PROJECT REPORT

ON

“COMPUTER SHOP MANAGEMENT SYSTEM”

BY **VINEET KHANNA** AND **KUNJAN VAGHELA**

BACHELOR OF SCIENCE

(INFORMATION TECHNOLOGY)

FOR THE ACADEMIC YEAR

2015 – 16

UNDER THE GUIDANCE OF

MS. HARSHALA CHAUDHARI

BHAVAN’S COLLEGE

MUNSHI NAGAR, ANDHERI (WEST),

MUMBAI - 400058

Corporate Infosystems (India) Pvt. Ltd.

TO WHOM SO EVER IT MAY CONCERN

This is to certify that Mr. Vineet Mahesh Khanna and Mr. Kunjan Hasmukh Vaghela of T.Y.B.Sc. (Information Technology), Bhavan's College, Andheri (W) has Successfully designed and Implemented the software "Computer Shop Management System" for Corporate Infosystems (India) Pvt. Ltd.

I appreciate the efforts taken by the students in developing the software and Wish them all the Success in their future.

Corporate Infosystems (India) Pvt. Ltd. Mumbai

Signature of Director

Date : 5th May 2016

26287626 / 26282561
32411188 / 32418811
9322597925

REGD. OFFICE :
G-2, OSIA FRIENDSHIP C. H. S., 4th GAOTHAN LANE, OFF J. P. ROAD, OPP RAM MANDIR, ANDHERI (WEST), MUMBAI-400 058.

corporate@mtnl.net.in
corp2001@vsnl.com
infosystemscorporate@gmail.com

Acknowledgement

With the complete work undertaken, I would like to take this opportunity to thank all those who helped me to complete this project with their guidance, co-operation and support. This project would not have been materialized without the help from many quarters. I am thankful to Miss Harshala Chaudhari & Miss Radhe Bane for their guidance.

Lastly, I need to extend my thanks to all those, who helped me directly or indirectly in completing this project.

TABLE OF CONTENTS

I) REQUIREMENT and ANALYSIS PHASE

1) Requirement Gathering

1.1) Company Profile

Name of the Organisation: Corporate Infosystems (India) Pvt. Ltd.
Address of the Organisation:
G-2 Osia Friendship Co-Operative Housing Society, 4 Goathan Lane, Opposite Ram Mandir, Off J.P. Road, Andheri (W), Mumbai – 400058

About the Organisation:
The organisation was founded in the year 1999 by Mr. Deepak G. Sarin. The company is a reseller for all Information Technology Products and offers Products and Services ranging from Hardware, Software and Consumable Products to Laptop Repair Services.

1.2) Problem Definition

Managing inventory, day to day transactions, client and supplier details, employee attendance etc. is done manually or on Tally ERP 9. Although, it provides most of the features with not so great GUI, it is not in accordance to the customer's need. The Director (Administrator) of the company wants a software specifically customised to suit his needs.

1.3) Objective and Scope of Project

Computer Shop Management System (CSMS) will fulfil all the requirements of the customer and is supposed to have the following features:

- The system provides login facility to the users.
- Manage and store suppliers' details including transaction.
- Manage and store clients' details including transaction.
- Inventory control i.e. stock.
- Payroll management.

1.4) Limitations of Current System

The system is time consuming in terms of data processing.
All the records are to be entered manually.
Possibility of human errors.
Lots of hard-copies increase the amount of paper-work to be maintained and also increase the cost of stationery used.
Increased labour cost as maximum human involvement is required.
The system is not secured because the records are sometimes accessible even to some unauthorized persons.

1.5) Advantages of Proposed System

- The records are secured and accessible only to authorized persons.
- Most paper-work is eliminated, creating an almost paper-free environment.
- Labour costs & time requirement are reduced.
- Data processing is faster since the system is computerized.
- Searching of records.
- Manual making of bill & calculation is totally eliminated.
- Manual calculations are eliminated.

1.6) Modules of Proposed System

1) Login
    - Add/Update/Delete user
    - Grant privileges to the user (optional)
2) Manage Client and Supplier details
    - Add/Update/Delete details
    - Track details in real time
3) Manage Inventory
    - Add/Update/Delete items
4) Manage Payroll
    - Add/Update/Delete employee details
5) Store and generate payroll information
6) Bill calculator and storage of bills
7) Generate and store bill related information

2) Feasibility Study

Feasibility study is essential to evaluate cost and benefit of proposed system.
This is very important step because on basis of this system decision is taken on whether to proceeds or postpone the project or to cancel the project
If the result of feasibility study is positive, then the cooperative can proceed to develop a business plan.
They provide in depth derails about the business to determine if and how it can succeed, and serve as valuable tool for developing a winning business plan.

2.1) Operational Feasibility

Operational Feasibility is measure of how well proposed system soles the problem, and takes advantage of opportunities identified during scope definition and how it satisfies the requirements analysis phase of system development.
Thus the proposed system is desirable within the existing framework of the institute.
After the successful development of the proposed system, at least one knowledgeable person will attend to and resolve the queries or doubts, if any, of the user regarding the system, which is expected to work smoothly otherwise.

2.2) Technical Feasibility

Technical feasibility is carried out to determine whether the company has the capability in term of software, hardware, personnel and expertise to handle the completion of project.
With new technology such as ‘C#.NET' and 'MySQL', it is practically possible to develop and update the proposed system since all the necessary software, hardware, technical resources and expertise is currently available.

2.3) Financial and Economic Feasibility

Financial and Economic Feasibility involves the initial one-time investment in hardware & software is bearable and thus is not a constraint for developing the system. Apart from this, the flying school does not need to invest any extra amount for the initial working of the system.
Future updates, if any, are also well within the financial capability of the institute.
Also the benefits out-weight cost and then decision is made o design & implements he system.

2.4) Organisational Feasibility

This demonstrates the Event management company capability & availability, employee involvement & their commitment.
This shows the Management Company and Organizational structure of the project.

3) Requirement Specifications

3.1) System Configuration

Hardware Requirements:

CPU:	1 gigahertz (GHz) or faster processor
RAM:	1 gigabyte (GB) RAM
Storage:	16 gigabyte (GB) available hard disk space

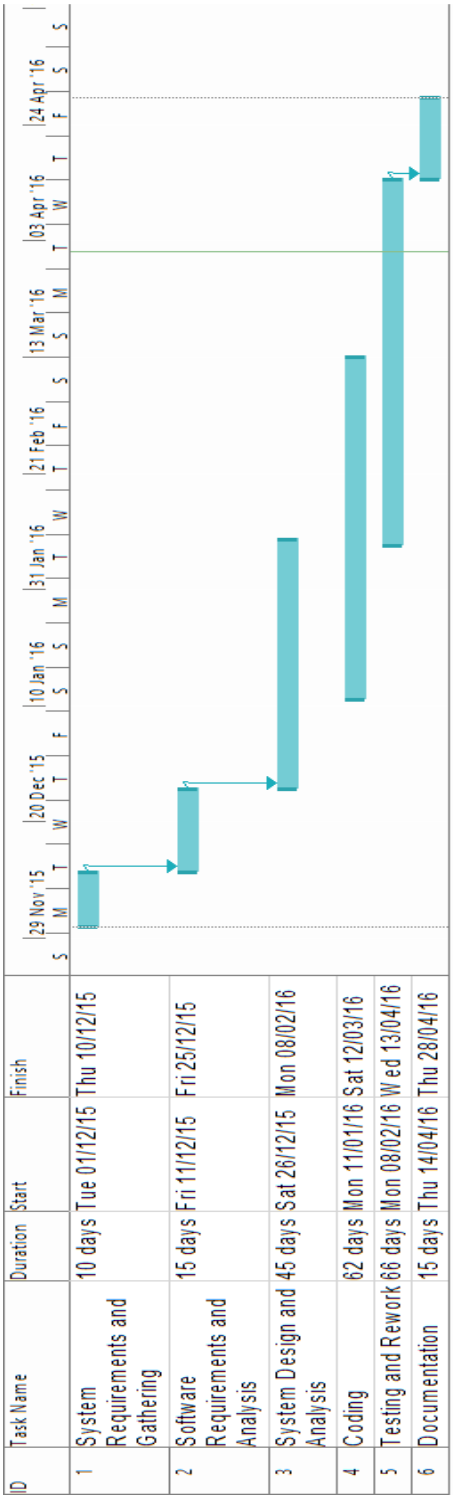
Software Requirements:

Operating System:	Microsoft Windows 7 SP1 and above with .NET Framework 4.0
Frontend:	Microsoft Visual Studio 2013
Backend:	MySQL 2014

II DESIGN PHASE

1) Detailed Life Cycle of Project

1.1) Gantt Chart



ID	Task Name	Duration	Start	Finish
1	System Requirements and Gathering	10 days	Tue 01/12/15	Thu 10/12/15
2	Software Requirements and Analysis	15 days	Fri 11/12/15	Fri 25/12/15
3	System Design and Analysis	45 days	Sat 26/12/15	Mon 08/02/16
4	Coding	62 days	Mon 11/01/16	Sat 12/03/16
5	Testing and Rework	66 days	Mon 08/02/16	Wed 13/04/16
6	Documentation	15 days	Thu 14/04/16	Thu 28/04/16

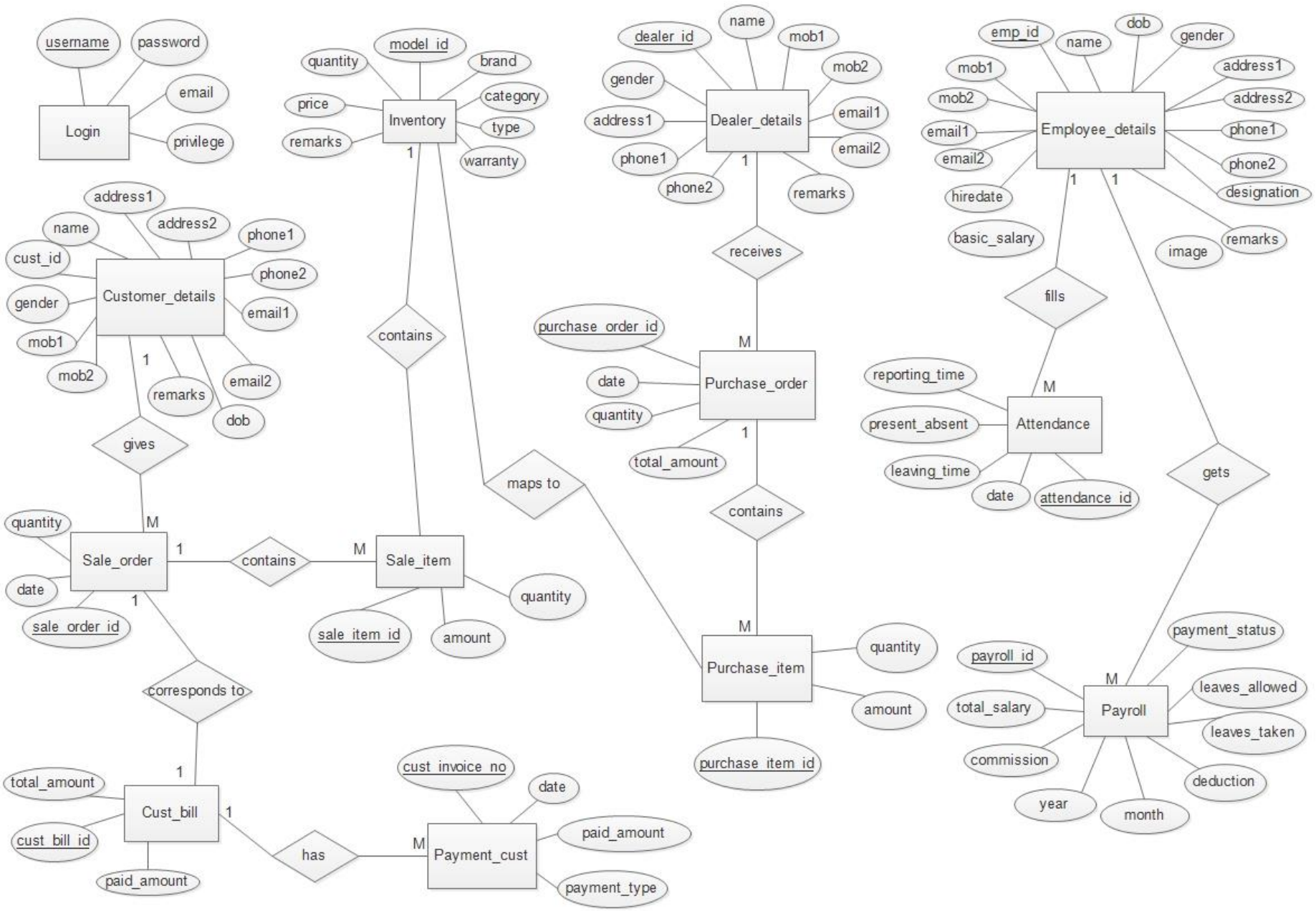
1.2) ER Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.
At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbols, and the meanings of those symbols, that make it unique.

Entities, which are represented by rectangles. An entity is an object or concept about which you want to store information.

Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

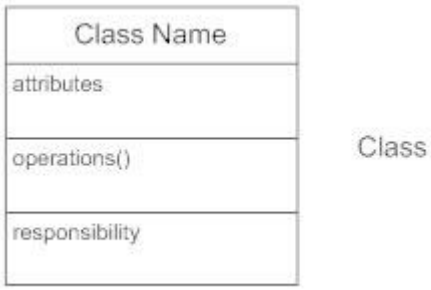
Actions, which are represented by diamond shapes, show how two entities share information in the database.


username
password
email
privilege
Login
model_id
quantity
brand
price
category
Inventory
type
remarks
warranty
dealer_id
name
mob1
mob2
gender
email1
address1
Dealer_details
email2
phone1
phone2
remarks
emp_id
name
dob
gender
mob1
address1
mob2
address2
email1
Employee_details
phone1
email2
phone2
hiredate
designation
basic_salary
image
remarks
address1
name
address2
phone1
cust_id
phone2
Customer_details
gender
email1
mob1
mob2
remarks
email2
dob
receives
fills
contains
gives
maps to
purchase_order_id
date
quantity
Purchase_order
total_amount
reporting_time
present_absent
Attendance
leaving_time
date
attendance_id
gets
quantity
Sale_order
contains
Sale_item
date
sale_order_id
sale_item_id
amount
quantity
contains
corresponds to
Purchase_item
quantity
amount
purchase_item_id
payment_status
payroll_id
leaves_allowed
total_salary
Payroll
leaves_taken
commission
deduction
year
month
total_amount
Cust_bill
cust_bill_id
paid_amount
has
cust_invoice_no
date
Payment_cust
paid_amount
payment_type
1
M

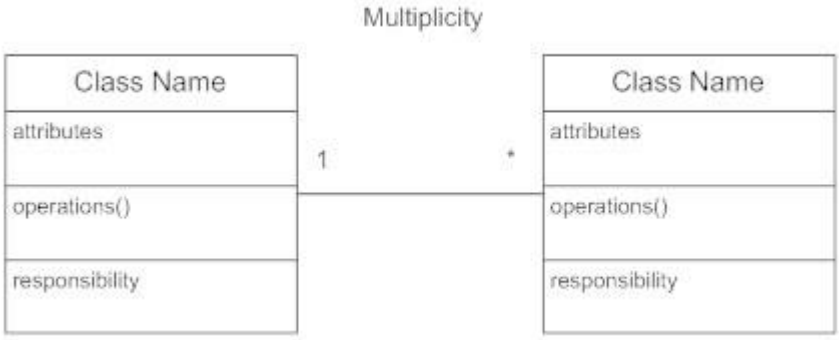
1.3) Class Diagram

A Class diagram models the static structure of a system. It shows relationships between classes, objects, attributes, and operations.

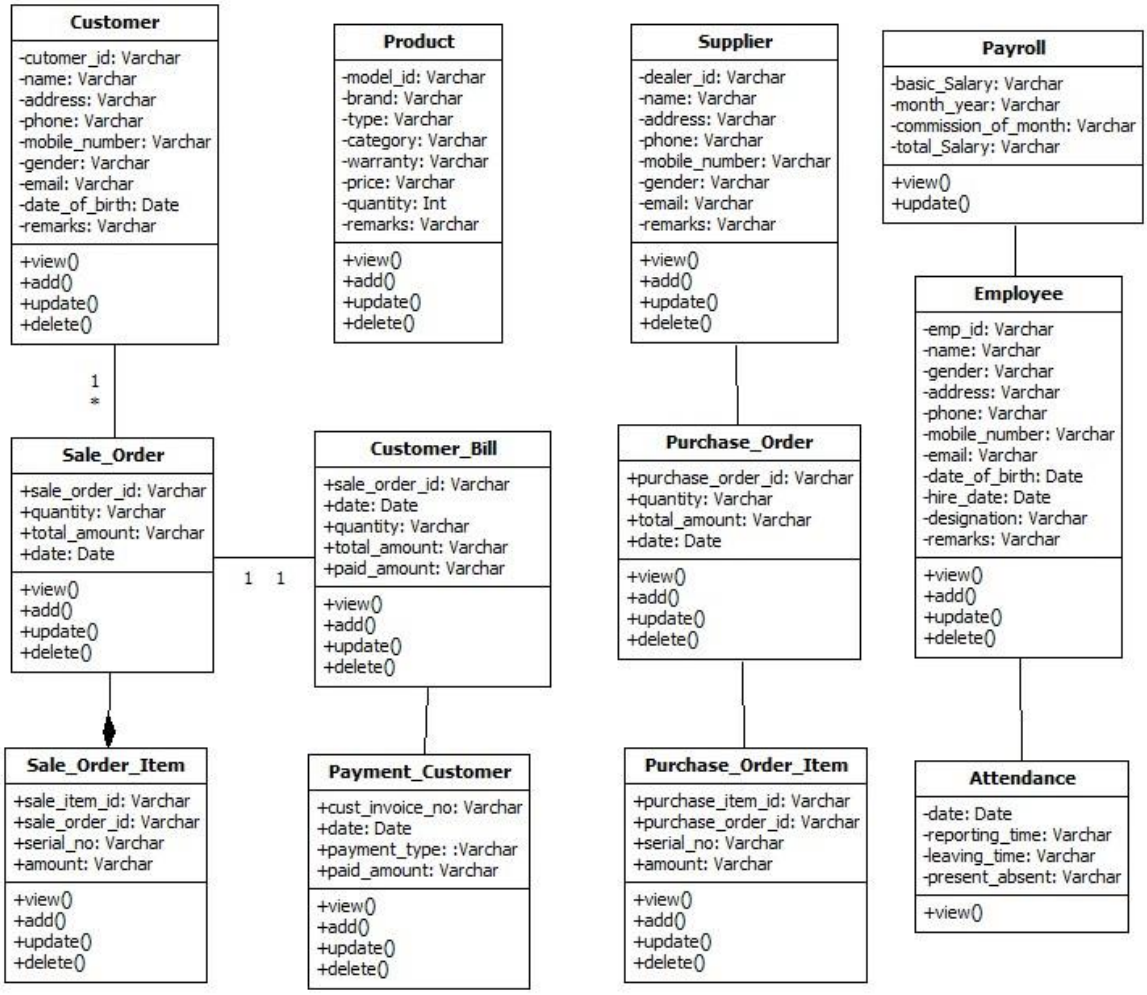
Classes
Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.



Multiplicity (Cardinality)
Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for just one company.



Indicator	Meaning
0..1	Zero or one
1	One only
0..*	0 or more
1..* *	1 or more
n	Only n (where n > 1)
0..n	Zero to n (where n >1)
1..n	One to n (where n > 1)


Customer
-cutomer_id: Varchar
-name: Varchar
-address: Varchar
-phone: Varchar
-mobile_number: Varchar
-gender: Varchar
-email: Varchar
-date_of_birth: Date
-remarks: Varchar
+view()
+add()
+update()
+delete()
Product
-model_id: Varchar
-brand: Varchar
-type: Varchar
-category: Varchar
-warranty: Varchar
-price: Varchar
-quantity: Int
-remarks: Varchar
+view()
+add()
+update()
+delete()
Supplier
-dealer_id: Varchar
-name: Varchar
-address: Varchar
-phone: Varchar
-mobile_number: Varchar
-gender: Varchar
-email: Varchar
-remarks: Varchar
+view()
+add()
+update()
+delete()
Payroll
-basic_Salary: Varchar
-month_year: Varchar
-commission_of_month: Varchar
-total_Salary: Varchar
+view()
+update()
1
*
Employee
-emp_id: Varchar
-name: Varchar
-gender: Varchar
-address: Varchar
-phone: Varchar
-mobile_number: Varchar
-email: Varchar
-date_of_birth: Date
-hire_date: Date
-designation: Varchar
-remarks: Varchar
+view()
+add()
+update()
+delete()
Sale_Order
+sale_order_id: Varchar
+quantity: Varchar
+total_amount: Varchar
+date: Date
+view()
+add()
+update()
+delete()
1 1
Customer_Bill
+sale_order_id: Varchar
+date: Date
+quantity: Varchar
+total_amount: Varchar
+paid_amount: Varchar
+view()
+add()
+update()
+delete()
Purchase_Order
+purchase_order_id: Varchar
+quantity: Varchar
+total_amount: Varchar
+date: Date
+view()
+add()
+update()
+delete()
Sale_Order_Item
+sale_item_id: Varchar
+sale_order_id: Varchar
+serial_no: Varchar
+amount: Varchar
+view()
+add()
+update()
+delete()
Payment_Customer
+cust_invoice_no: Varchar
+date: Date
+payment_type: :Varchar
+paid_amount: Varchar
+view()
+add()
+update()
+delete()
Purchase_Order_Item
+purchase_item_id: Varchar
+purchase_order_id: Varchar
+serial_no: Varchar
+amount: Varchar
+view()
+add()
+update()
+delete()
Attendance
-date: Date
-reporting_time: Varchar
-leaving_time: Varchar
-present_absent: Varchar
+view()

1.4) Use Case Diagram

A use case diagram is a dynamic or behaviour diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.

System
Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

System

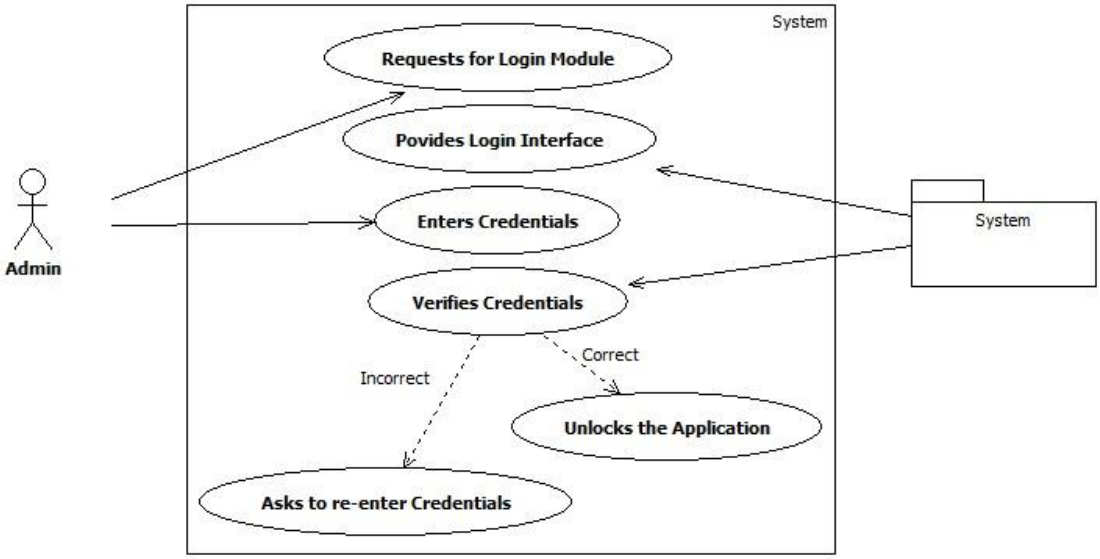
Use Case
Draw use cases using ovals. Label the ovals with verbs that represent the system's functions.

Use case

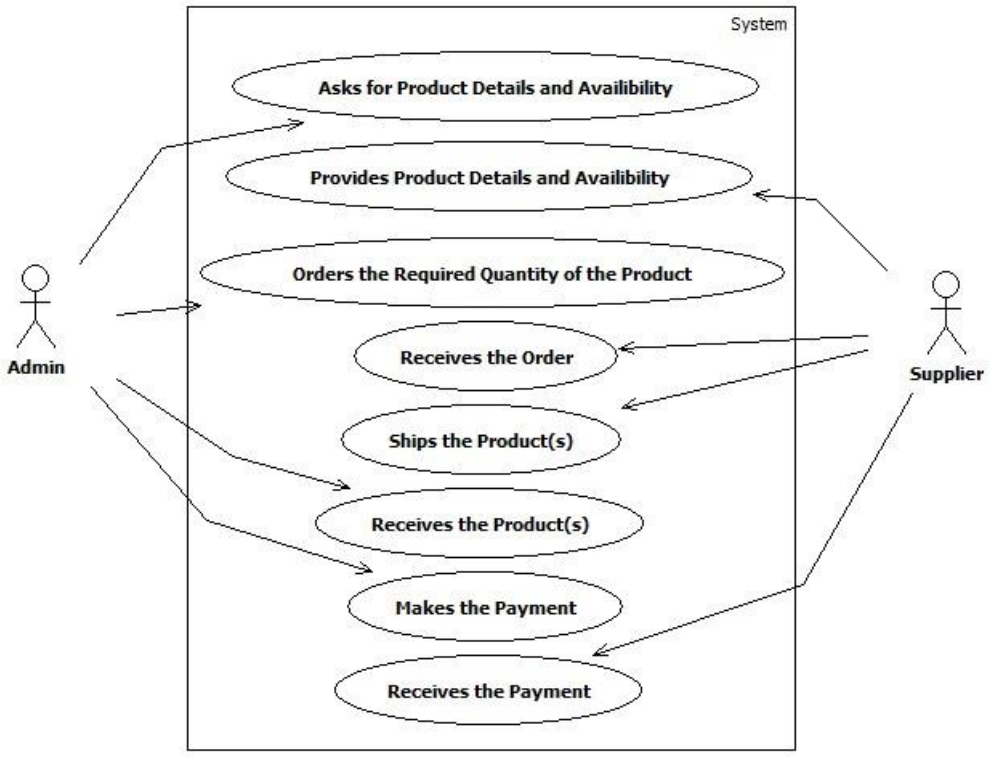
Actors
Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.

