



## **BVDU AMPLIFY DITM**

### **MINOR PROJECT – 1**

#### **TOPICS:**

- **VB.NET-**  
Customer and Product Management
- **C++**  
Employee Management System

**Made By-**

Abhik Rai

**Index:****Vb.Net Project**

Acknowledgement-----	1
About the project-----	4
User-Interface Sketch-----	5
Entity-Relationship Diagram-----	6
Design and Code implementation-----	7
Table Designs-----	13
Future Expansion and Summary-----	17

**C++ Project**

Introduction-----	18
OOP Features used-----	19
Design Output Flow-----	21
Code Description-----	24
Future Expansion and Summary-----	36

**Acknowledgement:**

I would like to thank all my teachers who guided me for this project.

My deepest gratitude to Bhaskar Sir and Chanda madam who helped me to overcome difficulties at certain points in my project.

I would also like to thank Mr. Prashant Hinduja Sir for providing us opportunity to build up our skills and talent.

I greatly acknowledge all the support provided by my friends who helped me to accomplish this project. I also thank my father for his help and support.

**Customer and Product Management:****Business Application Development in Windows Environment Project Report****Introduction:**

In this project I have created a business application for Energy Technics Company. In this application the admin can manage their customers and the products that they sell.

**Objective:**

The main objective of this project is to manage the customers and the products of a company. The admin can add a customer record, update customer record and delete a record and search a record from the customers list. In the same way the admin can add a product detail, update a product detail, delete a product detail and search for a product from the products list.

**Software Used:**

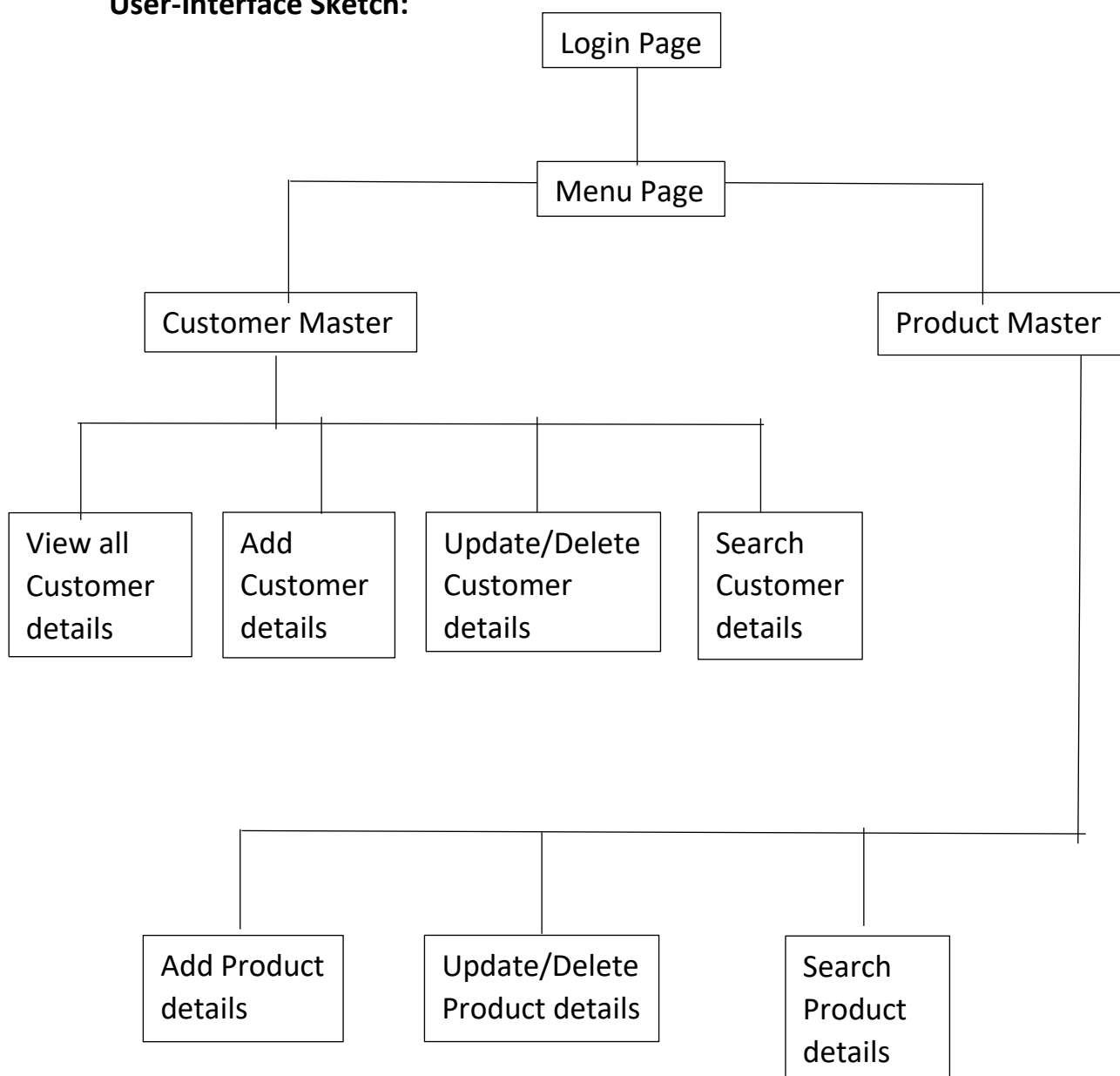
- The User Interface Design and the code has been written in Visual Basic: Community Edition 2016
- The database is stored and retrieved from the MS SQL Server 2014

