



POKHARA UNIVERSITY

Nobel College

Final Year Project

On

Inventory Management System

Submitted By

Anish Singh Maharjan (Roll No: 12080115)

Registration No: 2011-2-08-0031

Mandip Humagain (Roll No: 12080122)

Registration No: 2011-2-08-0038

In partial fulfillment of the requirement for the degree of

Bachelor of Computer Information System (BCIS)

Kathmandu, Nepal

January, 2016

POKHARA UNIVERSITY

Nobel College

Final Year Project

On

Inventory Management System

Submitted To

Nobel College,
Department of Management

Submitted By

Anish Singh Maharjan (Roll No: 12080115)

Registration No: 2011-2-08-0031

Mandip Humagain (Roll No: 12080122)

Registration No: 2011-2-08-0038

Under the supervision of

Mr. Dipesh Shrestha

Mr. Utsav Neupane

In partial fulfillment for the award of the Degree of
Bachelor in Computer Information System (BCIS)

January, 2016



Faculty of Management

POKHARA UNIVERSITY

RECOMMENDATION

This is to certify that the Final Year Project on
Inventory Management System

Submitted By

Anish Singh Maharjan (Roll No: 12080115)

Registration No: 2011-2-08-0031

Mandip Humagain (Roll No: 12080122)

Registration No: 2011-2-08-0038

Entitled

“INVENTORY MANAGEMENT SYSTEM”

Has been prepared and approved by this college.
This report is forwarded for examination to Pokhara University.

Mr. Kamal Pd. Regmi
Principal

Faculty of Management

Nobel College

DECLARATION

We hereby declare that the work reported in this project work on “**Inventory Management System**” submitted to Nobel College, Pokhara University is our original work done in the form of partial requirement for the degree of Bachelor’s in Computer Information System (BCIS) under the supervision of Mr. Dipesh Shrestha, Head of department (Management Faculty), Nobel College and supervisor for the final year project Mr. Utsav Neupane. The material contained in the report has not been submitted to any University or Institution for the award of any degree.

Anish Singh Maharjan (Roll No: 12080115)

Registration No: 2011-2-08-0031

Mandip Humagain (Roll No: 12080122)

Registration No: 2011-2-08-0038

Nobel College

Department Of Management

CERTIFICATE OF APPROVAL

We have the pleasure in forwarding the project of Mr. Anish Singh Maharjan (Roll No: 12080115) and Mr. Mandip Humagain (Roll No: 12080122) entitled “Inventory Management System” for the award of the degree of Bachelor of Computer Information System of this institute. Mr. Anish Singh Maharjan (Roll No: 12080115) and Mr. Mandip Humagain (Roll No: 12080122) have completed the project work for the full prescribed period under Pokhara University curriculum and the project embodied the result of his investigations conducted during the period they worked as a full time student of this department.

The Final year project has been approved by the following panel of examiners:

S.N	Name	Designation	Signature	Date
1	Mr. Dipesh Shrestha	Head of Department (Management Faculty)		
2	Mr Utsav Neupane	Supervisor		
3	Mr. Krian Kurmar Regmi	External Examiner		

ACKNOWLEDGEMENT

This project is prepared in the partial fulfillment of the requirement for the degree of Bachelor in Computer Information System (BCIS). The satisfaction and success of completion of this task would be incomplete without heartfelt thanks to people whose constant guidance, support and encouragement made this work successful. On doing this undergraduate project we have been fortunate to have help, support and encouragement from many people we would like to acknowledge them for their cooperation.

Our first thanks goes to Pokhara University for designing such a worthy syllabus and making us do this project. Our next batch of thanks goes to the faculty of Management of Nobel College without whose help our project would have been impossible. This list includes Principal of Nobel College, Mr. Kamal Pd. Regmi, Head of Department (Management Faculty), Mr. Dipesh Shrestha. Our very sincere and heartfelt thanks go to Mr. Utsav Neupane our project supervisors who constantly guided us through the project time period. Without his guidance, our project would have been impossible. Last but not the least we want to thank every direct and indirect hands that were involved in completion of this project.

Finally, our heartfelt thanks goes to Mr. Krian Kurmar Regmi, Training Manager of Leaf frog technology Nepal as an external examiner for our final year project and thanks for sharing his valuable time with us to evaluate our work and provide professional suggestion with us.

This project has been a wonderful experience where we have learnt and experienced many beneficial things.

With Regards

Anish Singh Maharjan (Roll No: 12080115)

Registration No: 2011-2-08-0031

Mandip Humagain (Roll No: 12080122)

Registration No: 2011-2-08-0038

ABSTRACT

This project is aimed at developing a desktop based application named Inventory Management System for managing the inventory system of any organization. The Inventory Management System (IMS) refers to the system and processes to manage the stock of organization with the involvement of Technology system. This system can be used to store the details of the inventory, stock maintenance, update the inventory based on the sales details, generate sales and inventory report daily or weekly based. This project is categorize individual aspects for the sales and inventory management system. In this system we are solving different problem affecting to direct sales management and purchase management. Inventory Management System is important to ensure quality control in businesses that handle transactions revolving around consumer goods. Without proper inventory control, a large retail store may run out of stock on an important item. A good inventory management system will alert the wholesaler when it is time to record. Inventory Management System is also on important means of automatically tracking large shipment. An automated Inventory Management System helps to minimize the errors while recording the stock.

LIST OF FIGURES

FIGURE 2.1: TIER ARCHITECTURE	5
FIGURE 3.6: GANTT CHART	10
FIGURE 4.1: IMS PROCESS FLOW DIAGRAM.....	11
FIGURE 4.2.1: IMS USE CASE DIAGRAM.....	13
FIGURE 5.2.1: .NET FRAMEWORK ARCHITECTURE.....	16
FIGURE 5.2.2: COMPILATION TO MANAGED CODE.....	17
FIGURE 6.6.1: ADMIN OR LOGIN PAGE.....	25
FIGURE 6.6.2: COMPANY/SHOP DETAILS.....	26
FIGURE 6.6.3: INVENTORY ENTRY SYSTEM	27
FIGURE 6.6.4: BACK UP AND RESTORE MENU	28
FIGURE 6.6.5: REPORT MENU	29
FIGURE 6.6.6: CREATING GODWOM	29
FIGURE 6.6.7: CREATING UNITS.....	30
FIGURE 6.6.8: CREATING PRODUCT TYPE.....	30
FIGURE 6.6.9: CREATING PRODUCT.....	31
FIGURE 6.6.10: INSERTING OPENING STOCK.....	31
FIGURE 6.6.11: PURCHASING PRODUCT FORM VENDOR.....	32
FIGURE 6.6.12 SALES PRODUCT TO CUSTOMER.....	32
FIGURE 6.6.13: PURCHASE RETURN.....	33
FIGURE 6.6.14: SALES RETURN	33
FIGURE: 6.6.15 REPORT OF CURRENT STOCK.....	34
FIGURE 6.6.16: BACK-UP DATA	34

LIST OF ABBREVIATIONS

IMS	Inventory Management System
DBMS	Database Management System
DFD	Data Flow Diagram
BCIS	Bachelor of Computer Information System
JIT	Just-In-Time
DML	Data Manipulation Language
DDL	Data Definition Language
DCL	Data Control Language
CLR	Common Language Runtime
SSMSE	SQL Server Management Studio Express
SQL	Structured Query Language

Table of Contents

RECOMMENDATION	ii
DECLARATION	iii
CERTIFICATE OF APPROVAL.....	iv
ACKNOWLEDGEMENT	v
ABSTRACT.....	vi
LIST OF FIGURES	vii
LIST OF ABBREVIATIONS.....	viii
CHAPTER-1: INTRODUCTION.....	1
1.1 Introduction to Inventory Management System	1
1.2 Literature Review.....	1
1.3 Problem Statement	2
1.4 Objective of the Project	2
1.4.1 Primary objective	2
1.4.2 Secondary objective	2
1.6 Features of Project	2
1.7 Scope of the Application.....	3
CHAPTER-2 BACKGROUND KNOWLEDGE	4
2.1 Architectural Review.....	4
2.1.1 Client tier	4
2.1.2 Business tier	4
2.1.3 Data tier.....	4
2.2 Database Theory.....	5
2.2.1 Relational Database	5
2.2.2 Structured Query Language (SQL)	6
2.3 ACID Property	6
CHAPTER-3: ANALYSIS AND DESIGN.....	8

3.1 Background Research	8
3.2 Requirement Analysis.....	8
3.3 IMS Requirement.....	8
3.4 Users Requirement.....	9
3.4.1 Admin	9
3.4.2 Inventory management.....	9
3.5 Feasibility Analysis.....	9
3.5.1 Economic Feasibility	9
3.5.2 Technical Feasibility	9
3.5.3 Operational Feasibility.....	10
3.5.4 Schedule Feasibility	10
3.6 Gantt chart.....	10
CHAPTER – 4: SYSTEM DESIGN.....	11
4.1 Process Flow Diagram	11
4.2 Use Case Diagram.....	11
4.2.1 Diagram Building Block	12
CHAPTER – 5: TOOLS AND TECHNOLOGY USED.....	14
5.1Development Tools.....	14
5.1.1 Microsoft visual Studio.....	14
5.1.2 Microsoft SQL server Management Studio Express	15
5.1.3 .NET Framework 4.5	15
5.2 Technology Used	15
5.2.1 .NET Framework Structure.....	15
5.2.2 Compilation to Manage Code	16
5.2.3 JIT compilation	17
5.2.4 The .NET Language.....	17
5.2.5 Data Provider	18

5.2.6 The Connection object	18
5.2.7 The command Object	18
5.2.8 The Data Reader object	19
5.3 Microsoft SQL Server	19
CHAPTER- 6: CODING IMPLEMENTATION	21
6.1 Application Code Structure	21
6.2 Logic	21
6.3 Code for Login page and validation	21
6.4 Backup Data code	23
6.5 Restore Data code	24
6.6 Project Screenshot	25
CHAPTER – 7: DEBUGGING AND TESTING	35
7.1 Purpose of Testing	35
7.2 Type of Testing	35
7.2.1 Units Test	35
CHAPTER – 8: CONCLUSION AND LESSON LEARNT	36
8.1 Project Limitation	36
8.2 Conclusion	36
8.3 Lesson Learnt	36
8.4 Future Enhancements	36
REFERENCES	38

CHAPTER-1: INTRODUCTION

1.1 Introduction to Inventory Management System

The project Inventory Management System is a complete desktop based application designed on .Net technology using Visual Studio Software. The main aim of the project is to develop Inventory Management System Model software in which all the information regarding the stock of the organization will be presented. It is an intranet based desktop application which has admin component to manage the inventory and maintenance of the inventory system.

