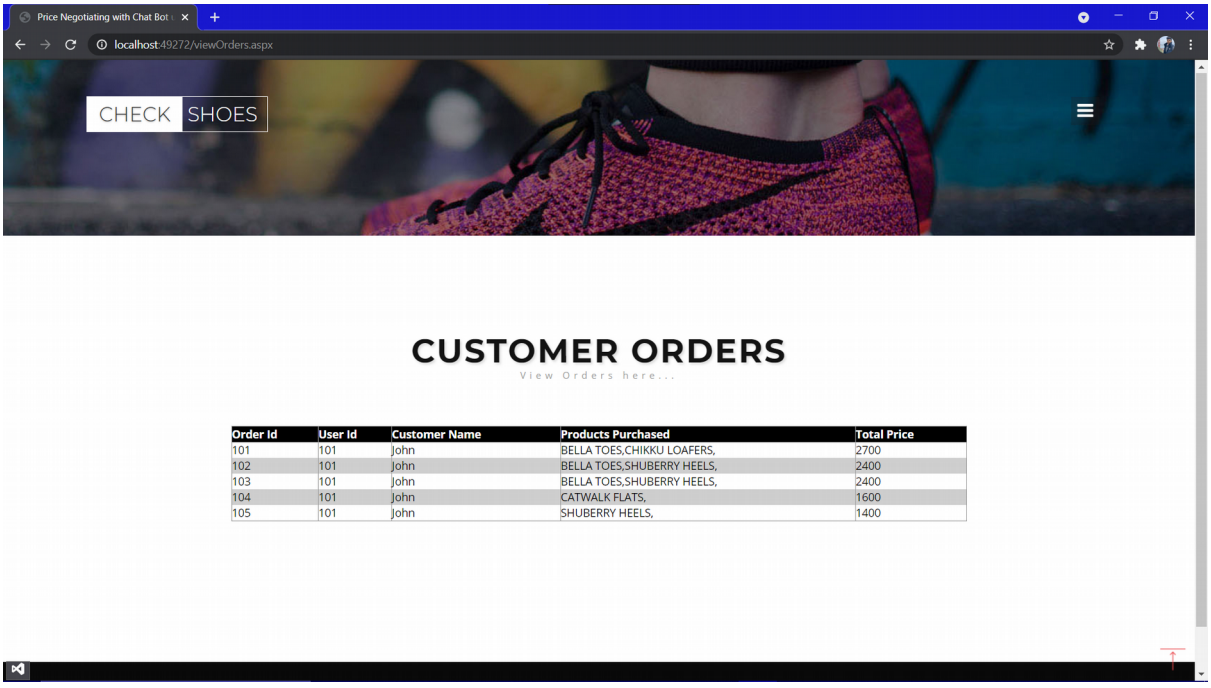


Fig.5.12 Order Details

Chapter 6: SYSTEM IMPLEMENTATION

Project Implementation Technology

The Project is loaded in Visual Studio 2019. We used Visual Studio for Design and coding of project. Created and maintained all databases into SQL Server 2020 in that we create tables, write query for store data or record of project.

❖ Hardware Requirement:

- i5 Processor Based Computer or higher
- Memory: 8 GB RAM
- Hard Drive: 50 GB
- Monitor
- Internet Connection

❖ Software Requirement:

- Windows 10
- Visual studio 2019.
- SQL Server 2020.

OVERVIEW OF TECHNOLOGIES USED

Front End Technology

Microsoft .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 remote, 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.

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.

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.

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.

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 post back 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.

Crystal Reports

Crystal Reports for Visual Basic .NET is the standard reporting tool for Visual Basic.NET; it brings the ability to create interactive, presentation-quality content — which has been the strength of Crystal Reports for years — to the .NET platform.

With Crystal Reports for Visual Basic.NET, you can host reports on Web and Windows platforms and publish Crystal reports as Report Web Services on a Web server.

To present data to users, you could write code to loop through record sets and print them inside your Windows or Web application. However, any work beyond basic formatting can be complicated: consolidations, multiple level totals, charting, and conditional formatting are difficult to program.

