

# **CHAPTER 1**

## **1. INTRODUCTION**

### **1.1 Inventory Management System**

An inventory management system is the combination of technology (hardware and software) and processes and procedures that oversee the monitoring and maintenance of stocked products, whether those products are company assets, raw materials and supplies, or finished products ready to be sent to vendors or end consumers.

This system can widely be used by normal shops, departmental stores or MNCs for keeping a proper track of the stock. It also consists of information like manager details, customer details etc.

With the help of this system, we can fix a minimum quantity of any inventory below which we need to place an order for that inventory. This will help us in good sales results and never the out-of-stock stage for any inventory. Each new stock is created and entitled with the named and the entry date of that stock and it can also be updating any time when required as per the transaction or the sales is returned in case.

### **1.2 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 the system easily managed and can be secured.
- To make the system easily managed and can be secured.
- The purpose of inventory management is to ensure availability of materials in sufficient quality and quantity.
- Thus, it is very essential to have proper control and management of inventory.

- This will help us in maintain the exact count of any product.
- Can help us to set minimum quantity of any product below which we can order the product from manufacturer.
- Can reduce duplicate entries.

### **1.3 Objectives**

The main objectives of inventory management are operational and financial. Listed below are some of the main objectives of inventory management.

- To have stock available as and when required.
- To maintain adequate accounting and understanding of inventory.
- To avoid both overstocking and under stocking of inventory.
- To ensure right quality goods at reasonable price.
- To design proper organization for inventory management.
- To bring down the inventory carrying cost.
- To facilitate furnishing of data for short term and long-term planning and control of inventory.
- To minimize losses through deterioration, pilferage, wastes and damages.
- To avoid duplication in ordering or replenishing stocks.
- To facilitate purchasing economies.
- To decide which item to stock and which item to procure on demand.

## **1.4 Scope and applications**

The scope of the study is extended to materials department in particular and purchases and stores department, in general further the study confiner to the Company.

This study was mainly concentrated on inventory management adopted by the company. The mode of selecting of the vendors, managing the orders and further actions for satisfying the requirement are analyzed.

The company's purchase procedure will influence more on the level of inventory of the company. So, the study of purchase has vital role to understand the efficient system of the company.

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

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

## 1.5 General and Unique Services in the database application

A database application for an inventory management system typically provides general services such as data storage, retrieval, and management of inventory-related information. Some of the general services that can be expected in an inventory management system are:

- Stock management: This involves keeping track of the quantity of products in stock, their location, and any movements in and out of the inventory.
- Order processing: This service involves receiving and processing orders from customers, checking inventory levels, and updating stock records accordingly.
- Reporting: Inventory management systems should provide various reporting features to help users analyze inventory levels, sales trends, and other critical information.
- Reorder management: This service helps users determine when to reorder products, how much to order, and from whom.

In addition to the general services, there may be unique services in a database application for an inventory management system, depending on the specific needs of the business. Some examples of unique services include:

- Integration with other systems: This service allows for the seamless integration of the inventory management system with other systems such as accounting, shipping, and purchasing.
- Mobile access: This service allows users to access inventory information from their mobile devices, enabling them to manage inventory on the go.
- Customizable interfaces: This service allows users to customize the interface to suit their specific needs, making the system more user-friendly and efficient.

## 1.6 Software Requirements Specification

The System aims at providing an efficient interface to the user for managing of inventory, it shall also provide the user varied options for managing the inventory through various functions at hand. The ingredient levels are continuously monitored based on their usage and are checked for the threshold levels in the inventory and accordingly the user is alerted about low levels of certain ingredients. The design is such that the user does not have to manually update the inventory every time, the System does it for the user.

### **Performance:**

- The system must not lag, because the workers using it don't have down-time to wait for it to complete an action.
- The system must complete updating the databases, adding of recipe, ingredient, vendor and occasions successfully every time the user requests such a process.

### **Interfacing:**

- The system must offer an easy and simple way of viewing the current inventory.
- The system must be able to display the relationships between vendors, ingredients, and recipes in an intuitive manner

## **CHAPTER 2**

### **2. LITERATURE SURVEY**

#### **2.1 Existing system:**

There is a number of Inventory Management System available in the market. After doing my research, I have come to know that most of them are limited to few products. Some others are lacking in good UI. Marketing points are not much focused on increasing sales.

Customer management system and Inventory Management system can't be linked due to different organization which leads to compromising the client satisfaction level. Most of them are not using the cloud computing concept but we are trying to develop such a system that is for everyone rather than for only big companies or for a small organization.