This desktop application is based on the management of stock of an organization. The application contains general organization profile, sales details, Purchase details and the remaining stock that are presented in the organization. There is a provision of updating the inventory also. This application also provides the remaining balance of the stock as well as the details of the balance of transaction.

Each new stock is created and entitled with the named and the entry date of that stock and it can also be update any time when required as per the transaction or the sales is returned in case. Here the login page is created in order to protect the management of the stock of organization in order to prevent it from the threads and misuse of the inventory.

1.2 Literature Review

Products are considered as the business resources for the organization. This includes managing the product with appropriate way to review any time as per the requirement. Therefore it is important to have a computer based IMS which has the ability to generate reports, maintain the balance of the stock, details about the purchase and sales in the organization. Before developing this application we came up with 2 Inventory Management System existing in the market, which helps to give the knowledge for the development of our project. These application software are only used by the large organization but so we came up with the application which can be used by the small company for the management of their stock in the production houses.

After analyzing the other inventory management system we decided to include some of common and key features that should be included in every inventory management system. So we decided to include those things that help the small organization in a way or other.

1.3 Problem Statement

After analyzing many existing IMS we have now the obvious vision of the project to be developed. Before we started to build the application team had many challenges.

We defined our problem statement as:

- To make desktop based application of IMS for small organization.
- To make the system easily managed and can be secured.
- To cover all the areas of IMS like purchase details, sales details and stock management.

1.4 Objective of the Project

1.4.1 Primary objective

The primary objectives of the project are mentioned below:

- To fulfill the requirement for achieving the Bachelor's degree of Computer Information System.
- To know the fundamentals of the .Net Technology and Visual Studio with the .Net Framework

1.4.2 Secondary objective

The secondary objectives of this project are mentioned below:

- To develop an application that deals with the day to day requirement of any production organization
- To develop the easy management of the inventory
- To handle the inventory details like sales details, purchase details and balance stock details.
- To provide competitive advantage to the organization.
- To provide details information about the stock balance.
- To make the stock manageable and simplify the use of inventory in the organization.

1.6 Features of Project

This application is used to show the stock remaining and details about the sales and purchase. It gives the details about the stock on daily based and weekly based. The details components are described below:

Login page: As application starts the login page appears. Admin login is determined by the username and password that has all the authority to add, update and delete the stock of the organization as per the requirement.

Create Godwom: We can create godwom if we need to extend or we have more than one godwom. We can create the godwom along with the date.

Sales details: It show the details about the sales and the remaining stock of sales. It also show the details about the sales in return.

Purchase details: It shows the details about the purchase made by the organization along with the price and dates.

1.7 Scope of the Application

Inventory Management System (IMS) is targeted to the small or medium organization which doesn't have many godwom or warehouses i.e. only to those organization that has single power of authority. Some of the scope are:

- Only one person is responsible in assigning the details or records
- It is security driven.
- Godown can be added as per the requirement.

CHAPTER-2 BACKGROUND KNOWLEDGE

2.1 Architectural Review

This desktop based application is based on 3-tier architecture of .Net Framework. The 3-tier includes the three hierarchy of the flow of programming logic from user interface to database and again database to user interface with the desired information requested by the clients. In between there involves the logic layer for effectively and correctly manipulating the request. The 3-tier includes the following:

2.1.1 Client tier

The visual part is implemented using all kinds of swing components, which does not make database calls. The main function of this tier is to display information to the user upon user's request generated by user's inputs such as firing button events. For example, inventory list will display when user click "display" button if he or she wants to know the list of stock remaining in the organization.

2.1.2 Business tier

The middle tier, business logic, is called by the client to make database queries. It provides core function of the system as well as connectivity to the data tier, which simplify tasks that were done by the clients tier.

2.1.3 Data tier

Data layer is also the class which gets the data from the business tier and sends it to the database or gets the data from the database and sends it to business tier. This is the actual DBMS access layer or object layer also called the business object. The database backend stores information which can be retrieved by using the mysql database Connectivity. Mysql database connectivity is used to manage the communication between the middle tier and the backend database by issuing complex database queries.

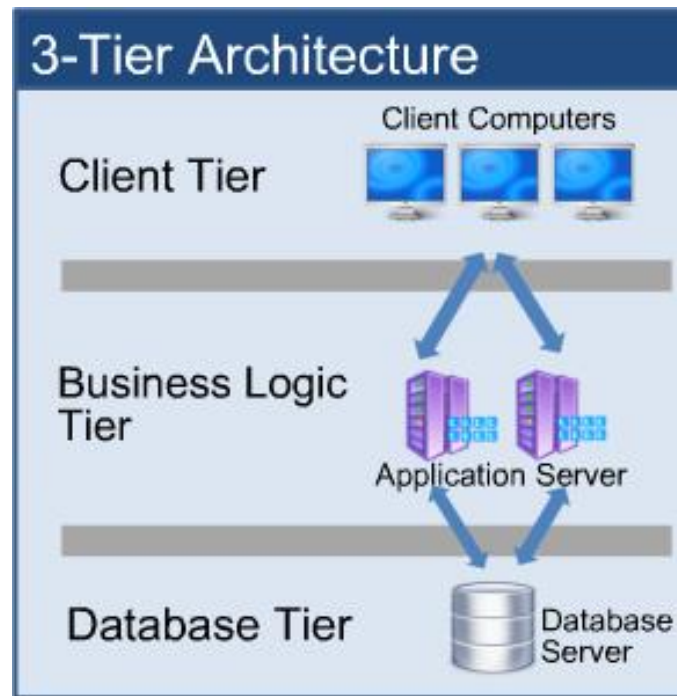


Figure 2.1: Tier Architecture

Source: <http://www.dbtalks.com/UploadFile/a7e1c8/3-tier-clientserver-architecture/>

2.2 Database Theory

A database is a collection of information that is organized so that it can easily be accessed, managed and updated. In one view, database can be classified according to types of content: bibliography, full-text, numeric, and image. In computing, database are sometime classified according to their organizational approach. A distributed database is one that can be dispersed or replicated among different points in a network.

2.2.1 Relational Database

IMS has the relational database model. A relational database is a digital database whose organization is based on the relational model of data. This model organizes data into one or more tables of rows and columns. These tables have the relation. The relation is maintained by the unique key defined in each row. The key can be primary and foreign depending on their nature of connection. The standard user and application program interface to a relational database is the structured query language (SQL). SQL statements are used both for interactive queries for information from relational database and for gathering data for reports.

Primary Key

The primary key of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique or it can be generated by the DBMS. A primary key's main features are:

- It must contain a unique value for each row of data.
- It cannot contain null value.

Foreign Key

A foreign key is a column or group of column in a relational database table that provides a link between data in two tables. In foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or column in another table thereby establishing a link between them. Creating a foreign key manually includes the following advantages:

- Changes to primary key constraints are checked with foreign key constraints in relation table.
- An index enables the Database Engine to quickly find related data in the foreign key tables.

2.2.2 Structured Query Language (SQL)

The structured Query language (SQL) is the set of instructions used to interact with a relational database. In fact, SQL is the only language the most database actually understand. Whenever you interact with such a database, the software translates your commands into SQL statement that the database knows how to interpret. SQL has three major Components:

- Data Manipulation Language (DML)
- Data Definition Language (DDL)
- Data Control Language (DCL)

2.3 ACID Property

Every database transaction obeys the following rules:

Atomicity – Either the effects of all or none of its operation remain (“all or nothing” semantics) when a transaction is completed (committed or aborted respectively). In other words, to the outside world a committed transaction appears (by its effects on

the database) to be indivisible, atomic, and an aborted transaction does not leave effects on the database at all, as if never existed.

Consistency – every transaction must leave the database in a consistent (correct) state, i.e., maintain the predetermined integrity rules of the database (constraints upon and among the database's objects). A transaction must transform a database from one consistent state to another consistent state (however, it is the responsibility of the transaction's programmer to make sure that the transaction itself is correct, i.e., performs correctly what it intends to perform (from the application's point of view) while the predefined integrity rules are enforced by the DBMS). Thus since a database can be normally changed only by transactions, all the database's states are consistent. An aborted transaction does not change the database state it has started from, as if it never existed (atomicity above).

Isolation – Transactions cannot interfere with each other (as an end result of their executions). Moreover, usually (depending on concurrency control method) the effects of an incomplete transaction are not even visible to another transaction. Providing isolation is the main goal of concurrency control.

Durability – Effects of successful (committed) transactions must persist through crashes (typically) by recording the transaction's effects and its commit event in a non-volatile memory.

CHAPTER-3: ANALYSIS AND DESIGN

3.1 Background Research

We started research by identifying the need of IMS in the organization. Initially we bounded our research to find the general reasons that emerged the needs of Inventory Management System. We used different techniques to collect the data that can clearly give us the overall image of the application. The techniques we used were interview with the developers, visiting online websites that are presented as the templates and visiting some organization to see their IMS application. Basically the following factors forced us to develop IMS application:

- Cost and affordability
- Lack of stock management.
- Effective flow of stock transfer and management.
- Difficulty in monitoring the stock management.

3.2 Requirement Analysis

We collected a number of requirements for project from our primitive research, website visits, and interview to the concerned personnel and their experiences regarding the concepts of its development. We have even visited some organization in Kathmandu valley and analyze its importance and try to develop the project by fulfilling all the weakness that were found in the application. We then decided to bulid same type of application with different logic flow and new language which will be suitable for the small organization.

3.3 IMS Requirement

The goal for the application is to manage the inventory management function of the organization. Once it is automated all the functions can be effectively managed and the organization can achieve the competitive advantage. Business requirement are discussed in the Scope section, with the following additional details:

- Helps to search the specific product and remaining stock.
- Details information about the product sales and purchase.
- Brief Information of the organization todays status in terms of news, number of present inventory as per the date entered.
- It helps to identify the total presented inventory in the company.