With Crystal Reports for Visual Studio .NET, you can quickly create complex and professional-looking reports. Instead of coding, you use the Crystal Report Designer interface to create and format the report you need. The powerful Report Engine processes the formatting, grouping, and charting criteria you specify.

Report Experts

Using the Crystal Report Experts, you can quickly create reports based on your development needs:

- Choose from report layout options ranging from standard reports to form letters, or build your own report from scratch.
- Display charts that users can drill down on to view detailed report data.
- Calculate summaries, subtotals, and percentages on grouped data.
- Show TopN or BottomN results of data.
- Conditionally format text and rotate text objects.

BACK END TECHNOLOGY:

About Microsoft SQL Server

Microsoft SQL Server is a Structured Query Language (SQL) based, client/server relational database. Each of these terms describes a fundamental part of the architecture of SQL Server.

Database

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

A database typically has two components: the files holding the physical database and the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including:

- Maintaining the relationships between data in the database.
- Ensuring that data is stored correctly and that the rules defining data relationships are not violated.
- Recovering all data to a point of known consistency in case of system failures.

Relational Database

There are different ways to organize data in a database but relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory).

When organizing data into tables, you can usually find many different ways to define tables. Relational database theory defines a process, normalization, which ensures that the set of tables you define will organize your data effectively.

Client/Server:

In a client/server system, the server is a relatively large computer in a central location that manages a resource used by many people. When individuals need to use the resource, they connect over the network from their computers, or clients, to the server.

Examples of servers are: In a client/server database architecture, the database files and DBMS software reside on a server. A communications component is provided so applications can run on separate clients and communicate to the database server over a network. The SQL Server

communication component also allows communication between an application running on the server and SQL Server.

Server applications are usually capable of working with several clients at the same time. SQL Server can work with thousands of client applications simultaneously. The server has features to prevent the logical problems that occur if a user tries to read or modify data currently being used by others.

While SQL Server is designed to work as a server in a client/server network, it is also capable of working as a stand-alone database directly on the client. The scalability and ease-of-use features of SQL Server allow it to work efficiently on a client without consuming too many resources.

Structured Query Language (SQL)

To work with data in a database, you must use a set of commands and statements (language) defined by the DBMS software. There are several different languages that can be used with relational databases; the most common is SQL.

Both the American National Standards Institute (ANSI) and the International Standards Organization (ISO) have defined standards for SQL. Most modern DBMS products support the Entry Level of SQL-92, the latest SQL standard (published in 1992).

SQL Server Features

Microsoft SQL Server supports a set of features that result in the following benefits:

Ease of installation, deployment, and use

SQL Server includes a set of administrative and development tools that improve your ability to install, deploy, manage, and use SQL Server across several sites.

Scalability

The same database engine can be used across platforms ranging from laptop computers running Microsoft Windows® 95/98 to large, multiprocessor servers running Microsoft Windows NT®, Enterprise Edition.

Data warehousing

SQL Server includes tools for extracting and analyzing summary data for online analytical processing (OLAP). SQL Server also includes tools for visually designing databases and analyzing data using English-based questions.

System integration with other server software

SQL Server integrates with e-mail, the Internet, and Windows.

Databases

A database in Microsoft SQL Server consists of a collection of tables that contain data, and other objects, such as views, indexes, stored procedures, and triggers, defined to support activities performed with the data.

The data stored in a database is usually related to a particular subject or process, such as inventory information for a manufacturing warehouse.

SQL Server can support many databases, and each database can store either interrelated data or data unrelated to that in the other databases. For example, a server can have one database that stores personnel data and another that stores product-related data. Alternatively, one database can store current customer order data, and another; related database can store historical customer orders that are used for yearly reporting. Before you create a database, it is important to understand the parts of a database and how to design these parts to ensure that the database performs well after it is implemented.

Normalization Theory:

Relations are to be normalized to avoid anomalies. In insert, update and delete operations. Normalization theory is built around the concept of normal forms. A relation is said to be in a particular form if it satisfies a certain specified set if constraints. To decide a suitable logical structure for given database design the concept of normalization, which are briefly described below.