The limitations of existing inventory management systems can vary depending on the specific software and the needs of the business. However, some common limitations include:

- **Scalability:** Some inventory management systems may not be able to scale as the business grows, which could result in the need for a new system in the future.
- **Customization:** Some systems may not offer enough customization options, making it difficult to tailor the system to the specific needs of the business.
- **Integration:** Some inventory management systems may not integrate well with other software or systems, which could create inefficiencies in the business processes.

## 2.2 Proposed system:

Proposed system is a software application which avoids more manual hours that need to spend in record keeping and generating reports. This application keeps the data in a centralized way which is available to all the users simultaneously. It is very easy to manage historical data in database. No specific training is required for the employees to use this application. They can easily use the tool that decreases manual hours spending for normal things and hence increases the performance. As the data is centralized it is very easy to maintain the stocks of the various items in all go downs.

Advantages:

The following are the advantages of proposed system

- Easy to manage all the daily transactions
- Can generate required reports easily
- Easy to manage historical data in a secure manner
- Centralized database helps in avoiding conflicts
- Easy to use GUI that does not requires specific training.

## CHAPTER 3

### 3. SYSTEM ARCHITECTURE AND DESIGN

#### 3.1 Architecture Diagram:

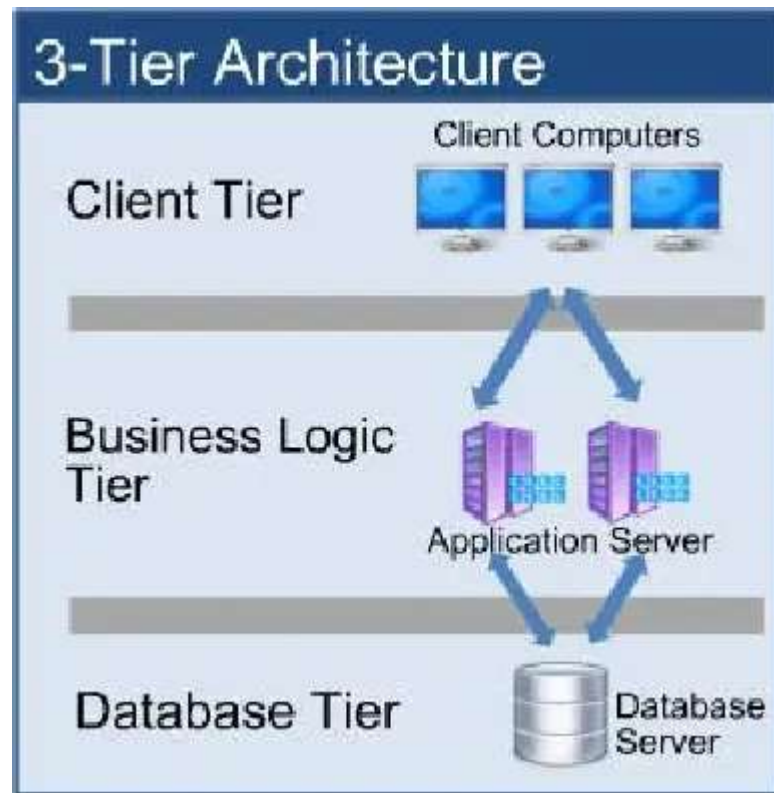


Figure 3.1 Tier Architecture

**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.

**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 tie, which simplify tasks that were done by the client's tier.

**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.



### 3.1.1 Front end (UI) design:

Microsoft Visual Studio comes with VB.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.

The VB.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#.

#### **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 provide three methods that are used to execute commands on the database.

- Execute Non-Query: Executes commands that have no return values such as INSERT, UPDATE AND DELETE.
- Execute Scalar: Returns a single value from a database query
- Execute Reader: Returns a result set by way of a Data Reader Objects.

### 3.1.2 Back end (Database Table) design:

An inventory management system backend using SQL can be designed in many ways depending on the specific requirements and business needs.

- Products Table: This table stores information about all the products that are available in the inventory.  
Columns: ProductID, ProductName, Category, Quantity, CostPrice, SellingPrice.
- Suppliers Table: This table stores information about all the suppliers who supply products to the inventory.  
Columns: SupplierID, SupplierName, Address, Phone, Email
- Orders Table: This table stores information about all the orders placed by the customers.  
Columns: OrderID, CustomerID, OrderDate, TotalAmount, Status
- Customers Table: This table stores information about all the customers who place orders.  
Columns: CustomerID, CustomerName, Address, Phone, Email