Actor

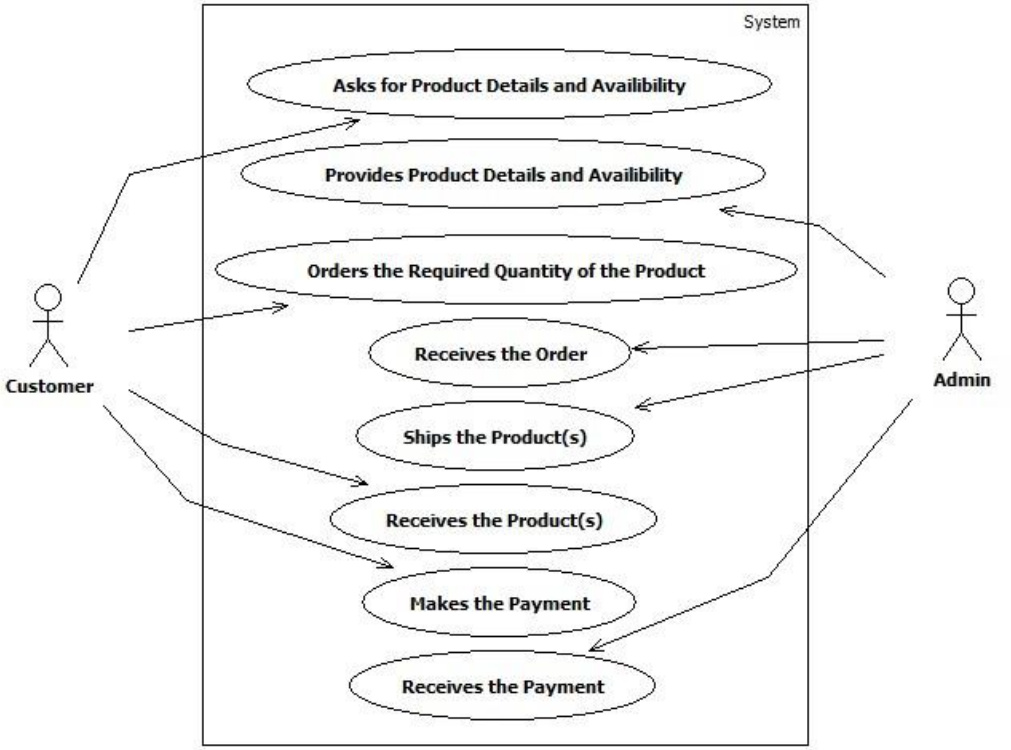
Login



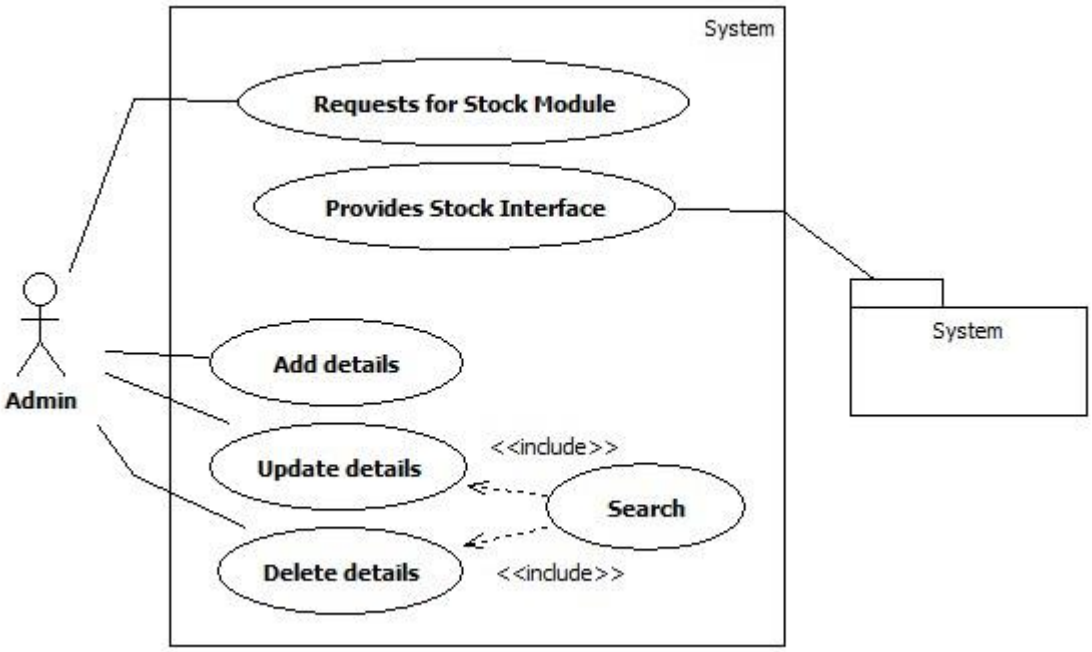
Admin - Supplier - Billing



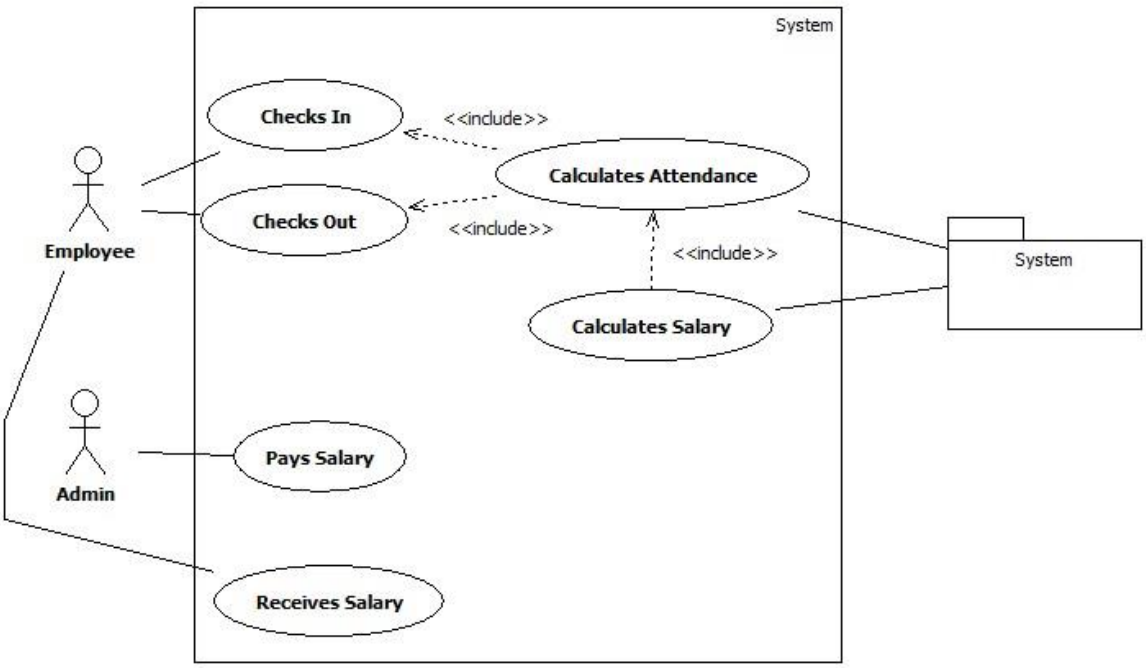
Admin - Customer - Billing



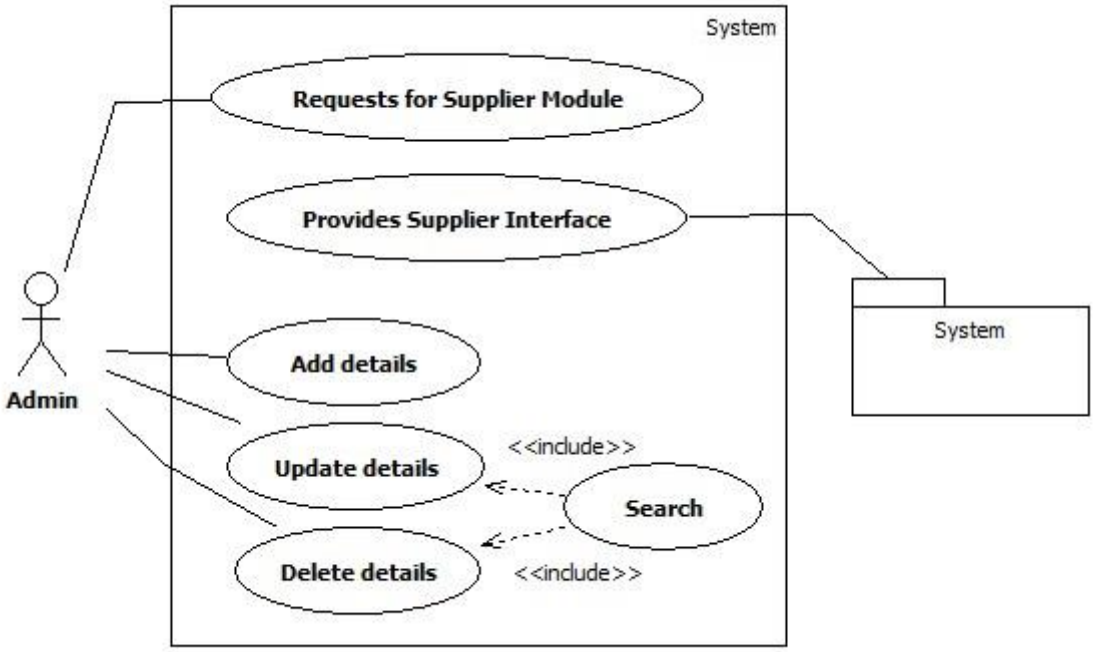
Stock Management



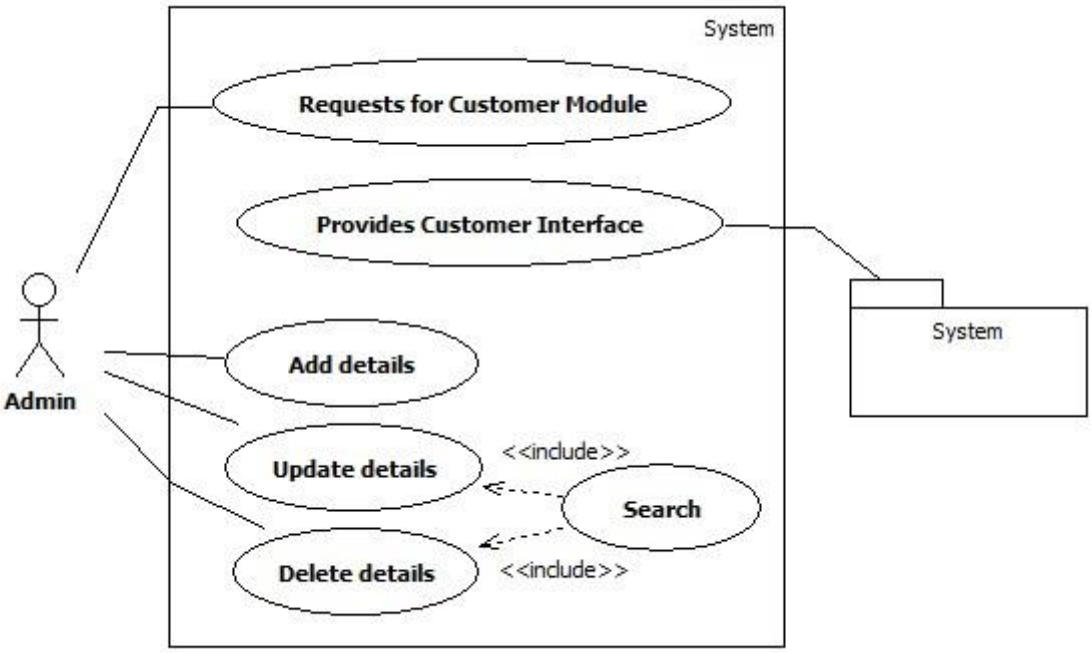
Payroll Management



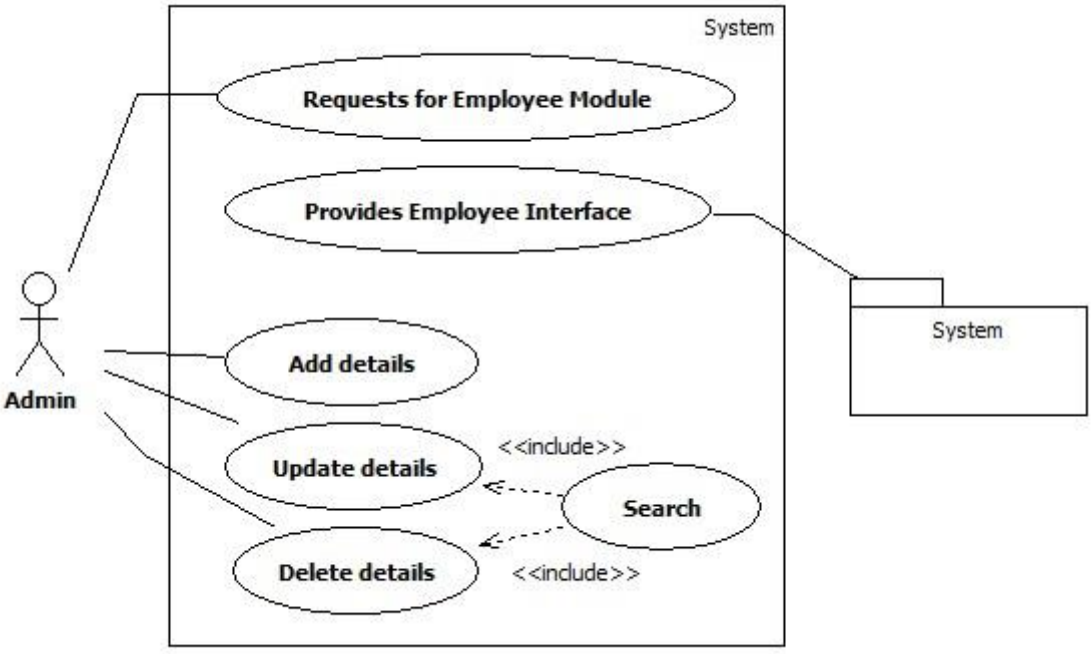
Supplier Management



Customer Management



Employee Management



1.5) Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

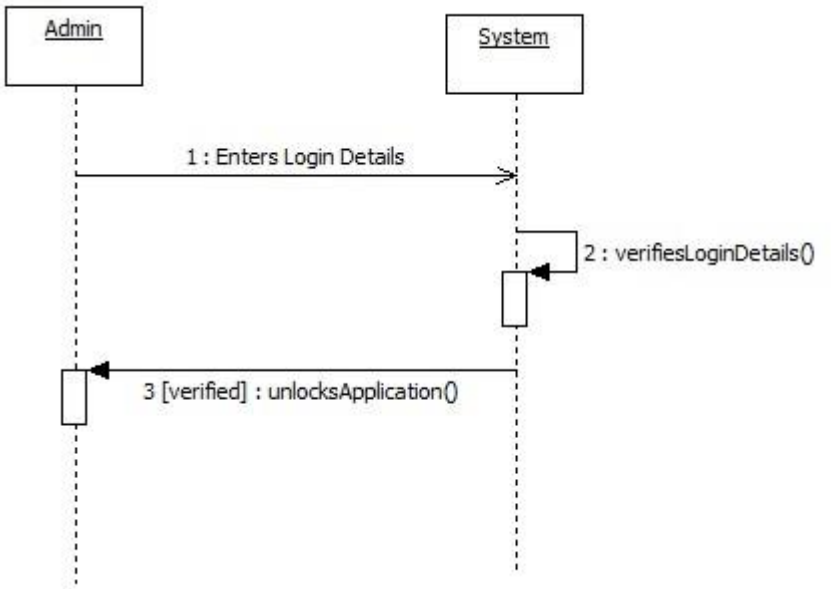
Class Roles or Participants
Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

component

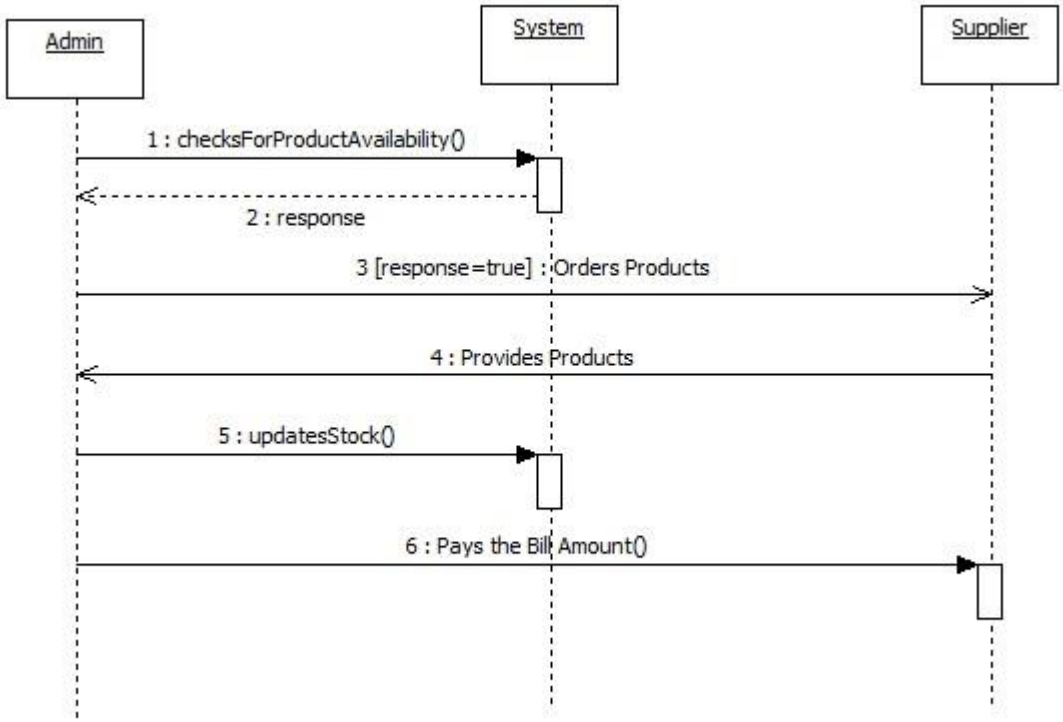
Activation or Execution Occurrence
Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.