**Technical Specification:**

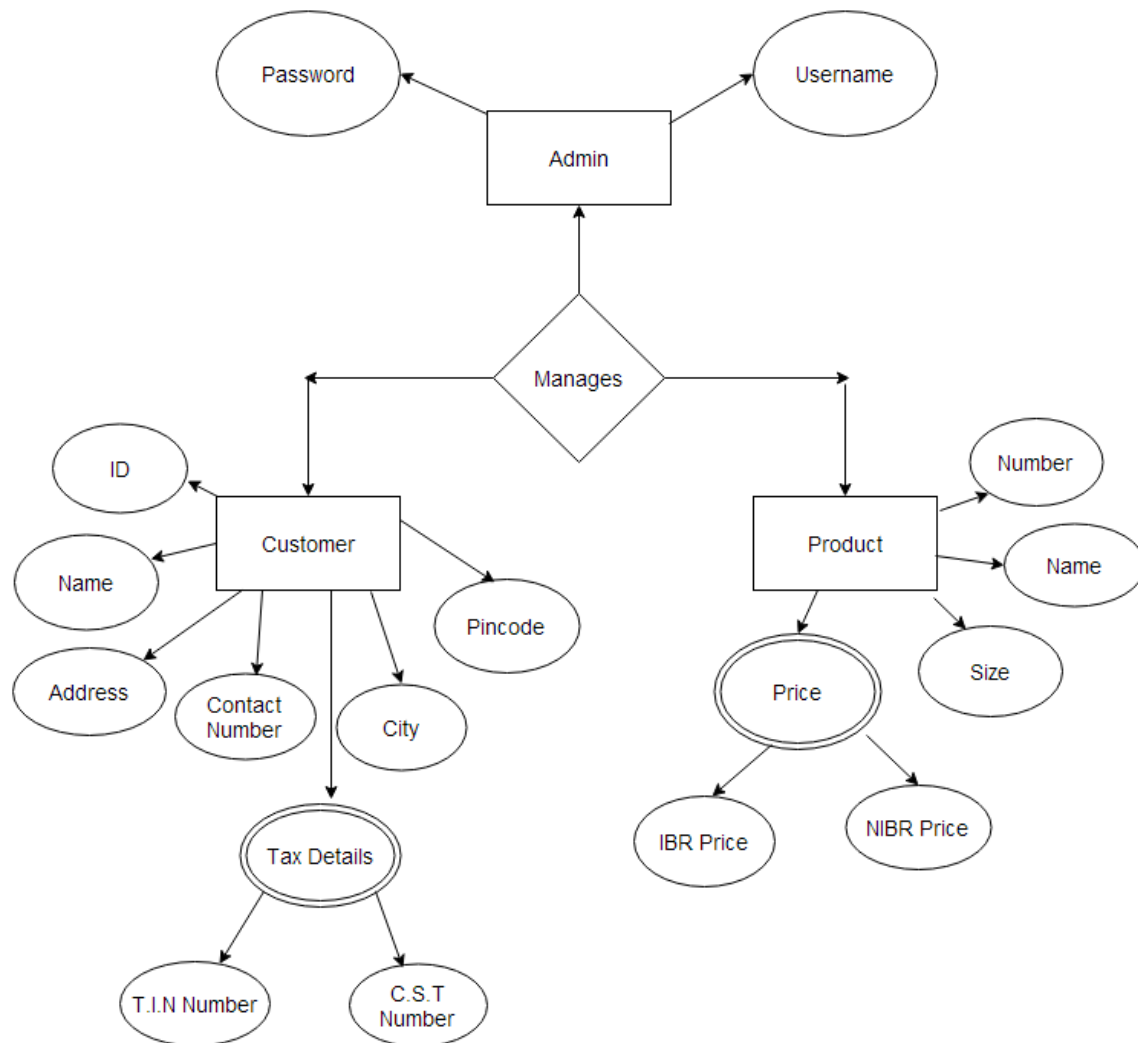
The data is being stored in the database in SQL SERVER 2014 when he customer or the products are added. The data is being retrieved from the database while updating, deleting and searching for a customer or product.

**Steps to achieve output:**

- Thinking about the project and highlighting its main points was my first task.
- Making a rough sketch and representing it diagrammatically (flowchart).
- Use of database.