### 3.2 ER Diagram and Use case Diagram:

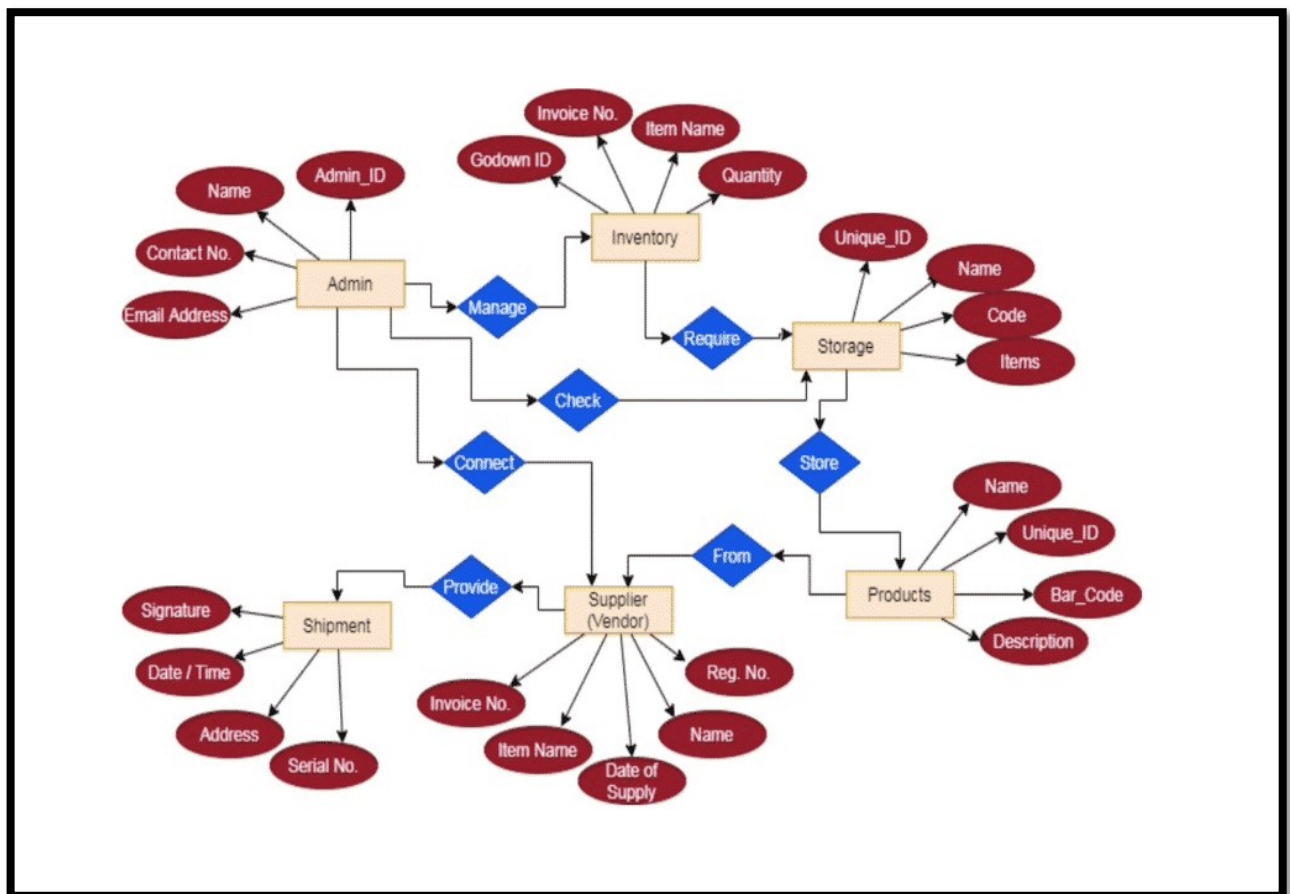
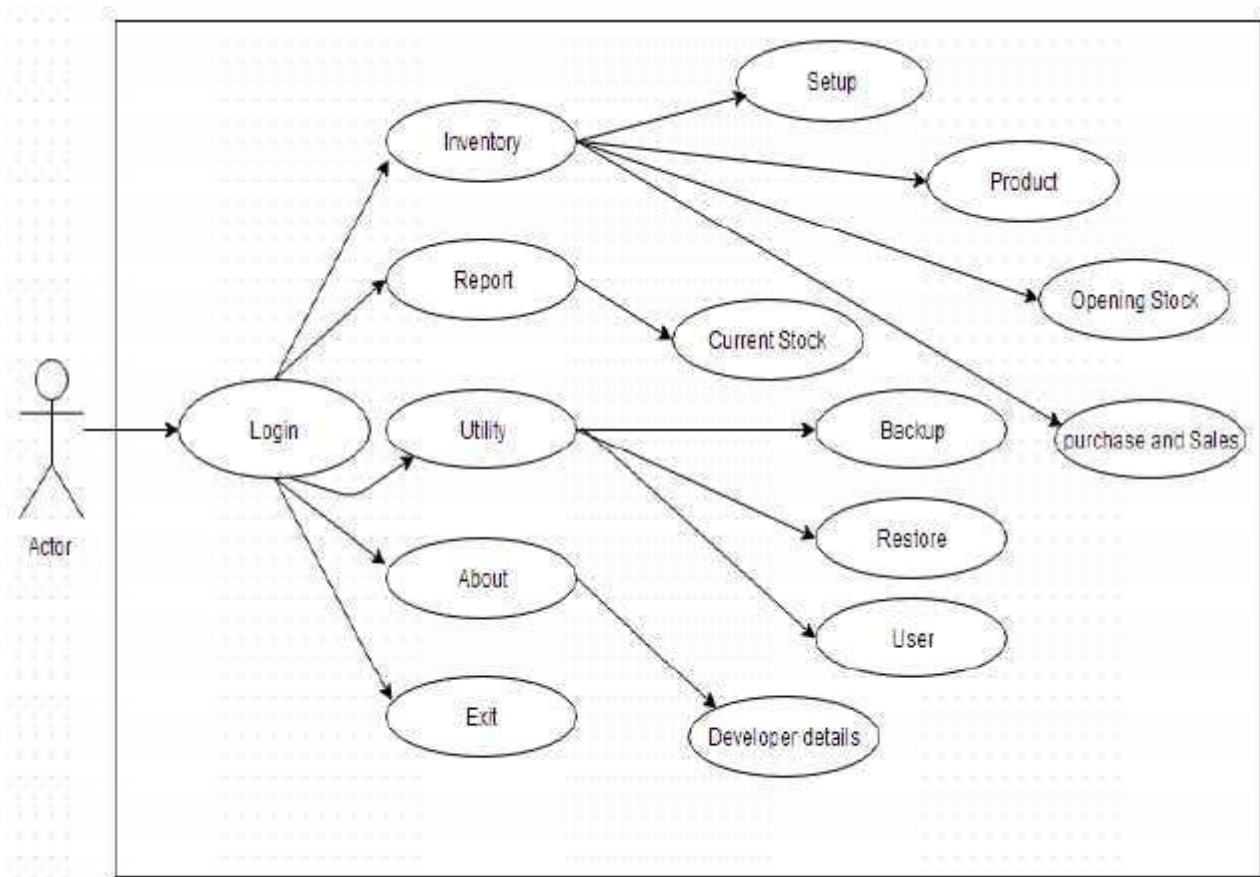