Activation or Execution Occurrence

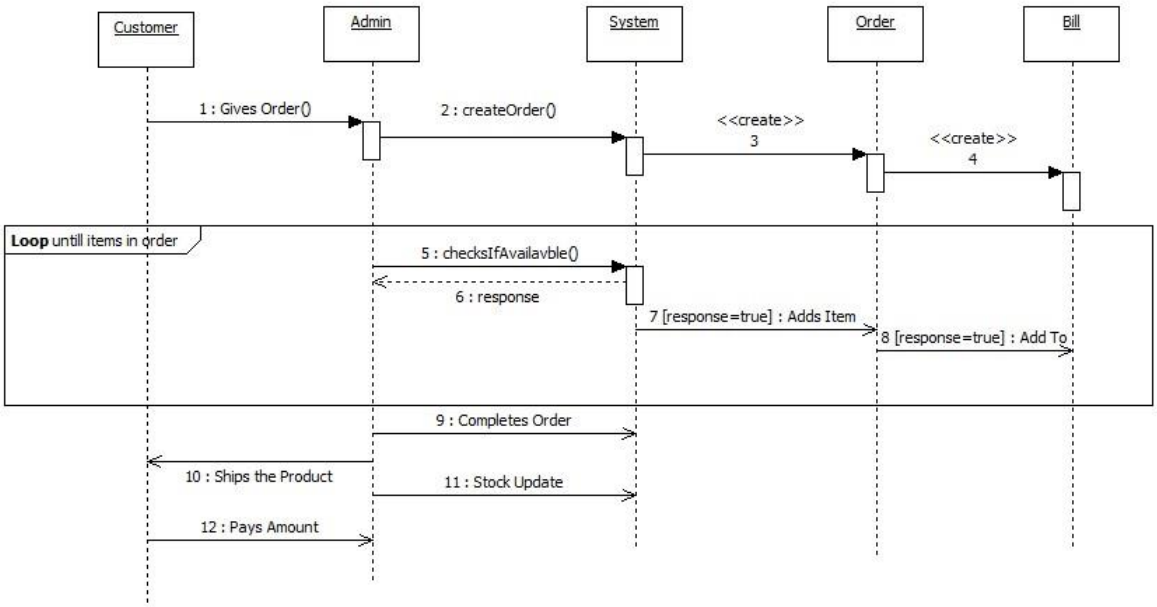
Login



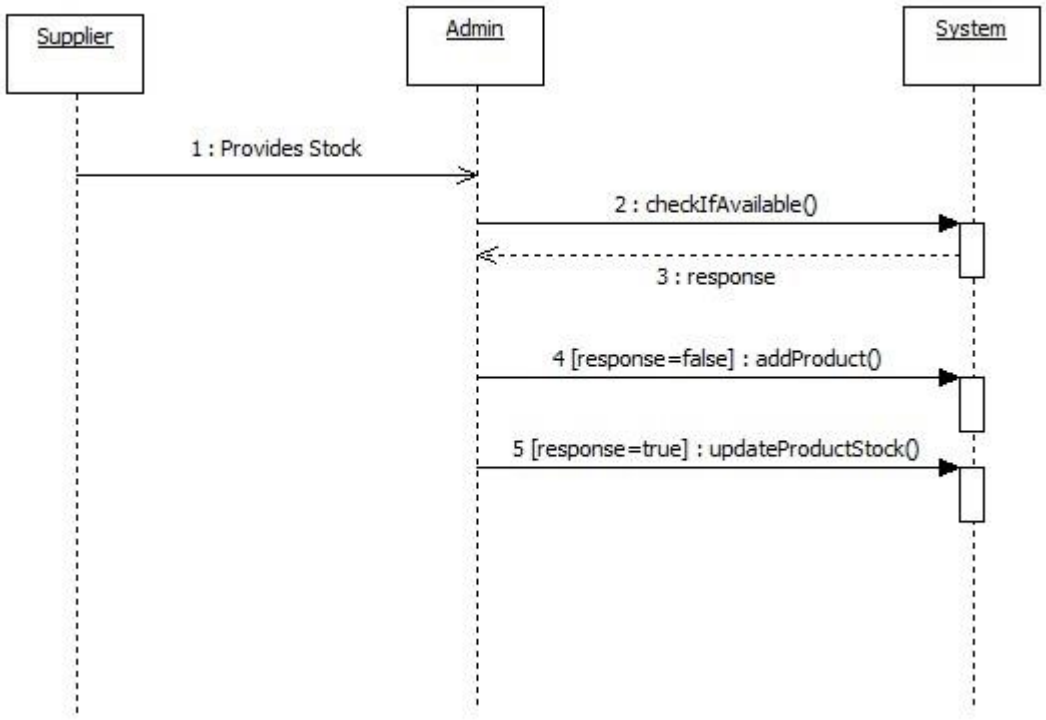
Admin - Supplier - Billing



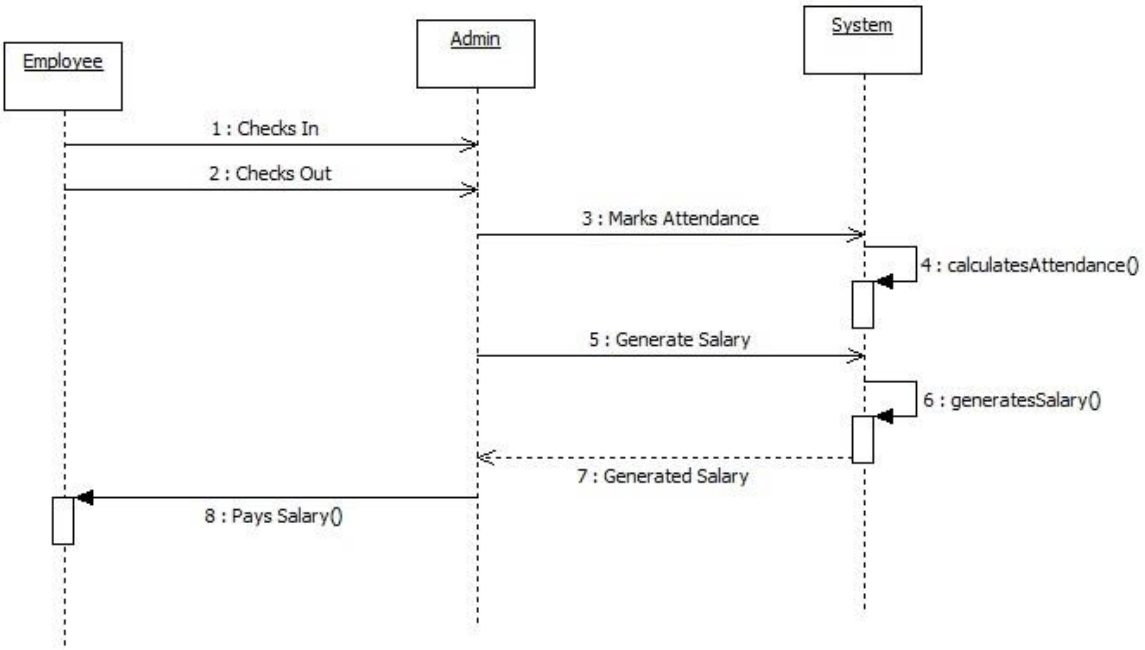
Admin - Customer - Billing



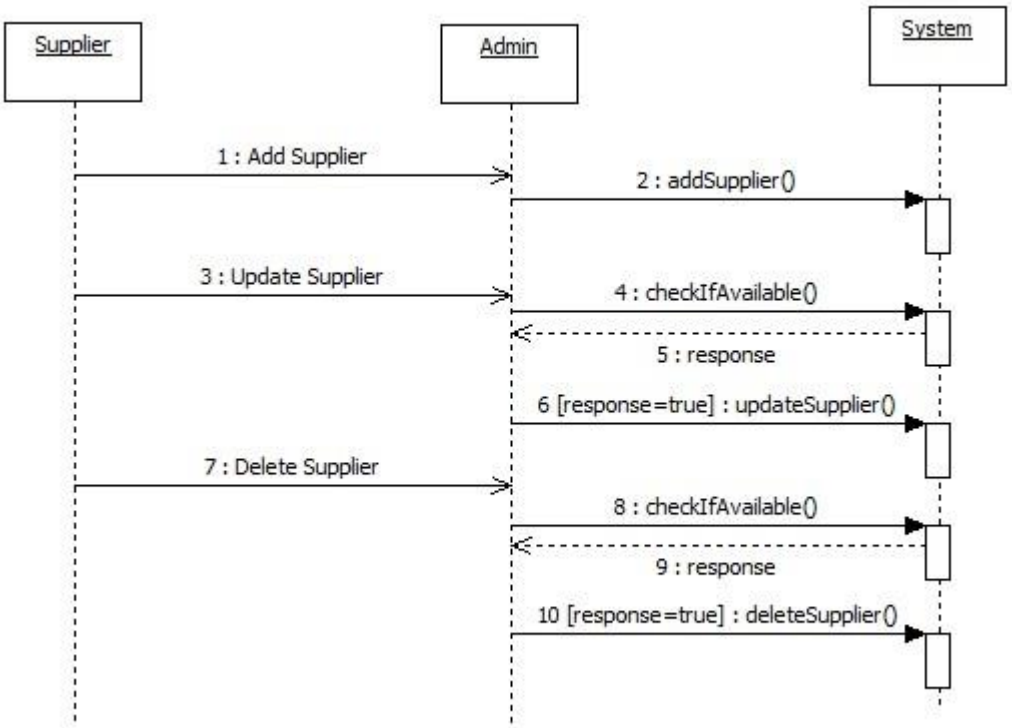
Stock Management



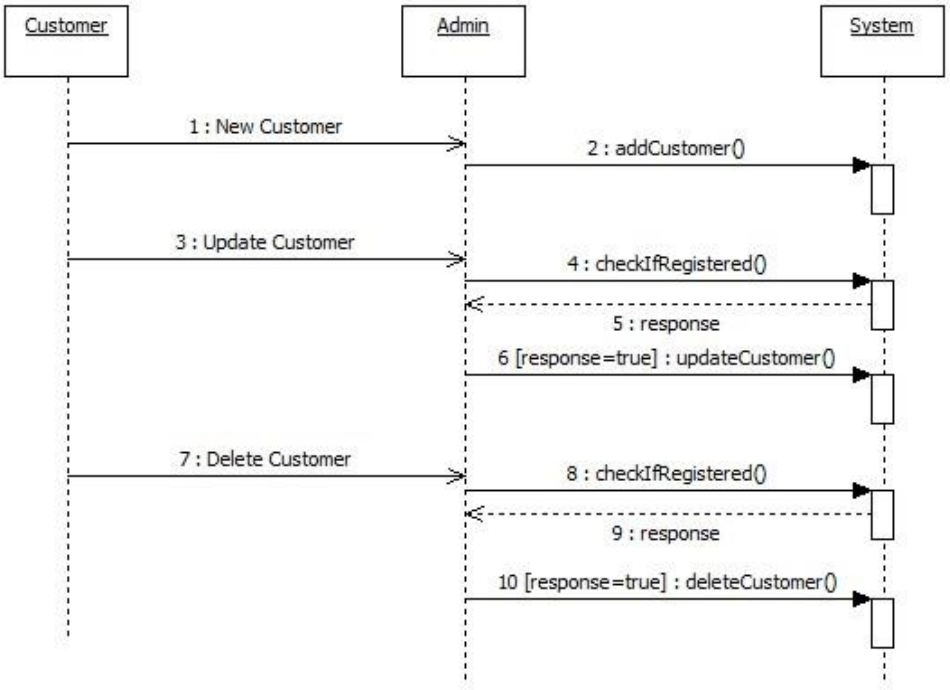
Payroll Management



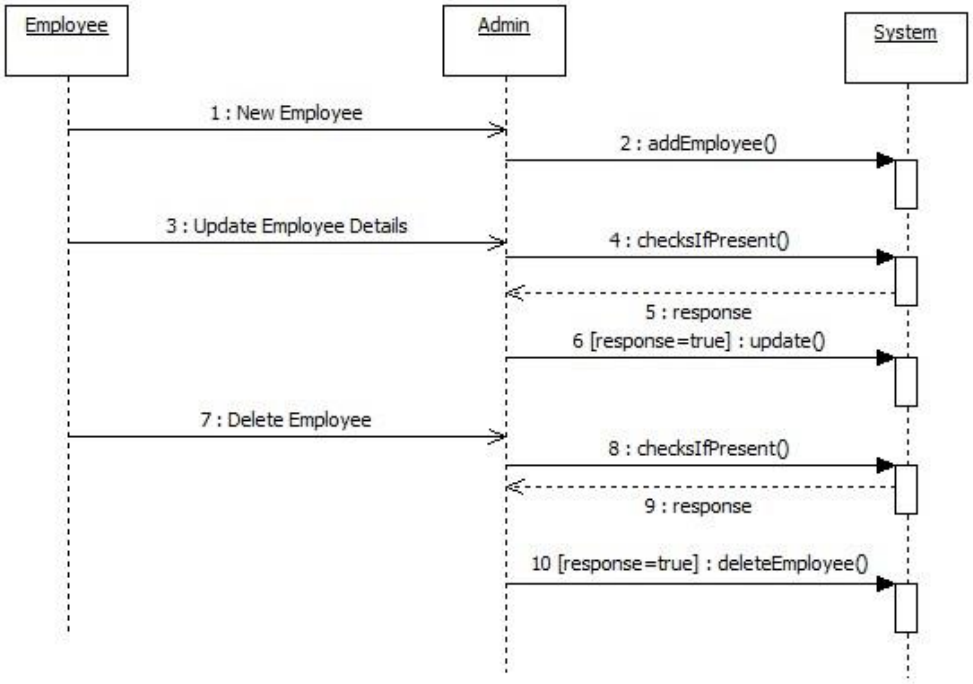
Supplier Management



Customer Management



Employee Management



1.6) Activity Diagram

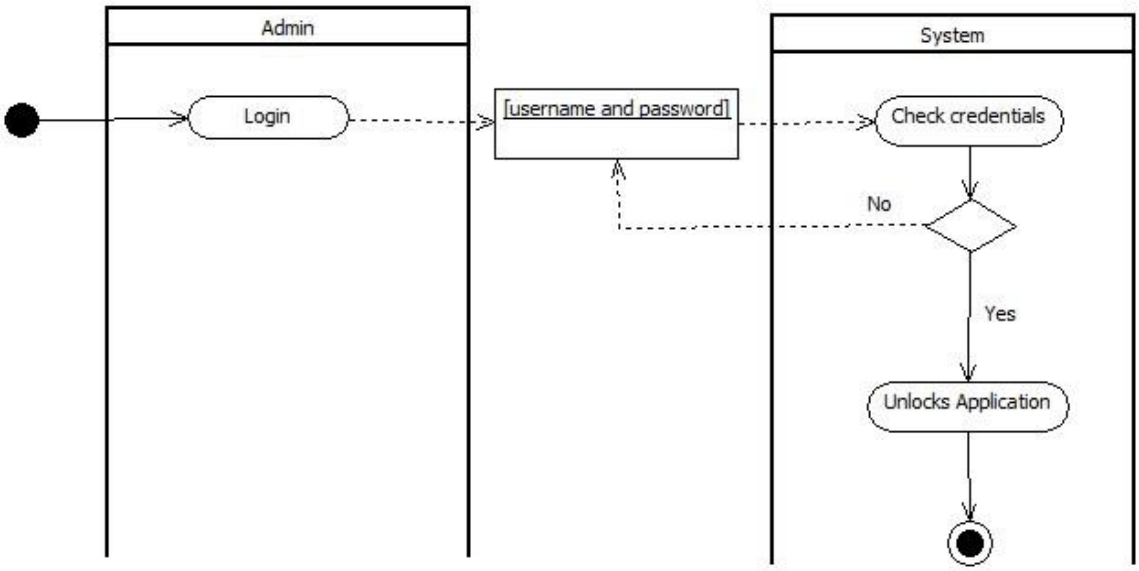
An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modelling. They can also describe the steps in a use case diagram. Activities modelled can be sequential and concurrent. In both cases an activity diagram will have a beginning and an end.

Initial State or Start Point
A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram. For activity diagram using swimlanes, make sure the start point is placed in the top left corner of the first column.

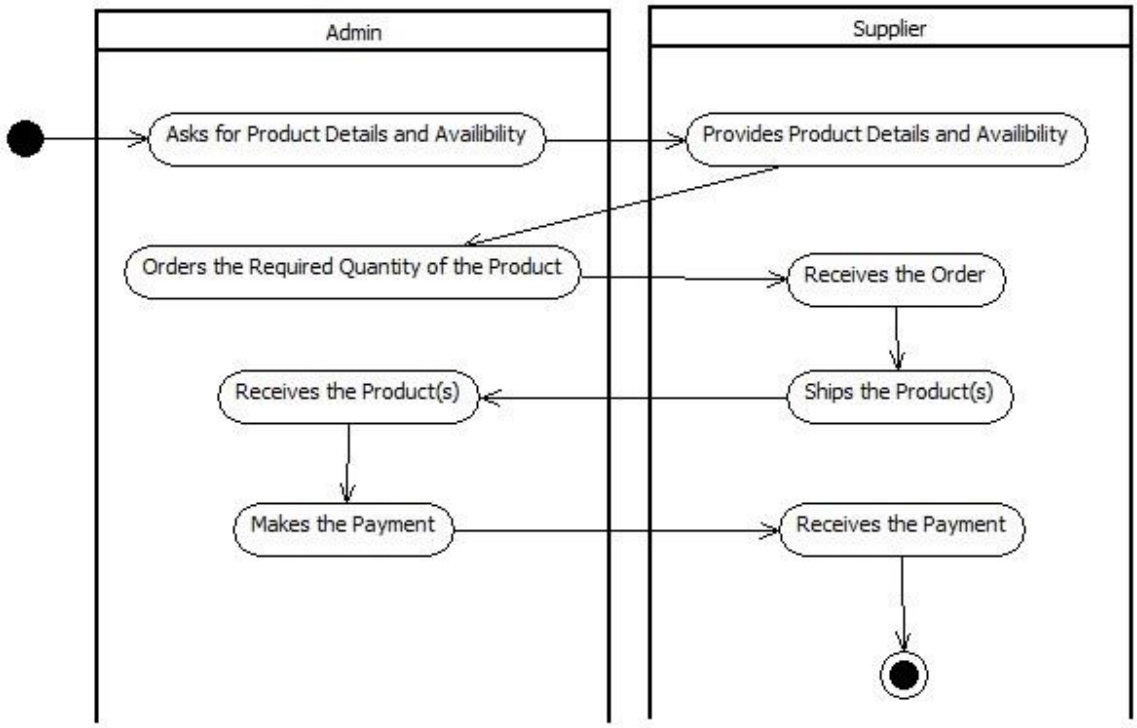
Activity or Action State
An action state represents the non-interruptible action of objects.

Decisions and Branching
A diamond represents a decision with alternate paths. When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities. The outgoing alternates should be labelled with a condition or guard expression. You can also label one of the paths "else."

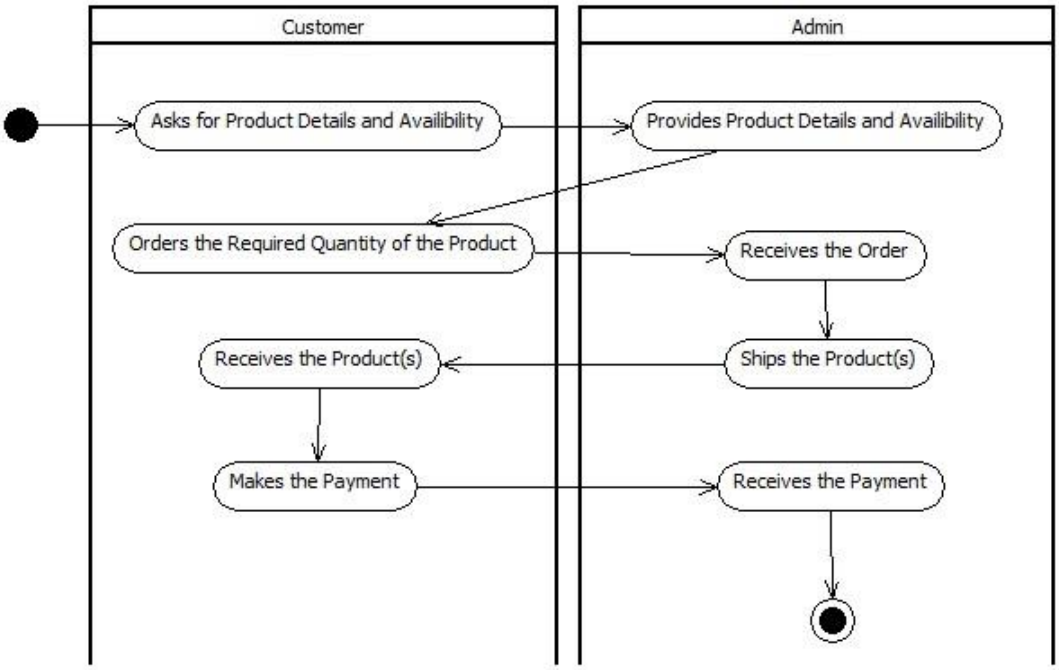
Login



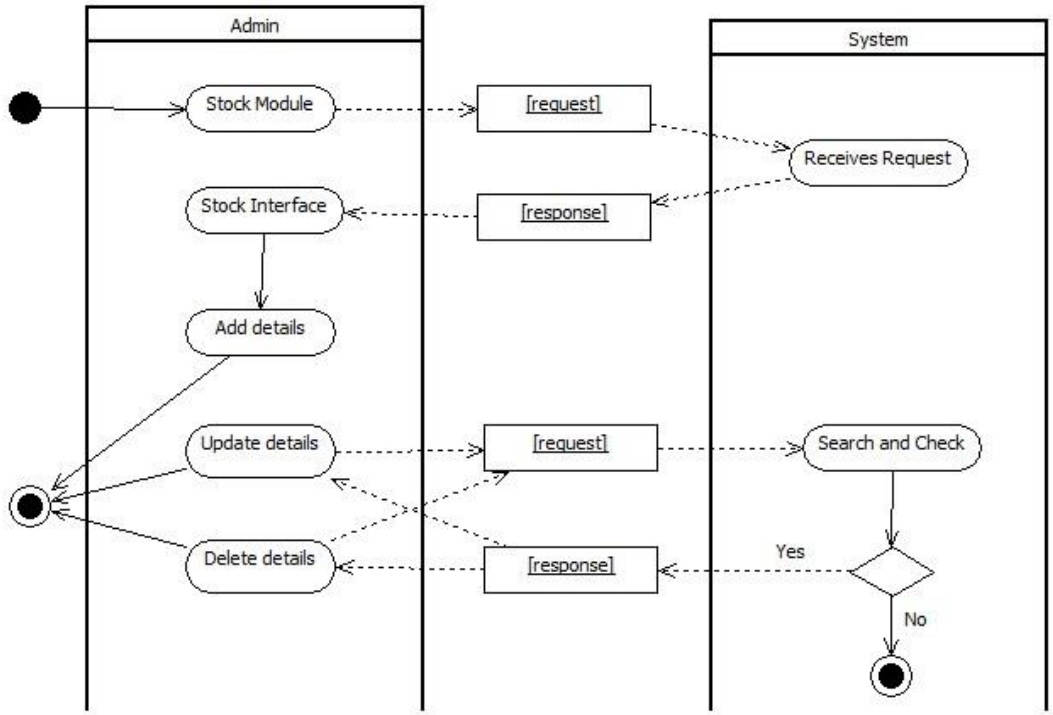
Admin - Supplier - Billing



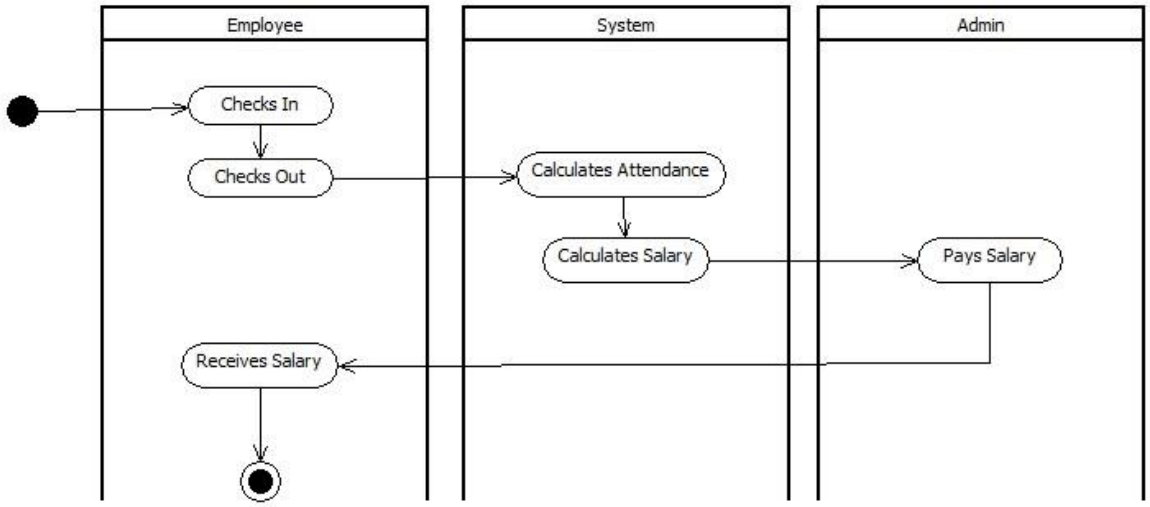
Admin - Customer - Billing



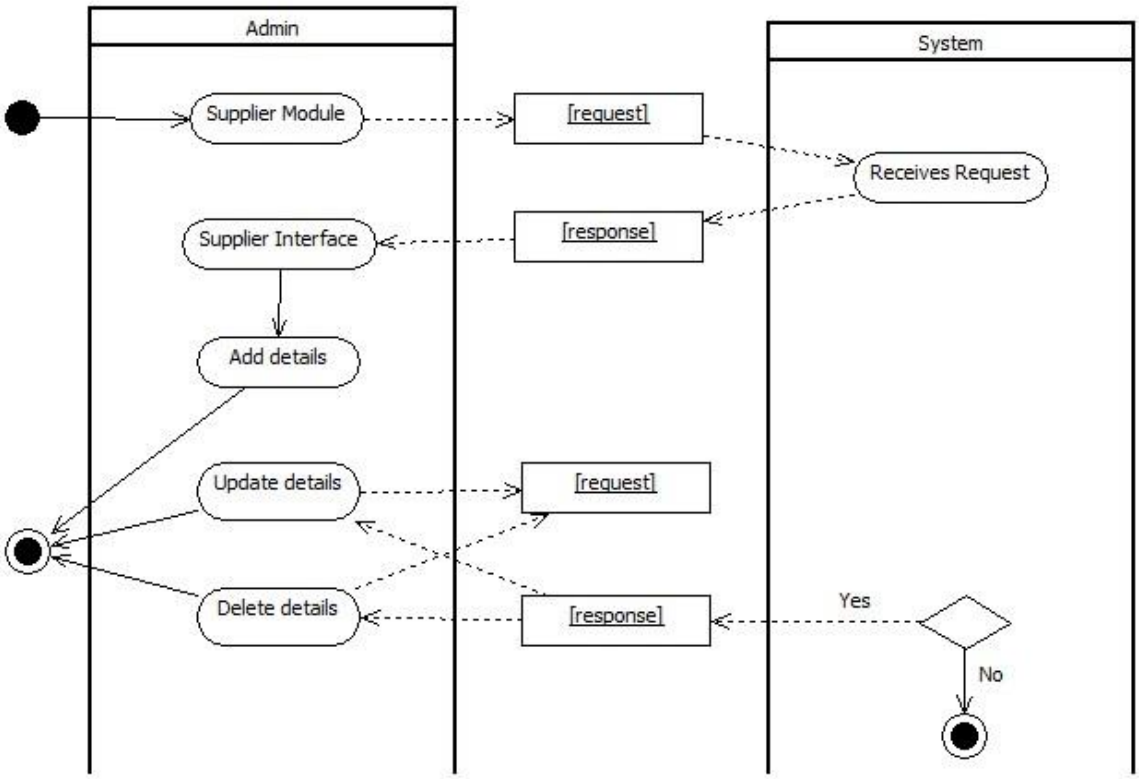
Stock Management



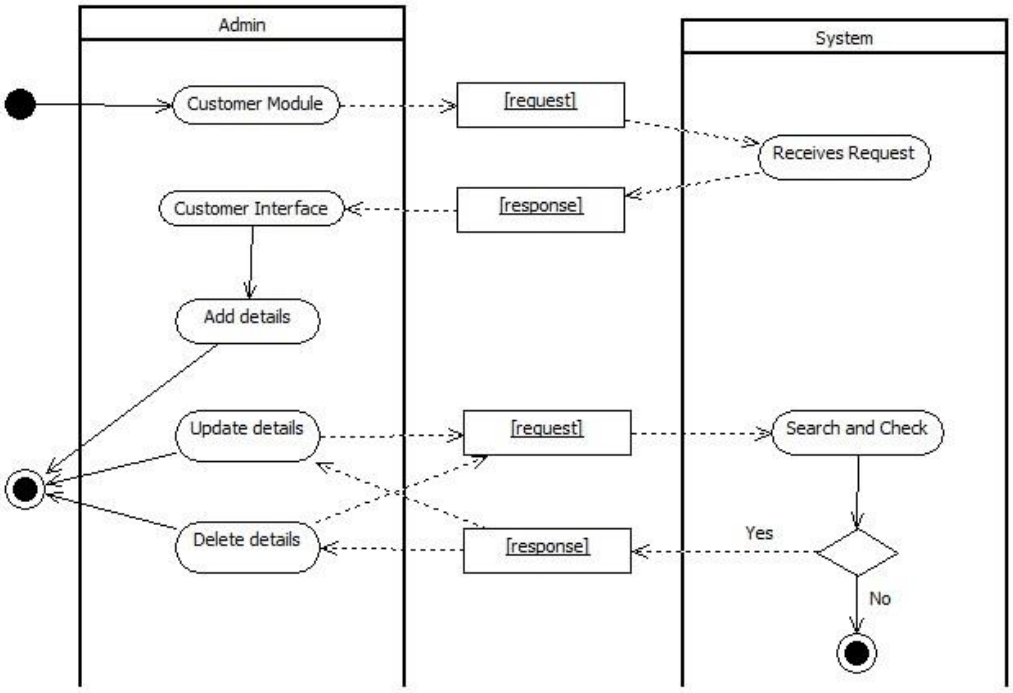
Payroll Management



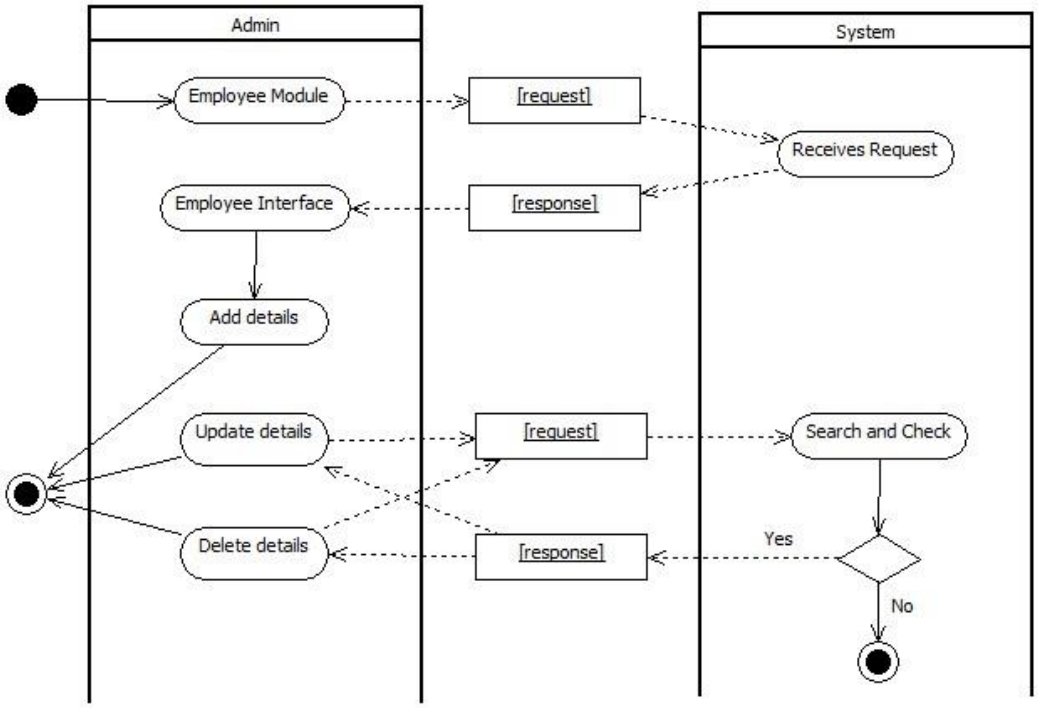
Supplier Management



Customer Management



Employee Management



2) Tables

Attendance

Column Name	**Data Type**	**Allow Nulls**
attendance_id (PRIMARY KEY)	VARCHAR(30)	No
emp_id	VARCHAR(30)	Yes
date	DATE	Yes
reporting_time	VARCHAR(10)	Yes
present_absent	VARCHAR(10)	Yes
leaving_time	VARCHAR(10)	Yes