- To know the balance and details of sales distributed in specific date.
- There is proper transaction management of inventory.
- All transaction have specific entry date along with quantity and rate.
- Only admin can login in the page.

3.4 Users Requirement

User requirement are categorized by the user type

3.4.1 Admin

- Able to create new godwom along with date.
- Able to edit the entry as per entry.
- Able to add, modify and delete the stock entry.

3.4.2 Inventory management

- Able to check the stock available.
- Able to check the balance payment.
- Able to view the remaining sales stock.

3.5 Feasibility Analysis

This software has been tested for various feasibility criterions from various point of views.

3.5.1 Economic Feasibility

The system is estimated to be economically affordable. The system is medium scale desktop application and has affordable price. The benefits include increased efficiency, effectiveness, and the better performance. Comparing the cost and benefits the system is found to be economically feasible.

3.5.2 Technical Feasibility

Development of the system requires tools like:

- Visual Studio 2015
- .NET Framework 4.5
- Microsoft SQL server 2008, etc

Which are easily available within the estimated cost and schedule.

3.5.3 Operational Feasibility

The system provides better solution to the libraries by adding the typical requirement and necessities. The solution provided by this system will be acceptable to ultimate solution for the stock management.

3.5.4 Schedule Feasibility

The organized schedule for the development of the system is presented in the schedule sub-section. The reasonable timeline reveals that the system development can be finished on desired time framework.

3.6 Gantt chart

It is one of the popular way to illustrate project schedule. A Gantt chart is a graphical representation of a project that shows each activity task as a horizontal bar whose length is proportional to its time for completion. A Gantt chart for the project deliverables within time frame. This project Gantt chart is shown below:

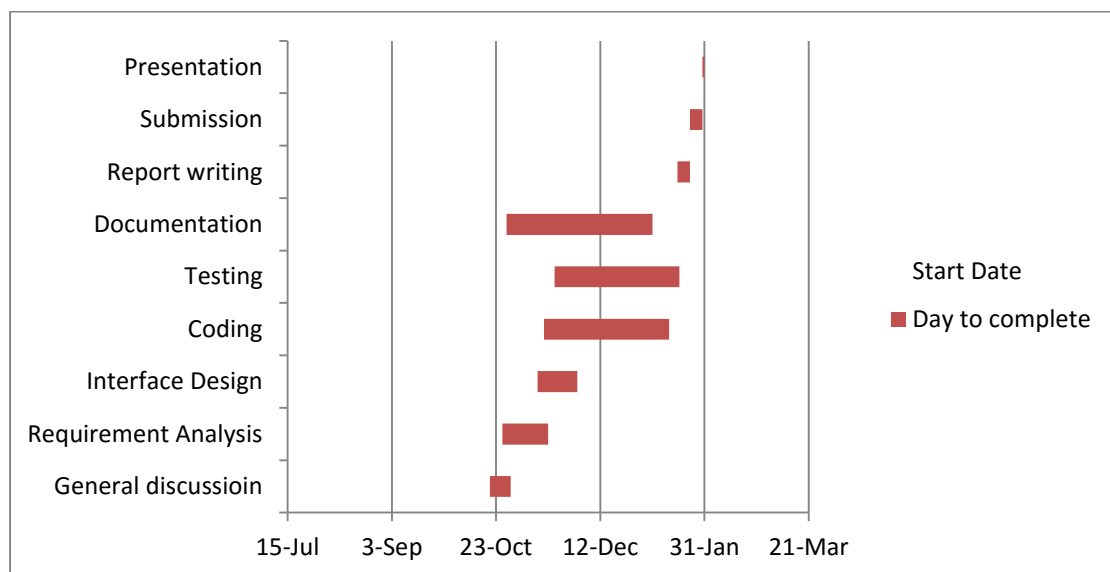


Figure 3.6: Gantt chart

CHAPTER – 4: SYSTEM DESIGN

4.1 Process Flow Diagram

Process Flow Diagram or Flowchart is a diagram which uses geometric symbols and arrows to define the relationships. It is a diagrammatic representation of the algorithm. The Process flow Diagram of our application is shown below:

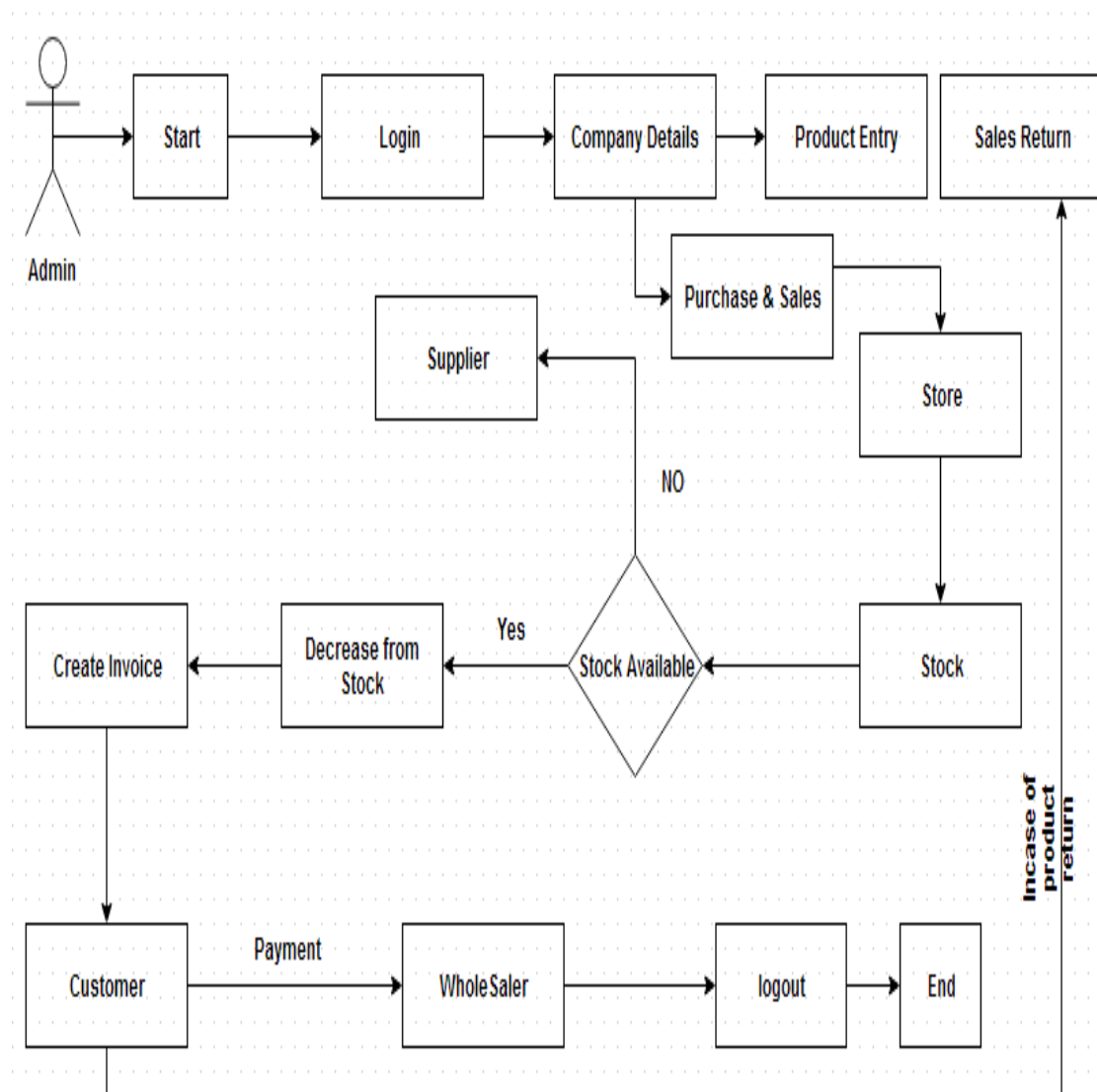


Figure 4.1: IMS Process flow diagram

4.2 Use Case Diagram

Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors and their goals.

The main purpose of a use case diagram is to show what system functions are performed for which actors.

4.2.1 Diagram Building Block

Use cases

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

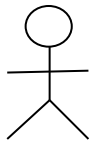
Actor

An actor is a person, organization or external system that plays a role in one or more interactions with the system

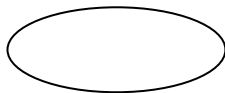
System boundary boxes (optional)

A rectangle is drawn around the use case called the system boundary box to indicate scope of the system.

Actor



Use case



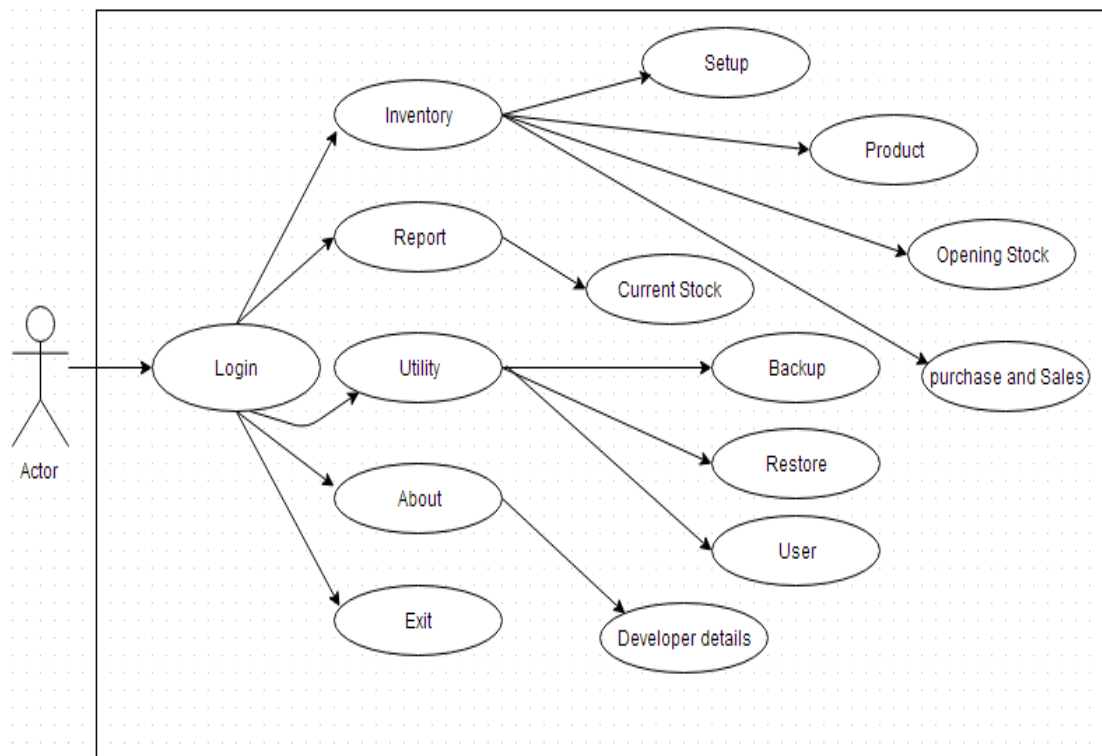


Figure 4.2.1: IMS Use Case Diagram

CHAPTER – 5: TOOLS AND TECHNOLOGY USED

5.1 Development Tools

5.1.1 Microsoft visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Form applications, websites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Window, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight. Microsoft Visual Studio simplifies the basic tasks of creating, debugging and deploying applications.