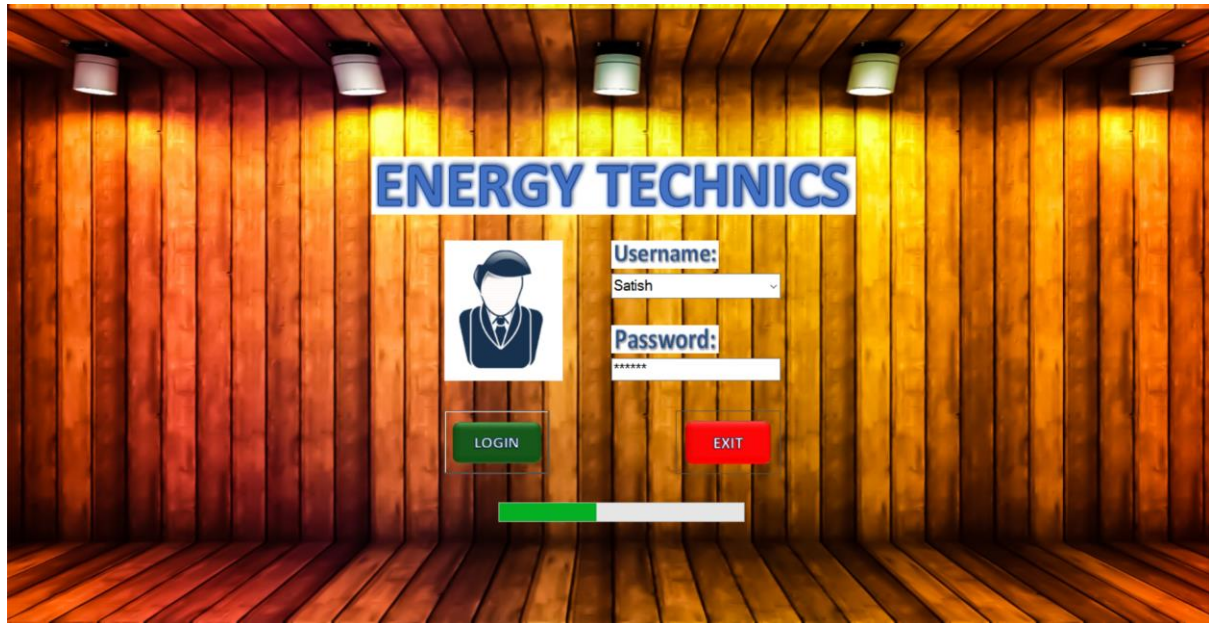
**User-Interface Sketch:**

## Entity-Relationship Diagram



## Design and Code Implementation:

### Login Page:



In this the user can select only two usernames from the combo box. Each username has a different password. If the user enters a wrong password a message box will be displayed saying wrong username or password. After clicking on login button the progress bar will start and the next page will be displayed.

### Code:

```
ProgressBar1.Show()  
    If ComboxUsername.Text = "Satish" And txtPassword.Text = "energy" Then  
        Timer1.Start()  
        ProgressBar1.Show()  
    ElseIf ComboxUsername.Text = "Uday" And txtPassword.Text = "technics"  
Then  
        Timer1.Start()  
        ProgressBar1.Show()  
    Else  
        ComboxUsername.Text = "Select"  
        txtPassword.Text = ""  
        MsgBox("You have entered the wrong password")  
    End If
```

## Menu Page:

[Customer Master](#) [Product Master](#) [Logout](#)



In this page the user can select the options from the menu strip. The options have further sub options like adding customer, deleting customer, etc.



## Adding Customer Details:

The screenshot shows a web form for adding customer details. It is divided into three main sections: Account Details, Sales Tax Details, and Contact Details. The Account Details section includes fields for Id, Alpha Name, Mailing Name, and Group. The Sales Tax Details section includes fields for TIN No. and C.S.T No. The Contact Details section includes fields for Address, Contact Number, City, Pin Code, and Country. A yellow button labeled 'ADD' is positioned to the right of the Sales Tax Details section. A green button labeled 'BACK' with a right-pointing arrow is located at the bottom right of the Contact Details section. The entire form is set against a blue and black checkered background.

In this page the user will enter the customer details. On clicking the add button. On clicking the add button a message box will be displayed saying "Customer details added".

## Code:

```
MyCn.ConnectionString = "Data Source=abhi-lenovo;Initial
Catalog=ETC;Integrated Security=True"
MyCn.Open()
Using cmd As New SqlCommand("INSERT INTO
[customer_table]([No.],[Alpha Name],[Mailing Name],[Group],[Address],[Contact
Number],[City],[Pin Code],[Country],[TIN No.],[C.S.T
No.]) VALUES ('" & txtbId.Text & "','" & txtbAlphaName.Text & "','" &
txtbMailName.Text & "','" & txtbGroup.Text & "','" & txtbAddress.Text & "','"
& txtbContactNumber.Text & "','" & txtbCity.Text & "','" & txtbPincode.Text &
'"','" & txtbCountry.Text & "','" & txtbTIN.Text & "','" & txtbCST.Text & "'
)', MyCn)
    rp = cmd.ExecuteNonQuery
End Using
If rp > 0 Then
    MsgBox("CUSTOMER DETAILS ADDED")
    clear()
End If
MyCn.Close()
```

## Updating/Deleting Customer Details:

No.	Alpha Name	Mailing Name	Group	Address	Contact Number
1	Tetra Pak				

In this page the user can update and delete a customer details. After clicking on a cell on the data grid view the details will be displayed in the textbox and then edit any detail and click on update button. For deleting the user can click on the arrow in the table which selects the entire row and then click on the delete button.