1. **1st Normal Form (1 N.F):** A relation is said to be in 1 NF is and only if all unaligned domains contain values only. That is the fields of an n-set should have no group items and no repeating groups.
2. **2nd Normal Form (2 N.F):** A relation is said to be in 2 NF is and only if it is in 1 NF and every non key attribute is fully dependent on primary key. This normal takes care of functional dependencies on non-key attributes.
3. **3rd Normal Form (3 N.F):** A relation is said to be in 3 NF is and only if it is in 2 NF and every non key attribute is non transitively dependent on the primary key. This normal form avoids the transitive dependencies on the primary key.

4. **Boyce code Normal Form (BCNF):** This is a stronger definition than that of NF. A relation is said to be in BCNF if and only if every determinant is a Candidate key.

5. **4th Normal Form (4 NF):** A relation is said to be in 4 NF if and only if whenever there exists a multi valued dependency in a relation say A->->B then all of the relation are also functionally dependent on A (i.e. A->X for all attributes x of the relation.).
6. **5th Normal Form (5 NF) OR Projection Join Normal Form (PJNF):** A relation R is in 5 NF. if and only if every join dependency in R is implied by the candidate key on R. A relation can't be non-loss split into two tables but can be split into three tables. This is called Join Dependency.

Middleware Technology

Active Data Objects.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, Data Reader, and Data Adapter.

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 record set 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 Data Adapter 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 Data Adapter,

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, Data Reader and Data Adapter. 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.
- **Data Readers.** For reading a forward-only stream of data records from a SQL Server data source.
- **Datasets.** For storing, removing and programming against flat data, XML data and relational data.
- **Data Adapters.** 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).

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 Data Adapter, are provided for these scenarios. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
3. There is a lot more information about ADO.NET in the documentation.
4. 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.
5. Also, you can use a Dataset to bind to the data, move through the data, and navigate data relationships.

Chapter 7: CODING

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.SqlClient;
using System.Data;

public partial class single : System.Web.UI.Page
{
   //String flag = "0";
   SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\
MSSQLLocalDB;AttachDbFilename='C:\Users\prade\Desktop\FY project\Price Negotiating
with Chat Bot (Pradeep Ramesh)\Price Negotiating with Chat Bot
using Artificial Intelligence On E-Commerce Website\Project\ProjectN\App_Data\
priceNegotiate.mdf';Integrated Security=True");
   protected void Page_Load(object sender, EventArgs e)
   {
      string pid = Request.QueryString["pid"];
      string sel1 = "select * from products where pid='" + pid + "'";
      SqlDataAdapter sda = new SqlDataAdapter(sel1, con);
      DataSet ds = new DataSet();
      sda.Fill(ds);

      ListView1.DataSource = ds;
      ListView1.DataBind();
   }



   protected void Button1_Click(object sender, EventArgs e)
   {
      string uid = (string)Session["uid"];
      if (uid == null)
      {
         Page.ClientScript.RegisterStartupScript(GetType(), "msgbox", "<script>alert('Login
First!')</script>");

         Response.Redirect("mainPage.aspx#login");
      }
      else
      {
         string pid = Request.QueryString["pid"];
```

```
            string sel1 = "select * from products where pid='" + pid + "'";
            SqlDataAdapter sda = new SqlDataAdapter(sel1, con);
            DataSet ds = new DataSet();
            sda.Fill(ds);


            string pname = ds.Tables[0].Rows[0][2].ToString();
            string price = ds.Tables[0].Rows[0][7].ToString();

            string ins = "insert into cart values('" + uid + "','" + pid + "','" + pname + "','" + price +
"')";
            SqlCommand cmd = new SqlCommand(ins, con);
            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
            Response.Redirect("single.aspx?pid=" + pid);
        }
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        string uid = (string)Session["uid"];
        if (uid == null)
        {
            Page.ClientScript.RegisterStartupScript(GetType(), "msgbox", "<script>alert('Login
First!')</script>");