Customer_Bill

Column Name	Data Type	Allow Nulls
cust_bill_id (PRIMARY KEY)	VARCHAR(30)	No
sale_order_id	VARCHAR(30)	Yes
total_amount	VARCHAR(20)	Yes
paid_amount	VARCHAR(20)	Yes

Customer_Details

Column Name	Data Type	Allow Nulls
cust_id (PRIMARY KEY)	VARCHAR(30)	No
name	VARCHAR(45)	Yes
address1	VARCHAR(100)	Yes
address2	VARCHAR(100)	Yes
phone1	VARCHAR(20)	Yes
phone2	VARCHAR(20)	Yes
mob1	VARCHAR(20)	Yes
mob2	VARCHAR(20)	Yes
gender	VARCHAR(1)	Yes
email1	VARCHAR(80)	Yes
email2	VARCHAR(80)	Yes
dob	DATE	Yes
remarks	VARCHAR(100)	Yes

Dealer_Details

Column Name	Data Type	Allow Nulls
dealer_id (PRIMARY KEY)	VARCHAR(30)	No
name	VARCHAR(45)	Yes
address1	VARCHAR(100)	Yes
phone1	VARCHAR(20)	Yes
phone2	VARCHAR(20)	Yes
mob1	VARCHAR(20)	Yes
mob2	VARCHAR(20)	Yes
gender	VARCHAR(1)	Yes
email1	VARCHAR(80)	Yes
email2	VARCHAR(80)	Yes
remarks	VARCHAR(100)	Yes

Employee_Details

Column Name	Data Type	Allow Nulls
emp_id (PRIMARY KEY)	VARCHAR(30)	No
name	VARCHAR(45)	Yes
gender	VARCHAR(1)	Yes
address1	VARCHAR(100)	Yes
address2	VARCHAR(100)	Yes
phone1	VARCHAR(20)	Yes
phone2	VARCHAR(20)	Yes
mob1	VARCHAR(20)	Yes
mob2	VARCHAR(20)	Yes
dob	DATE	Yes
hiredate	DATE	Yes
designation	VARCHAR(45)	Yes
remarks	VARCHAR(100)	Yes
email1	VARCHAR(80)	Yes
email2	VARCHAR(80)	Yes
basic_salary	VARCHAR(45)	Yes
image	MEDIUMBLOB	Yes

Login

Column Name	Data Type	Allow Nulls
username (PRIMARY KEY)	VARCHAR(30)	No
password	VARCHAR(30)	Yes
email	VARCHAR(80)	Yes
privilege	VARCHAR(1)	Yes

Payment_Cust

Column Name	Data Type	Allow Nulls
cust_invoice_no (PRIMARY KEY)	VARCHAR(30)	No
date	DATE	Yes
cust_bill_id	VARCHAR(30)	Yes
payment_type	VARCHAR(20)	Yes
paid_amount	VARCHAR(20)	Yes

Payroll

Column Name	Data Type	Allow Nulls
payroll_id (PRIMARY KEY)	VARCHAR(30)	No
emp_id	VARCHAR(30)	Yes
year	VARCHAR(45)	Yes
month	VARCHAR(45)	Yes
leave_taken	VARCHAR(45)	Yes
leave_allowed	VARCHAR(45)	Yes
Commission	VARCHAR(45)	Yes
Deduction	VARCHAR(45)	Yes
Total_salary	VARCHAR(45)	Yes
Payment_Status	VARCHAR(45)	Yes

Purchase_Item

Column Name	Data Type	Allow Nulls
purchase_item_id (PRIMARY KEY)	VARCHAR(30)	No
purchase_order_id	VARCHAR(30)	Yes
amount	VARCHAR(20)	Yes
model_id	VARCHAR(30)	Yes
quantity	INT(6)	Yes

Purchase_Order

Column Name	Data Type	Allow Nulls
purchase_order_id (PRIMARY KEY)	VARCHAR(30)	No
dealer_id	VARCHAR(30)	Yes
quantity	VARCHAR(20)	Yes
total_amount	VARCHAR(20)	Yes
date	DATE	Yes

Sale_Item

Column Name	Data Type	Allow Nulls
sale_item_id (PRIMARY KEY)	VARCHAR(30)	No
sale_order_id	VARCHAR(30)	Yes
amount	VARCHAR(20)	Yes
model_id	VARCHAR(30)	Yes
quantity	INT(6)	Yes

Sale_Order

Column Name	Data Type	Allow Nulls
sale_order_id (PRIMARY KEY)	VARCHAR(30)	No
cust_id	VARCHAR(30)	Yes
quantity	VARCHAR(20)	Yes
date	DATE	Yes

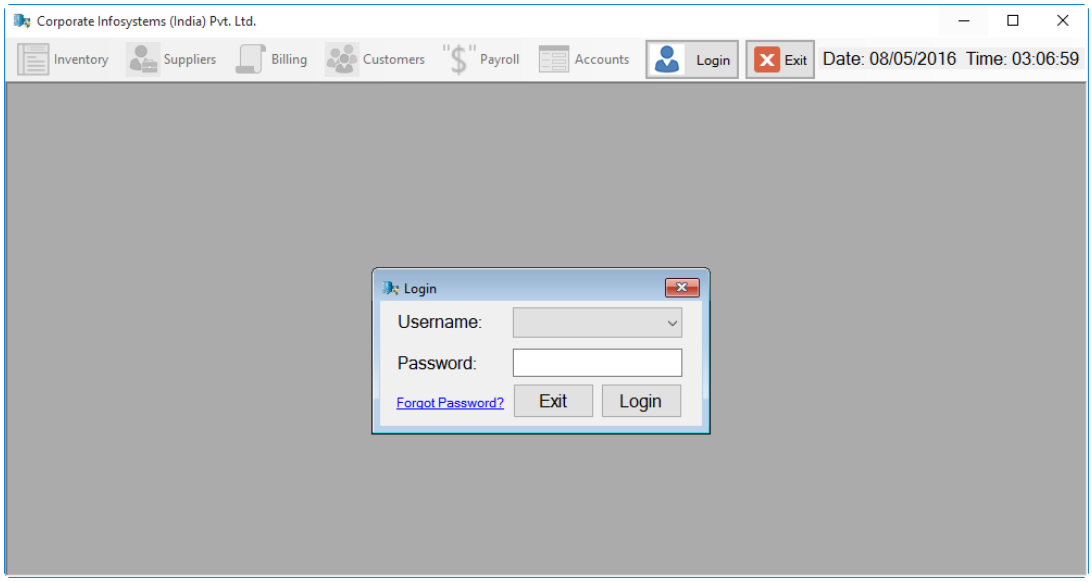
Inventory

Column Name	Data Type	Allow Nulls
model_id (PRIMARY KEY)	VARCHAR(30)	No
brand	VARCHAR(45)	Yes
type	VARCHAR(45)	Yes
category	VARCHAR(45)	Yes
warranty	VARCHAR(20)	Yes
price	VARCHAR(20)	Yes
quantity	INT(6)	Yes
remarks	VARCHAR(100)	Yes