Figure 3.2.1 ER Diagram for Inventory Management System



**Figure 3.2.2 Use Case Diagram for Inventory Management System**

## CHAPTER 4

### 4. Modules and Functionalities

#### 4.1 MODULES:

1. **Product Catalogue:** A database that stores information about each product such as its name, description, price, SKU, and supplier details.
2. **Inventory Tracking:** This module helps you track the quantity of products you have in stock, the location of each item, and any changes in stock levels. You can also set up alerts to notify you when stock levels are running low.
3. **Order Management:** This module allows you to manage incoming orders, allocate stock, and track the status of each order from purchase to delivery.
4. **Purchasing:** This module helps you manage the purchasing process, including supplier selection, order placement, and receiving goods.
5. **Reporting and Analytics:** This module provides you with insights into your inventory levels, sales trends, and other key metrics to help you make informed business decisions.

#### 4.2 Functionalities:

1. **Barcode Scanning:** This functionality allows you to use barcode scanners to track inventory movements, conduct stock takes, and update inventory records.
2. **Integration with E-commerce Platforms:** If you sell products online, you can integrate your inventory management system with your e-commerce platform to keep track of stock levels, automate order processing, and manage customer information.
3. **Multi-Location Inventory Management:** If you have multiple warehouses or store locations, this module allows you to manage inventory levels and stock movements across all locations.
4. **Reorder Point and Safety Stock:** This functionality helps you determine when to reorder products based on minimum and maximum stock levels, and set aside safety stock to prevent stockouts.
5. **Serial Number Tracking:** This feature allows you to track individual products by serial number, which is useful for products with warranties or for tracking specific batches or lots of products.

## CHAPTER 5

### 5. CODING AND TESTING

#### 5.1 CODING:

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.