            Response.Redirect("mainPage.aspx#login");
        }
        else
        {
            Label1.Text += "<label style='background:#eee'>" + chat.Text +
"</label><br/><br/>";
            string query = chat.Text.ToLower();
            string sel1 = "";

            string[] words = query.Split(' ');
            string prodId = "";

            string sel0 = "select pid,name from products";
            SqlDataAdapter sda0 = new SqlDataAdapter(sel0, con);
            DataSet ds0 = new DataSet();
            sda0.Fill(ds0);

            int count = ds0.Tables[0].Rows.Count;

            for (int i = 0; i < count; i++)
            {
```

```
            foreach (string word in words)
            {
               string pname = ds0.Tables[0].Rows[i][1].ToString().ToLower();
               string srchWord = word.ToLower();
               if (pname.Contains(srchWord))
               {
                  prodId = ds0.Tables[0].Rows[i][0].ToString();
                  hiddenflag.Value = "1";
                  hiddenpid.Value = prodId;
                  break;
               }

            }

         }


         if (hiddenflag.Value == "1")
         {
            if (query.Contains("hello") || query.Contains("hi") || query.Contains("hey") ||
query.Contains("hii") || query.Contains("hiii"))
            {
               Label1.Text += "Hello ! What would u like to Know?";
            }
            if (query.Contains("feature") || query.Contains("features") ||
query.Contains("detail")||query.Contains("details") || query.Contains("description"))
            {
               sel1 = "select features from products  where pid='" + hiddenpid.Value + "'";
               SqlDataAdapter sda1 = new SqlDataAdapter(sel1, con);
               DataSet ds1 = new DataSet();
               sda1.Fill(ds1);

               if (ds1.Tables[0].Rows.Count > 0)
               {

                  Label1.Text += "<b>BOT :</b>" + ds1.Tables[0].Rows[0][0].ToString() +
"<br/><br/>";
               }

            }
            else
          if (query.Contains("warranty") || query.Contains("waranty"))
            {
               sel1 = "select warranty from products  where pid='" + hiddenpid.Value + "'";
               SqlDataAdapter sda1 = new SqlDataAdapter(sel1, con);
               DataSet ds1 = new DataSet();
               sda1.Fill(ds1);
```

```
                if (ds1.Tables[0].Rows.Count > 0)
                {

                    Label1.Text += "<b>BOT :</b>" + ds1.Tables[0].Rows[0][0].ToString() +
"<br/><br/>";
                }

            }

            else
            if (query.Contains("discount") || query.Contains("discounts") ||
query.Contains("free"))
            {
                sel1 = "select discount from products  where pid='" + hiddenpid.Value + "'";
                SqlDataAdapter sda1 = new SqlDataAdapter(sel1, con);
                DataSet ds1 = new DataSet();
                sda1.Fill(ds1);

                if (ds1.Tables[0].Rows.Count > 0)
                {

                    Label1.Text += "<b>BOT :</b>" + ds1.Tables[0].Rows[0][0].ToString() +
"<br/><br/>";
                }

            }

            else
            if (query.Contains("delivery") || query.Contains("deliver") ||
query.Contains("delivered"))
            {
                sel1 = "select delivery from products  where pid='" + hiddenpid.Value + "'";
                SqlDataAdapter sda1 = new SqlDataAdapter(sel1, con);
                DataSet ds1 = new DataSet();
                sda1.Fill(ds1);

                if (ds1.Tables[0].Rows.Count > 0)
                {
                    Label1.Text += "<b>BOT :</b>" + ds1.Tables[0].Rows[0][0].ToString() +
"<br/><br/>";
                }

            }

            else
            if (query.Contains("price") || query.Contains("cost") || query.Contains("costs") ||
query.Contains("rate") || query.Contains("costing"))
            {
```