Microsoft Visual Studio comes with .NET Framework and supports applications targeting Windows. It supports IBM DB2 and Oracle databases, in addition to Microsoft SQL Server. It has integrated support for developing Microsoft Silverlight applications, including an interactive designer. Microsoft Visual Studio offers several tools to make parallel programming simpler: in addition to the Parallel Extensions for the .NET Framework and the Parallel Patterns Library for native code, Visual Studio includes tools for debugging parallel applications.

The Visual Studio code editor now highlights references; whenever a symbol is selected; all other usages of the symbol are highlighted. It also offers a Quick Search feature to incrementally search across all symbols in C++, C# and VB.NET projects. Quick Search supports substring matches and camel Case searches. The Call Hierarchy feature allows the developers to see all the methods that are called from a current method as well as the methods that call the current one. IntelliSense in Visual Studio supports a consume-first mode which developers can opt into. In this mode, IntelliSense will not auto-complete identifiers; this allows the developer to use undefined identifiers (like variable or method names) and define those later. Visual Studio can also help in this by automatically defining them, if it can infer their types from usage.

We have used Visual Studio Community 2015, v 14.0.23107.10 for developing the Inventory Management System Application.

5.1.2 Microsoft SQL server Management Studio Express

Microsoft SQL Server Management Studio Express (SSMSE) provides a graphical management tool for SQL Server Express Edition. SSMSE user interface is a subset of SQL Management Studio that is available with other editions of SQL Server. SSMSE can also manage instance of the SQL Server Database Engine created by any edition of SQL Server. Inventory Management System is developed using Microsoft SQL Server 2008.

5.1.3 .NET Framework 4.5

The .NET Framework is a development platform for building apps for Windows, Windows Phone, Windows Server, and Microsoft Azure. It consists of the common language runtime (CLR) and the .NET Framework class library, which includes classes, interfaces, and value types that support an extensive range of technologies. The .NET Framework provides a managed execution environment, simplified development and deployment, and integration with a variety of programming languages, including Visual Basic and Visual C#.

5.2 Technology Used

5.2.1 .NET Framework Structure

The .Net architecture is basically segregated into three layers namely top, middle and bottom layer. The bottom layer is CLR, it is the heart of .NET Framework which provides the runtime environment in which programs are executed. The middle layer comprises the next generation of standard system services are brought under the control of the framework, making them universally available and standardizing their usage across languages. The top layer includes user and program interfaces as figure:

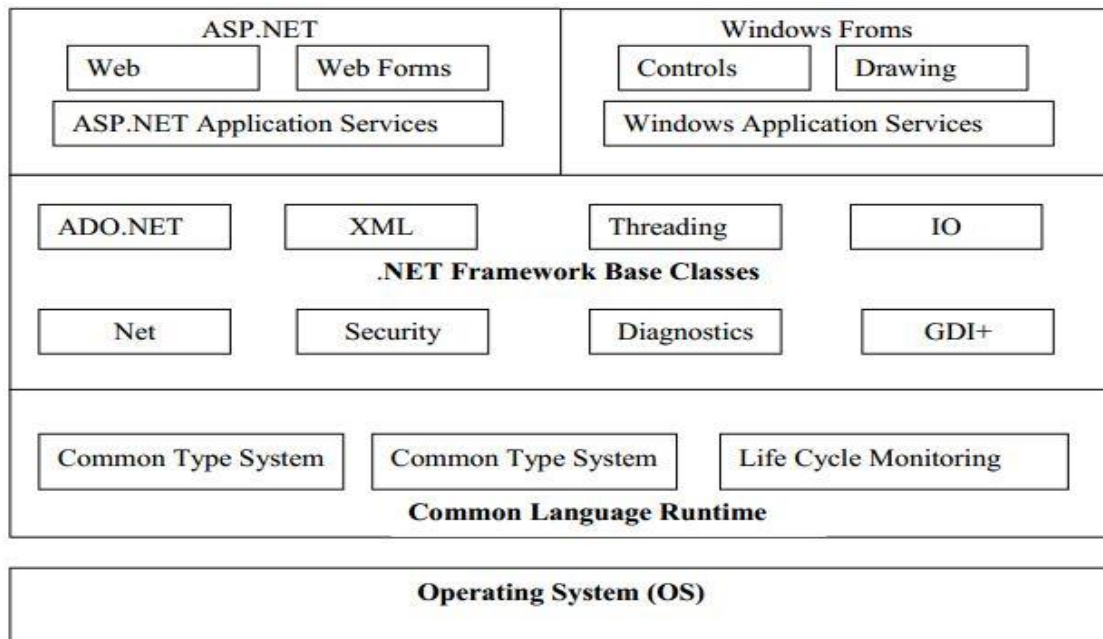


Figure 5.2.1: .Net Framework Architecture

Source: Teacher's Handout

The first thing that you should notice when looking at this diagram is that the .NET Framework sits on top of the operating system. There has also been a lot of talk about .NET being ported over by some third-party companies so that a majority of the .NET Framework could run on other platforms as well.

At the base of the .NET Framework is the Common Language Runtime (CLR). The CLR is the engine that manages the execution of the code. The next layer up is the .NET Framework Base Classes. This layer contains classes, value types, and interfaces that you will use often in your development process. Most notably within the .NET Framework Base Classes is ADO.NET, which provides access to and management of data.

The third layer of the framework is ASP.NET and Windows Forms. ASP.NET should not be viewed as the next version of Active Server Pages after ASP 3.0, but as a dramatically new shift in Web application development. Using ASP.NET, it's now possible to build robust Web applications that are even more functional than Win32 applications of the past.

5.2.2 Compilation to Manage Code

Code that is compiled and targeted to the CLR is known as managed code. Managed code provides metadata that is needed for the CLR to provide the services of multi-

language support, code security, object lifetime management, and memory management. The .NET Framework requires that you use a language compiler that is targeted at the CLR, such as the Visual Basic .NET, C#, C++ .NET, or Jscript .NET compilers provided by Microsoft. So how does the code that you typed into Visual Studio .NET become the code that the user receives when he is using your application? It is fairly simple and straightforward. Figure below shows a diagram of the compilation process.

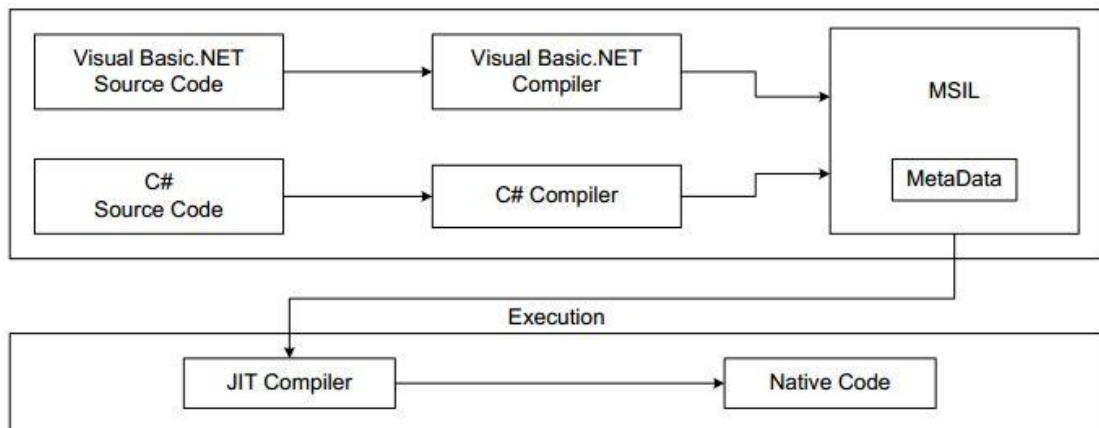


Figure 5.2.2: Compilation to managed code

Source: Teacher's handout

The IL is CPU-independent. This means that IL code is not reliant on the specific computer that generated it. In other words, it can be moved from one computer to another (as long as the computer supports the .NET Framework) without any complications. This is what makes XCopy, or just copying over the application, possible.

5.2.3 JIT compilation

The .NET Framework contains one or more JIT compilers that compile your IL code down to machine code. This is done when the application is executed for the first time.

5.2.4 The .NET Language

In the past, you chose the development language for an application based upon the functionality that you were trying to perform. Some languages were more powerful than others, but at the same time they might have required a higher level of understanding and were, in most cases, more difficult to program in.

Now the .NET Framework provides you with a language-independent programming platform. You do not have to decide which language would provide a better solution. All languages are now on a level playing field. In .NET, no one language is superior to any of the other languages. They all have equal access to everything that .NET offers.

To be part of the .NET Framework, a language only has to follow certain rules. The biggest and most important rule for inclusion is that the language needs to be an object-oriented language. Microsoft provides four languages with the .NET Framework:

Visual Basic .NET

C#

C++.NET and

Jscript .NET.

Microsoft also provides J# (pronounced J-sharp), but in order to use this new language that is basically Java for .NET, you need to download the language to install it on your server.