## Code:

### Update Button

```
MyCn.ConnectionString = "Data Source=abhi-lenovo;Initial Catalog=ETC;Integrated Security=True"
Using cmd As New SqlClient.SqlCommand("UPDATE [customer_table] SET [No.] = '' & id & '',[Alpha Name] = ''
& alphaname & '',[Mailing Name] = '' & mailname & '',[Group] = '' & group & '',[Address] = '' & address &
'',[Contact Number] = '' & contact & '',[City] = '' & city & '',[Pin Code] = '' & pincode & '',[Country]
= '' & country & '',[TIN No.] = '' & tin & '',[C.S.T No.] = '' & cst & '' WHERE [No.] = '' & id & ''",
MyCn)
    rp = cmd.ExecuteNonQuery()
End Using
If rp > 0 Then
    MsgBox("CUSTOMER DETAILS UPDATED")
    clear()
End If
MyCn.Close()
```

### Delete Button

```
MyCn.ConnectionString = "Data Source=abhi-lenovo;Initial Catalog=ETC;Integrated Security=True"
MyCn.Open()
Dim num As String = dg1.SelectedRows(0).Cells(0).Value
Using cmd As New SqlClient.SqlCommand("DELETE FROM [customer_table] WHERE [No.] = " & num & "",
MyCn)
    'dg1.SelectedRows(0).Cells(0).Value
    rp = cmd.ExecuteNonQuery()
End Using
If rp > 0 Then
    MsgBox("CUSTOMER DETAILS DELETED")
    clear()
End If
MyCn.Close()
```

## Search Customer details:

No_	Alpha Name	Mailing Name	Group	Address	Contact Number	City	Pin Code	Country	TIN No_	C_S_T No_
1	Tetra Pak						0			

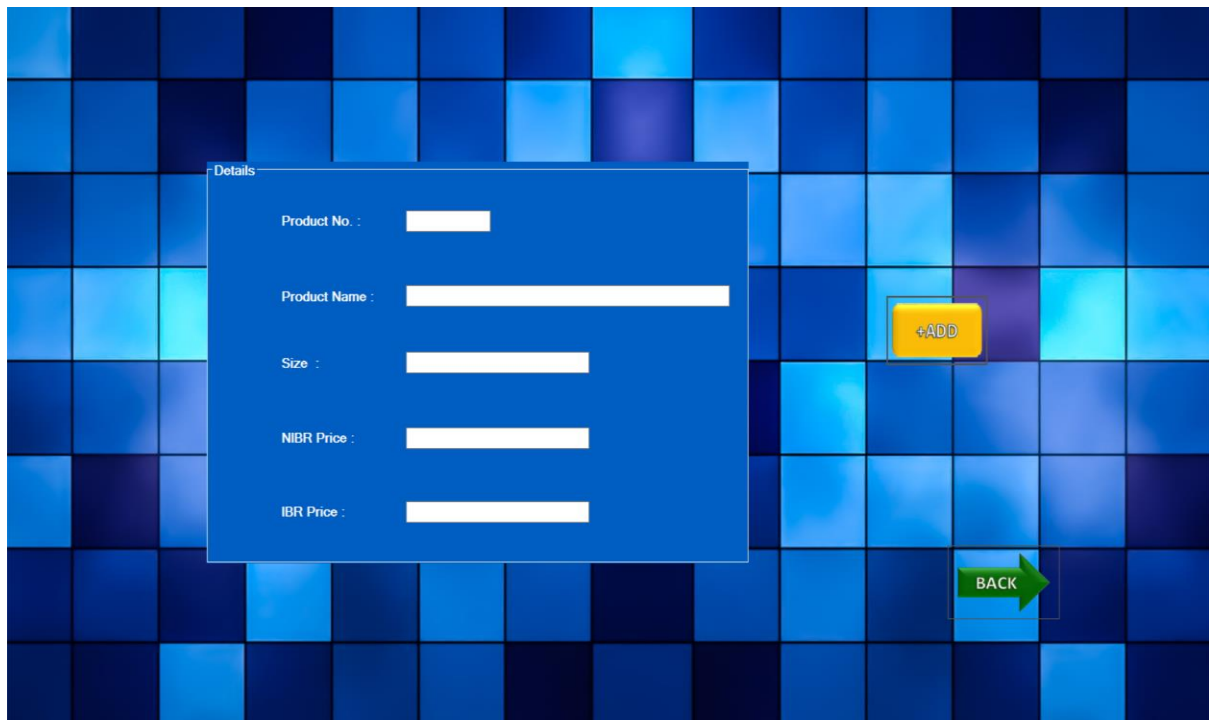
In this page the user can search for a customer details. The user can enter the first name or enter the first alphabet of the name.