```
            if (query.Contains("reduce") || query.Contains("lower") ||
query.Contains("lowered") || query.Contains("decrease") || query.Contains("less") ||
query.Contains("lesser"))
            {
                sel1 = "select price,lastPrice from products  where pid='" + hiddenpid.Value +
"'";
                SqlDataAdapter sda1 = new SqlDataAdapter(sel1, con);
                DataSet ds1 = new DataSet();
                sda1.Fill(ds1);
                if (Session["counter"] == null)
                {
                    Session["counter"] = 1;
                }
                else
                {
                    Session["counter"] = (int)Session["counter"] + 1;
                }
                int counter = (int)Session["counter"];
                int dif = 0; double div = 0.0; double pr = 0.0;
                int price = Convert.ToInt16(ds1.Tables[0].Rows[0][0]);
                int finalprice = Convert.ToInt16(ds1.Tables[0].Rows[0][1]);
                if (counter == 1)
                {
                    dif = price - finalprice;
                    div = dif / 3;
                    pr = price - Math.Ceiling(div);
                }
                if (counter == 2)
                {
                    dif = price - finalprice;
                    div = dif / 2;
                    pr = price - Math.Ceiling(div);
                }
                if (counter == 3)
                {
                    dif = price - finalprice;
                    div = dif / 1;
                    pr = price - Math.Ceiling(div);
                }
                if (counter > 3)
                {
                    pr = finalprice;
                    Label1.Text += "<b>BOT : This is our final price:</b>" + pr + "<br/>";
                    Label1.Text += "<label style='color:red;font-weight:bold'>Deal or no Deal :
<a href='payment.aspx?pid=" + hiddenpid.Value + "&price=" + pr +
"'>Deal</label><br/><br/>";
                    Session["counter"] = null;
                }
```

```
                else
                {
                    Label1.Text += "<b>BOT :</b>" + pr + "<br/>";
                    Label1.Text += "<label style='color:red;font-weight:bold'>Deal or no Deal :
<a href='payment.aspx?pid=" + hiddenpid.Value + "&price=" + pr +
"'>Deal</label><br/><br/>";

                }


            }
            else
            {
                sel1 = "select price from products where pid='" + hiddenpid.Value + "'";
                SqlDataAdapter sda1 = new SqlDataAdapter(sel1, con);
                DataSet ds1 = new DataSet();
                sda1.Fill(ds1);

                if (ds1.Tables[0].Rows.Count > 0)
                {
                    Label1.Text += "<b>BOT :</b>" + ds1.Tables[0].Rows[0][0].ToString() +
"<br/><br/>";
                }
            }

        }


        else
        {
            Label1.Text += "Kindly Contact our Customer Support Service Team at :<label
style='color:blue'>7777777777</label><br/>";

        }
    }
    if (hiddenflag.Value == "0")
    {
        Label1.Text += "Kindly Contact our Customer Support Service Team at :<label
style='color:blue'>7777777777</label><br/>";
    }