5.2.5 Data Provider

The data provider is responsible for providing and maintaining the connection to the database. A database provider is a set of related components that works together to provide in an efficient and performance driven manner. Each Data provider consists of the following components classes:

- The command object which is used to execute a command.
- The Connection object which provides a connection to the database.
- The Data Reader object which provides a ready only, connects recordset.

5.2.6 The Connection object

The connection object created the connection to the database. Microsoft Visual Studio .NET provides two types of connection classes: the SqlConnection object, which is designed specifically to connect to Microsoft SQL Server.

5.2.7 The command Object

The command object is represented by corresponding classes: SQL Command. Command object are used to execute commands to a database across a data connection. The command objects provides three methods that are used to execute commands on the database.

- **ExecuteNonQuery:** Executes commands that have no return values such as INSERT, UPDATE AND DELETE
- **ExecuteScalar:** Returns a single value from a database query
- **ExecuteReader:** Returns a result set by way of a DataReader Objects.

5.2.8 The Data Reader object

The DataReader object provides a read-only, connected stream recordset from a database. Unlike other components of the Data Provider, DataReader objects cannot be directly instantiated. Rather, the DataReader is returned as the result of the Command object's ExecuteReader method. The DataReader can provide rows of data directly to application logic when one does not need to keep the data cached in memory. Because only one row is in memory in time, the DataReader provides the lowest overhead in terms of system performance but requires the exclusive use of an open Connection object for the life time of the DataReader.

5.3 Microsoft SQL Server

Microsoft SQL Server is an application used to create computer databases for the Microsoft Windows family of server operating systems. Microsoft SQL Server provides an environment used to generate database that can be accessed from workstations, the Internet, or other media such as a personal digital assistant (PDA). Microsoft SQL Server is used to create desktop, enterprise, and web-based database applications. It is used at different levels and with various goals.

SQL Server makes simpler and easier to deploy, manage, and optimize enterprise data and analytical applications. An enterprise data management platform, it performs a single management console that enables data administrators anywhere in your organization to monitor, manage, and tune all of the databases and associated services across your enterprise. It provides an extensible management infrastructure that can be easily programmed by using SQL management objects, enabling users to customize and extend their management environment and independent software vendors to build additional tools and functionality to further extend the capabilities that come out of the box.

SQL Server simplifies management by providing integrated management console to monitor and manage the SQL Server relational database as well as integration services, analysis services, reporting services, notification services and SQL Server Mobile Edition across large number of distributed servers and databases. Database

administrator can perform several tasks at the same time, such as authorizing and executing a query, viewing server objects, managing an object, monitoring system activities, and viewing online help.

SQL Server expose more than 70 new measures of internal database performance and resource usages, ranging from memory, locking, and scheduling to transactions and network and disk I/O. these dynamic management views provide greater transparency and visibility into the database and a powerful infrastructure for proactive monitoring of database health and performance. The major characteristics are listed below:

Reliability: achieve a more secure deployment. SQL Server provides rich security features to protect data and network resources.

Confidentiality: Protect your data. SQL Server clustering supports Kerberos authentication on a virtual Server and Microsoft-style policies on standard logins so that a consistent policy is applied across all accounts in the domain.

Integrity: SQL Server supports encryption capabilities within database itself, fully integrated with a key management infrastructure. By default, client server communications are in encrypted.

CHAPTER- 6: CODING IMPLEMENTATION

6.1 Application Code Structure

Inventory Management System was designed using Visual Studio as mentioned earlier following the three tier application architecture. It provided us with the code editor as a white blank space and the solution explorer where every code files were kept. Code Editor is where the logical were developed into code and kept safe in the solution explorer. In solution explorer we kept every code file by creating the folder and adding those files in a folder that are similar in nature. The main folder was the Inventory Management System. Following are the list of the folders, sub-folders and their corresponding files:

6.2 Logic

Logic is the main component of any application portrayed through the code. Every modules in the application includes logic. Most of the logic are common and understandable as we call 3-tier architecture based system.

6.3 Code for Login page and validation

frmLogin.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace InventoryManagementSystem
{
    public partial class frmLogin : Form
    {
        public frmLogin()
        {
            InitializeComponent();

            private void txtUserName_KeyDown(object sender, KeyEventArgs e)
            {
                clsGlobalFunction.Tab_Function(e);
                clsGlobalFunction.Escape_Function(e, btnCancel);
            }

            private void txtPassword_KeyDown(object sender, KeyEventArgs e)
            {

```

```

        clsGlobalFunction.Tab_Function(e);
        clsGlobalFunction.Escape_Function(e, btnCancel);
    }

    private void txtFiscalYear_KeyDown(object sender, KeyEventArgs e)
    {
        clsGlobalFunction.Tab_Function(e);
        clsGlobalFunction.Escape_Function(e, btnCancel);
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    private void btnCancel_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyCode == Keys.Escape) { Application.Exit(); }
    }

    private void btnOK_Click(object sender, EventArgs e)
    {
        clsGlobalFunction.temp_text = "Select * from user_login where
user_name='" + txtUserName.Text + "' and password='" +
clsGlobalFunction.Encrypt(txtPassword.Text) + "'";
        SqlDataAdapter da = new SqlDataAdapter(clsGlobalFunction.temp_text,
clsGlobalFunction.cn);
        DataSet ds = new DataSet();
        da.Fill(ds, "table0");
        if (ds.Tables[0].Rows.Count > 0)
        {
            this.Hide();
        }
        else
        {
            MessageBox.Show("User Name or Password is not Correct");
            txtUserName.Focus();
        }
    }

}

private void txtUserName_Validated(object sender, EventArgs e)
{
}

private void txtPassword_Validated(object sender, EventArgs e)
{
}

private void txtFiscalYear_Validated(object sender, EventArgs e)
{
}

private void frmLogin_Load(object sender, EventArgs e)
{
    try
    {
        clsGlobalFunction.temp_text = "select * from fiscal_year";
    }
}

```

```

        SqlDataAdapter da = new
SqlDataAdapter(clsGlobalFunction.temp_text, clsGlobalFunction.cn);
        DataSet ds = new DataSet();
        da.Fill(ds, "table0");
        txtFiscalYear.Items.Clear();
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            txtFiscalYear.Items.Add(ds.Tables[0].Rows[i]["fyear"].ToString());
            txtFiscalYear.Text =
ds.Tables[0].Rows[i]["fyear"].ToString();
        }
    }
    catch { }

}

private void txtUserName_Validating(object sender, CancelEventArgs e)
{
    if (txtUserName.Text == "") { MessageBox.Show("Please Enter User
Name", "Inventory Management System", MessageBoxButtons.OK,
MessageBoxIcon.Information); txtUserName.Focus(); return; }
}

private void txtFiscalYear_SelectedIndexChanged(object sender,
EventArgs e)
{
}

private void label3_Click(object sender, EventArgs e)
{
}
}
}

```

6.4 Backup Data code

```

SaveFileDialog saveBACKUP = new SaveFileDialog();
try
{
    DialogResult Dr;
    saveBACKUP.Filter = "File format (*.bak)|*.bak";
    saveBACKUP.FileName = "INVENTORY ( BACKUP ) " +
DateTime.Now.Year.ToString() + "-" + DateTime.Now.Month.ToString() + "-" +
DateTime.Now.Day.ToString();
    Dr = saveBACKUP.ShowDialog();
    if (Dr == DialogResult.OK)
    {
        string s = null;
        s = saveBACKUP.FileName;
        SqlCommand cmd = new SqlCommand("Backup database " +
clsGlobalFunction.DatabaseName + " to disk='" + s + "' WITH STATS",
clsGlobalFunction.cnMaster);
        cmd.ExecuteNonQuery();
        clsGlobalFunction.MessageBoxDisplay("Sucussfully Created
Backup !!!");
    }
}
catch (Exception ex)

```

```

{
    clsGlobalFunction.MessageBoxDisplay(ex.Message);
}

```

6.5 Restore Data code

```

OpenFileDialog RestoreBACKUP = new OpenFileDialog();
try
{
    DialogResult Dr;
    RestoreBACKUP.Filter = "File format (*.bak)|*.bak";
    Dr = RestoreBACKUP.ShowDialog();
    if (Dr == DialogResult.OK)
    {
        string s = null;
        s = RestoreBACKUP.FileName;
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = clsGlobalFunction.cnMaster;