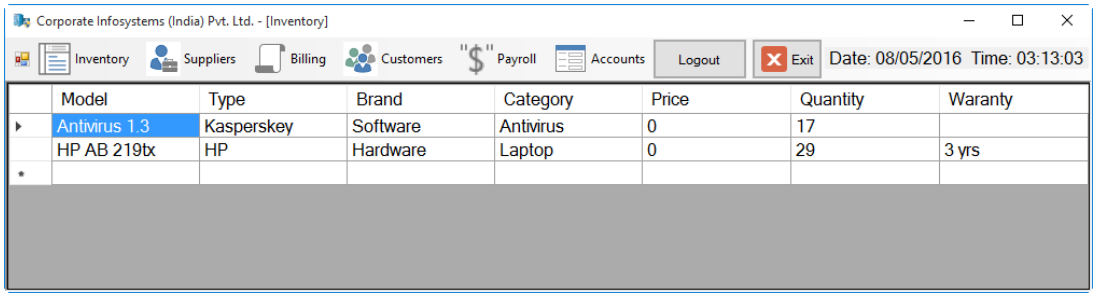
III Implementation Phase

1.1) Screen Layouts

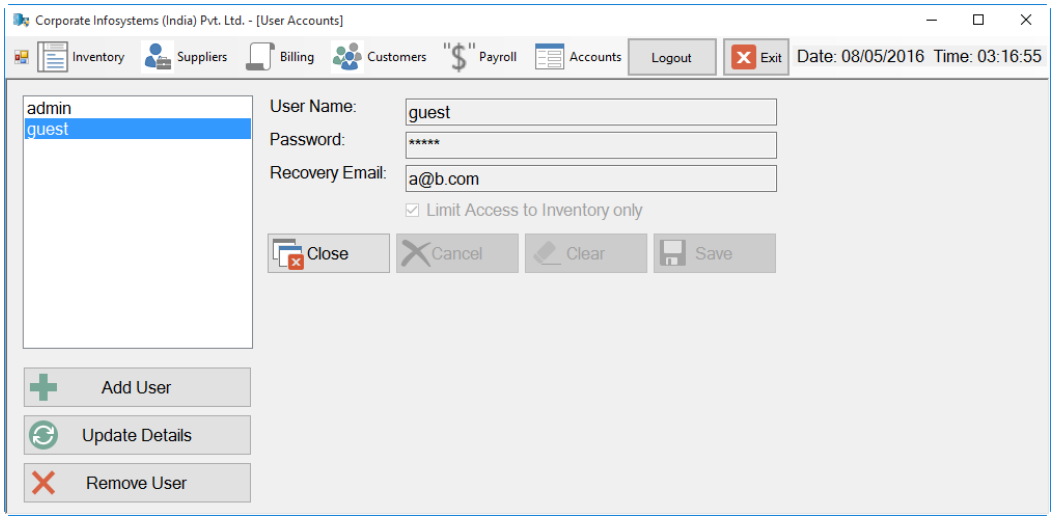
Login



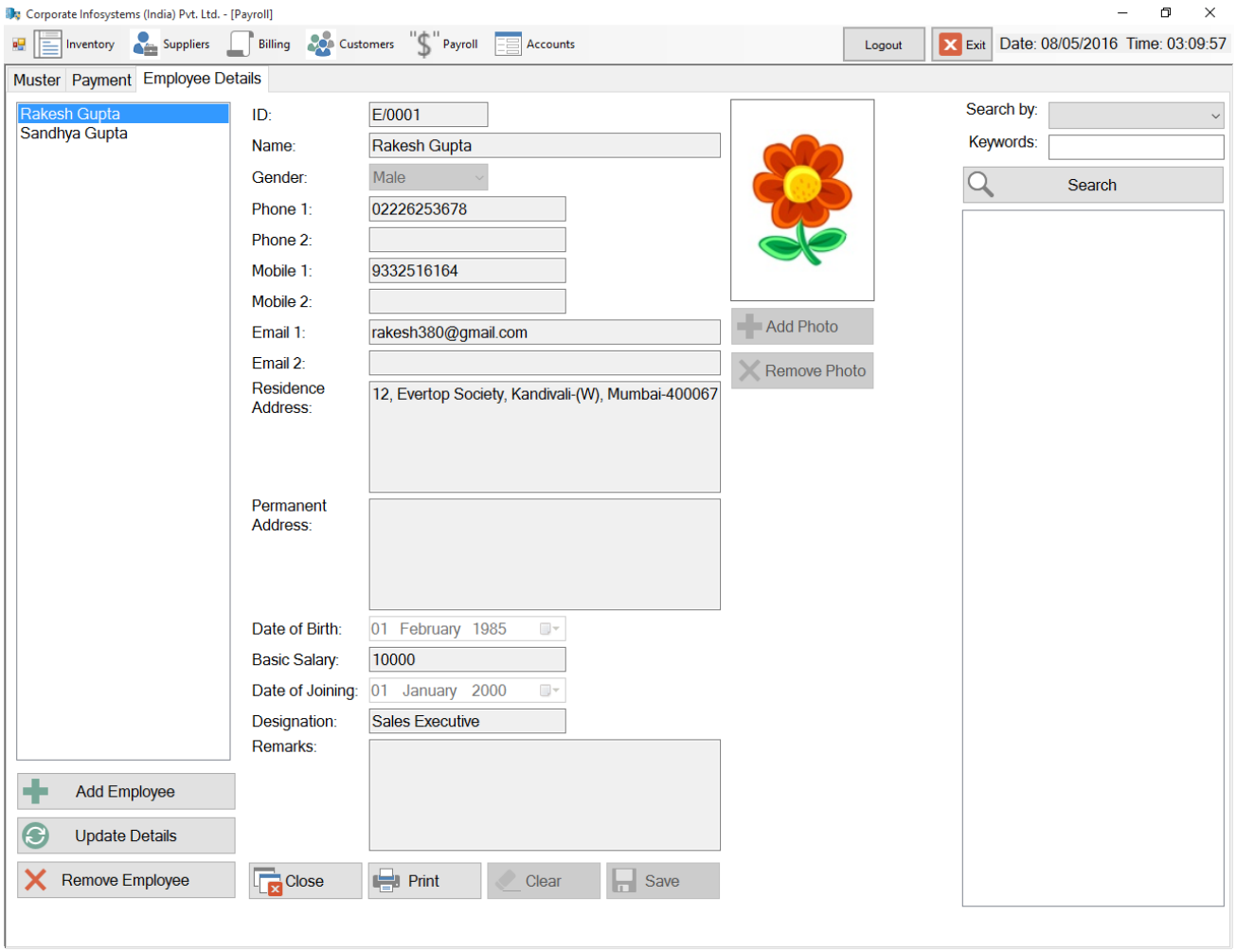
Inventory



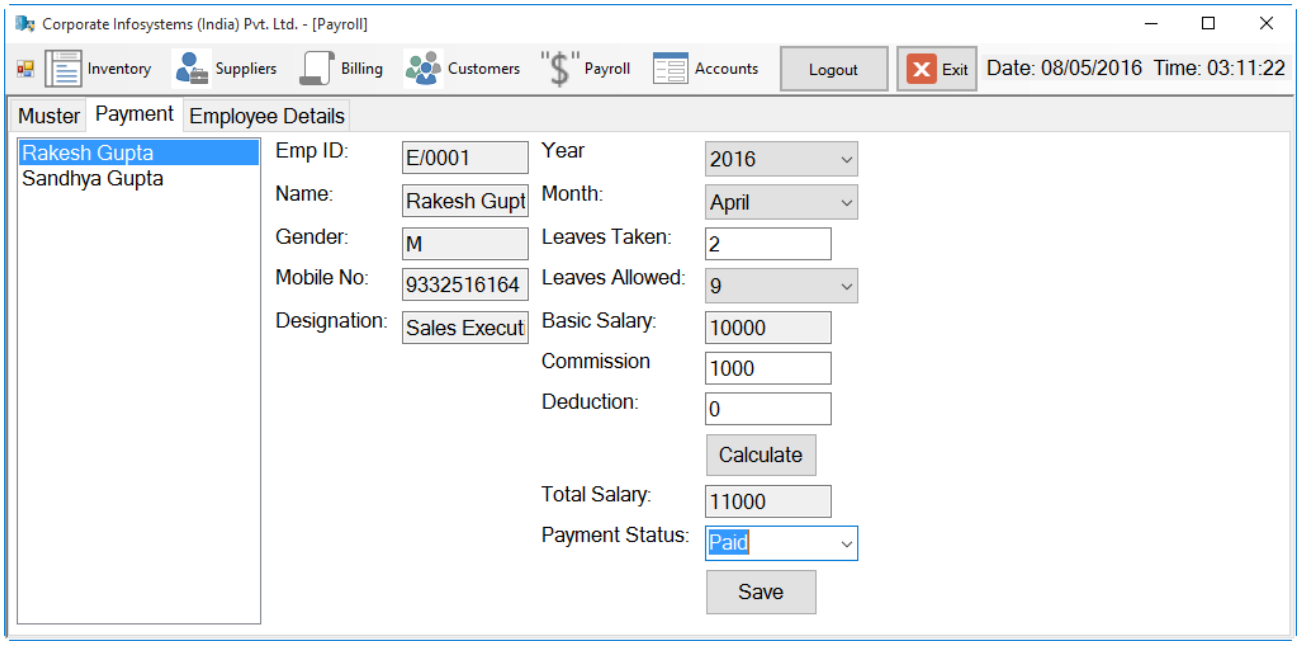
Accounts



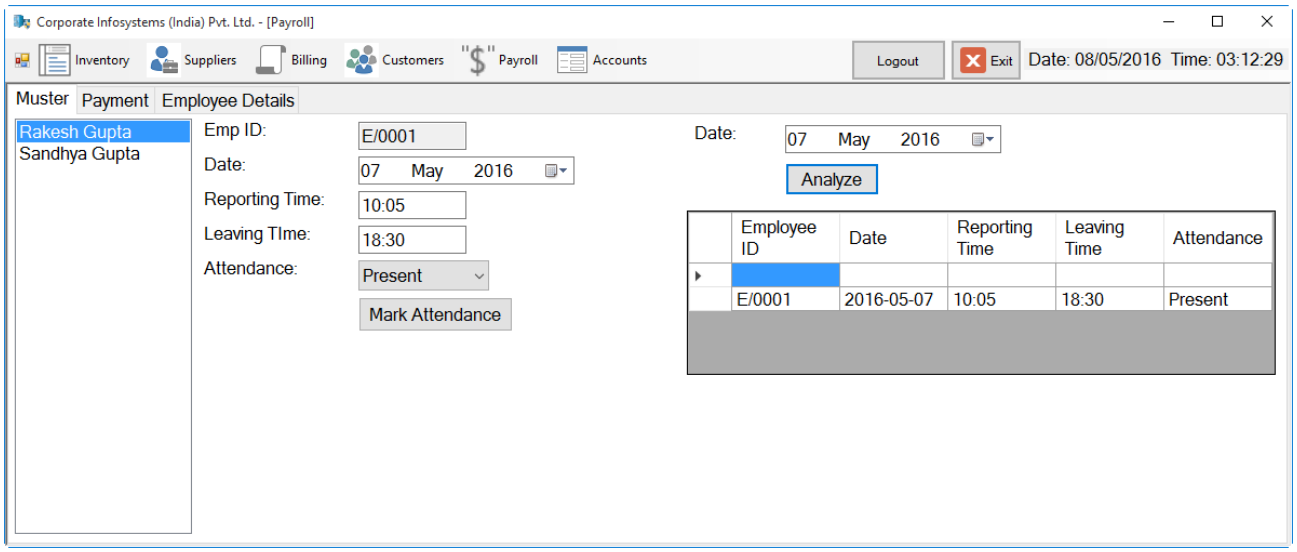
Employee Details



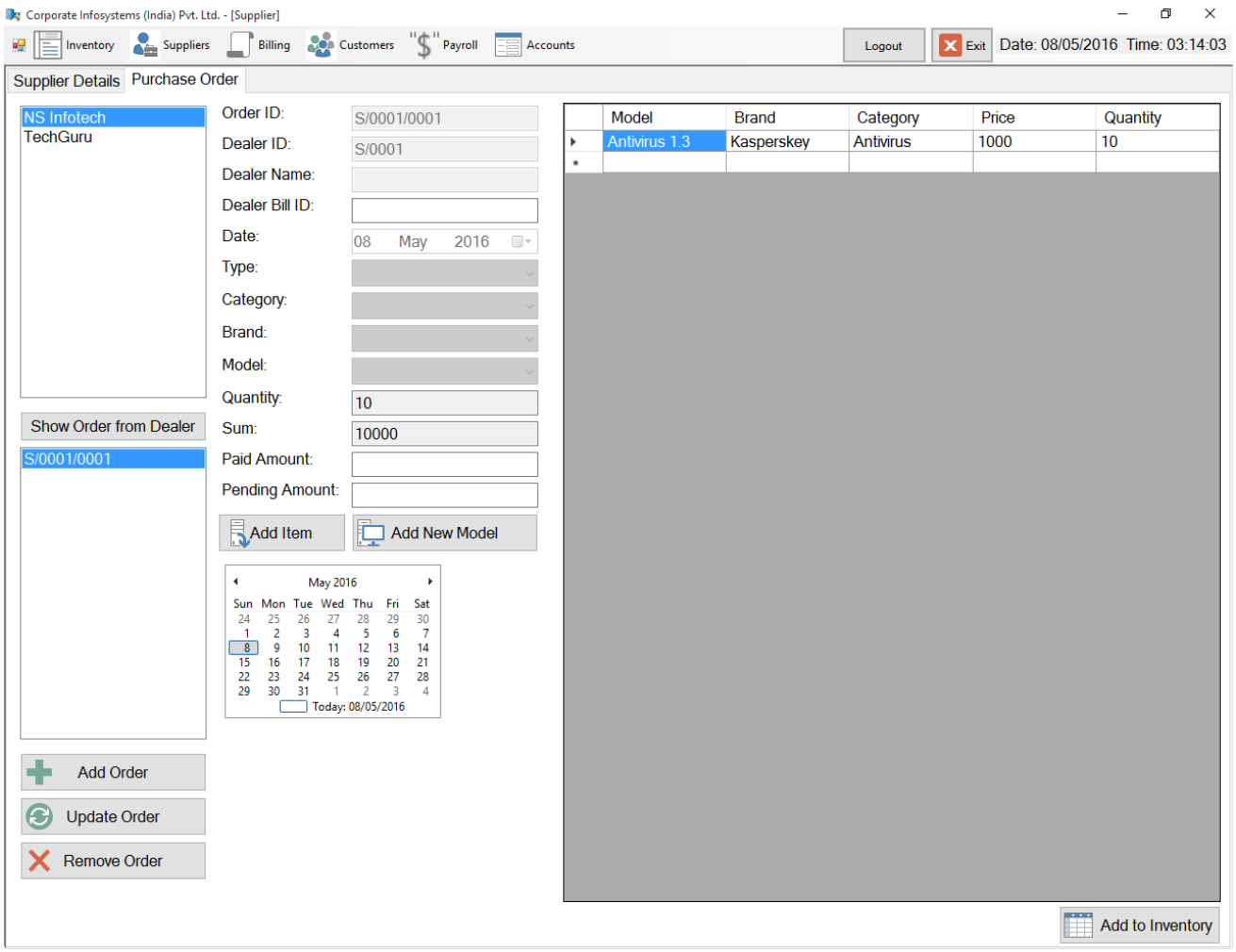
Employee Payment



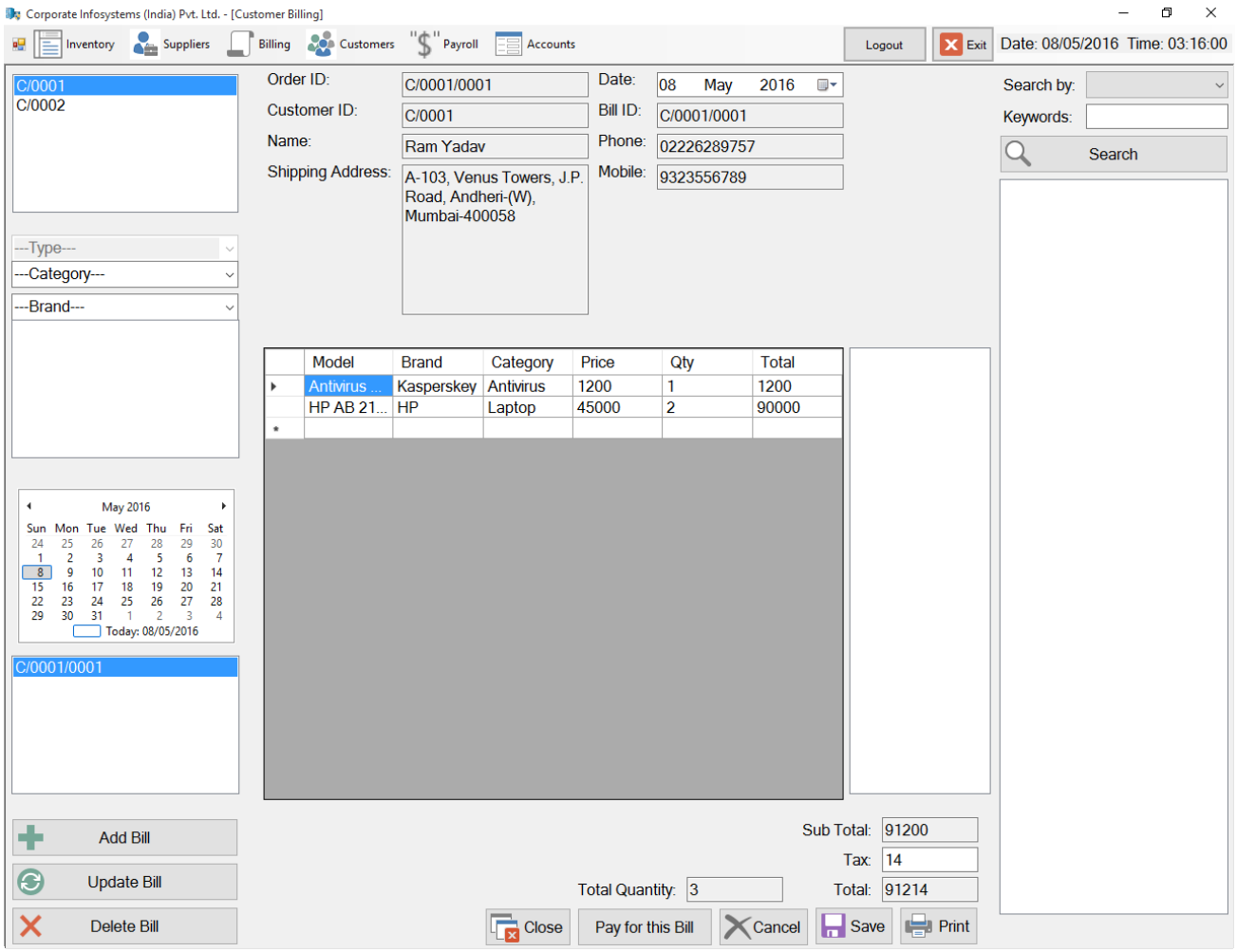
Employee Muster



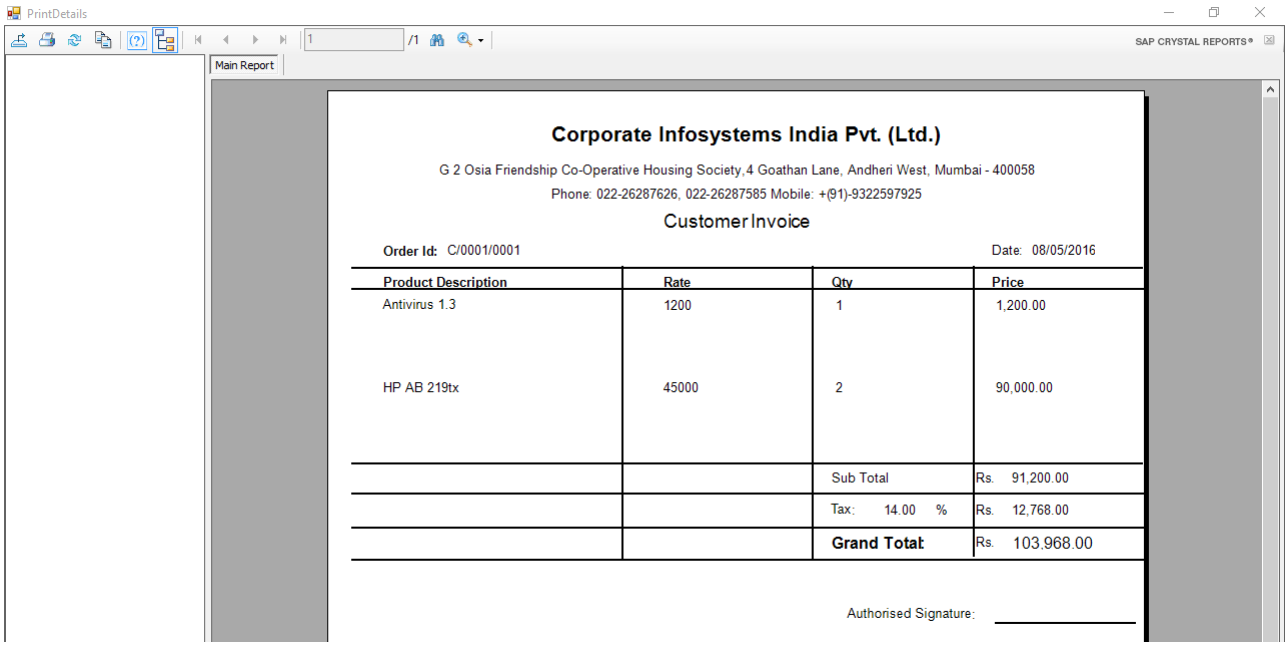
Supplier Purchase Order



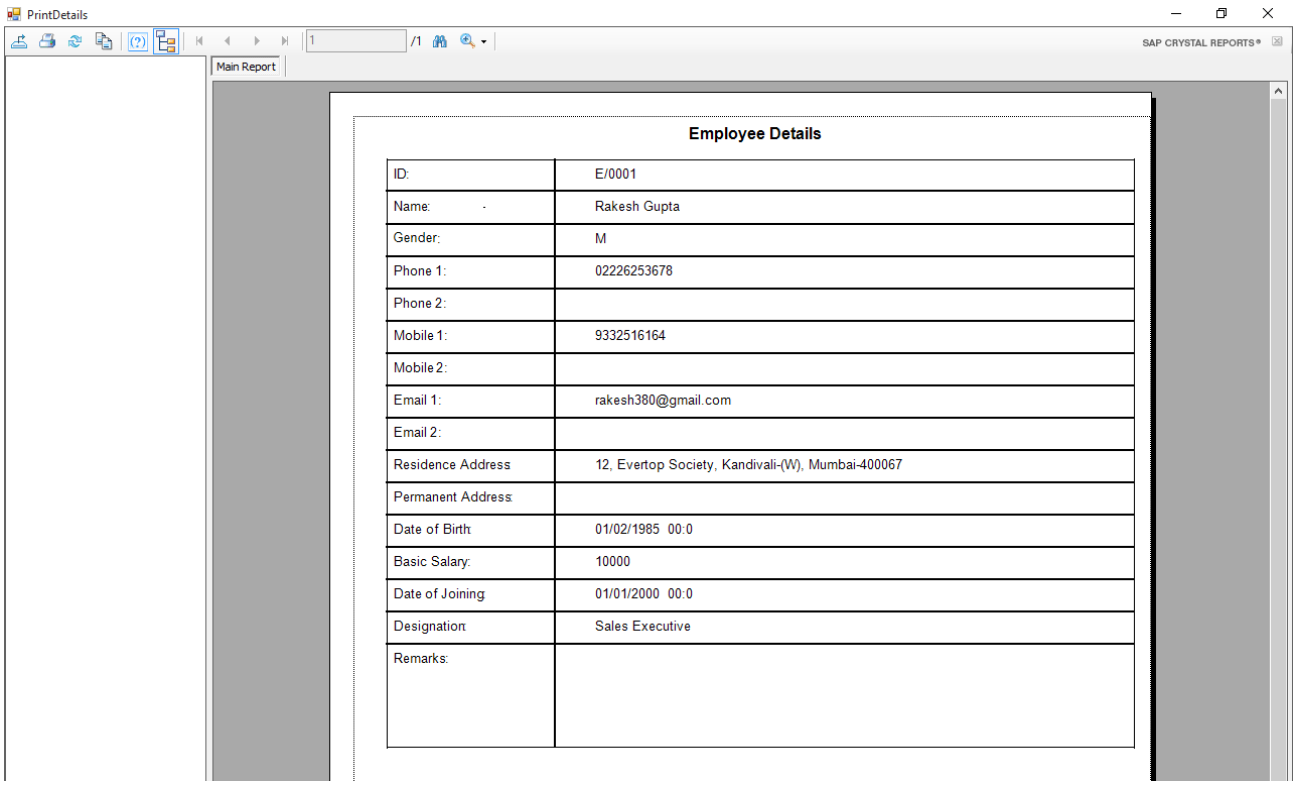
Customer Billing



Report – Invoice



Report - Employee Details



1.2) Coding

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace Test_GUI_for_Inventory_App
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]

        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }

    }

    static class Validations
    {
        public static string flagReport = "";
        public static void OnlyTextAllowed(KeyPressEventArgs e)
        {
            if (char.IsLetter(e.KeyChar))
            {
            }

            else
            {
                if (e.KeyChar == (char)Keys.Back || e.KeyChar == (char)Keys.Space)
                {
                }

                else
                {
                    e.Handled = e.KeyChar != (char)Keys.Back;
                    MessageBox.Show("Only text allowed!");
                }
            }
        }
        public static void OnlyNosAllowed(KeyPressEventArgs e)
        {
            if (char.IsNumber(e.KeyChar))
            {
            }

            else
            {
```

```
                if (e.KeyChar == (char)Keys.Back)
                {
                }

                else
                {
                    e.Handled = e.KeyChar != (char)Keys.Back;
                    MessageBox.Show("Only numbers allowed!");
                }
            }
        }
        public static void ValidateEmail(EventArgs e, TextBox txtEmail)
        {
            Regex myRegularExpression = new Regex(@"^(?!\.)[\.\w!#$%&'*+\-
/=?\^_`{|}~]{1,64}(?<!\.)@((([\-\w]+\.)+[a-zA-Z]{2,4})|(([0-9]{1,3}\.){3}[0-9]{1,3}))$");

            if (myRegularExpression.IsMatch(txtEmail.Text))
            {
            }
            else
            {
                if (txtEmail.Text != "")
                {
                    MessageBox.Show("Invalid email");
                    txtEmail.Focus();
                    txtEmail.Clear();
                }
            }
        }
    }
}
```

Login.cs

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

namespace Test_GUI_for_Inventory_App
{
    public partial class MainForm : Form
    {
        LoginForm lf = new LoginForm();
        public MainForm()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            lblCurrDate.Text = "Date: " + DateTime.Now.ToShortDateString();

            lf.Show();
            mainMenuStrip.Enabled = false;
```

```
            lf.MdiParent = this;
            mainMenubtnLogout.Visible = false;
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            timerCurrTime.Start();
            this.lblCurrTime.Text = "Time: " + DateTime.Now.ToString("HH:mm:ss");
        }

        private void MainForm_MdiChildActivate(object sender, EventArgs e)
        {
            if (lf.Visible == false && lf.flag == 1)
            {
                if (lf.privilege == "Y")
                {
                    mainMenuStrip.Enabled = true;
                    mainMenubtnLogin.Visible = false;
                    mainMenubtnLogout.Visible = true;
                    mainMenubtnLogout.BringToFront();
                }
                else
                {
                    mainMenuStrip.Enabled = true;
                    InventorytoolStripMenuItem1.Enabled = true;
                    accountsToolStripMenuItem.Enabled = false;
                    payrollToolStripMenuItem.Enabled = false;
                    billingToolStripMenuItem.Enabled = false;
                    customersToolStripMenuItem.Enabled = false;
                    suppliersToolStripMenuItem.Enabled = false;

                    mainMenubtnLogin.Visible = false;
                    mainMenubtnLogout.Visible = true;
                    mainMenubtnLogout.BringToFront();
                }
            }
            else
            {
                mainMenuStrip.Enabled = false;
            }
        }
        private void mainMenubtnLogin_Click(object sender, EventArgs e)
        {
            Application.Restart();
        }
        private void mainMenubtnLogout_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to Logout?",
"Confirm Action", MessageBoxButtons.YesNo);

            if(result==DialogResult.Yes)
            {
                Application.Restart();
            }
        }
        private void mainMenubtnExit_Click(object sender, EventArgs e)
        {
            DialogResult result = MessageBox.Show("Are you sure you want to Quit?",
"Confirm Action", MessageBoxButtons.YesNo);
```

```
            if (result == DialogResult.Yes)
            {
                Application.Exit();
            }
        }
        private void InventorytoolStripMenuItem_Click(object sender, EventArgs e)
        {
            Inventory inventory = new Inventory();
            inventory.instance.MdiParent = this;
            inventory.instance.Show();
            inventory.instance.WindowState = FormWindowState.Maximized;
        }
        private void suppliersToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Supplier supplier = new Supplier();
            supplier.instance.MdiParent = this;
            supplier.instance.Show();
            supplier.instance.WindowState = FormWindowState.Maximized;
        }
        private void billingToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Customer_Billing bill = new Customer_Billing();
            bill.instance.MdiParent = this;
            bill.instance.Show();
            bill.instance.WindowState = FormWindowState.Maximized;
        }
        private void customersToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Customer customer = new Customer();
            customer.instance.MdiParent = this;
            customer.instance.Show();
            customer.instance.WindowState = FormWindowState.Maximized;
        }
        private void payrollToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Payroll payroll = new Payroll();
            payroll.instance.MdiParent = this;
            payroll.instance.Show();
            payroll.instance.WindowState = FormWindowState.Maximized;
        }
        private void accountsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            UserAccounts useraccount = new UserAccounts();
            useraccount.instance.MdiParent = this;
            useraccount.instance.Show();
            useraccount.instance.WindowState = FormWindowState.Maximized;
        }
    }
}
```

Login.cs

```
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Test_GUI_for_Inventory_App
{
    public partial class LoginForm : Form
    {
        string MyConnectionString;
        string pwd;
        public string privilege;
        MySqlConnection con;
        MySqlDataReader rdr;

        public int flag = 0;
        public string access = "";

        public LoginForm()
        {
            InitializeComponent();
        }
        public void connectingCsmsLoginDetails()
        {
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }
        public void reloadComboBoxUserList()
        {
            try
            {
                connectingCsmsLoginDetails();
                string stm = "SELECT username FROM login";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();

                while (rdr.Read())
                {
                    comboUserName.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if(con.State==ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
```

```
        private void btnLogin_Click(object sender, EventArgs e)
        {
            try
            {
                connectingCsmsLoginDetails();

                string stm = "SELECT password, privilege FROM login where username='" +
comboUserName.Text + "'";

                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();

                if (comboUserName.Text == "")
                {
                    MessageBox.Show("Select a username!");
                }
                else if (txtPassword.Text == "")
                {
                    MessageBox.Show("Enter a password!");
                }
                else
                {
                    while (rdr.Read())
                    {
                        pwd = rdr.GetString(0);
                        access = rdr.GetString(1);
                        if (txtPassword.Text == pwd)
                        {
                            flag = 1;
                            privilege = access;
                            this.Close();
                        }
                        else
                        {
                            flag = 0;
                            MessageBox.Show("Invalid Username or Password", "Login Error");
                            txtPassword.Clear();
                            txtPassword.Focus();
                        }
                    }
                }
            }
            catch(Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        private void LoginForm_Load(object sender, EventArgs e)
        {
            reloadComboBoxUserList();
        }
        private void button1_Click(object sender, EventArgs e)
```

```
        {
            this.Close();
        }
        private void linkLblForgotPassword_LinkClicked_1(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            if (comboUserName.Text == "")
            {
                MessageBox.Show("Select a username!");
            }
            else
            {
                PasswordRecoveryForm pass_recovery = new PasswordRecoveryForm();
                pass_recovery.pass_values = comboUserName.Text;
                pass_recovery.ShowDialog();
            }
        }
        private void txtPassword_KeyPress_1(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)Keys.Return)
            {
                btnLogin.PerformClick();
            }
        }
    }
}
```

Inventory.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Test_GUI_for_Inventory_App
{
    public partial class Inventory : Form
    {
        private static Inventory _instance;
        MySqlConnection con;
        MySqlCommand cmd;
        MySqlDataReader rdr;
        String MyConnectionString;

        public Inventory()
        {
            InitializeComponent();
        }
        private void Inventory_Load(object sender, EventArgs e)
        {
            loadInventory();
        }
        public Inventory instance
        {
```

```
            get
            {
                if (Inventory._instance == null)
                {
                    Inventory._instance = new Inventory();
                }
                return Inventory._instance;
            }
        }
        private void Inventory_FormClosed(object sender, FormClosedEventArgs e)
        {
            Inventory._instance = null;
        }
        private void loadInventory()
        {
            try
            {
                dataGridViewInventory.Rows.Clear();
                connectingCsmsInventory();
                String stm = "SELECT * from stock;";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();

                while (rdr.Read())
                {
                    DataGridViewRow row =
(DataGridViewRow)dataGridViewInventory.Rows[0].Clone();
                    row.Cells[0].Value = rdr.GetString(0);
                    row.Cells[1].Value = rdr.GetString(1);
                    row.Cells[2].Value = rdr.GetString(2);
                    row.Cells[3].Value = rdr.GetString(3);
                    row.Cells[4].Value = rdr.GetString(5);
                    row.Cells[5].Value = rdr.GetString(6);
                    row.Cells[6].Value = rdr.GetString(4);
                    dataGridViewInventory.Rows.Add(row);
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        public void connectingCsmsInventory()
        {
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }
        private void dataGridViewInventory_DoubleClick(object sender, EventArgs e)
        {
            loadInventory();
```