## Code:

```
MyCn.ConnectionString = "Data Source=abhi-lenovo;Initial
Catalog=ETC;Integrated Security=True"
MyCn.Open()
Dim alphasrch As String = TxtSearch.Text
Dim Sqlcmd As SqlCommand = New SqlCommand("SELECT * " &
"FROM
[customer_table]" &
"WHERE [Alpha
Name] LIKE '" & alphasrch & "%' ", MyCn)
da = New SqlDataAdapter(Sqlcmd)
ds = New DataSet
da.Fill(ds)
MyCn.Close()

dg1.DataSource = ds.Tables(0)
dg1.Refresh()
```

## Adding Product Details:



In this page the user can enter product details.

## Code:

```
MyCn.ConnectionString = "Data Source=abhi-lenovo;Initial
Catalog=ETC;Integrated Security=True"
MyCn.Open()
Using cmd As New SqlClient.SqlCommand("INSERT INTO
[product_tbl]([No.],[Name],[Size],[NIBR Price],[IBR Price]) VALUES ('"
& txtbNumber.Text & "','" & txtbName.Text & "','" & txtbSize.Text &
"', '" & txtbNIBRprice.Text & "','" & txtbIBRprice.Text & "' )", MyCn)
rp = cmd.ExecuteNonQuery
End Using
If rp > 0 Then
    MsgBox("PRODUCT DETAILS ADDED")
    clear()
End If
MyCn.Close()
```

## Updating/Deleting Product Details:

No.	Name	Size	NIBR Price	IBR Price
1	Air Vent	15	7650.0000	8250.0000
2	Air Eliminator	15	8400.0000	0.0000
3	Float Trap	15	31000.0000	32850.0000
4	Float Trap	20	32200.0000	34100.0000
5	Bucket Trap	15	14100.0000	15100.0000
6	Bucket Trap	20	14300.0000	15250.0000
7	Bucket Trap	25	25600.0000	26500.0000

In this page the user can update and delete the product details.

## Code:

### Update Button

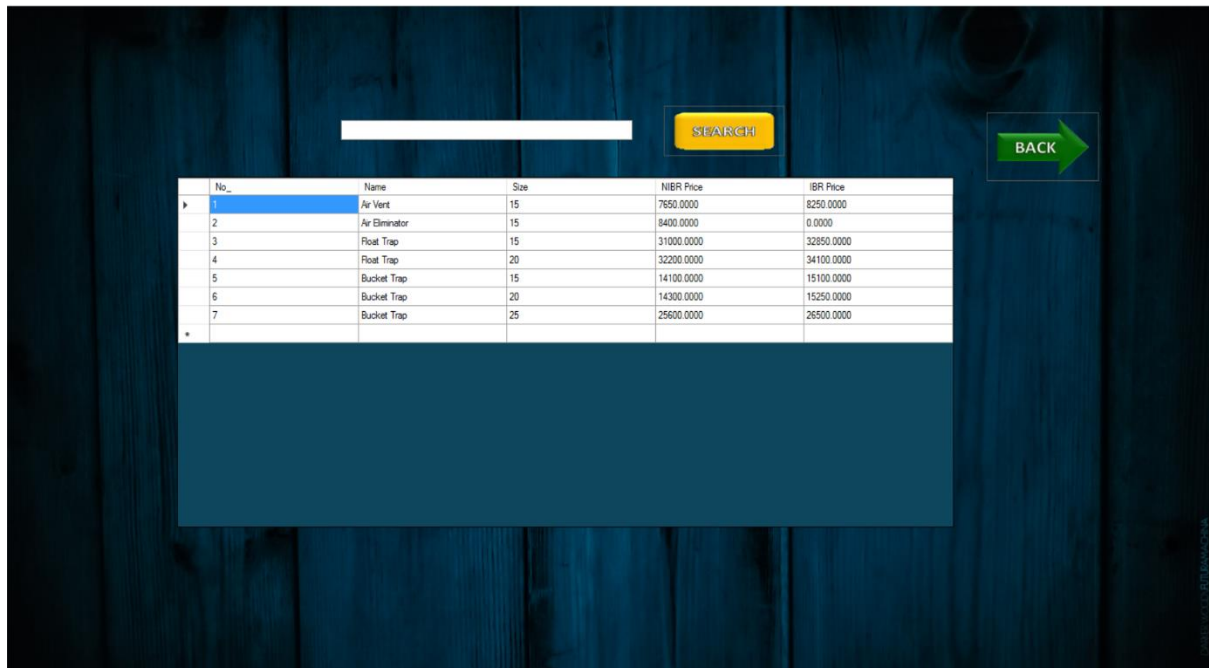
```
MyCn.ConnectionString = "Data Source=abhi-lenovo;Initial Catalog=ETC;Integrated Security=True"
MyCn.Open()
Dim num As String = txtbNumber.Text
Dim nam As String = txtbName.Text
Dim siz As String = txtbSize.Text
Dim nibr As String = txtbNIBRprice.Text
Dim ibr As String = txtbIBRprice.Text

Using cmd As New SqlClient.SqlCommand("UPDATE [product_tbl] SET [Name] = '" & nam & "',[Size] = '" & siz & "',[NIBR Price] = '" & nibr & "',[IBR Price] = '" & ibr & "' WHERE [No.] = '" & num & "' ",
MyCn)
    rp = cmd.ExecuteNonQuery
End Using
If rp > 0 Then
    MsgBox("PRODUCT DETAILS UPDATED")
    clear()
End If
MyCn.Close()
```