        try
        {
            cmd.CommandText = "Alter Database " +
clsGlobalFunction.DatabaseName + " SET SINGLE_USER With ROLLBACK IMMEDIATE";
            cmd.ExecuteNonQuery();
            cmd.CommandText = "RESTORE DATABASE " +
clsGlobalFunction.DatabaseName + " FROM DISK = '" + s + "'" WITH REPLACE";
            cmd.ExecuteNonQuery();
            cmd.CommandText = "Alter Database " +
clsGlobalFunction.DatabaseName + " SET MULTI_USER";
            cmd.ExecuteNonQuery();
            clsGlobalFunction.MessageBoxDisplay("Sucussfully
Created Restored.Application is Restarted !!!");
            Application.Restart();
        }
        catch (Exception ex)
        {
            clsGlobalFunction.MessageBoxDisplay(ex.Message); }
    }
}

```

6.6 Project Screenshot

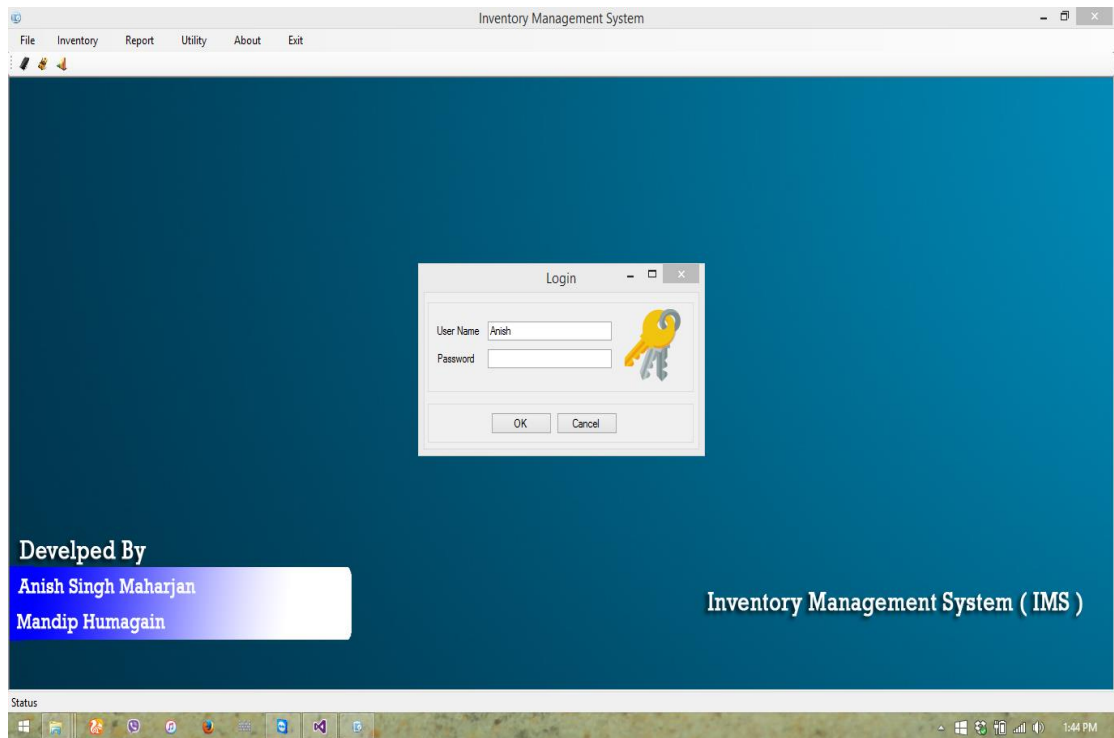


Figure 6.6.1: Admin or login page

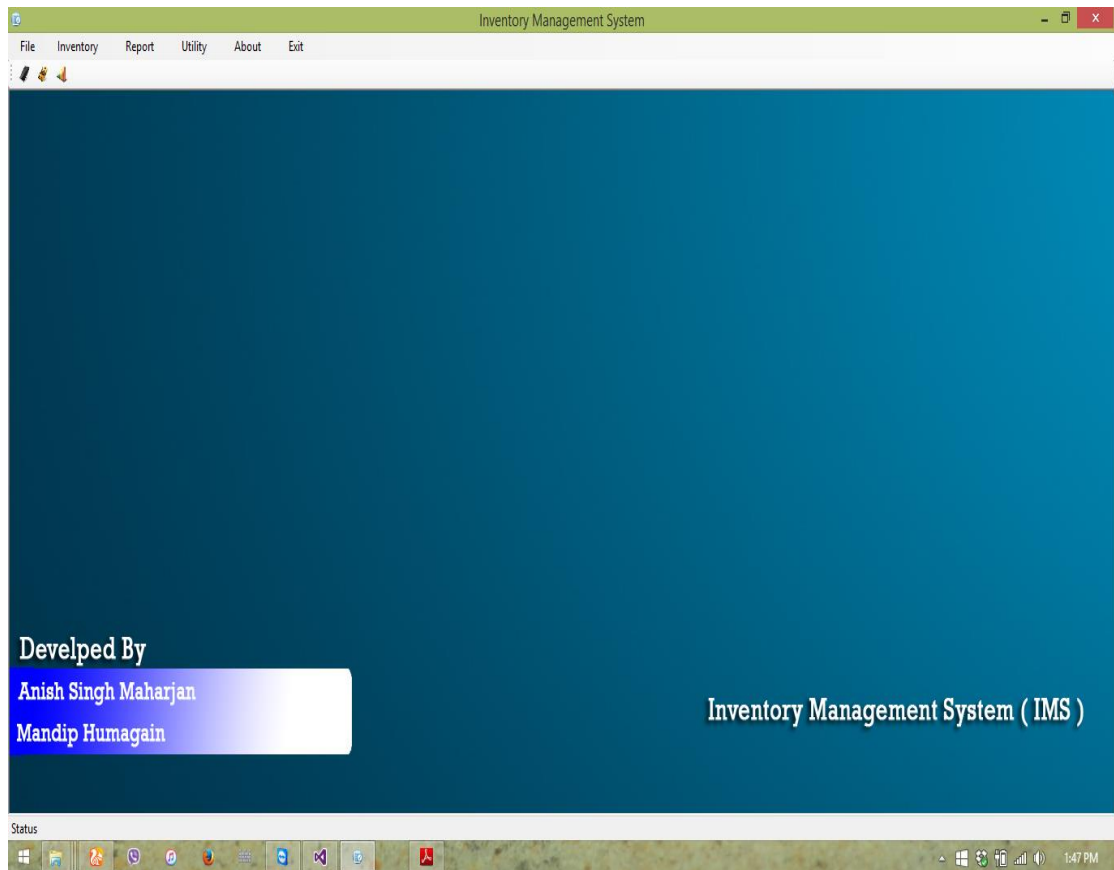


Figure 6.6.2: Company/Shop details

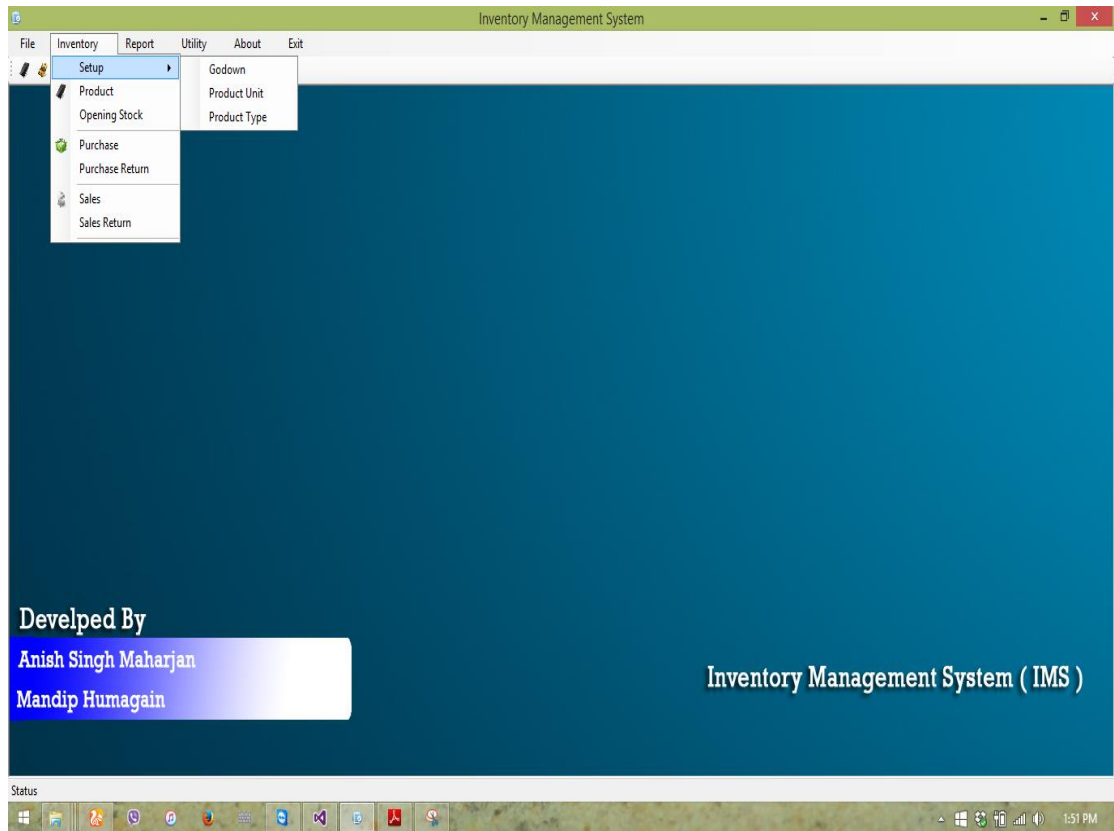


Figure 6.6.3: Inventory Entry System

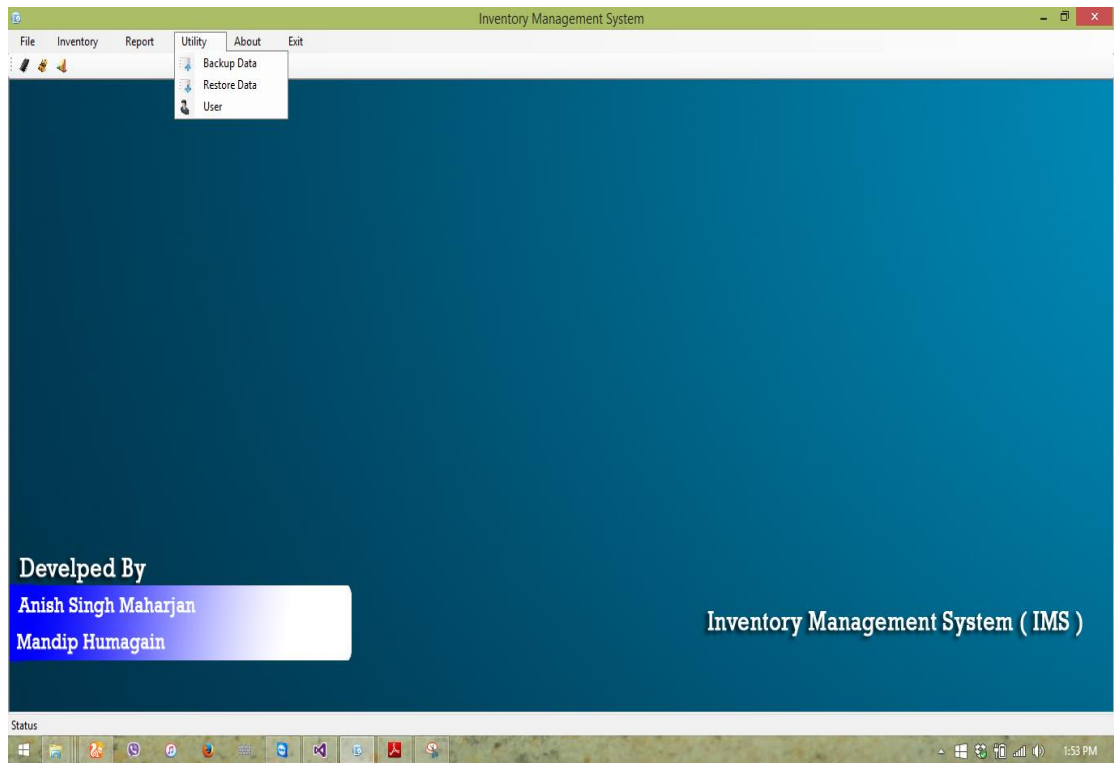


Figure 6.6.4: Back up and Restore Menu

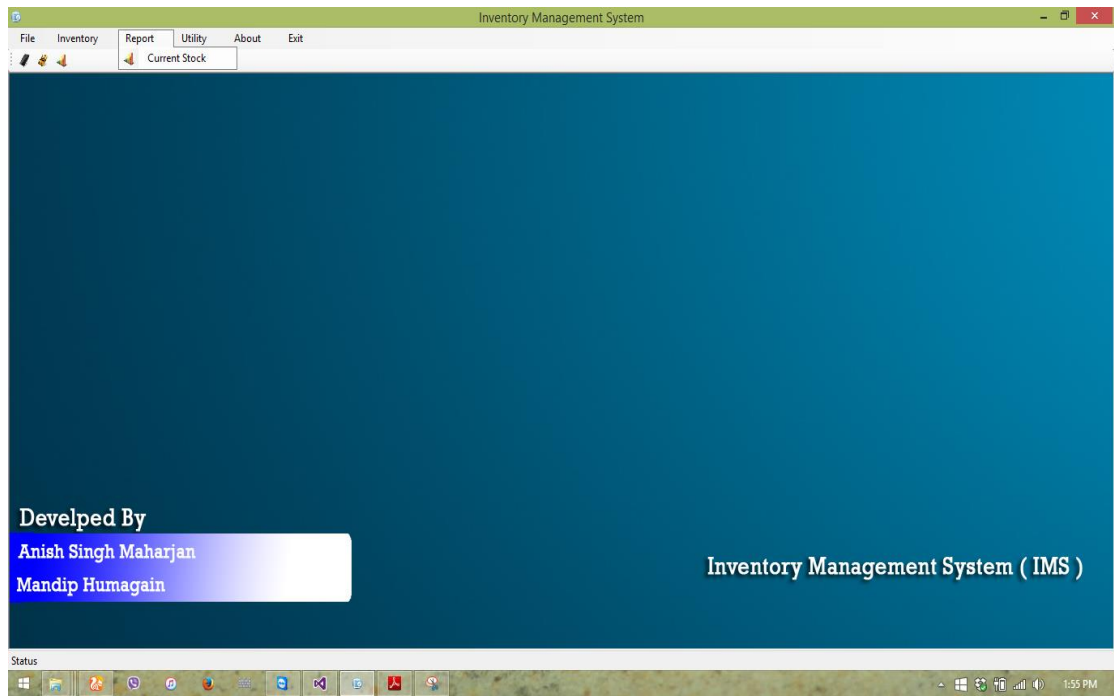


Figure 6.6.5: Report Menu

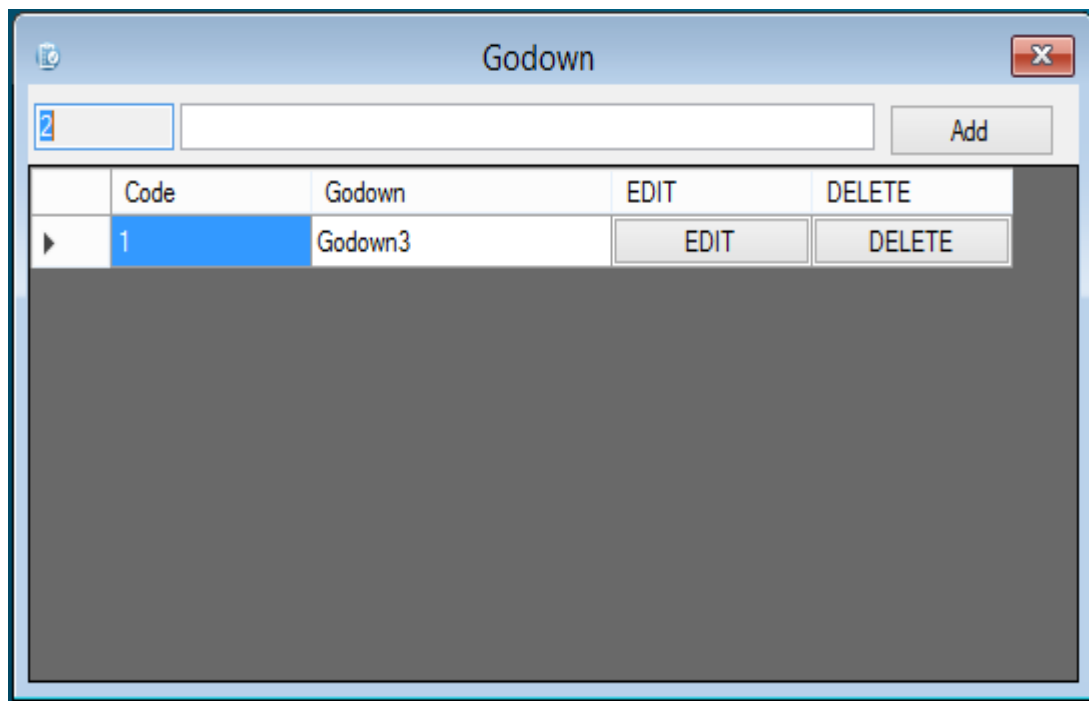
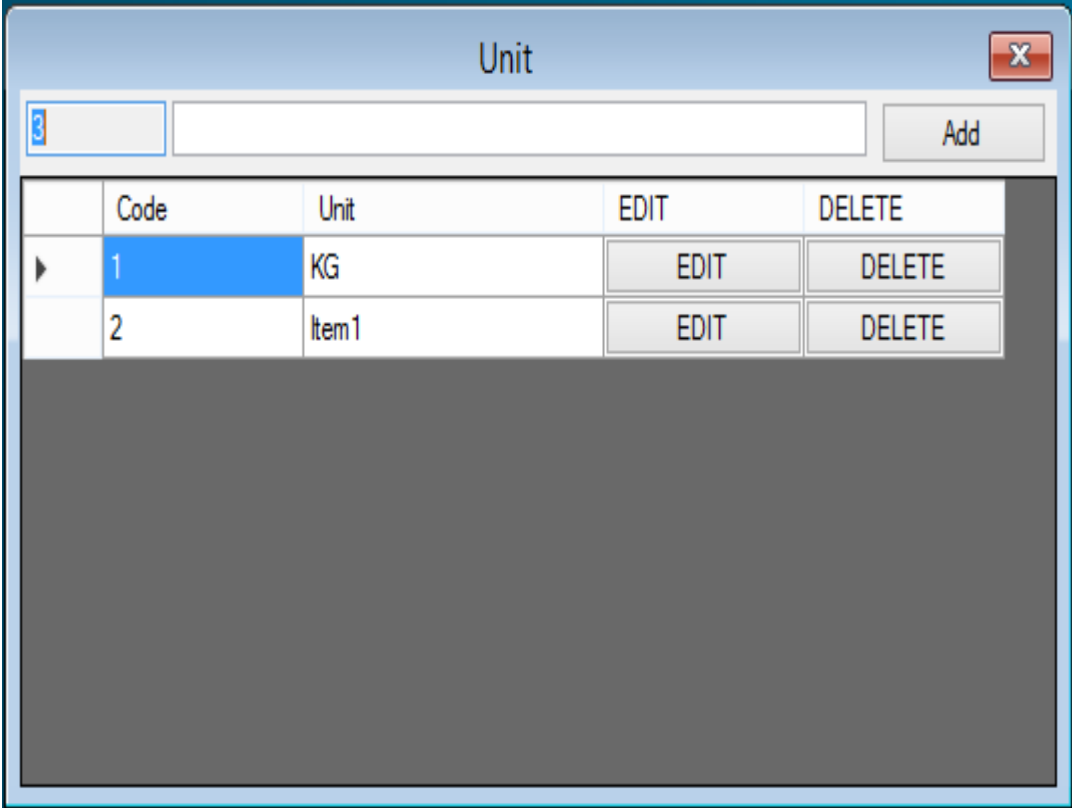


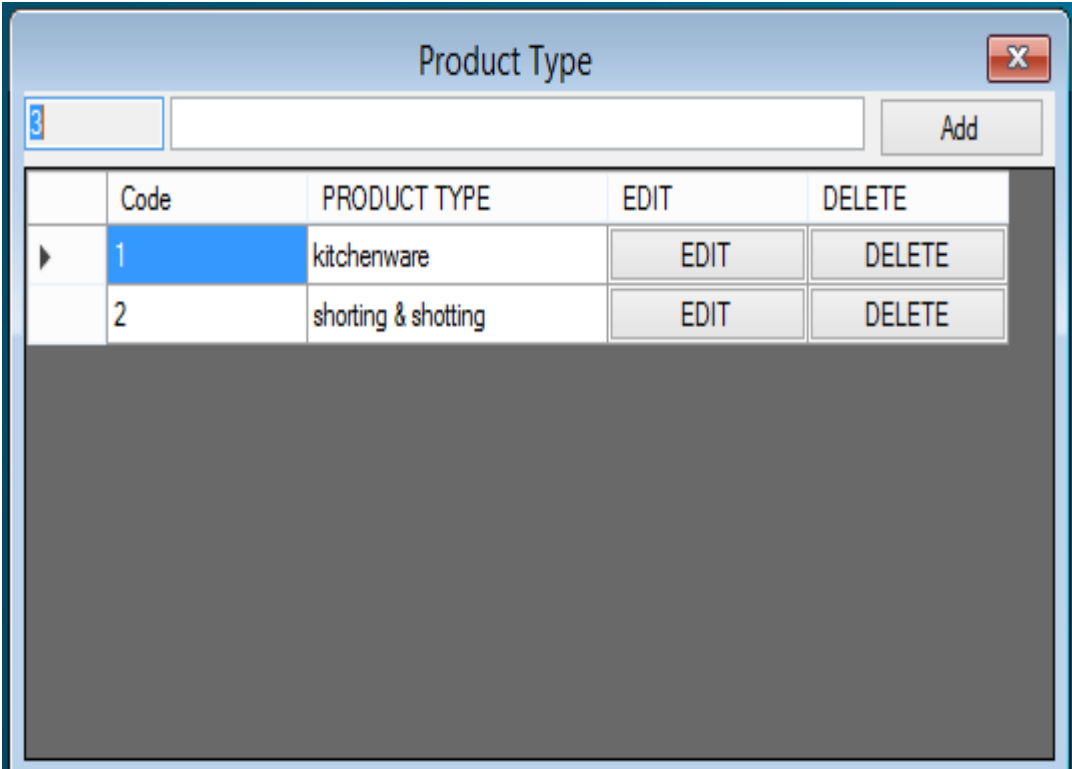
Figure 6.6.6: Creating Godwom



The 'Unit' window features a title bar with a close button. Below the title bar is a small input field containing the number '3', followed by a larger empty text field and an 'Add' button. The main area contains a table with columns: Code, Unit, EDIT, and DELETE. The first row has Code '1', Unit 'KG', and buttons 'EDIT' and 'DELETE'. The second row has Code '2', Unit 'Item1', and buttons 'EDIT' and 'DELETE'. The rest of the table area is a solid grey block.

	Code	Unit	EDIT	DELETE
▶	1	KG	EDIT	DELETE
	2	Item1	EDIT	DELETE

Figure 6.6.7: Creating Units



The 'Product Type' window features a title bar with a close button. Below the title bar is a small input field containing the number '3', followed by a larger empty text field and an 'Add' button. The main area contains a table with columns: Code, PRODUCT TYPE, EDIT, and DELETE. The first row has Code '1', PRODUCT TYPE 'kitchenware', and buttons 'EDIT' and 'DELETE'. The second row has Code '2', PRODUCT TYPE 'shorting & shotting', and buttons 'EDIT' and 'DELETE'. The rest of the table area is a solid grey block.

	Code	PRODUCT TYPE	EDIT	DELETE
▶	1	kitchenware	EDIT	DELETE
	2	shorting & shotting	EDIT	DELETE

Figure 6.6.8: Creating Product Type

Product Entry

Product Code:

Product Name:

Product Type:

Product Unit:

Sales Rate:

Product Image
Click Here to Insert Picture

	Code	Product	EDIT	DELETE
▶	1	product 1	<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
	3	product 3	<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>