```
        }
    }
}
```

Suppplire.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Test_GUI_for_Inventory_App
{
    public partial class Supplier : Form
    {
        public static String modID;
        String MyConnectionString;
        String dealer_id, name, phone1, phone2, mob1, mob2, email1, email2, address1,
address2, remarks;
        Char gender;
        MySqlConnection con;
        MySqlCommand cmd;
        MySqlDataReader rdr;
        String flag, flag2;

        private static Supplier _instance;


        public Supplier()
        {
            InitializeComponent();
        }
        private void Supplier_Load(object sender, EventArgs e)
        {
            reloadListBoxSuppliers();
            reloadlistBoxPurchaseOrderOrderList();
            suppdetails_editable(false);
            reloadComboType();
            flag = "";
            flag2 = "";
        }
        public Supplier instance
        {
            get
            {
                if (Supplier._instance == null)
                {
                    Supplier._instance = new Supplier();
                }
                return Supplier._instance;
            }
        }
```

```
        private void Supplier_FormClosed(object sender, FormClosedEventArgs e)
        {
            Supplier._instance = null;
        }

        /*
         *
         * Supplier Details Tab Coding
         *
         *
         */

        private void listBoxSupplierList_SelectedIndexChanged(object sender, EventArgs e)
        {
            suppdetails_editable(false);
            if (listBoxSupplierList.SelectedIndex > -1)
            {
                String selectedValue = listBoxSupplierList.SelectedItem.ToString();
                connectingCsmsSuppDetails();
                try
                {
                    string stm = "SELECT * FROM dealer_details where name='" +
selectedValue + "'";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    populatingForm();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void btnAddSupp_Click(object sender, EventArgs e)
        {
            btnSuppDetailsSave.Enabled = true;
            clearAllFieldsSupplierDetails();
            suppdetails_editable(true);
            flag = "Add";

            btnSuppDetailsCancel.BringToFront();
            ctrls_enabled(true);
            suppIDGeneration();
        }
        private void suppIDGeneration()
        {
            String suppID = "", finalSuppID = "";
            int suppNo = 1;
            connectingCsmsSuppDetails();
            try
            {
                String stm = "SELECT dealer_id from dealer_details;";
```

```
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                if (rdr.Read())
                {
                    rdr.Close();
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        suppID = rdr.GetString(0);
                        string t = suppID.Substring(2, 4);
                        suppNo = int.Parse(t) + 1;
                    }
                    finalSuppID = "S" + "/" + suppNo.ToString("D4");
                }
                else
                {
                    finalSuppID = "S" + "/0001";
                }
                txtSuppID.Text = finalSuppID;
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        private void btnUpdateSuppDetails_Click(object sender, EventArgs e)
        {
            if (listBoxSupplierList.SelectedIndex > -1 ||
listBoxSuppDetailsSearchResults.SelectedIndex > 0)
            {
                btnSuppDetailsSave.Enabled = true;
                suppdetails_editable(true);
                flag = "Update";

                btnSuppDetailsCancel.BringToFront();
                ctrls_enabled(true);
            }
        }
        private void btnRemoveSupp_Click(object sender, EventArgs e)
        {
            if (txtSuppID.Text != null) //((listBoxCustomerList.SelectedIndex > -1 ||
listBoxCustDetailsSearchResults.SelectedIndex > 0) && (txtCustID!=null))
            {
                try
                {
                    connectingCsmsSuppDetails();
                    String stm = "SELECT dealer_id,name FROM dealer_details where dealer_id
= '" + txtSuppID.Text + "'";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
```

```
                        if (MessageBox.Show("Do you want to delete Supplier with name " +
rdr.GetString(1) + "?", "Delete Supplier?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
                        {
                            cmd = con.CreateCommand();
                            cmd.CommandText = "delete from dealer_details where
dealer_id='" + rdr.GetString(0) + "';";
                            rdr.Close();
                            cmd.ExecuteNonQuery();
                        }
                    }
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                    reloadListBoxSuppliers();
                    clearAllFieldsSupplierDetails();
                }
            }
        }
        private void btnSuppDetailsSave_Click(object sender, EventArgs e)
        {
            //Extracting data from fields
            dealer_id = txtSuppID.Text;
            name = txtSuppName.Text;
            if (comboBoxSuppGender.Text == "Male")
                gender = 'M';
            else if (comboBoxSuppGender.Text == "Female")
                gender = 'F';
            else
                gender = ' ';
            phone1 = txtSuppPhone1.Text;
            phone2 = txtSuppPhone2.Text;
            mob1 = txtSuppMobile1.Text;
            mob2 = txtSuppMobile2.Text;
            email1 = txtSuppEmail1.Text;
            email2 = txtSuppEmail2.Text;
            address1 = txtSuppAddress.Text;
            remarks = txtSuppRemarks.Text;

            //Inserting data
            connectingCsmsSuppDetails();
            try
            {
                cmd = con.CreateCommand();
                if (flag.Equals("Add"))
                {
                    cmd.CommandText = "insert into dealer_details
values(@dealer_id,@name,@add1,@ph1,@ph2,@mob1,@mob2,@gender,@email1,@email2,@remarks)";
                }
                else if (flag.Equals("Update"))
                {
```

```
                    cmd.CommandText = "UPDATE dealer_details SET
name=@name,address1=@add1,phone1=@ph1,phone2=@ph2,mob1=@mob1,mob2=@mob2,gender=@gender,emai
l1=@email1,email2=@email2,remarks=@remarks WHERE dealer_id = @dealer_id";
                }
                cmd.Parameters.AddWithValue("@dealer_id", dealer_id);
                cmd.Parameters.AddWithValue("@name", name);
                cmd.Parameters.AddWithValue("@add1", address1);
                cmd.Parameters.AddWithValue("@ph1", phone1);
                cmd.Parameters.AddWithValue("@ph2", phone2);
                cmd.Parameters.AddWithValue("@mob1", mob1);
                cmd.Parameters.AddWithValue("@mob2", mob2);
                cmd.Parameters.AddWithValue("@gender", gender);
                cmd.Parameters.AddWithValue("@email1", email1);
                cmd.Parameters.AddWithValue("@email2", email2);
                cmd.Parameters.AddWithValue("@remarks", remarks);
                cmd.ExecuteNonQuery();
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
                reloadListBoxSuppliers();
            }
            suppdetails_editable(false);
            btnSuppDetailsSave.Enabled = false;

            ctrls_enabled(false);
            btnSuppDetailsCancel.SendToBack();
        }
        private void btnSuppDetailsClear_Click(object sender, EventArgs e)
        {
            clearAllFieldsSupplierDetails();
        }
        private void btnSuppDetailsCancel_Click(object sender, EventArgs e)
        {
            clearAllFieldsSupplierDetails();
            suppdetails_editable(false);
            ctrls_enabled(false);
            btnSuppDetailsCancel.SendToBack();
        }
        private void btnSuppDetailsClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void btnSuppDetailsSearch_Click(object sender, EventArgs e)
        {
            listBoxSuppDetailsSearchResults.Items.Clear();
            if (comboBoxSuppDetailsSearchBy.SelectedIndex > -1)
            {
                string stm;
                if (comboBoxSuppDetailsSearchBy.SelectedItem.ToString() == "ID")
                {
```

```
                    stm = "SELECT dealer_id FROM dealer_details where dealer_id like '%" +
txtSuppDetailsSearchKeywords.Text + "%'";
                    listBoxSuppDetailsSearchResults.Items.Add("Based on ID:");
                }
                else if (comboBoxSuppDetailsSearchBy.SelectedItem.ToString() == "Name")
                {
                    stm = "SELECT name FROM dealer_details where name like '%" +
txtSuppDetailsSearchKeywords.Text + "%'";
                    listBoxSuppDetailsSearchResults.Items.Add("Based on Name:");
                }
                else if (comboBoxSuppDetailsSearchBy.SelectedItem.ToString() == "Phone 1")
                {
                    stm = "SELECT phone1 FROM dealer_details where phone1 like '%" +
txtSuppDetailsSearchKeywords.Text + "%'";
                    listBoxSuppDetailsSearchResults.Items.Add("Based on Phone 1:");
                }
                else if (comboBoxSuppDetailsSearchBy.SelectedItem.ToString() == "Phone 2")
                {
                    stm = "SELECT phone2 FROM dealer_details where phone2 like '%" +
txtSuppDetailsSearchKeywords.Text + "%'";
                    listBoxSuppDetailsSearchResults.Items.Add("Based on Phone 2:");
                }
                else if (comboBoxSuppDetailsSearchBy.SelectedItem.ToString() == "Mobile 1")
                {
                    stm = "SELECT mob1 FROM dealer_details where mob1 like '%" +
txtSuppDetailsSearchKeywords.Text + "%'";
                    listBoxSuppDetailsSearchResults.Items.Add("Based on Mobile 1:");
                }
                else if (comboBoxSuppDetailsSearchBy.SelectedItem.ToString() == "Mobile 2")
                {
                    stm = "SELECT mob2 FROM dealer_details where mob2 like '%" +
txtSuppDetailsSearchKeywords.Text + "%'";
                    listBoxSuppDetailsSearchResults.Items.Add("Based on Mobile 2:");
                }
                else if (comboBoxSuppDetailsSearchBy.SelectedItem.ToString() == "Email 1")
                {
                    stm = "SELECT email1 FROM dealer_details where email1 like '%" +
txtSuppDetailsSearchKeywords.Text + "%'";
                    listBoxSuppDetailsSearchResults.Items.Add("Based on Email 1:");
                }
                else // if (comboBoxCustDetailsSearchBy.SelectedItem == "Email 2")
                {
                    stm = "SELECT email2 FROM dealer_details where email2 like '%" +
txtSuppDetailsSearchKeywords.Text + "%'";
                    listBoxSuppDetailsSearchResults.Items.Add("Based on Email 2:");
                }
                try
                {
                    connectingCsmsSuppDetails();
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        listBoxSuppDetailsSearchResults.Items.Add(rdr.GetString(0));
                    }
                }
                catch (Exception)
                {
                    throw;
```

```
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void listBoxSuppDetailsSearchResults_SelectedIndexChanged(object sender,
EventArgs e)
        {
            suppdetails_editable(false);
            if (listBoxSuppDetailsSearchResults.SelectedIndex > 0)
            {
                String selectedValue =
listBoxSuppDetailsSearchResults.SelectedItem.ToString();
                String stm;
                connectingCsmsSuppDetails();
                if (listBoxSuppDetailsSearchResults.Items[0].ToString() == "Based on ID:")
                {
                    stm = "SELECT * FROM dealer_details where dealer_id = '" +
selectedValue + "'";
                }
                else if (listBoxSuppDetailsSearchResults.Items[0].ToString() == "Based on
Name:")
                {
                    stm = "SELECT * FROM dealer_details where name = '" + selectedValue +
"'";
                }
                else if (listBoxSuppDetailsSearchResults.Items[0].ToString() == "Based on
Phone 1:")
                {
                    stm = "SELECT * FROM dealer_details where phone1 = '" + selectedValue +
"'";
                }
                else if (listBoxSuppDetailsSearchResults.Items[0].ToString() == "Based on
Phone 2:")
                {
                    stm = "SELECT * FROM dealer_details where phone2 = '" + selectedValue +
"'";
                }
                else if (listBoxSuppDetailsSearchResults.Items[0].ToString() == "Based on
Mobile 1:")
                {
                    stm = "SELECT * FROM dealer_details where mob1 = '" + selectedValue +
"'";
                }
                else if (listBoxSuppDetailsSearchResults.Items[0].ToString() == "Based on
Mobile 2:")
                {
                    stm = "SELECT * FROM dealer_details where mob2 = '" + selectedValue +
"'";
                }
                else if (listBoxSuppDetailsSearchResults.Items[0].ToString() == "Based on
Email 1:")
                {
```

```
                    stm = "SELECT * FROM dealer_details where email1 = '" + selectedValue +
"'";
                }
                else
                {
                    stm = "SELECT * FROM dealer_details where email2 = '" + selectedValue +
"'";
                }
                try
                {
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    populatingForm();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        public void reloadListBoxSuppliers()
        {
            listBoxSupplierList.Items.Clear();
            listBoxPurchaseOrderSupplierList.Items.Clear();
            try
            {
                connectingCsmsSuppDetails();
                string stm = "SELECT name FROM dealer_details";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();

                while (rdr.Read())
                {
                    listBoxSupplierList.Items.Add(rdr.GetString(0));
                    listBoxPurchaseOrderSupplierList.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        private void populatingForm()                //of Supplier Details form
        {
            while (rdr.Read())
```

```
    {
        txtSuppID.Text = rdr.GetString(0);
        txtSuppName.Text = rdr.GetString(1);
        txtSuppAddress.Text = rdr.GetString(2);
        txtSuppPhone1.Text = rdr.GetString(3);
        txtSuppPhone2.Text = rdr.GetString(4);
        txtSuppMobile1.Text = rdr.GetString(5);
        txtSuppMobile2.Text = rdr.GetString(6);
        if (rdr.GetChar(7) == 'M')
            comboBoxSuppGender.Text = "Male";
        else if (rdr.GetChar(7) == 'F')
            comboBoxSuppGender.Text = "Female";
        else
            comboBoxSuppGender.Text = "";
        txtSuppEmail1.Text = rdr.GetString(8);
        txtSuppEmail2.Text = rdr.GetString(9);
        txtSuppRemarks.Text = rdr.GetString(10);
    }
}
private void suppdetails_editable(bool choice)
{
    choice = !choice;
    txtSuppName.ReadOnly = choice;
    comboBoxSuppGender.Enabled = !choice;
    txtSuppPhone1.ReadOnly = choice;
    txtSuppPhone2.ReadOnly = choice;
    txtSuppMobile1.ReadOnly = choice;
    txtSuppMobile2.ReadOnly = choice;
    txtSuppEmail1.ReadOnly = choice;
    txtSuppEmail2.ReadOnly = choice;
    txtSuppAddress.ReadOnly = choice;
    txtSuppRemarks.ReadOnly = choice;
}
private void clearAllFieldsSupplierDetails()
{
    txtSuppID.Text = "";
    txtSuppName.Text = "";
    txtSuppAddress.Text = "";
    txtSuppPhone1.Text = "";
    txtSuppPhone2.Text = "";
    txtSuppMobile1.Text = "";
    txtSuppMobile2.Text = "";
    comboBoxSuppGender.SelectedIndex = 0;
    txtSuppEmail1.Text = "";
    txtSuppEmail2.Text = "";
    txtSuppRemarks.Text = "";
}
private void ctrls_enabled(bool choice)
{
    tableLayoutPanelSuppDisplay.Enabled = !choice;
    tableLayoutPanelSuppSearchCtrls.Enabled = !choice;

    btnSuppDetailsCancel.Enabled = choice;
    btnSuppDetailsClose.Enabled = !choice;
    btnSuppDetailsClear.Enabled = choice;
    btnSuppDetailsSave.Enabled = choice;
}

private void txtSuppName_KeyPress(object sender, KeyPressEventArgs e)
```

```
        {
            Test_GUI_for_Inventory_App.Validations.OnlyTextAllowed(e);
        }

        private void txtSuppPhone1_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }

        private void txtSuppPhone2_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }

        private void txtSuppMobile1_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }

        private void txtSuppMobile2_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }

        private void txtSuppEmail1_Leave(object sender, EventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.ValidateEmail(e, txtSuppEmail1);
        }

        private void txtSuppEmail2_Leave(object sender, EventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.ValidateEmail(e, txtSuppEmail2);
        }

        /*
         *
         * Supplier Purchase Order Tab Coding
         *
         *
         */

        private void listBoxPurchaseOrderSupplierList_DoubleClick(object sender, EventArgs
e)
        {
            if (listBoxPurchaseOrderSupplierList.SelectedIndex > -1)
            {
                if (flag2.Equals("update"))
                {
                    purchase_order_editable(false);
                    clearAllFieldsPurchaseOrder();
                }
                if (flag2.Equals("add"))
                {
                    String selectedValue =
listBoxPurchaseOrderSupplierList.SelectedItem.ToString();
                    connectingCsmsSuppDetails();
                    try
                    {
```

```
                        string stm = "SELECT dealer_id,name FROM dealer_details where
name='" + selectedValue + "'";
                        MySqlCommand cmd = new MySqlCommand(stm, con);
                        rdr = cmd.ExecuteReader();
                        while (rdr.Read())
                        {
                            txtDealerID.Text = rdr.GetString(0);
                            txtDealerName.Text = rdr.GetString(1);
                        }
                        rdr.Close();
                        orderIDGeneration();
                    }
                    catch (Exception)
                    {
                        throw;
                    }
                    finally
                    {
                        if (con.State == ConnectionState.Open)
                        {
                            con.Close();
                        }
                    }
                }
            }
        }
        private void btnShowDealerOrder_Click(object sender, EventArgs e)
        {
            String stm, dealerNameHere, dealerIdHere = "";
            if (listBoxPurchaseOrderSupplierList.SelectedIndex > -1)
            {
                listBoxPurchaseOrderOrderList.Items.Clear();
                dealerNameHere = listBoxPurchaseOrderSupplierList.SelectedItem.ToString();
                connectingCsmsSuppDetails();
                try
                {
                    stm = "SELECT dealer_id from dealer_details where name='" +
dealerNameHere + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        dealerIdHere = rdr.GetString(0);
                    }
                    rdr.Close();
                    stm = "SELECT purchase_order_id from purchase_order where dealer_id =
'" + dealerIdHere + "';";
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        listBoxPurchaseOrderOrderList.Items.Add(rdr.GetString(0));
                    }
                }
                catch (Exception)
                {
                    throw;
                }
                finally
```

```
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void listBoxPurchaseOrderOrderList_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (listBoxPurchaseOrderOrderList.SelectedIndex > -1)
            {
                purchase_order_editable(false);
                String selectedValue =
listBoxPurchaseOrderOrderList.SelectedItem.ToString();
                try
                {
                    connectingCsmsSuppDetails();
                    clearDataGridViewProductOrdering();
                    String stm = "SELECT * from purchase_order where purchase_order_id = '"
+ selectedValue + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        txtOrderID.Text = rdr.GetString(0);
                        txtDealerID.Text = rdr.GetString(1);
                        dateTimePickerOrderDate.Value = rdr.GetDateTime(4);
                    }
                    rdr.Close();
                    stm = "SELECT a.amount,a.quantity,a.model_id,b.category,b.brand from
purchase_item a,stock b where a.model_id=b.model_id and a.purchase_order_id ='" +
txtOrderID.Text + "';";
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();

                    while (rdr.Read())
                    {
                        DataGridViewRow row =
(DataGridViewRow)dataGridViewProductOrdering.Rows[0].Clone();
                        row.Cells[0].Value = rdr.GetString(2);
                        row.Cells[1].Value = rdr.GetString(4);
                        row.Cells[2].Value = rdr.GetString(3);
                        row.Cells[3].Value = rdr.GetString(0);
                        row.Cells[4].Value = rdr.GetString(1);
                        dataGridViewProductOrdering.Rows.Add(row);
                    }
                    countAmountQuantity();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
```

```
                }
            }
        }
    }
    private void btnAddCust_Click(object sender, EventArgs e)
    {
        purchase_order_editable(true);
        clearAllFieldsPurchaseOrder();
        flag2 = "add";
    }
    private void btnUpdateOrderDetails_Click(object sender, EventArgs e)
    {
        if (txtOrderID.Text != "")
        {
            purchase_order_editable(true);
            flag2 = "update";
        }
    }
    private void btnRemoveOrder_Click(object sender, EventArgs e)
    {
        if (listBoxPurchaseOrderOrderList.SelectedIndex > -1 && txtOrderID.Text ==
listBoxPurchaseOrderOrderList.SelectedItem.ToString())
        {
            MessageBox.Show(listBoxPurchaseOrderOrderList.SelectedItem.ToString());
            try
            {
                connectingCsmsSuppDetails();
                int r = dataGridViewProductOrdering.RowCount;
                for (int i = 1; i < r; i++)
                {
                    if (dataGridViewProductOrdering.Rows[i -
1].Cells[0].Value.ToString() != null)
                    {
                        int t;
                        t = int.Parse(dataGridViewProductOrdering.Rows[i -
1].Cells[4].Value.ToString());
                        cmd = con.CreateCommand();
                        cmd.CommandText = "UPDATE stock SET quantity=quantity-@quantity
where model_id=@model_id";
                        cmd.Parameters.AddWithValue("@model_id",
dataGridViewProductOrdering.Rows[i - 1].Cells[0].Value.ToString());
                        int quantity = Int32.Parse(dataGridViewProductOrdering.Rows[i -
1].Cells[4].Value.ToString());
                        cmd.Parameters.AddWithValue("@quantity", quantity);
                        cmd.ExecuteNonQuery();
                        //Removing from purchase_item
                        cmd = con.CreateCommand();
                        cmd.CommandText = "DELETE from purchase_item where
purchase_order_id=@purchase_order_id";
                        cmd.Parameters.AddWithValue("@purchase_order_id",
txtOrderID.Text);
                        cmd.ExecuteNonQuery();
                    }
                }
                cmd = con.CreateCommand();
                cmd.CommandText = "DELETE from purchase_order where
purchase_order_id=@p_o_i";
                cmd.Parameters.AddWithValue("@p_o_i", txtOrderID.Text);
                cmd.ExecuteNonQuery();
```

```
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                        clearAllFieldsPurchaseOrder();
                        clearDataGridViewProductOrdering();
                        reloadListBoxSuppliers();
                        reloadlistBoxPurchaseOrderOrderList();
                    }
                }
            }
        }
        private void comboType_SelectedIndexChanged(object sender, EventArgs e)
        {
            connectingCsmsSuppDetails();
            comboCategory.Items.Clear();
            comboBrand.Items.Clear();
            comboModel.Items.Clear();
            String stm = "Select DISTINCT category from stock where type = '" +
(String)comboType.SelectedItem + "';";
            try
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    comboCategory.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                rdr.Close();
                con.Close();
            }
        }
        private void comboCategory_SelectedIndexChanged(object sender, EventArgs e)
        {
            connectingCsmsSuppDetails();
            String stm = "Select DISTINCT brand from stock where type = '" +
(String)comboType.SelectedItem + "' and category ='" + comboCategory.SelectedItem + "';";
            try
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    comboBrand.Items.Add(rdr.GetString(0));
                }
            }
```

```
            catch (Exception)
            {
                throw;
            }
            finally
            {
                rdr.Close();
                con.Close();
            }
        }
        private void comboBrand_SelectedIndexChanged(object sender, EventArgs e)
        {
            connectingCsmsSuppDetails();
            String stm = "Select model_id from stock where type = '" +
(String)comboType.SelectedItem + "' and category ='" + comboCategory.SelectedItem + "' and
brand = '" + comboBrand.SelectedItem + "';";
            try
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    comboModel.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                rdr.Close();
                con.Close();
            }

        }
        private void btnAddToDataGrid_Click(object sender, EventArgs e)
        {
            if (comboModel.SelectedIndex > -1)
            {
                DataGridViewRow row =
(DataGridViewRow)dataGridViewProductOrdering.Rows[0].Clone();
                row.Cells[0].Value = comboModel.SelectedItem;
                row.Cells[1].Value = comboBrand.SelectedItem;
                row.Cells[2].Value = comboCategory.SelectedItem;
                dataGridViewProductOrdering.Rows.Add(row);
            }
        }
        private void btnNewModel_Click(object sender, EventArgs e)
        {
            AddModel addModel = new AddModel();
            addModel.ShowDialog(this);
            reloadComboType();
        }
        private void dataGridViewProductOrdering_CellEndEdit(object sender,
DataGridViewCellEventArgs e)
        {
            for (int rows = 0; rows < dataGridViewProductOrdering.Rows.Count; rows++)
            {
```

```
                if (dataGridViewProductOrdering.Rows.Count > 0 &&
dataGridViewProductOrdering.Rows[rows].Cells[3] != null &&
dataGridViewProductOrdering.Rows[rows].Cells[4] != null &&
dataGridViewProductOrdering.Rows[rows].Cells[0] != null)
                {
                    if (dataGridViewProductOrdering.Rows[rows].Cells[0].Value == null)
                    {
                        if (dataGridViewProductOrdering.Rows[rows].Cells[3].Value != null)
                        {
                            dataGridViewProductOrdering.Rows[rows].Cells[3].Value = "";
                        }
                        if (dataGridViewProductOrdering.Rows[rows].Cells[4].Value != null)
                        {
                            dataGridViewProductOrdering.Rows[rows].Cells[4].Value = "";
                        }
                    }
                }
            }
            if (flag2.Equals("update"))
            {
                int inStock, inPurchaseItem;
                try
                {
                    connectingCsmsSuppDetails();

                    int r = dataGridViewProductOrdering.RowCount;
                    for (int i = 1; i < r; i++)
                    {
                        if (dataGridViewProductOrdering.Rows[i -
1].Cells[0].Value.ToString() != null)
                        {
                            String stm = "SELECT a.quantity,b.quantity from stock a,
purchase_item b where a.model_id=b.model_id and b.purchase_order_id='" + txtOrderID.Text +
"' and b.model_id='" + dataGridViewProductOrdering.Rows[i - 1].Cells[0].Value + "' and
b.amount='" + dataGridViewProductOrdering.Rows[i - 1].Cells[3].Value + "';";
                            MySqlCommand cmd = new MySqlCommand(stm, con);
                            rdr = cmd.ExecuteReader();
                            while (rdr.Read())
                            {
                                inStock = int.Parse(rdr.GetString(0));
                                inPurchaseItem = int.Parse(rdr.GetString(1));
                                int t1 = inPurchaseItem -
int.Parse(dataGridViewProductOrdering.Rows[i - 1].Cells[4].Value.ToString());
                                int t = inStock - t1;
                                if (t < 0)
                                {
                                    //MessageBox.Show("In if");
                                    dataGridViewProductOrdering.Rows[i - 1].Cells[4].Value
= inStock;
                                }
                            }
                            rdr.Close();
                        }
                    }

                }
                catch (Exception)
                {
                    throw;
```

```
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
            countAmountQuantity();
        }
        private void btnAddToInventory_Click(object sender, EventArgs e)
        {
            String orderID = txtOrderID.Text;
            String dealerID = txtDealerID.Text;
            String dealerBillID = txtDealerBillID.Text;
            int quantity = int.Parse(txtQuantity.Text);
            String totalAmount = txtAmount.Text;
            DateTime orderDate = dateTimePickerOrderDate.Value.Date;

            connectingCsmsSuppDetails();

            if (txtOrderID.Text != "")
            {
                //inserting into Purchase_Order table
                try
                {
                    String stm = "select dealer_id from dealer_details where dealer_id = '"
+ dealerID + "';";
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
                        rdr.Close();
                        stm = "select purchase_order_id from purchase_order where
purchase_order_id = '" + orderID + "';";
                        cmd = new MySqlCommand(stm, con);
                        rdr = cmd.ExecuteReader();
                        if (rdr.Read() && flag2.Equals("add"))
                        {
                            MessageBox.Show("OrderID is not unique. Please choose unique
OrderId");
                            rdr.Close();
                        }
                        else
                        {
                            rdr.Close();
                            cmd = con.CreateCommand();
                            if (flag2.Equals("add"))
                            {
                                cmd.CommandText = "insert into purchase_order
values(@purchase_order_id,@dealer_id,@quantity,@total_amount,@date);";
                                cmd.Parameters.AddWithValue("@purchase_order_id", orderID);
                                cmd.Parameters.AddWithValue("@dealer_id", dealerID);
                                cmd.Parameters.AddWithValue("@quantity", quantity);
                                cmd.Parameters.AddWithValue("@total_amount", totalAmount);
                                cmd.Parameters.AddWithValue("@date", orderDate);
                                cmd.ExecuteNonQuery();
                            }
```

```
                            else if (flag2.Equals("update"))
                            {
                                cmd.CommandText = "UPDATE purchase_order SET
quantity=@quantity,total_amount=@quantity,date=@date WHERE
purchase_order_id=@purchase_order_id;";
                                cmd.Parameters.AddWithValue("@purchase_order_id", orderID);
                                cmd.Parameters.AddWithValue("@quantity", quantity);
                                cmd.Parameters.AddWithValue("@total_amount", totalAmount);
                                cmd.Parameters.AddWithValue("@date", orderDate);
                                cmd.ExecuteNonQuery();
                            }
                            MessageBox.Show("Inserted into purchase_order");

                            int r = dataGridViewProductOrdering.RowCount;

                            if (flag2.Equals("update"))    //Decrementing stock previously
order
                            {
                                for (int i = 1; i < r; i++)
                                {
                                    if (dataGridViewProductOrdering.Rows[i -
1].Cells[0].Value.ToString() != null)
                                    {
                                        int t;
                                        t = int.Parse(dataGridViewProductOrdering.Rows[i -
1].Cells[4].Value.ToString());
                                        cmd = con.CreateCommand();
                                        cmd.CommandText = "UPDATE stock SET
quantity=quantity-@quantity where model_id=@model_id";
                                        cmd.Parameters.AddWithValue("@model_id",
dataGridViewProductOrdering.Rows[i - 1].Cells[0].Value.ToString());
                                        quantity =
Int32.Parse(dataGridViewProductOrdering.Rows[i - 1].Cells[4].Value.ToString());
                                        cmd.Parameters.AddWithValue("@quantity", quantity);
                                        cmd.ExecuteNonQuery();
                                        //Removing from purchase_item
                                        cmd = con.CreateCommand();
                                        cmd.CommandText = "DELETE from purchase_item where
purchase_order_id=@purchase_order_id";
                                        cmd.Parameters.AddWithValue("@purchase_order_id",
txtOrderID.Text);
                                        cmd.ExecuteNonQuery();
                                    }
                                }
                            }

                            for (int i = 1; i < r; i++)
                            {
                                //MessageBox.Show("i = " + i);
                                if (dataGridViewProductOrdering.Rows[i - 1].Cells[0].Value
!= null)
                                {
                                    //MessageBox.Show("Model ID is not null");
                                    stm = "select model_id,quantity from stock where
model_id = '" + dataGridViewProductOrdering.Rows[i - 1].Cells[0].Value.ToString() + "';";
                                    cmd = new MySqlCommand(stm, con);
                                    rdr = cmd.ExecuteReader();
                                    if (rdr.Read())
                                    {
```

```
                                        modID = rdr.GetString(0);
                                        rdr.Close();
                                        //MessageBox.Show("Model ID present!");
                                        cmd = con.CreateCommand();
                                        cmd.CommandText = "INSERT into purchase_item
(purchase_order_id,amount,model_id,quantity)
values(@purchase_order_id,@amount,@model_id,@quantity)";
                                        cmd.Parameters.AddWithValue("@purchase_order_id",
txtOrderID.Text);
                                        //MessageBox.Show("Amount : " +
dataGridViewProductOrdering.Rows[i - 1].Cells[3].Value.ToString());
                                        cmd.Parameters.AddWithValue("@amount",
dataGridViewProductOrdering.Rows[i - 1].Cells[3].Value.ToString());
                                        cmd.Parameters.AddWithValue("@model_id",
dataGridViewProductOrdering.Rows[i - 1].Cells[0].Value.ToString());
                                        quantity =
Int32.Parse(dataGridViewProductOrdering.Rows[i - 1].Cells[4].Value.ToString());
                                        cmd.Parameters.AddWithValue("@quantity", quantity);
                                        cmd.ExecuteNonQuery();
                                        for (int j = 0; j < quantity; j++)
                                        {
                                            cmd = con.CreateCommand();
                                            cmd.CommandText = "update stock set
quantity=quantity+1 where model_id=@model_id";
                                            cmd.Parameters.AddWithValue("@model_id",
Test_GUI_for_Inventory_App.Supplier.modID);
                                            cmd.ExecuteNonQuery();
                                            cmd.CommandText = "";
                                        }
                                        MessageBox.Show("Updated Stock");
                                    }
                                    //MessageBox.Show("Exited 1st if");
                                }
                                //MessageBox.Show("Exited 2nd if");
                            }
                            //MessageBox.Show("Exited for");
                        }
                    }
                    else
                    {
                        MessageBox.Show("Error: Dealer not present!!");
                        rdr.Close();
                    }
                }
                catch
                {
                    throw;
                }
                finally
                {
                    con.Close();
                }
            }
            else
            {
                MessageBox.Show("Please add a new order or choose an existing order!");
            }
        }
```

```
        private void countAmountQuantity()
        {
            int quantity, price, total_amount = 0, total_quantity = 0;
            for (int rows = 0; rows < dataGridViewProductOrdering.Rows.Count; rows++)
            {
                if (dataGridViewProductOrdering.Rows[rows].Cells[3].Value != null &&
dataGridViewProductOrdering.Rows[rows].Cells[4].Value != null)
                {
                    if
(Int32.TryParse(dataGridViewProductOrdering.Rows[rows].Cells[3].Value.ToString(), out
price) && Int32.TryParse(dataGridViewProductOrdering.Rows[rows].Cells[4].Value.ToString(),
out quantity))
                    {
                        total_quantity += quantity;
                        total_amount += price * quantity;
                    }
                }
            }
            txtAmount.Text = total_amount.ToString();
            txtQuantity.Text = total_quantity.ToString();
        }
        public void reloadlistBoxPurchaseOrderOrderList()
        {
            listBoxPurchaseOrderOrderList.Items.Clear();
            try
            {
                connectingCsmsSuppDetails();
                string stm = "SELECT purchase_order_id FROM purchase_order";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();

                while (rdr.Read())
                {
                    listBoxPurchaseOrderOrderList.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        private void reloadComboType()
        {
            comboType.Items.Clear();
            connectingCsmsSuppDetails();
            String stm = "Select DISTINCT type from stock";
            try
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
```

```
                    comboType.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            { }
            finally
            {
                rdr.Close();
                con.Close();
            }
        }
        private void purchase_order_editable(bool choice)
        {
            comboType.Enabled = choice;
            comboCategory.Enabled = choice;
            comboBrand.Enabled = choice;
            comboModel.Enabled = choice;
            dateTimePickerOrderDate.Enabled = choice;
            dataGridViewProductOrdering.Enabled = choice;
        }
        private void clearDataGridViewProductOrdering()
        {
            dataGridViewProductOrdering.Rows.Clear();
        }
        private void clearAllFieldsPurchaseOrder()
        {
            txtOrderID.Text = "";
            txtDealerID.Text = "";
            txtDealerName.Text = "";
            txtDealerBillID.Text = "";
            dateTimePickerOrderDate.Value = DateTime.Today;
            comboType.SelectedIndex = -1;
            comboCategory.SelectedIndex = -1;
            comboBrand.SelectedIndex = -1;
            comboModel.SelectedIndex = -1;
            txtQuantity.Text = "";
            txtAmount.Text = "";
            txtPurchaseOrderPaidAmount.Text = "";
            txtPurchaseOrderPendingAmount.Text = "";
            clearDataGridViewProductOrdering();
        }
        private void orderIDGeneration()
        {
            String ordID = "", finalOrdID = "";
            int ordNo = 1;
            if (txtDealerID.Text != "")
            {
                connectingCsmsSuppDetails();
                try
                {
                    String stm = "SELECT purchase_order_id from purchase_order where
dealer_id = '" + txtDealerID.Text + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
                        rdr.Close();
                        rdr = cmd.ExecuteReader();
                        while (rdr.Read())
```

```
                        {
                            //MessageBox.Show("Passed while");
                            ordID = rdr.GetString(0);
                            int initialIndex = txtDealerID.Text.Length + 1;
                            //MessageBox.Show("ordId = "+ordID);
                            //MessageBox.Show("initialIndex = " + initialIndex);
                            string t = ordID.Substring(initialIndex, 4);
                            //MessageBox.Show("t = " + t);
                            ordNo = int.Parse(t) + 1;
                            //MessageBox.Show("ordNo = " + ordNo);
                        }
                        finalOrdID = txtDealerID.Text + "/" + ordNo.ToString("D4");
                    }
                    else
                    {
                        finalOrdID = txtDealerID.Text + "/0001";
                    }
                    txtOrderID.Text = finalOrdID;
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        public void connectingCsmsSuppDetails()
        {
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }
        private void btnSuppDetailsPrint_Click(object sender, EventArgs e)
        {
            PrintDetails pd = new PrintDetails();
            Test_GUI_for_Inventory_App.Validations.flagReport = "Supp";
            pd.pass_values_SuppID = txtSuppID.Text;
            pd.Show();
        }
        private void txtPurchaseOrderPaidAmount_KeyPress(object sender, KeyPressEventArgs
e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }
        private void txtPurchaseOrderPendingAmount_KeyPress(object sender,
KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }
    }
}
```