    chat.Text = "";
  }
 }
}
```

single.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" MasterPageFile="~/User.master"
CodeFile="single.aspx.cs" Inherits="single" %>

<asp:Content ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
   <table width="90%"><tr><td width="50%">
   <asp:ListView ID="ListView1" runat="server">
      <ItemTemplate>
   <div class="ads-grid_shop">
                <div class="shop_inner_inf">
                        <div class="col-md-4 single-right-left ">
                                <div class="grid images_3_of_2">
                                        <div class="flexslider">

                                                <ul class="slides" style="list-style-type: none;">
                                                        <li data-thumb="images/d2.jpg"
style="text-decoration:none">
                                                                <div class="thumb-image"> <img
src="<%#Eval("image") %>" data-imagezoom="true" class="img-responsive"> </div>
                                                        </li>

                                                </ul>
                                                <div class="clearfix"></div>
                                        </div>
                                </div>
                        </div>
                        <div class="col-md-8 single-right-left simpleCart_shelfItem">
                                <h3><%#Eval("name") %></h3>
                                <p><span class="item_price"><%#Eval("price") %></span>

                                </p>
                                <div class="rating1">

                                </div>
                                <div class="description">
                                        <h5><%#Eval("features") %></h5>

                                </div>


                                        </div>


                                </div>
```

```
                    </div>
                    <div class="clearfix"> </div>
                    <!--/tabs-->



                        <!-- //womens -->
                        <div class="clearfix"></div>
                    </div>
                    <!--//new_arrivals-->


                </div>
            </ItemTemplate>
            </asp:ListView>
          <center>
       <asp:Button ID="Button1" runat="server" Text="Add to cart"  class="button add"
OnClick="Button1_Click"/>
                    <br /><br /><br /><br /></center>
        </td>

        <td width="50%">
            <div class="ads-grid_shop">
                <div class="shop_inner_inf">
                    <h3 class="head">ASK Here</h3>
            <br />
            <br />

                    <p class="head_para">WELCOME...</p>
                    <div class="inner_section_w3ls">
                        <div class="col-md-12 contact_grid_right">
                            <div class="col-md-6 col-sm-6
contact_left_grid">
                                <asp:HiddenField id="hiddenflag"
runat="server" Value="0"/>
                                <asp:HiddenField id="hiddenpid"
runat="server" Value="0"/>

                    <asp:Panel ID="Panel1" runat="server" ScrollBars="Vertical"
Height="347px" Width="451px" style="padding: 40px 40px 10px 20px; border: 1px solid
#d7cbcb;">
                    <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
                       </asp:Panel>
                    <br />
                                <asp:TextBox ID="chat" runat="server"
TextMode="MultiLine" Rows="3" Columns="75" placeholder="Type
here..."></asp:TextBox>
```

```
                    <asp:Button ID="Button2" runat="server" Text="SEND"
OnClick="Button2_Click" BackColor="Black" BorderColor="Black" ForeColor="White"></
asp:Button>
                     <br />
                    <br /><br /><br />
                                                    </div></div></div></div></div>
      </td>
        </tr></table>

   </asp:Content>
```

FEASIBILITY REPORT

Feasibility Study is a high level capsule version of the entire process intended to answer a number of questions like: What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? Feasibility study is conducted once the problem clearly understood. Feasibility study is necessary to determine that the proposed system is Feasible by considering the technical, Operational, and Economical factors. By having a detailed feasibility study the management will have a clear-cut view of the proposed system.

The following feasibilities are considered for the project in order to ensure that the project is variable and it does not have any major obstructions. Feasibility study encompasses the following things:

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

In this phase, we study the feasibility of all proposed systems, and pick the best feasible solution for the problem. The feasibility is studied based on three main factors as follows.

❖ Technical Feasibility

In this step, we verify whether the proposed systems are technically feasible or not. i.e., all the technologies required to develop the system are available readily or not. Technical Feasibility determines whether the organization has the technology and skills necessary to carry out the project and how this should be obtained. The system can be feasible because of the following grounds:

- All necessary technology exists to develop the system.
- This system is too flexible and it can be expanded further.
- This system can give guarantees of accuracy, ease of use, reliability and the data security.
- This system can give instant response to inquire.

Our project is technically feasible because, all the technology needed for our project is readily available.

Operating System **:** Windows 7 or higher

Languages **:** Asp.Net with C#

Database System **:** MS-SQL Server 2008

Documentation Tool: MS - Word 2013

❖ Economic Feasibility

Economically, this project is completely feasible because it requires no extra financial investment and with respect to time, it's completely possible to complete this project in 6 months.

In this step, we verify which proposal is more economical. We compare the financial benefits of the new system with the investment. The new system is economically feasible only when the financial benefits are more than the investments and expenditure. Economic Feasibility determines whether the project goal can be within the resource limits allocated to it or not. It must determine whether it is worthwhile to process with the entire project or whether the benefits obtained from the new system are not worth the costs. Financial benefits must be equal or exceed the costs. In this issue, we should consider:

- The cost to conduct a full system investigation.
- The cost of h/w and s/w for the class of application being considered.
- The development tool.
- The cost of maintenance etc...

Our project is economically feasible because the cost of development is very minimal when compared to financial benefits of the application.