Figure 6.6.9: Creating Product

Opening Stock

	Code	product	qty
▶	1	product 1	70
	3	product 3	90

Figure 6.6.10: Inserting Opening Stock

Purchase

Purchase Code: 5

Product Name: [dropdown]

Sales Rate: [input] Qty: 1 Total: [input] [Add]

Product Code	Product	Rate	Qty	Total	Remove
--------------	---------	------	-----	-------	--------

Total Purchase: 0 [Save]

Purchase No: 4 [EDIT] [DELETE]

Figure 6.6.11: Purchasing product form vendor

Sales

Sales Code: 1

Product Name: [dropdown] Sales Rate: [input] Qty: 1 Total: [input] [Add]

Product Code	Product	Rate	Qty	Total	Remove
3	product 3	800.00	3.00	2400.00	[Remove]

Total Sales: 0 [Save]

Sales No: 1 [EDIT] [DELETE]

Figure 6.6.12 Sales product to Customer

Purchase Return

Purchase Return Code: 2

Product Name: Sales Rate: 0 Qty: 1 Total: 0

Add

Product Code	Product	Rate	Qty	Total	Remove
3	product 3	800.00	1.00	800.00	Remove
3	product 3	800.00	1.00	800.00	Remove

Total Purchase Return: 0

Save

Sales No	EDIT	DELETE
1	EDIT	DELETE
2	EDIT	DELETE

Figure 6.6.13: Purchase Return

Sales Return

Sales Return Code: 3

Product Name: Sales Rate: 0 Qty: 1 Total:

Add

Product Code	Product	Rate	Qty	Total	Remove
3	product 3	800.00	1	800.00	Remove

Total Sales Return: 800.00

Save

Sales No	EDIT	DELETE
1	EDIT	DELETE
2	EDIT	DELETE

Figure 6.6.14: Sales Return

	code	product	OPENING STOCK (+)	PURCHASE (+)	SALES (-)	SALES RETURN (+)	PURCHASE_RETURN (-)	TOTAL STOCK
▶	1	product1	70	9.00	0.00	1.00	0.00	80.00
	3	product 3	90	5.00	3.00	2.00	4.00	90.00

Figure: 6.6.15 Report of Current Stock

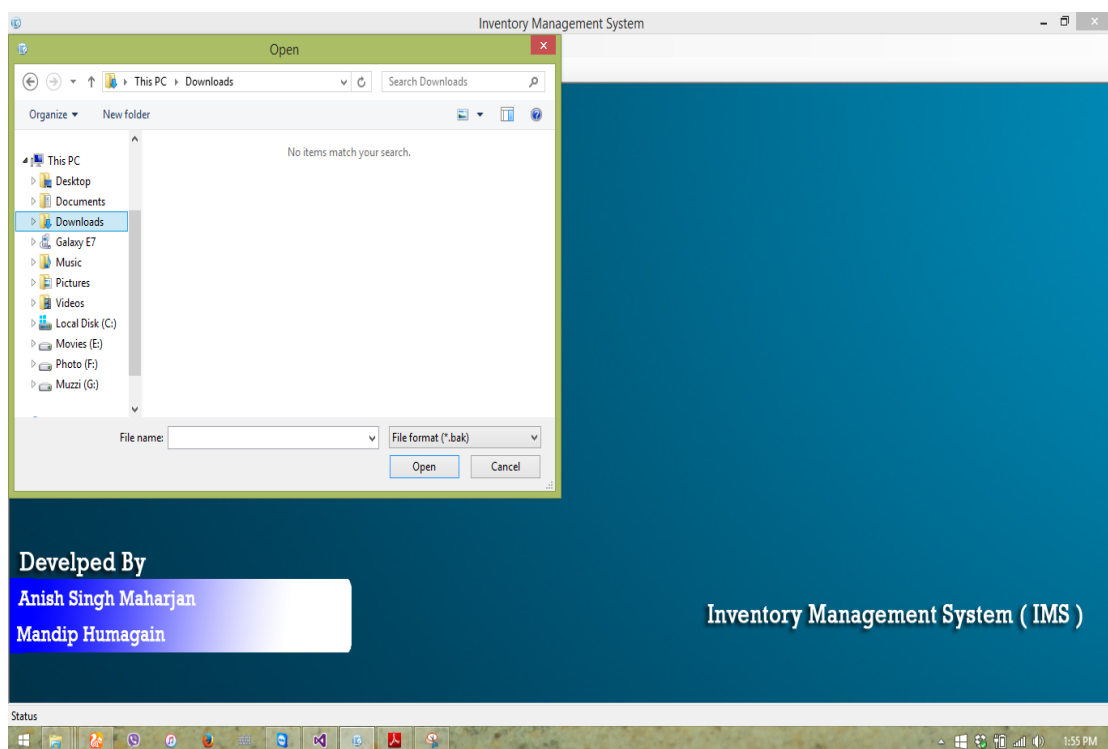


Figure 6.6.16: Back-up Data

CHAPTER – 7: DEBUGGING AND TESTING

7.1 Purpose of Testing

The purpose of software testing is to access or evaluate the capabilities or attributes of a software program's ability to adequately meet the applicable standards and application need. Testing does not ensure quality and the purpose of testing is not to find bugs. Testing can be verification and validation or reliability estimation. The primary objective of testing includes:

- To identifying defects in the application.
- The most important role of testing is simply to provide information.
- to check the proper working of the application while inserting updating and deleting the entry of the products.

7.2 Type of Testing

We have used one type of testing to ensure the error free features of our software application:

7.2.1 Units Test

This type of testing is the testing of individual software components. It is typically done by the programmer and not by the testers. It requires details information and knowledge about the internal program design and code to perform this.

During unit testing, we carried out various testing task such as the reflection of the unit data on database and its interface. Various types of bugs associated with the component were identified and fixed. We use various functional keys to test our software.