### Delete Button

```
MyCn.ConnectionString = "Data Source=abhi-lenovo;Initial Catalog=ETC;Integrated Security=True"
MyCn.Open()
Dim num As String = dg1.SelectedRows(0).Cells(0).Value
Using cmd As New SqlClient.SqlCommand("DELETE FROM [product_tbl] WHERE [No.] = '" & num & "'",
MyCn)
    'dg1.SelectedRows(0).Cells(0).Value
    rp = cmd.ExecuteNonQuery
End Using
If rp > 0 Then
    MsgBox("PRODUCT DETAILS DELETED")
    clear()
End If
MyCn.Close()
```

## Search a Product Detail:



In this page the user can search for a product detail.

## Code:

```

MyCn.ConnectionString = "Data Source=abhi-lenovo;Initial
Catalog=ETC;Integrated Security=True"
MyCn.Open()


Dim pronamesrch As String = txtbSearch.Text
Dim Sqlcmd As SqlCommand = New SqlCommand("SELECT * " &
"FROM
[product_tbl]" &
"WHERE [Name]
LIKE '" & pronamesrch & "%' ", MyCn)
da1 = New SqlDataAdapter(Sqlcmd)
ds1 = New DataSet
da1.Fill(ds1)
MyCn.Close()


dg1.DataSource = ds1.Tables(0)
dg1.Refresh()

```



**Product Table:**

	Name	Data Type	Allow Nulls	Default
	No.	int	<input type="checkbox"/>	
	Name	varchar(50)	<input checked="" type="checkbox"/>	
	Size	float	<input checked="" type="checkbox"/>	
	NIBR Price	money	<input checked="" type="checkbox"/>	
	IBR Price	money	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

	No.	Name	Size	NIBR Price	IBR Price
	1	Air Vent	15	7650.0000	8250.0000
	2	Air Eliminat...	15	8400.0000	0.0000
	3	Float Trap	15	31000.0000	32850.0000
	4	Float Trap	20	32200.0000	34100.0000
	5	Bucket Trap	15	14100.0000	15100.0000
	6	Bucket Trap	20	14300.0000	15250.0000
	7	Bucket Trap	25	25600.0000	26500.0000
*	NULL	NULL	NULL	NULL	NULL



**Future Expansion:**

- Will add billing option in the project where the admin can select a customer and click on the billing option and then a bill will be created with the selected name.

**Conclusion:**

- The overall project was very informative and helped me explore my abilities.
- I learned many features of VB.Net and SQL Server 2014.

**Summary:**

Coming to the end I would like to summarize the whole project as an opportunity to increase one's intellect. It also makes one's concepts clearer about whatever has been used in the project.

**References:**

I took help from the books available in the library and also browsed the internet for information.

- [www.google.com](http://www.google.com)
- [www.stackoverflow.com](http://www.stackoverflow.com)
- GitHub
- [www.howtostartprogramming.com](http://www.howtostartprogramming.com)
- [www.Visualbasic.about.com](http://www.Visualbasic.about.com)
- All About .net

## **C++ Project**

### **Topic:**

Employee Management System

### **Description:**

In this project the user can manage the employee records like salary, name, id, department and post.

### **Steps to achieve output:**

- Thinking about the project and highlighting its main points in prior.
- Making a rough sketch and representing it diagrammatically (flowchart).
- Making the documentation to understand and get a clear idea of the project.
- Acquiring knowledge from different sources about the functions and features.

### **Functionalities:**

- Manages the Employee records.
- View all Employee records.
- Delete a record of the employee.
- Can add an Employee record.
- Can update a record of an employee.
- Can view all records of employee by a particular department.

### **Software Used:**

Turbo C++ 4.0 (64-bit version)

## OOP Features:

### Class and Objects:

The class will contain all data member and methods to use inventory items. Using objects of this class, we can invoke all its members.

### Abstraction:

When we will create the object of the class and use it to call member function, the function calling method will not know about how the process is carried out. It will only be concerned with the result of method invoked.

### Inheritance:

The write class in this code will inherit all the properties from login class. The inheritance will be of type Single. There will be one another single inheritance implemented.

### File Handling:

Concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory (Hard disk).

- For permanent storage.
- The transfer of input - data or output - data from one computer to another can be easily done by using files.

- 

Data Type	Description
<b>ofstream</b>	This data type represents the output file stream and is used to create files and to write information to files.
<b>ifstream</b>	This data type represents the input file stream and is used to read information from files.
<b>fstream</b>	This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

Mode Flag	Description
<b>ios::app</b>	Append mode. All output to that file to be appended to the end.
<b>ios::ate</b>	Open a file for output and move the read/write control to the end of the file.
<b>ios::in</b>	Open a file for reading.
<b>ios::out</b>	Open a file for writing.
<b>ios::trunc</b>	If the file already exists, its contents will be truncated before opening the file.