Customer_Billing.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Test_GUI_for_Inventory_App
{
    public partial class Customer_Billing : Form
    {
        public static String BillId;
        String MyConnectionString, stm;
        MySqlConnection con;
        MySqlCommand cmd;
        MySqlDataReader rdr;
        String flag;
        int searchLevel;

        private static Customer_Billing _instance;

        public Customer_Billing()
        {
            InitializeComponent();
        }
        private void Customer_Billing_Load(object sender, EventArgs e)
        {
            reloadComboType();
            reloadListBoxCustList();
            reloadListBoxBills();
            flag = "";
            searchLevel = 0;
        }
        public Customer_Billing instance
        {
            get
            {
                if (Customer_Billing._instance == null)
                {
                    Customer_Billing._instance = new Customer_Billing();
                }
                return Customer_Billing._instance;
            }
        }
        private void Customer_Billing_FormClosed(object sender, FormClosedEventArgs e)
        {
            Customer_Billing._instance = null;
        }
        private void reloadComboType()
        {
            comboBoxBillingOrderType.Items.Clear();
            connectingCust_Billing();
            String stm = "Select DISTINCT type from stock";
            try
```

```
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    comboBoxBillingOrderType.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            { }
            finally
            {
                comboBoxBillingOrderType.SelectedIndex = -1;
                comboBoxBillingOrderType.Text = "---Type---";
                comboBoxBillingOrderCategory.SelectedIndex = -1;
                comboBoxBillingOrderCategory.Text = "---Category---";
                comboBoxBillingOrderBrand.SelectedIndex = -1;
                comboBoxBillingOrderBrand.Text = "---Brand---";
                rdr.Close();
                con.Close();
            }
        }
        private void reloadListBoxCustList()
        {
            listBoxBillingCustList.Items.Clear();
            connectingCust_Billing();
            String stm = "SELECT cust_id from cust_details";
            try
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    listBoxBillingCustList.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            { }
            finally
            {
                rdr.Close();
                con.Close();
            }
        }
        public void connectingCust_Billing()
        {
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }
        private void comboBoxBillingOrderType_SelectedIndexChanged(object sender, EventArgs
e)
        {
            connectingCust_Billing();
            comboBoxBillingOrderCategory.Items.Clear();
            comboBoxBillingOrderBrand.Items.Clear();
            String stm = "Select DISTINCT category from stock where type = '" +
(String)comboBoxBillingOrderType.SelectedItem + "';";
            try
```

```
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    comboBoxBillingOrderCategory.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                rdr.Close();
                con.Close();
            }
        }
        private void comboBoxBillingOrderCategory_SelectedIndexChanged(object sender,
EventArgs e)
        {
            connectingCust_Billing();
            String stm = "Select DISTINCT brand from stock where type = '" +
(String)comboBoxBillingOrderType.SelectedItem + "' and category ='" +
comboBoxBillingOrderCategory.SelectedItem + "';";
            try
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    comboBoxBillingOrderBrand.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                rdr.Close();
                con.Close();
            }
        }
        private void comboBoxBillingOrderBrand_SelectedIndexChanged(object sender,
EventArgs e)
        {
            connectingCust_Billing();
            listBoxBillingOrderInventory.Items.Clear();
            String stm = "Select model_id from stock where type = '" +
(String)comboBoxBillingOrderType.SelectedItem + "' and category ='" +
comboBoxBillingOrderCategory.SelectedItem + "' and brand = '" +
comboBoxBillingOrderBrand.SelectedItem + "';";
            try
            {
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
```

```
                    listBoxBillingOrderInventory.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                rdr.Close();
                con.Close();
            }
        }
        private void listBoxBillingOrderInventory_DoubleClick(object sender, EventArgs e)
        {
            for(int i=0; i<dataGridView1.RowCount;i++)
            {
                if (dataGridView1.Rows[i].Cells[0].Value != null)
                {
                    if
(dataGridView1.Rows[i].Cells[0].Value.Equals(listBoxBillingOrderInventory.SelectedItem.ToSt
ring()))
                    {
                        return;
                    }
                }
            }
            DataGridViewRow row = (DataGridViewRow)dataGridView1.Rows[0].Clone();
            row.Cells[0].Value = listBoxBillingOrderInventory.SelectedItem.ToString();
            row.Cells[1].Value = comboBoxBillingOrderBrand.SelectedItem.ToString();
            row.Cells[2].Value = comboBoxBillingOrderCategory.SelectedItem;
            dataGridView1.Rows.Add(row);
        }
        private void listBoxBillingCustList_DoubleClick(object sender, EventArgs e)
        {
            if (listBoxBillingCustList.SelectedIndex > -1 && flag.Equals("add"))
            {
                emptyAll();
                fillCustomerDetails();
                orderIDGeneration();
                //dataGridView1.ReadOnly = true;
            }
        }
        private void emptyAll()
        {
            txtBillingCustName.Text = "";
            txtBillingShippingAddress.Text = "";
            txtCustMobile.Text = "";
            txtCustPhone.Text = "";
            dataGridView1.Rows.Clear();
            txtBillingTax.Clear();
            txtBillingSubTotal.Clear();
            txtBillingTotal.Clear();
        }
        private void fillCustomerDetails()
        {
            try
            {
                connectingCust_Billing();
```

```
                stm = "SELECT * from cust_details where cust_id = '" +
listBoxBillingCustList.SelectedItem.ToString() + "';";
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    txtCustID.Text = rdr.GetString(0);
                    txtBillingCustName.Text = rdr.GetString(1);
                    if (rdr.GetString(3) != null && rdr.GetString(3) != "")
                    {
                        txtBillingShippingAddress.Text = rdr.GetString(3);
                    }
                    else if (rdr.GetString(2) != null && rdr.GetString(2) != "")
                    {
                        txtBillingShippingAddress.Text = rdr.GetString(2);
                    }
                    else
                    {
                        MessageBox.Show("Shipping Address is empty.\nPlease first fill that
in Add Customer form.");
                    }
                    if (rdr.GetString(4) != null && rdr.GetString(4) != "")
                    {
                        txtCustPhone.Text = rdr.GetString(4);
                    }
                    else if (rdr.GetString(5) != null && rdr.GetString(5) != "")
                    {
                        txtCustPhone.Text = rdr.GetString(5);
                    }
                    if (rdr.GetString(6) != null && rdr.GetString(6) != "")
                    {
                        txtCustMobile.Text = rdr.GetString(6);
                    }
                    else if (rdr.GetString(7) != null && rdr.GetString(7) != "")
                    {
                        txtCustMobile.Text = rdr.GetString(7);
                    }
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        private void populatingForm()
        {
            try
            {
                emptyAll();
                connectingCust_Billing();
                int sub_total = 0;
                int quantity = 0;
```

```
                String stm = "SELECT cust_bill_id from cust_bill where sale_order_id='" +
txtOrderID.Text + "';";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    txtBillID.Text = rdr.GetString(0);
                }
                rdr.Close();
                stm = "SELECT a.amount,a.quantity,a.model_id,b.category,b.brand from
sale_item a,stock b where a.model_id=b.model_id and a.sale_order_id ='" + txtOrderID.Text +
"';";
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    DataGridViewRow row = (DataGridViewRow)dataGridView1.Rows[0].Clone();
                    row.Cells[0].Value = rdr.GetString(2);
                    row.Cells[1].Value = rdr.GetString(4);
                    row.Cells[2].Value = rdr.GetString(3);
                    row.Cells[3].Value = rdr.GetString(0);
                    row.Cells[4].Value = rdr.GetString(1);
                    row.Cells[5].Value = Int32.Parse(rdr.GetString(0)) *
Int32.Parse(rdr.GetString(1));
                    sub_total += Int32.Parse(rdr.GetString(0)) *
Int32.Parse(rdr.GetString(1));
                    dataGridView1.Rows.Add(row);
                    quantity += Int32.Parse(rdr.GetString(1));
                }
                rdr.Close();
                txtBillingQuantity.Text = quantity.ToString();
                stm = "SELECT total_amount from cust_bill where cust_bill_id = '" +
txtBillID.Text + "';";
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    txtBillingTotal.Text = rdr.GetString(0);
                }
                rdr.Close();
                stm = "SELECT cust_id,date from sale_order where sale_order_id = '" +
txtOrderID.Text + "';";
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                txtBillingSubTotal.Text = sub_total.ToString();
                while (rdr.Read())
                {
                    if (listBoxBillingCustList.Items.Contains(rdr.GetString(0)))
                    {
                        listBoxBillingCustList.SelectedIndex =
listBoxBillingCustList.Items.IndexOf(rdr.GetString(0));
                        dateTimePicker1.Value = rdr.GetDateTime(1);
                        fillCustomerDetails();
                        return;
                    }
                }
            }
            catch (Exception)
```

```
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    rdr.Close();
                    con.Close();
                }
                reloadListBoxPaymentsDone();
            }
        }
        private void orderIDGeneration()
        {
            String ordID = "", finalOrdID = "";
            int ordNo = 1;
            if (txtCustID.Text != "")
            {
                connectingCust_Billing();
                try
                {
                    String stm = "SELECT sale_order_id from sale_order where cust_id = '" +
txtCustID.Text + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
                        rdr.Close();
                        rdr = cmd.ExecuteReader();
                        while (rdr.Read())
                        {
                            ordID = rdr.GetString(0);
                            int initialIndex = txtCustID.Text.Length + 1;
                            string t = ordID.Substring(initialIndex, 4);
                            ordNo = int.Parse(t) + 1;
                        }
                        finalOrdID = txtCustID.Text + "/" + ordNo.ToString("D4");
                    }
                    else
                    {
                        finalOrdID = txtCustID.Text + "/0001";
                    }
                    txtOrderID.Text = finalOrdID;
                    txtBillID.Text = finalOrdID;
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
```

```
        private void dataGridView1_CellEndEdit(object sender, DataGridViewCellEventArgs e)
        {
            checkPriceQuantityTax();
            for (int rows = 0; rows < dataGridView1.Rows.Count; rows++)
            {
                if (dataGridView1.Rows.Count > 0 && dataGridView1.Rows[rows].Cells[3] !=
null && dataGridView1.Rows[rows].Cells[4] != null && dataGridView1.Rows[rows].Cells[0] !=
null)
                {
                    if (dataGridView1.Rows[rows].Cells[0].Value == null)
                    {
                        if (dataGridView1.Rows[rows].Cells[3].Value != null)
                        {
                            dataGridView1.Rows[rows].Cells[3].Value = "";
                        }
                        if (dataGridView1.Rows[rows].Cells[4].Value != null)
                        {
                            dataGridView1.Rows[rows].Cells[4].Value = "";
                        }
                    }
                }
            }
            countAmountQuantity();
        }
        private void countAmountQuantity()
        {
            int quantity, price, total_amount = 0, total_quantity = 0;
            for (int rows = 0; rows < dataGridView1.Rows.Count; rows++)
            {
                if (dataGridView1.Rows[rows].Cells[3].Value != null &&
dataGridView1.Rows[rows].Cells[4].Value != null)
                {
                    if (Int32.TryParse(dataGridView1.Rows[rows].Cells[3].Value.ToString(),
out price) && Int32.TryParse(dataGridView1.Rows[rows].Cells[4].Value.ToString(), out
quantity))
                    {
                        total_quantity += quantity;
                        total_amount += price * quantity;
                        dataGridView1.Rows[rows].Cells[5].Value = price * quantity;
                    }
                }
            }
            txtBillingSubTotal.Text = total_amount.ToString();
            txtBillingQuantity.Text = total_quantity.ToString();
        }
        private void txtBillingTax_TextChanged(object sender, EventArgs e)
        {
        }
        private void txtBillingTax_Leave(object sender, EventArgs e)
        {
            int x, y;
            if (!txtBillingTax.Equals("") && !txtBillingSubTotal.Equals(""))
            {
                if ((Int32.TryParse(txtBillingSubTotal.Text, out x)) &&
(Int32.TryParse(txtBillingTax.Text, out y)))
                {
                    txtBillingTotal.Text = (Int32.Parse(txtBillingSubTotal.Text) +
Int32.Parse(txtBillingTax.Text)).ToString();
```

```
                }
                else
                {
                    txtBillingTotal.Text = "";
                }
            }
            else
            {
                txtBillingTotal.Text = "";
            }
        }
        private void btnAddBill_Click(object sender, EventArgs e)
        {
            flag = "add";
            dataGridView1.Rows.Clear();
            comboBoxBillingOrderType.Enabled = true;
        }
        private void btnSave_Click(object sender, EventArgs e)
        {
            String modID;
            String orderID = txtOrderID.Text;
            String custID = txtCustID.Text;
            String custBillID = txtBillID.Text;
            int quantity = int.Parse(txtBillingQuantity.Text);
            String tax = txtBillingTax.Text;
            String totalAmount = txtBillingTotal.Text;
            int paidAmount = 0;
            DateTime orderDate = dateTimePicker1.Value.Date;

            connectingCust_Billing();

            //inserting into Purchase_Order table
            try
            {
                //for paid_amount calculation
                String stm = "SELECT paid_amount from payment_cust where cust_bill_id = '"
+ txtBillID.Text + "';";
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    paidAmount += rdr.GetInt32(0);
                }
                rdr.Close();
                stm = "select cust_id from cust_details where cust_id = '" + custID + "';";
                cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                if (rdr.Read())
                {
                    rdr.Close();
                    stm = "select sale_order_id from sale_order where sale_order_id = '" +
orderID + "';";
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read() && flag.Equals("add"))
                    {
                        MessageBox.Show("OrderID is not unique. Please choose unique
OrderId");
                        rdr.Close();
```

```
                    }
                    else
                    {
                        rdr.Close();
                        cmd = con.CreateCommand();
                        if (flag.Equals("add"))
                        {
                            cmd.CommandText = "insert into sale_order
values(@sale_order_id,@cust_id,@quantity,@date);";
                            cmd.Parameters.AddWithValue("@sale_order_id", orderID);
                            cmd.Parameters.AddWithValue("@cust_id", custID);
                            cmd.Parameters.AddWithValue("@quantity", quantity);
                            cmd.Parameters.AddWithValue("@date", orderDate);
                            cmd.ExecuteNonQuery();
                            cmd = con.CreateCommand();
                            cmd.CommandText = "insert into cust_bill
values(@cust_bill_id,@sale_order_id,@total_amount,@paid_amount);";
                            cmd.Parameters.AddWithValue("@cust_bill_id", custBillID);
                            cmd.Parameters.AddWithValue("@sale_order_id", orderID);
                            cmd.Parameters.AddWithValue("@total_amount", totalAmount);
                            cmd.Parameters.AddWithValue("@paid_amount", "0");
                            cmd.ExecuteNonQuery();
                        }
                        else if (flag.Equals("update"))
                        {
                            cmd.CommandText = "UPDATE sale_order SET
quantity=@quantity,date=@date WHERE sale_order_id=@sale_order_id;";
                            cmd.Parameters.AddWithValue("@sale_order_id", orderID);
                            cmd.Parameters.AddWithValue("@quantity", quantity);
                            cmd.Parameters.AddWithValue("@date", orderDate);
                            cmd.ExecuteNonQuery();
                            cmd = con.CreateCommand();
                            cmd.CommandText = "UPDATE cust_bill SET
total_amount=@total_amount, paid_amount=@paid_amount WHERE cust_bill_id=@cust_bill_id;";
                            cmd.Parameters.AddWithValue("@cust_bill_id", custBillID);
                            cmd.Parameters.AddWithValue("@sale_order_id", orderID);
                            cmd.Parameters.AddWithValue("@total_amount", totalAmount);
                            cmd.Parameters.AddWithValue("@paid_amount", paidAmount);
                            cmd.ExecuteNonQuery();
                        }
                        MessageBox.Show("Inserted into sale_order");

                        int r = dataGridView1.RowCount;

                        if (flag.Equals("update"))    //Decrementing stock previously order
                        {
                            for (int i = 1; i < r; i++)
                            {
                                if (dataGridView1.Rows[i - 1].Cells[0].Value != null)
                                {
                                    int t;
                                    t = int.Parse(dataGridView1.Rows[i -
1].Cells[4].Value.ToString());
                                    cmd = con.CreateCommand();
                                    cmd.CommandText = "UPDATE stock SET quantity=quantity-
@quantity where model_id=@model_id";
                                    cmd.Parameters.AddWithValue("@model_id",
dataGridView1.Rows[i - 1].Cells[0].Value.ToString());
```

```
                                    quantity = Int32.Parse(dataGridView1.Rows[i -
1].Cells[4].Value.ToString());
                                    cmd.Parameters.AddWithValue("@quantity", quantity);
                                    cmd.ExecuteNonQuery();
                                    //Removing from purchase_item
                                    cmd = con.CreateCommand();
                                    cmd.CommandText = "DELETE from sale_item where
sale_order_id=@sale_order_id";
                                    cmd.Parameters.AddWithValue("@sale_order_id",
txtOrderID.Text);
                                    cmd.ExecuteNonQuery();
                                }
                            }
                        }

                        for (int i = 1; i < r; i++)
                        {
                            //MessageBox.Show("i = " + i);
                            if (dataGridView1.Rows[i - 1].Cells[0].Value != null)
                            {
                                //MessageBox.Show("Model ID is not null");
                                stm = "select model_id,quantity from stock where model_id =
'" + dataGridView1.Rows[i - 1].Cells[0].Value.ToString() + "';";
                                cmd = new MySqlCommand(stm, con);
                                rdr = cmd.ExecuteReader();
                                if (rdr.Read())
                                {
                                    modID = rdr.GetString(0);
                                    rdr.Close();
                                    //MessageBox.Show("Model ID present!");
                                    cmd = con.CreateCommand();
                                    cmd.CommandText = "INSERT into sale_item
(sale_order_id,amount,model_id,quantity)
values(@sale_order_id,@amount,@model_id,@quantity)";
                                    cmd.Parameters.AddWithValue("@sale_order_id",
txtOrderID.Text);
                                    MessageBox.Show("Amount : " + dataGridView1.Rows[i -
1].Cells[3].Value.ToString());
                                    cmd.Parameters.AddWithValue("@amount",
dataGridView1.Rows[i - 1].Cells[3].Value.ToString());
                                    cmd.Parameters.AddWithValue("@model_id",
dataGridView1.Rows[i - 1].Cells[0].Value.ToString());
                                    quantity = Int32.Parse(dataGridView1.Rows[i -
1].Cells[4].Value.ToString());
                                    cmd.Parameters.AddWithValue("@quantity", quantity);
                                    cmd.ExecuteNonQuery();
                                    for (int j = 0; j < quantity; j++)
                                    {
                                        cmd = con.CreateCommand();
                                        cmd.CommandText = "UPDATE stock set
quantity=quantity+1 where model_id=@model_id";
                                        cmd.Parameters.AddWithValue("@model_id", modID);
                                        cmd.ExecuteNonQuery();
                                        cmd.CommandText = "";
                                    }
                                    MessageBox.Show("Updated Stock");
                                }
                                //MessageBox.Show("Exited 1st if");
                            }
```

```
                            //MessageBox.Show("Exited 2nd if");
                        }
                        //MessageBox.Show("Exited for");
                    }
                }
                else
                {
                    MessageBox.Show("Error: Customer not present!!");
                    rdr.Close();
                }
            }
            catch
            {
                throw;
            }
            finally
            {
                con.Close();
                flag = "";
            }
        }
        private void btnUpdateBill_Click(object sender, EventArgs e)
        {
            flag = "update";
        }
        private void btnDeleteBill_Click(object sender, EventArgs e)
        {
            if (!txtOrderID.Text.Equals(""))
            {
                try
                {
                    connectingCust_Billing();
                    int r = dataGridView1.RowCount;
                    for (int i = 1; i < r; i++)
                    {
                        if (dataGridView1.Rows[i - 1].Cells[0].Value.ToString() != null)
                        {
                            int t;
                            t = int.Parse(dataGridView1.Rows[i -
1].Cells[4].Value.ToString());
                            cmd = con.CreateCommand();
                            cmd.CommandText = "UPDATE stock SET quantity=quantity-@quantity
where model_id=@model_id";
                            cmd.Parameters.AddWithValue("@model_id", dataGridView1.Rows[i -
1].Cells[0].Value.ToString());
                            int quantity = Int32.Parse(dataGridView1.Rows[i -
1].Cells[4].Value.ToString());
                            cmd.Parameters.AddWithValue("@quantity", quantity);
                            cmd.ExecuteNonQuery();
                            //Removing from sale_item
                            cmd = con.CreateCommand();
                            cmd.CommandText = "DELETE from sale_item where
sale_order_id=@sale_order_id";
                            cmd.Parameters.AddWithValue("@sale_order_id", txtOrderID.Text);
                            cmd.ExecuteNonQuery();
                        }
                    }
                    cmd = con.CreateCommand();
                    cmd.CommandText = "DELETE from cust_bill where sale_order_id=@s_o_i";
```

```
            cmd.Parameters.AddWithValue("@s_o_i", txtOrderID.Text);
            cmd.ExecuteNonQuery();
            cmd = con.CreateCommand();
            cmd.CommandText = "DELETE from sale_order where sale_order_id=@s_o_i";
            cmd.Parameters.AddWithValue("@s_o_i", txtOrderID.Text);
            cmd.ExecuteNonQuery();
        }
        catch (Exception)
        {
            throw;
        }
        finally
        {
            if (con.State == ConnectionState.Open)
            {
                con.Close();
                emptyAll();
                reloadListBoxBills();
                reloadListBoxCustList();
            }
        }
    }
}
private void reloadListBoxBills()
{
    listBoxBills.Items.Clear();
    try
    {
        connectingCust_Billing();
        stm = "SELECT sale_order_id from sale_order";
        MySqlCommand cmd = new MySqlCommand(stm, con);
        rdr = cmd.ExecuteReader();
        while (rdr.Read())
        {
            listBoxBills.Items.Add(rdr.GetString(0));
        }
    }
    catch (Exception)
    {
        throw;
    }
    finally
    {
        if (con.State == ConnectionState.Open)
        {
            con.Close();
        }
    }
}
private void listBoxBills_DoubleClick(object sender, EventArgs e)
{
    if (listBoxBills.SelectedIndex > -1)
    {
        flag = "";
        emptyAll();
        txtOrderID.Text = listBoxBills.SelectedItem.ToString();
        populatingForm();
    }
}
```