In our software unit testing is concerned with the stock units, opening stock units and product units validation as well as the validation of product units.

CHAPTER – 8: CONCLUSION AND LESSON LEARNT

8.1 Project Limitation

Since this is our first project it has some limitation. Due to less knowledge in particular fields and limited time we were not able to fulfill all our expectations that we expected we could do while the project got started. We hope this limitations are considerable. Some of the project limitations are:

- This application is not suitable for those organization where there is large quantity of product and different level of warehouses
- This software application is able to generate only simple reports.
- Single admin panel is only made.
- It is not suitable for large organization.

8.2 Conclusion

To conclude, Inventory Management System is a simple desktop based application basically suitable for small organization. It has every basic items which are used for the small organization. Our team is successful in making the application where we can update, insert and delete the item as per the requirement. This application also provides a simple report on daily basis to know the daily sales and purchase details.

This application matches for small organization where there small limited if godwoms.

Through it has some limitations, our team strongly believes that the implementation of this system will surely benefit the organization.

8.3 Lesson Learnt

Doing something for long time periods always gives good lesson. Some of the things that our team learnt are listed as below:

- Basically we learnt to work in team.
- Learnt about the IMS process.
- Learnt about .NET technology, its components and ways to implement them
- Learnt to work in pressure and to be patient.
- Learnt to manage the database under Microsoft SQL server 2008.

8.4 Future Enhancements

Since this project was started with very little knowledge about the Inventory Management System, we came to know about the enhancement capability during the

process of building it. Some of the scope we can increase for the betterment and effectiveness oar listed below:

- Interactive user interface design.
- Manage Stock Godown wise.
- Use of Oracle as its database.
- Online payment system can be added.
- Making the system flexible in any type.
- Sales and purchase return system will be added in order to make return of products.
- Lost and breakage

REFERENCES

Software Reference

- Swatik Accounting And Inventory Software

High-tech Software, Kalimati

- Inventory Management Software

Sagar International, Balkhu

Website

Visual Studio Official Site: <https://msdn.microsoft.com/en-us/library/dd492171.aspx>