- **Logic:**

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

- **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)
    {

    }

}
}
}

```

## 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); }
    }
}

```

CREATE TABLE:

SQL> create table brands(

2 bid number(5),

3 bname varchar(20)

4 );

Table created.

SQL> alter table brands

2 add primary

key(bid); Table altered.

SQL> create table inv\_user(

2 user\_id varchar(20),

3 name varchar(20),

4 password varchar(20),

5 last\_login timestamp,

6 user\_type varchar(10)

7 );

Table created.

SQL> create table categories(

2 cid number(5),

3 category\_name varchar(20)

4 );

Table created.

SQL> alter table categories

2 add primary

key(cid); Table altered.

SQL> alter table inv\_user

2 add primary key(user\_id);

Table altered.

SQL> create table product(

2 pid number(5) primary key,

3 cid number(5) references  
categories(cid),

4 bid number(5) references  
brands(bid),

5 sid number(5),

6 pname varchar(20),

7 p\_stock number(5),

8 price number(5),

9 added\_date date);

Table created.

SQL> create table customer(

2 cust\_id number(5) primary key,

3 name varchar(20),

4 mobno number(10)

5 );

Table created.

SQL> create table report(

2 item\_no number(5),

3 product\_name varchar(20),

4 quantity number(5),

5 net\_price number(5),



```
6 transaction_id number(5)references
transaction(id)
```

```
7 );
```

INSERTION:

INSERT INTO BRANDS:

```
SQL> insert into brands values(
```

```
2 '&bid'
```

```
3 ,
```

```
4 '&bname');
```

Enter value for bid: 1

old 2: '&bid'

new 2: '1'

Enter value for bname: Apple

old 4: '&bname')

new 4: 'Apple')

1 row created.

1 row created.

```
SQL> insert into brands
values(2,'Samsung');
```

1 row created.

```
SQL> insert into brands
values(3,'Nike');
```

1 row created.

```
SQL> insert into brands
values(4,'Fortune');
```

1 row created.

INSERT INTO INV\_USER:

```
SQL> insert into inv_user values(
```

```
2 '&user_id',
```

```
3 '&name',
```

```
4 '&password',
```

Enter value for user\_id:  
vidit@gmail.com

old 2: '&user\_id',

new 2: 'vidit@gmail.com',

Enter value for name: vidit

old 3: '&name',

new 3: 'vidit',

Enter value for password: 1234

old 4: '&password',

new 4: '1234',

1 row created.

```
SQL> insert into inv_user
values('harsh@gmail.com','Harsh
Khanelwal','1111','30-oct-18
10:20','Manager');
```

1 row created.

```
SQL> insert into inv_user
values('prashant@gmail.com','Prasha
nt','0011','29-oct-18
```

```
10:20','Accountant');
```

1 row created.

INSERT INTO CATEGORIES:

```
SQL> insert into categories values(
```

```
2 '&cid',
```

3 '&category\_name');

Enter value for cid: 1

old 2: '&cid',

new 2: '1',

Enter value for category\_name:  
Electroincs

old 3: '&category\_name')

new 3: 'Electroincs')

1 row created.

SQL> insert into categories  
values(2,'Clothing');

1 row created.

SQL> insert into categories  
values(3,'Grocey');

1 row created.

INSERT INTO PRODUCT:

SQL> insert into product values(

2 '&pid',

3 '&cid',

4 '&bid',

5 '&sid',

6 '&pname',

7 '&p\_stock',

8 '&price',

9 '&added\_date');

Enter value for pid: 1

old 2: '&pid',

new 2: '1',

Enter value for cid: 1

old 3: '&cid',

new 3: '1',

Enter value for bid: 1

old 4: '&bid',

new 4: '1',

Enter value for sid: 1

old 5: '&sid',

new 5: '1',

Enter value for pname: IPHONE

old 6: '&pname',

new 6: 'IPHONE',

Enter value for p\_stock: 4

old 7: '&p\_stock',

new 7: '4',

Enter value for price: 45000

old 8: '&price',

new 8: '45000',

Enter value for added\_date: 31-oct-18

old 9: '&added\_date')

new 9: '31-oct-18')

1 row created.

SQL> insert into product  
values(2,1,1,1,'Airpods',3,19000,'27-  
oct18'); 1 row created.

```
SQL> insert into product
values(3,1,1,1,'Smart
Watch',3,19000,'27-oct-18');
```

1 row created.

```
SQL> insert into product
values(4,2,3,2,'Air Max',6,7000,'27-
oct-18');
```

1 row created.

```
SQL> insert into product
values(5,3,4,3,'REFINED
OIL',6,750,'25-oct-18');
```

1 row created.