## Design Output Flow:

### Menu Page:

```
=====EMPLOYEE MANAGEMENT SYSTEM=====

1. Add a new record
2. Search record
3. List Employee by a particular department
4. Display all employee records
5. Update record of an employee
6. Delete record of an employee
7. Exit

Enter your choice: _
```

### Add Employee Record:

```
Employee #ID: 1
Employee Name: abhi
Employee's Post: emp
Employee's Department: technical
Salary:
30000

New record added successfully
Do you want to continue....(y/n)?

_
```

### Search Employee Record:

Enter #ID of employee to be searched: 1

Record found

ID	Name	Post	Department	Salary
1	abhi	emp	technical	30000

Do you want to continue....(y/n)?

### Display all Employee Records:

Record for employee

ID	Name	Post	Department	Salary
1	abhi	emp	technical	30000

1Records found  
Do you want to continue....(y/n)? \_

### Update Employee Record:

```
Updating file.....
Enter employee #ID: 1
The old record of employee having #ID1is:
-----
      1          abhi          emp      technical      30000
Enter new record for employee having #ID: 1
Employee #ID: 1
Employee Name: abhik
Employee's Post: hod
Employee's Department: technial
Salary: 50000
Do you want to continue....(y/n)?
```

### Delete an Employee record:

```
Enter employment #ID to be deleted: 1
The old record of employee having #ID 1 is:
-----
      1          abhik          hod          technial      50000
Do you want to continue....(y/n)?
```

**Code Description:****Header Files in this program:**

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<stdlib.h>
#include<iomanip.h>
#include<string.h>
#include<stdio.h>
```

**An Employee class has been created:**

```
class Employee
{
private:
int EmpID;
char EmpName[50], Post[50], Department[10];
float Salary;
public:
void ReadData();
int GetID();
void DisplayRecord();
char* GetDepartment();
};
```



**The data members will be used to save all the characteristics of the employee.**

**The member functions will**

```
void Employee::ReadData()
```

```
{
```

```
cout<<endl<<"Employee #ID: ";
```

```
cin>>EmpID;
```

```
cout<<"Employee Name: ";
```

```
cin>>EmpName;
```

```
cout<<"Employee's Post: ";
```

```
cin>>Post;
```

```
cout<<"Employee's Department: ";
```

```
cin>>Department;
```

```
cout<<"Salary: ";
```

```
cin>>Salary;
```

```
}
```

```
void Employee::DisplayRecord()
```

```
{
```

```
cout<<endl<<"-----";
```

```
cout<<endl<<setw(5)<<EmpID<<setw(15)<<EmpName<<setw(15)<<Post<<setw(15)<<Department<<setw(8)<<Salary;
```

```
}
```

```
int Employee::GetID()
```

```
{
```

```
return EmpID;
```

```
}
```

```
char* Employee::GetDepartment()
```



```
cout<<"\t\t"_____"<<endl;
;
cout<<endl;
cout<<endl;
cout<<"Enter your choice: ";
cin>>option;
system("cls");
```

**Switch case is used in which the user can select the options**

```
switch(option)
{
```

**In this case (1) the user will add the employee details and the details will be stored in a file. If the user enters an id which is already present in the file, the details will not be added in the file and an error message will be displayed.**

case 1:

```
emp.ReadData();
file.seekg(0,ios::beg);
isFound=0;
file.read((char*)&e,sizeof(e));
while(!file.eof())
{
if(e.GetID()==emp.GetID())
{
cout<<"This ID already exist's Try another #ID";
isFound=1;
break;
```

```
}  
file.read((char*)&e,sizeof(e));  
}  
if(isFound==1)  
break;  
file.clear();  
file.seekp(0,ios::end);  
file.write((char*)&emp, sizeof(emp));  
cout<<endl<<"New record added successfully";  
system("cls");  
break;
```

**In this case (2) the user can search for an employee record by entering the employee id which is already stored in the file. If the entered id is not present in the file no records will be shown**

```
case 2:  
isFound=0;  
cout<<endl<<"Enter #ID of employee to be searched: ";  
cin>>ID;  
file.seekg(0,ios::beg);  
file.read((char*)&e,sizeof(e));  
while(!file.eof())  
{  
if(e.GetID()==ID)  
{  
cout<<endl<<"Record found"<<endl;
```

```
cout<<endl<<setw(5)<<"ID"<<setw(15)<<"Name"<<setw(15)<<"Post"<<setw(15)<<"Department"<<setw(8)<<"Salary";

e.DisplayRecord();

isFound=1;

break;

}

file.read((char*)&e,sizeof(e));

}

file.clear();

if(isFound==0)

cout<<endl<<"No data found for this employee #ID"<<ID;

break;
```

**In this case (3) the user can search for employee records by a specific department from the file. If there are no employee in the department or if the department does not exist no records will be shown.**