❖ Operational Feasibility

In this step, we verify different operational factors of the proposed systems like man-power, time etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement. Operational Feasibility determines if the proposed system satisfied user objectives could be fitted into the current system operation.

- The methods of processing and presentation are completely accepted by the clients since they can meet all user requirements.
- The clients have been involved in the planning and development of the system.
- The proposed system will not cause any problem under any circumstances.

Our project is operationally feasible because the time requirements and personnel requirements are satisfied. We are a team of four members and we worked on this project for three working months.

Chapter 8: TESTING

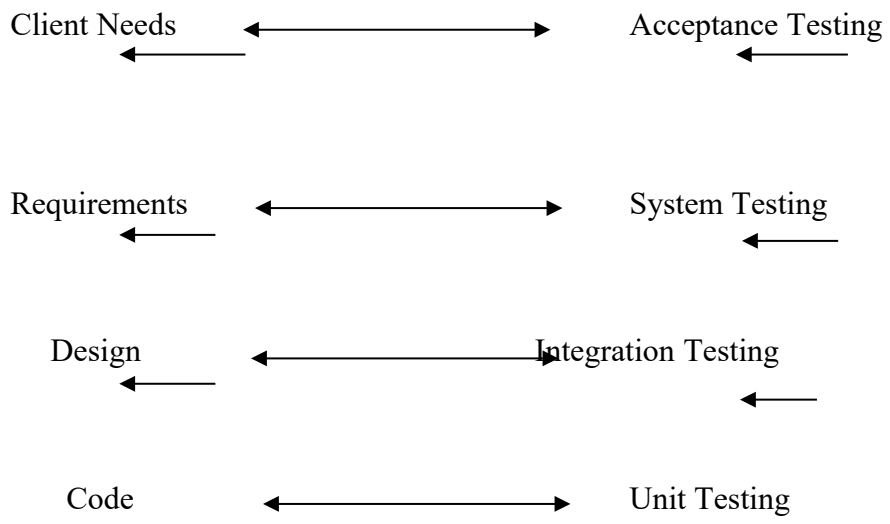
As the project is on bit large scale, we always need testing to make it successful. If each components work properly in all respect and gives desired output for all kind of inputs then project is said to be successful. So the conclusion is-to make the project successful, it needs to be tested.

The testing done here was System Testing checking whether the user requirements were satisfied. The code for the new system has been written completely using ASP .NET with C# as the coding language, C# as the interface for front-end designing. The new system has been tested well with the help of the users and all the applications have been verified from every nook and corner of the user.

Although some applications were found to be erroneous these applications have been corrected before being implemented. The flow of the forms has been found to be very much in accordance with the actual flow of data.

Levels of Testing

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are:



A series of testing is done for the proposed system before the system is ready for the user acceptance testing.

The steps involved in Testing are:

- ✓ **Unit Testing**

  Unit testing focuses verification efforts on the smallest unit of the software design, the module. This is also known as "Module Testing". The modules are tested separately. This testing carried out during programming stage itself. In this testing each module is found to be working satisfactorily as regards to the expected output from the module.

- ✓ **Integration Testing**

  Data can be grossed across an interface; one module can have adverse efforts on another. Integration testing is systematic testing for construction the program structure while at the same time conducting tests to uncover errors associated with in the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here correction is difficult because the isolation of cause is complicate by the vast expense of the entire program. Thus in the integration testing stop, all the errors uncovered are corrected for the text testing steps.

- ✓ **System testing**

  System testing is the stage of implementation that is aimed at ensuring that the system works accurately and efficiently for live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, then goal will be successfully achieved.

- ✓ **Validation Testing**

  At the conclusion of integration testing software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begins, validation test begins. Validation test can be defined in many ways. But the simple definition is that validation succeeds when the software

function in a manner that can reasonably expected by the customer. After validation test has been conducted one of two possible conditions exists.

One is the function or performance characteristics confirm to specifications and are accepted and the other is deviation from specification is uncovered and a deficiency list is created. Proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

- ✓ **Output Testing**

After performing validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated by the system under consideration. Here the output format is considered in two ways, one is on the screen and other is the printed format. The output format on the screen is found to be correct as the format was designed in the system designed phase according to the user needs.