INSERT INTO TRANSACTIONS:

```
SQL> insert into transaction values(
```

2 '&id',

3 '&total\_amount',

4 '&paid',

5 '&due',

6 '&gst',

7 '&discount',

8 '&payment\_method',

9 '&cart\_id');

Enter value for id: 1

old 2: '&id',

new 2: '1',

Enter value for total\_amount: 57000

old 3: '&total\_amount',

new 3: '25000',

Enter value for paid: 2000

old 4: '&paid',

new 4: '20000',

Enter value for due: 5000

old 5: '&due',

new 5: '5000',

Enter value for gst: 350

old 6: '&gst',

new 6: '350',

Enter value for discount: 350

old 7: '&discount',

new 7: '350',

Enter value for payment\_method: card

old 8: '&payment\_method',

new 8: 'card',

Enter value for cart\_id: 1

old 9: '&cart\_id')

new 9: '1')

1 row created.

```
insert into transaction
values(2,57000,57000,0,570,570,'cas
h',2);
```

```
SQL> insert into transaction
values(3,19000,17000,2000,190,190,'
cash',3);
```

1 row created. SQL> insert into transaction

```
values(3,19000,17000,2000,190,190,'
```

cash',3);

1 row created.

PL/SQL

Functions:

SQL> declare

2 due1 number(7);

3 cart\_id1 number(7);

4       function       get\_cart(c\_id  
number)return number is

5 begin

6 return (c\_id);

7 end;

8 begin

9 cart\_id1:=get\_cart('&c\_id');

10 select due into due1 from  
transaction where cart\_id=cart\_id1;

11 dbms\_output.put\_line(due1);

12 end;

13 /

Enter value for c\_id: 1

old 9: cart\_id1:=get\_cart('&c\_id');

new 9: cart\_id1:=get\_cart('1');

5000

PL/SQL procedure successfully  
completed.

Cursors:

SQL> DECLARE

2 p\_id product.pid%type;

3 p\_name product.pname%type;

4 p\_stock product.p\_stock%type;

5 cursor p\_product is

6 select pid,pname ,p\_stock from  
product;

7 begin

8 open p\_product;

9 loop

10 fetch p\_product into  
p\_id,p\_name,p\_stock;

11 exit when p\_product%notfound;

12 dbms\_output.put\_line(p\_id||  
'||p\_name||' ||p\_stock);

13 end loop;

14 close p\_product;

15 end;

16 /

1 IPHONE 4

2 AirPods 3

3 Smart Watch 3

4 Air Max 6

5 REFINED OIL 6

PL/SQL procedure successfully  
completed

## **5.2 TESTING:**

The purpose of software testing is to assess 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.