```
case 3:

isFound=0;

cout<<"Enter department name to list all employee in it: ";

cin>>Dept;

file.seekg(0,ios::beg);

file.read((char*)&e,sizeof(e));

while(!file.eof())

{

if(strcmp(e.GetDepartment(),Dept)==0)

{

cout<<endl<<"Record found for this department"<<endl;
```

```
cout<<endl<<setw(5)<<"ID"<<setw(15)<<"Name"<<setw(15)<<"Post"<<setw(15)<<"Department"<<setw(8)<<"Salary";
e.DisplayRecord();
isFound=1;
break;
}
file.read((char*)&e,sizeof(e));
}
file.clear();
if(isFound==0)
cout<<endl<<"No data found for this department"<<Dept;
break;
```

**In this case (4) all the employee records will be displayed from the file and if there are no employee records a message will appear saying no records found.**

```
case 4:
cout<<endl<<"Record for employee";
file.clear();
file.seekg(0,ios::beg);
int counter=0;
file.read((char*)&e,sizeof(e));
while(!file.eof())
{
counter++;
if(counter==1)
{
```

```
cout<<endl<<setw(5)<<"ID"<<setw(15)<<"Name"<<setw(15)<<"Post"<<setw(15)<<"Department"<<setw(8)<<"Salary";
}
e.DisplayRecord();
file.read((char*)&e,sizeof(e));
}
cout<<endl<<counter<<"Records found";
file.clear();
break;
```

**In this case (5) the user can update an employee record and the updated record will be saved in the file.**

case 5:

```
int recordNo=0;
cout<<endl<<"Updating file.....";
cout<<endl<<"Enter employee #ID: ";
cin>>ID;
isFound=0;
file.seekg(0,ios::beg);
file.read((char*)&e,sizeof(e));
while(!file.eof())
{
recordNo++;
if(e.GetID()==ID)
{
cout<<"The old record of employee having #ID"<<ID<<"is: ";
e.DisplayRecord();
```

```
isFound=1;
break;
}
file.read((char*)&e,sizeof(e));
}
if(isFound==0)
{
cout<<endl<<"No data found for this employee #ID"<<ID;
break;
}
file.clear();
int location=(recordNo-1)*sizeof(e);
file.seekp(location,ios::beg);
cout<<endl<<"Enter new record for employee having #ID: "<<ID;
e.ReadData();
file.write((char*)&e, sizeof(e));
break;
```

**In this case (6) the user can delete an employee record from the file.**

```
case 6:
recordNo=0;
cout<<endl<<"Enter employment #ID to be deleted: ";
cin>>ID;
isFound=0;
file.seekg(0,ios::beg);
file.read((char*)&e,sizeof(e));
```



```
while(!file.eof())
{
    recordNo++;
    if(e.GetID()==ID)
    {
        cout<<" The old record of employee having #ID "<<ID<<" is: ";
        e.DisplayRecord();
        isFound=1;
        break;
    }
    file.read((char*)&e,sizeof(e));
}
char tempFile[]="temp.txt";
fstream temp(tempFile,ios::out|ios::binary);
if(isFound==0)
{
    cout<<endl<<"No data found for this #ID"<<ID;
    break;
}
else
{
    file.clear();
    file.seekg(0,ios::beg);
    file.read((char*)&e,sizeof(e));
    while(!file.eof())
    {
```

```
if(e.GetID()!=ID)
temp.write((char*)&e,sizeof(e));
file.read((char*)&e,sizeof(e));
}
file.close();
temp.close();
temp.open(tempFile,ios::in|ios::binary);
file.open(fileName,ios::out|ios::binary);
temp.read((char*)&e,sizeof(e));
while(!temp.eof())
{
file.write((char*)&e,sizeof(e));
temp.read((char*)&e,sizeof(e));
}
}
temp.close();
file.close();
remove(tempFile);
file.open(fileName,ios::ate|ios::in|ios::out|ios::binary);
break;

case 7:
exit(0);
break;

default:
```

```
cout<<"Invalid Option, please enter a valid option";  
}  
cout<<"\nDo you want to continue....(y/n)? ";  
cin>>ch;  
}while(ch!='n');  
}
```

**Feasibility Study:**

The system has been tested and the amendments have been made in order to make it more efficient.

The purpose of this project is to develop a console -based system which facilitates management of Employees of a company.

**Future Expansion:**

Will add a sort option in which the user can sort the employees according to the name or salary.

**Conclusion:**

- The overall project was very informative and helped me explore my abilities.
- It brushed up my concepts regarding inheritance, encapsulation, abstraction, file handling and many more.

**Summary:**

It was a great opportunity to work on such a project which increases one's level of thinking. It also makes one's concepts crystal clear about whatever has been used in the project. This project served as a platform to gain knowledge which can be applied in future prospects.

**References:**

- Object Oriented Programming by E BALAGURUSAMY
- Let Us C by YASHWANT KANETKAR
- Used Internet to study about topics in detail.
- [www.stackoverflow.com](http://www.stackoverflow.com)
- [www.github.com](http://www.github.com)
- [www.quora.com](http://www.quora.com)