For the hard copy also the output comes as the specified requirements by the users. Hence output testing does not result any corrections in the system.

- ✓ **User Acceptance Testing**

User acceptance of a system is the key factor of the success of any system. The system under study is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

Test Cases

User Login/Registration: To begin with login, user need to register by filling up basic registration details. There are multiple fields in registration page and every field has to fill by user. User cannot use character in the login id field.

Admin Login: -Admin login id and password is kept compulsory fields, and if the admin id or password doesn't match then it will show an error message.

VALIDATION CRITERIA

1. In each form, no field which is not null able should be left blank.
2. All numeric fields should be checked for non-numeric values. Similarly, text fields like names should not contain any numeric characters.
3. All primary keys should be automatically generated to prevent the user from entering any existing key.
4. Use of error handling for each Save, Edit, delete and other important operations.
5. Whenever the user Tabs out or Enter from a text box, the data should be validated and if it is invalid, focus should again be sent to the text box with proper message.

Chapter 9: ADVANTAGES OF SYSTEM

- Helps shops to automate selling online.
- Consumers can negotiate with the AI Bot with respect to product price.
- After price negotiation, payment link with negotiated price is sent over mail to consumer.
- Helps shops to take cc payments.
- Provides email confirmation on payment success.

DISADVANTAGES OF SYSTEM

- It requires active internet connection else, error may occur.
- Does not keep track of stock.

APPLICATIONS OF SYSTEM

- This system can be used in single shop.
- This system can be used to sell like chain of furniture shops from a single site.

FEATURES OF SYSTEM

1) Load Balancing:
   Since the system will be available only the admin logs in the amount of load on server will be limited to time period of admin access.

2) Easy Accessibility:
   Records can be easily accessed and store and other information respectively.

3) User Friendly:
   The website will be giving a very user-friendly approach for all user.

4) Efficient and reliable:

Maintaining the all secured and database on the server which will be accessible according the user requirement without any maintenance cost will be a very efficient as compared to storing all the customer data on the spreadsheet or in physically in the record books.

5) Easy maintenance:

Price Negotiating with Chat Bot website is design as easy way. So maintenance is also easy.

Chapter 10: CONCLUSION

This was our project of System Design about "**Price Negotiating with Chat Bot**" is a web application based on Asp .Net language. The Development of this system takes a lot of efforts from us. We think this system gave a lot of satisfaction to all of us. Though every task is never said to be perfect in this development field even more improvement may be possible in this application. We learned so many things and gained a lot of knowledge about development field. We hope this will prove fruitful to us.

Chapter 11: REFERENCES

- ✓ https://expertsystem.com/chat bot/
- ✓ Shabana Tadvi, Sultanat Rangari, and Ammar Rohe, L,2020. HR Based Interactive Chat bot (PowerBot).
- ✓ "ELIZA - Wikipedia,the free encyclopedia.," [Online]. Available:
- ✓ http://en.wikipedia.org/wiki/ELIZA. [Accessed March 2021].
- ✓ J.Weizenbaum, "ELIZA - A Computer Program for the study of Natural
- ✓ Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullende
- ✓ R. Carpenter, "jabberwacky - live chat bot," [Online]. Available:
- ✓ "PARRY - Wikipedia,the free encyclopedia.," [Online]. Available: http://en.wikipedia.org/wiki/PARRY.
- ✓ "Cleverbot - Wikipedia,the free encyclopedia.," [Online]. Available:
- ✓ http://en.wikipedia.org/wiki/Cleverbot. [Accessed March 2021].
- ✓ "Cleverbot.com," [Online]. Available: http://www.cleverbot.com/. [Accessed March 2021].
- ✓ http://ijcsit.com/docs/Volume%206/vol6issue02/ ijcsit20150602125.pdf
- ✓ An E-Commerce Website based Chatbot Siddharth Gupta#1, Deep Borkar#2, Chevelyn De Mello#3, Saurabh Patil#4