## CHAPTER 6

### 6. RESULTS AND DISCUSSIONS

#### 6.1 OUTPUTS:

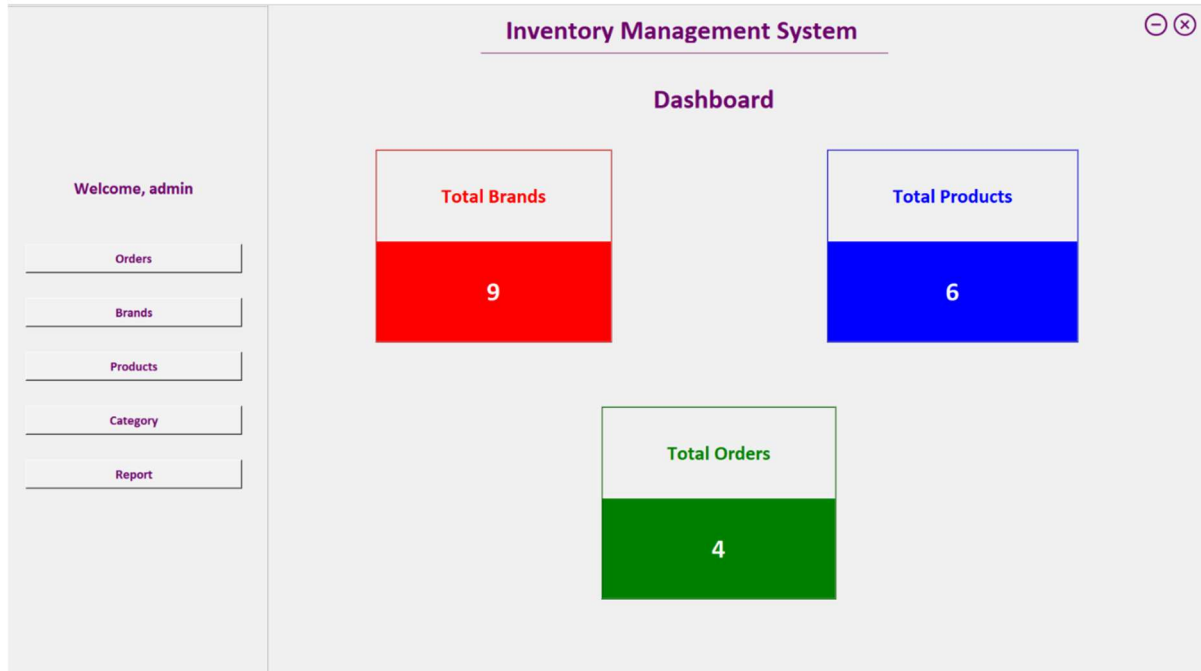
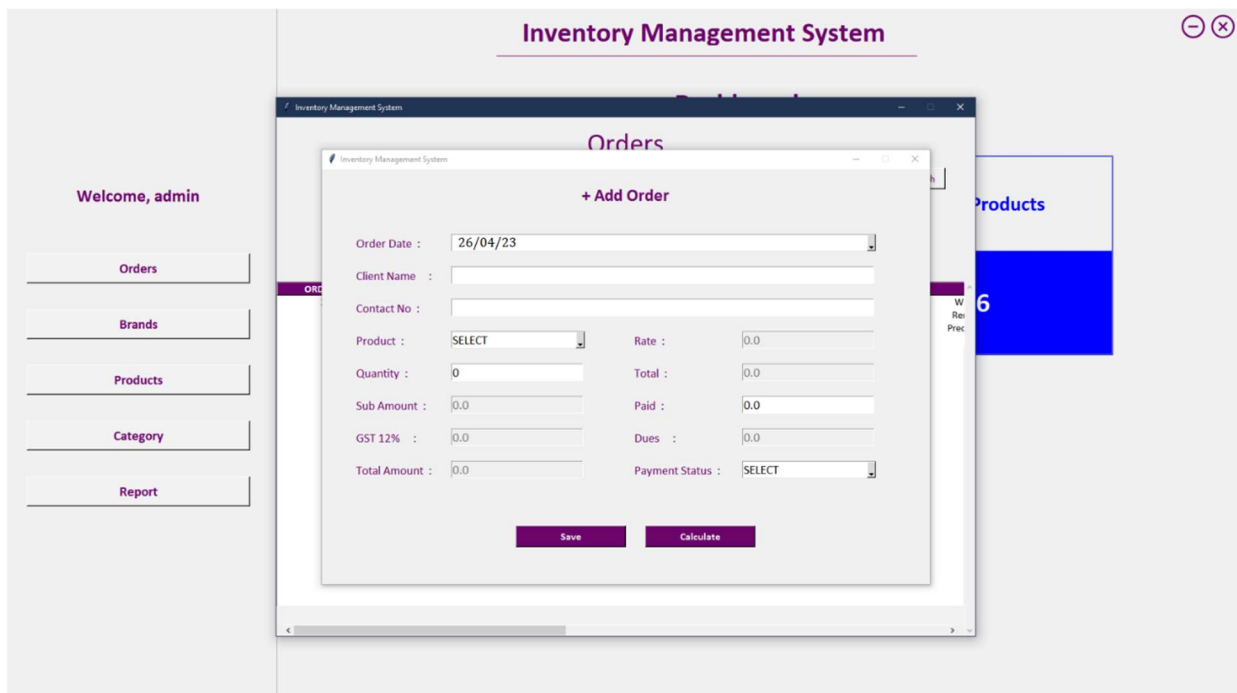
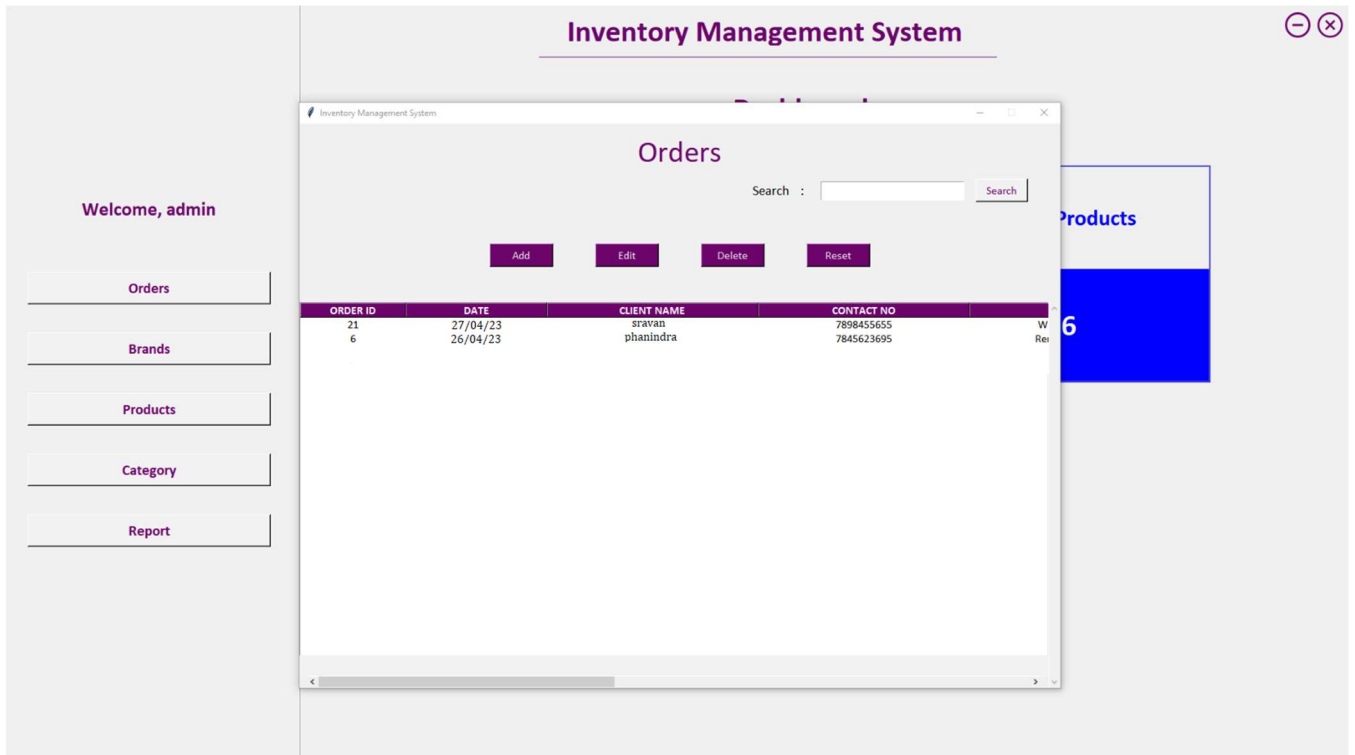


Figure 6.1.1 Home Page for placing order



The screenshot shows the 'Add Order' form within the 'Inventory Management System'. The form is titled '+ Add Order' and contains the following fields: 'Order Date' (26/04/23), 'Client Name', 'Contact No', 'Product' (a dropdown menu), 'Quantity' (0), 'Sub Amount' (0.0), 'GST 12%' (0.0), 'Total Amount' (0.0), 'Rate' (0.0), 'Total' (0.0), 'Paid' (0.0), 'Dues' (0.0), and 'Payment Status' (a dropdown menu). There are 'Save' and 'Calculate' buttons at the bottom. The background shows the dashboard with 'Total Products' (6) visible.

Figure 6.1.2 Product and Price Details



**Figure 6.1.3 Customer and Order Details**

## 6.2 DISCUSSION ABOUT OUTCOME:

In terms of discussions, it's important to note that the benefits of an inventory management system can vary depending on the specific needs and requirements of the business. Factors such as the size of the inventory, the complexity of the supply chain, and the types of products being sold can all impact the effectiveness of an inventory management system.

Additionally, while an inventory management system can provide many benefits, it's important to also consider the potential drawbacks, such as the cost of implementation and ongoing maintenance, as well as the potential for errors or system failures.

## **CHAPTER 7**

### **7. CONCLUSION AND FUTURE ENHANCEMENT**

#### **7.1 CONCLUSION:**

To conclude, Inventory Management System is a simple desktop-based application basically suitable for small organization. It has every basic item 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 go downs.

#### **7.2 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

- [1] Visual Studio Official Site: <https://msdn.microsoft.com/en-us/library/dd492171.aspx>
- [2] [https://www.academia.edu/26003928/Final\\_Year\\_Project\\_On\\_Inventory\\_Management\\_System\\_Submitted\\_By](https://www.academia.edu/26003928/Final_Year_Project_On_Inventory_Management_System_Submitted_By)
- [3] <https://docs.oracle.com/en/solutions/build-governance-app-oracle-paas/create-front-end-application-using-visual-builder-cloud-service1.html>
- [4] <https://www.slideshare.net/DivyaBaghel111/synopsis-on-inventorymanagementsystem>