```
        private void btnPay_Click(object sender, EventArgs e)
        {
            if (!txtOrderID.Equals(""))
            {
                try
                {
                    connectingCust_Billing();
                    stm = "SELECT cust_bill_id from cust_bill where cust_bill_id='" +
txtBillID.Text + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        BillId = rdr.GetString(0);
                    }
                    rdr.Close();
                    PaymentCustomer pc = new PaymentCustomer();
                    pc.Show();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void reloadListBoxPaymentsDone()
        {
            listBoxPaymentsDone.Items.Clear();
            try
            {
                connectingCust_Billing();
                String stm2 = "SELECT cust_invoice_no from payment_cust where
cust_bill_id='" + txtBillID.Text + "';";
                MySqlCommand cmd = new MySqlCommand(stm2, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    listBoxPaymentsDone.Items.Add(rdr.GetString(0));
                }
                rdr.Close();
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
```

```
            }
        }
        private void btnBillSearch_Click(object sender, EventArgs e)
        {
            listBoxBillSearchResults.Items.Clear();
            searchLevel = 0;
            if (comboBoxBillSearchBy.SelectedIndex > -1)
            {
                string stm = "";
                if (comboBoxBillSearchBy.SelectedItem.ToString() == "Order ID")
                {
                    stm = "SELECT sale_order_id FROM sale_order where sale_order_id like
'%" + txtBillSearchKeywords.Text + "%'";
                    listBoxBillSearchResults.Items.Add("Order ID's:");
                }
                else if (comboBoxBillSearchBy.SelectedItem.ToString() == "Bill ID")
                {
                    stm = "SELECT cust_bill_id FROM cust_bill where cust_bill_id like '%" +
txtBillSearchKeywords.Text + "%'";
                    listBoxBillSearchResults.Items.Add("Bill ID's:");
                }
                else if (comboBoxBillSearchBy.SelectedItem.ToString() == "Customer ID")
                {
                    stm = "SELECT cust_id FROM sale_order where cust_id like '%" +
txtBillSearchKeywords.Text + "%'";
                    listBoxBillSearchResults.Items.Add("Based on Customer ID:");
                }
                else if (comboBoxBillSearchBy.SelectedItem.ToString() == "Customer Name")
                {
                    stm = "SELECT a.name FROM cust_details a, sale_order b where
a.cust_id=b.cust_id and a.name like '%" + txtBillSearchKeywords.Text + "%'";
                    listBoxBillSearchResults.Items.Add("Based on Customer Name:");
                }
                else
                {
                    MessageBox.Show("Select something valid from the drop down list");
                }
                try
                {
                    connectingCust_Billing();
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        listBoxBillSearchResults.Items.Add(rdr.GetString(0));
                    }
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
```

```
        }
        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void txtBillingTax_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }
        private void btnPrintPreview_Click(object sender, EventArgs e)
        {
            BillprintPreviewDialog.Document = BillprintDocument;
            BillprintPreviewDialog.ShowDialog();

        }
        private void checkPriceQuantityTax()
        {
            int check;
            if(txtBillingTax.Text!="")
            {
                if(!Int32.TryParse(txtBillingTax.Text,out check))
                {
                    MessageBox.Show("Tax must be integer");
                    txtBillingTax.Text = "";
                }
            }
            int r = dataGridView1.RowCount;
            for (int i = 1; i < r; i++)
            {
                if (dataGridView1.Rows[i - 1].Cells[0].Value != null)
                {
                    if (dataGridView1.Rows[i - 1].Cells[3].Value != null &&
dataGridView1.Rows[i - 1].Cells[4].Value != null)
                    {
                        if (!Int32.TryParse(dataGridView1.Rows[i -
1].Cells[3].Value.ToString(), out check))
                        {
                            MessageBox.Show("Price must be integer");
                            dataGridView1.Rows[i - 1].Cells[3].Value = "";
                        }
                        if (!Int32.TryParse(dataGridView1.Rows[i -
1].Cells[4].Value.ToString(), out check))
                        {
                            MessageBox.Show("Quantity must be integer");
                            dataGridView1.Rows[i - 1].Cells[4].Value = "";
                        }
                    }
                }
            }
        }
        private void listBoxBillSearchResults_DoubleClick(object sender, EventArgs e)
        {
            if (searchLevel == 0)
            {
                if (listBoxBillSearchResults.SelectedIndex > 0)
                {
                    string stm = "";
                    if (comboBoxBillSearchBy.SelectedItem.ToString() == "Order ID")
                    {
```

```
                    txtOrderID.Text = listBoxBillSearchResults.SelectedItem.ToString();
                    populatingForm();
                }
                else if (comboBoxBillSearchBy.SelectedItem.ToString() == "Bill ID")
                {
                    stm = "SELECT sale_order_id FROM cust_bill where cust_bill_id = '"
+ listBoxBillSearchResults.SelectedItem.ToString() + "'";
                    try
                    {
                        connectingCust_Billing();
                        MySqlCommand cmd = new MySqlCommand(stm, con);
                        rdr = cmd.ExecuteReader();
                        while (rdr.Read())
                        {
                            txtOrderID.Text = rdr.GetString(0);
                        }
                    }
                    catch (Exception)
                    {
                        throw;
                    }
                    populatingForm();
                }
                else if (comboBoxBillSearchBy.SelectedItem.ToString() == "Customer ID")
                {
                    stm = "SELECT sale_order_id, cust_id FROM sale_order where cust_id
= '" + listBoxBillSearchResults.SelectedItem.ToString() + "'";
                    try
                    {
                        connectingCust_Billing();
                        MySqlCommand cmd = new MySqlCommand(stm, con);
                        listBoxBillSearchResults.Items.Clear();
                        listBoxBillSearchResults.Items.Add("Based on Customer ID:");
                        rdr = cmd.ExecuteReader();
                        if (rdr.Read())
                        {
                            listBoxBillSearchResults.Items.Add("Orders from " +
rdr.GetString(1) + ":");
                            rdr.Close();
                            rdr = cmd.ExecuteReader();
                            while (rdr.Read())
                            {
                                listBoxBillSearchResults.Items.Add(rdr.GetString(0));
                            }
                        }
                    }
                    catch (Exception)
                    {
                        throw;
                    }
                    searchLevel = 1;
                }
                else if (comboBoxBillSearchBy.SelectedItem.ToString() == "Customer
Name")
                {
                    stm = "SELECT a.sale_order_id, b.name FROM sale_order a,
cust_details b where a.cust_id=b.cust_id and b.name = '" +
listBoxBillSearchResults.SelectedItem.ToString() + "'";
                    try
```

```
                        {
                            connectingCust_Billing();
                            MySqlCommand cmd = new MySqlCommand(stm, con);
                            listBoxBillSearchResults.Items.Clear();
                            listBoxBillSearchResults.Items.Add("Based on Customer Name:");
                            rdr = cmd.ExecuteReader();
                            if (rdr.Read())
                            {
                                listBoxBillSearchResults.Items.Add("Orders from " +
rdr.GetString(1) + ":");
                                rdr.Close();
                                rdr = cmd.ExecuteReader();
                                while (rdr.Read())
                                {
                                    listBoxBillSearchResults.Items.Add(rdr.GetString(0));
                                }
                            }
                        }
                        catch (Exception)
                        {
                            throw;
                        }
                        searchLevel = 1;
                    }
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
            else
            {
                if (comboBoxBillSearchBy.SelectedItem.ToString() == "Customer ID")
                {
                    if (listBoxBillSearchResults.SelectedIndex > 1)
                    {
                        txtOrderID.Text = listBoxBillSearchResults.SelectedItem.ToString();
                        populatingForm();
                    }
                }
                else if (comboBoxBillSearchBy.SelectedItem.ToString() == "Customer Name")
                {
                    if (listBoxBillSearchResults.SelectedIndex > 1)
                    {
                        txtOrderID.Text = listBoxBillSearchResults.SelectedItem.ToString();
                        populatingForm();
                    }
                }
            }
        }
        private void btnPrint_Click(object sender, EventArgs e)
        {
            PrintDetails pd = new PrintDetails();
            Test_GUI_for_Inventory_App.Validations.flagReport = "Bill";
            pd.pass_values_OrderID = txtOrderID.Text;
            pd.Show();
        }
    }
}
```

Customer.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Test_GUI_for_Inventory_App
{
    public partial class Customer : Form
    {
        String MyConnectionString;
        String cust_id, name, phone1, phone2, mob1, mob2, email1, email2, address1,
address2, remarks;
        Char gender;
        DateTime dob;
        MySqlConnection con;
        MySqlCommand cmd;
        MySqlDataReader rdr;
        String flag;

        public string flagReport = "2";

        private static Customer _instance;

        Payroll P_Roll = new Payroll();

        public Customer()
        {
            InitializeComponent();
        }
        private void Customer_Load(object sender, EventArgs e)
        {
            reloadListBoxCustomerList();
            custdetails_editable(false);
        }
        public Customer instance
        {
            get
            {
                if (Customer._instance == null)
                {
                    Customer._instance = new Customer();
                }
                return Customer._instance;
            }
        }
        private void Customer_FormClosed(object sender, FormClosedEventArgs e)
        {
            Customer._instance = null;
        }
        /*
```

```
        *
        * Customer Details Tab Coding
        *
        *
        */
        private void listBoxCustomerList_SelectedIndexChanged(object sender, EventArgs e)
        {
            custdetails_editable(false);
            if (listBoxCustomerList.SelectedIndex > -1)
            {
                String selectedValue = listBoxCustomerList.SelectedItem.ToString();
                connectingCsmsCustDetails();
                try
                {
                    string stm = "SELECT * FROM cust_Details where name='" + selectedValue
+ "'";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    populatingForm();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void btnAddCust_Click(object sender, EventArgs e)
        {
            btnCustDetailsSave.Enabled = true;
            clearAllFields();
            custdetails_editable(true);
            flag = "Add";

            btnCustDetailsCancel.BringToFront();
            ctrls_enabled(true);
            custIDGeneration();
        }
        private void custIDGeneration()
        {
            String custID = "", finalCustID = "";
            int CustNo = 1;
            connectingCsmsCustDetails();
            try
            {
                String stm = "SELECT cust_id from cust_details;";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                if (rdr.Read())
                {
                    rdr.Close();
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
```

```
                    {
                        custID = rdr.GetString(0);
                        string t = custID.Substring(2, 4);
                        CustNo = int.Parse(t) + 1;
                    }
                    finalCustID = "C" + "/" + CustNo.ToString("D4");
                }
                else
                {
                    finalCustID = "C" + "/0001";
                }
                txtCustID.Text = finalCustID;
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        private void btnUpdateCustDetails_Click(object sender, EventArgs e)
        {
            if (listBoxCustomerList.SelectedIndex > -1 ||
listBoxCustDetailsSearchResults.SelectedIndex > 0)
            {
                btnCustDetailsSave.Enabled = true;
                custdetails_editable(true);
                flag = "Update";

                btnCustDetailsCancel.BringToFront();
                ctrls_enabled(true);
            }
        }
        private void btnRemoveCust_Click(object sender, EventArgs e)
        {
            if (txtCustID.Text != null) //((listBoxCustomerList.SelectedIndex > -1 ||
listBoxCustDetailsSearchResults.SelectedIndex > 0) && (txtCustID!=null))
            {
                try
                {
                    connectingCsmsCustDetails();
                    String stm = "SELECT cust_id,name FROM cust_Details where cust_id = '"
+ txtCustID.Text + "'";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
                        if (MessageBox.Show("Do you want to delete Customer with name " +
rdr.GetString(1) + "?", "Delete Customer?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
                        {
                            cmd = con.CreateCommand();
                            cmd.CommandText = "delete from cust_details where cust_id=" +
rdr.GetString(0) + ";";
```

```
                            rdr.Close();
                            cmd.ExecuteNonQuery();
                        }
                    }
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                    reloadListBoxCustomerList();
                    clearAllFields();
                }
            }
        }
        private void btnCustDetailsSave_Click(object sender, EventArgs e)
        {
            //Extracting data from fields
            cust_id = txtCustID.Text;
            name = txtCustName.Text;
            if (comboBoxCustGender.Text == "Male")
                gender = 'M';
            else if (comboBoxCustGender.Text == "Female")
                gender = 'F';
            else
                gender = ' ';
            phone1 = txtCustPhone1.Text;
            phone2 = txtCustPhone2.Text;
            mob1 = txtCustMobile1.Text;
            mob2 = txtCustMobile2.Text;
            email1 = txtCustEmail1.Text;
            email2 = txtCustEmail2.Text;
            address1 = txtCustAddress.Text;
            address2 = txtCustShippingAddress.Text;
            dob = dateTimePickerCustDOB.Value.Date;
            remarks = txtCustRemarks.Text;

            //Inserting data
            connectingCsmsCustDetails();
            try
            {
                cmd = con.CreateCommand();
                if (flag.Equals("Add"))
                {
                    cmd.CommandText = "insert into cust_details
values(@cust_id,@name,@add1,@add2,@ph1,@ph2,@mob1,@mob2,@gender,@email1,@email2,@dob,@remar
ks)";
                }
                else if (flag.Equals("Update"))
                {
                    cmd.CommandText = "UPDATE cust_Details SET
name=@name,address1=@add1,address2=@add2,phone1=@ph1,phone2=@ph2,mob1=@mob1,mob2=@mob2,gend
er=@gender,email1=@email1,email2=@email2,dob=@dob,remarks=@remarks WHERE cust_id =
@cust_id";
```

```
                }
                cmd.Parameters.AddWithValue("@cust_id", cust_id);
                cmd.Parameters.AddWithValue("@name", name);
                cmd.Parameters.AddWithValue("@add1", address1);
                cmd.Parameters.AddWithValue("@add2", address2);
                cmd.Parameters.AddWithValue("@ph1", phone1);
                cmd.Parameters.AddWithValue("@ph2", phone2);
                cmd.Parameters.AddWithValue("@mob1", mob1);
                cmd.Parameters.AddWithValue("@mob2", mob2);
                cmd.Parameters.AddWithValue("@gender", gender);
                cmd.Parameters.AddWithValue("@email1", email1);
                cmd.Parameters.AddWithValue("@email2", email2);
                cmd.Parameters.AddWithValue("@dob", dob);
                cmd.Parameters.AddWithValue("@remarks", remarks);
                cmd.ExecuteNonQuery();
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
                reloadListBoxCustomerList();
            }
            custdetails_editable(false);
            btnCustDetailsSave.Enabled = false;

            ctrls_enabled(false);
            btnCustDetailsCancel.SendToBack();
        }
        private void btnCustDetailsClear_Click(object sender, EventArgs e)
        {
            clearAllFields();
        }
        private void btnCustDetailsCancel_Click(object sender, EventArgs e)
        {
            clearAllFields();
            custdetails_editable(false);
            ctrls_enabled(false);
            btnCustDetailsCancel.SendToBack();
        }
        private void btnCustDetailsClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void btnCustDetailsSearch_Click(object sender, EventArgs e)
        {
            listBoxCustDetailsSearchResults.Items.Clear();
            if (comboBoxCustDetailsSearchBy.SelectedIndex > -1)
            {
                string stm;
                if (comboBoxCustDetailsSearchBy.SelectedItem.ToString() == "ID")
                {
                    stm = "SELECT cust_id FROM cust_Details where cust_id like '%" +
txtCustDetailsSearchKeywords.Text + "%'";
```

```
                    listBoxCustDetailsSearchResults.Items.Add("Based on ID:");
                }
                else if (comboBoxCustDetailsSearchBy.SelectedItem.ToString() == "Name")
                {
                    stm = "SELECT name FROM cust_Details where name like '%" +
txtCustDetailsSearchKeywords.Text + "%'";
                    listBoxCustDetailsSearchResults.Items.Add("Based on Name:");
                }
                else if (comboBoxCustDetailsSearchBy.SelectedItem.ToString() == "Phone 1")
                {
                    stm = "SELECT phone1 FROM cust_Details where phone1 like '%" +
txtCustDetailsSearchKeywords.Text + "%'";
                    listBoxCustDetailsSearchResults.Items.Add("Based on Phone 1:");
                }
                else if (comboBoxCustDetailsSearchBy.SelectedItem.ToString() == "Phone 2")
                {
                    stm = "SELECT phone2 FROM cust_Details where phone2 like '%" +
txtCustDetailsSearchKeywords.Text + "%'";
                    listBoxCustDetailsSearchResults.Items.Add("Based on Phone 2:");
                }
                else if (comboBoxCustDetailsSearchBy.SelectedItem.ToString() == "Mobile 1")
                {
                    stm = "SELECT mob1 FROM cust_Details where mob1 like '%" +
txtCustDetailsSearchKeywords.Text + "%'";
                    listBoxCustDetailsSearchResults.Items.Add("Based on Mobile 1:");
                }
                else if (comboBoxCustDetailsSearchBy.SelectedItem.ToString() == "Mobile 2")
                {
                    stm = "SELECT mob2 FROM cust_Details where mob2 like '%" +
txtCustDetailsSearchKeywords.Text + "%'";
                    listBoxCustDetailsSearchResults.Items.Add("Based on Mobile 2:");
                }
                else if (comboBoxCustDetailsSearchBy.SelectedItem.ToString() == "Email 1")
                {
                    stm = "SELECT email1 FROM cust_Details where email1 like '%" +
txtCustDetailsSearchKeywords.Text + "%'";
                    listBoxCustDetailsSearchResults.Items.Add("Based on Email 1:");
                }
                else // if (comboBoxCustDetailsSearchBy.SelectedItem == "Email 2")
                {
                    stm = "SELECT email2 FROM cust_Details where email2 like '%" +
txtCustDetailsSearchKeywords.Text + "%'";
                    listBoxCustDetailsSearchResults.Items.Add("Based on Email 2:");
                }
                try
                {
                    connectingCsmsCustDetails();
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        listBoxCustDetailsSearchResults.Items.Add(rdr.GetString(0));
                    }
                }
                catch (Exception)
                {
                    throw;
                }
                finally
```

```
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void listBoxCustDetailsSearchResults_SelectedIndexChanged(object sender,
EventArgs e)
        {
            custdetails_editable(false);
            if (listBoxCustDetailsSearchResults.SelectedIndex > 0)
            {
                String selectedValue =
listBoxCustDetailsSearchResults.SelectedItem.ToString();
                String stm;
                connectingCsmsCustDetails();
                if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on ID:")
                {
                    stm = "SELECT * FROM cust_Details where cust_id = '" + selectedValue +
"'";
                }
                else if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on
Name:")
                {
                    stm = "SELECT * FROM cust_Details where name = '" + selectedValue +
"'";
                }
                else if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on
Phone 1:")
                {
                    stm = "SELECT * FROM cust_Details where phone1 = '" + selectedValue +
"'";
                }
                else if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on
Phone 2:")
                {
                    stm = "SELECT * FROM cust_Details where phone2 = '" + selectedValue +
"'";
                }
                else if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on
Mobile 1:")
                {
                    stm = "SELECT * FROM cust_Details where mob1 = '" + selectedValue +
"'";
                }
                else if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on
Mobile 2:")
                {
                    stm = "SELECT * FROM cust_Details where mob2 = '" + selectedValue +
"'";
                }
                else if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on
Email 1:")
                {
                    stm = "SELECT * FROM cust_Details where email1 = '" + selectedValue +
"'";
                }
```

```
                else //if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on
Email 2:")
                {
                    stm = "SELECT * FROM cust_Details where email2 = '" + selectedValue +
"'";
                }
                try
                {
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    populatingForm();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        public void reloadListBoxCustomerList()
        {
            listBoxCustomerList.Items.Clear();
            try
            {
                connectingCsmsCustDetails();
                string stm = "SELECT name FROM cust_Details";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();

                while (rdr.Read())
                {
                    listBoxCustomerList.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        private void populatingForm()
        {
            while (rdr.Read())
            {
                txtCustID.Text = rdr.GetString(0);
                txtCustName.Text = rdr.GetString(1);
                txtCustAddress.Text = rdr.GetString(2);
```

```
            txtCustShippingAddress.Text = rdr.GetString(3);
            txtCustPhone1.Text = rdr.GetString(4);
            txtCustPhone2.Text = rdr.GetString(5);
            txtCustMobile1.Text = rdr.GetString(6);
            txtCustMobile2.Text = rdr.GetString(7);
            if (rdr.GetChar(8) == 'M')
                comboBoxCustGender.Text = "Male";
            else if (rdr.GetChar(8) == 'F')
                comboBoxCustGender.Text = "Female";
            else
                comboBoxCustGender.Text = "";
            txtCustEmail1.Text = rdr.GetString(9);
            txtCustEmail2.Text = rdr.GetString(10);
            dateTimePickerCustDOB.Value = rdr.GetDateTime(11);
            txtCustRemarks.Text = rdr.GetString(12);
        }
    }
    private void custdetails_editable(bool choice)
    {
        choice = !choice;
        txtCustName.ReadOnly = choice;
        comboBoxCustGender.Enabled = !choice;
        txtCustPhone1.ReadOnly = choice;
        txtCustPhone2.ReadOnly = choice;
        txtCustMobile1.ReadOnly = choice;
        txtCustMobile2.ReadOnly = choice;
        txtCustEmail1.ReadOnly = choice;
        txtCustEmail2.ReadOnly = choice;
        txtCustAddress.ReadOnly = choice;
        txtCustShippingAddress.ReadOnly = choice;
        dateTimePickerCustDOB.Enabled = !choice;
        txtCustRemarks.ReadOnly = choice;
    }
    private void clearAllFields()
    {
        txtCustID.Text = "";
        txtCustName.Text = "";
        txtCustAddress.Text = "";
        txtCustShippingAddress.Text = "";
        txtCustPhone1.Text = "";
        txtCustPhone2.Text = "";
        txtCustMobile1.Text = "";
        txtCustMobile2.Text = "";
        comboBoxCustGender.SelectedIndex = 0;
        txtCustEmail1.Text = "";
        txtCustEmail2.Text = "";
        dateTimePickerCustDOB.Value = DateTime.Now;
        txtCustRemarks.Text = "";
    }
    private void ctrls_enabled(bool choice)
    {
        tableLayoutPanelCustDisplay.Enabled = !choice;
        tableLayoutPanelCustSearchCtrls.Enabled = !choice;

        btnCustDetailsCancel.Enabled = choice;
        btnCustDetailsClose.Enabled = !choice;
        btnCustDetailsClear.Enabled = choice;
        btnCustDetailsSave.Enabled = choice;
    }
```

```
        private void txtCustName_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyTextAllowed(e);
        }

        private void txtCustPhone1_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }

        private void txtCustPhone2_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }

        private void txtCustMobile1_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }

        private void txtCustMobile2_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }

        private void txtCustEmail1_Leave(object sender, EventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.ValidateEmail(e, txtCustEmail1);
        }

        private void txtCustEmail2_Leave(object sender, EventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.ValidateEmail(e, txtCustEmail2);
        }

        public void connectingCsmsCustDetails()
        {
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            PrintDetails pd = new PrintDetails();
            Test_GUI_for_Inventory_App.Validations.flagReport = "Cust";
            pd.pass_values_CustID = txtCustID.Text;
            pd.Show();
        }
    }
}
```

Payroll.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Text.RegularExpressions;
using MySql.Data.MySqlClient;
using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.ReportSource;
using System.IO;
using System.Drawing.Imaging;

namespace Test_GUI_for_Inventory_App
{


    public partial class Payroll : Form
    {
        string MyConnectionString;
        String emp_id, name, phone1, phone2, mob1, mob2, email1, email2, address1,
address2, designation, remarks, basic_salary;
        Char gender;
        DateTime dob, hiredate;
        MySqlConnection con;
        MySqlCommand cmd;
        MySqlDataReader rdr;
        String flag;

        public string flagReport = "1";

        private static Payroll _instance;

        public Payroll()
        {
            InitializeComponent();
        }
        private void Payroll_Load(object sender, EventArgs e)
        {
            reloadListBoxEmployeeList();
            empdetails_editable(false);
            dataGridViewAttendanceShow.Rows.Add();
        }
        public Payroll instance
        {
            get
            {
                if (Payroll._instance == null)
                {
                    Payroll._instance = new Payroll();
                }
                return Payroll._instance;
            }
        }
```

```
        private void Payroll_FormClosed(object sender, FormClosedEventArgs e)
        {
            Payroll._instance = null;
        }
        /*
         *
         * Employee Details Tab Coding
         *
         *
         */
        private void listBoxEmployeeList_SelectedIndexChanged_1(object sender, EventArgs e)
        {
            empdetails_editable(false);
            if (listBoxEmployeeList.SelectedIndex > -1)
            {
                String selectedValue = listBoxEmployeeList.SelectedItem.ToString();
                connectingCsmsEmpDetails();
                try
                {
                    string stm = "SELECT * FROM employee_details where name='" +
selectedValue + "'";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    populatingForm();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void btnAddEmp_Click(object sender, EventArgs e)
        {
            btnEmpDetailsSave.Enabled = true;
            clearAllFields();
            empdetails_editable(true);
            flag = "Add";

            btnEmpDetailsCancel.BringToFront();
            ctrls_enabled(true);
            suppIDGeneration();
        }
        private void suppIDGeneration()
        {
            String empID = "", finalEmpID = "";
            int empNo = 1;
            connectingCsmsEmpDetails();
            try
            {
                String stm = "SELECT emp_id from employee_details;";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
```

```
                if (rdr.Read())
                {
                    rdr.Close();
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        empID = rdr.GetString(0);
                        string t = empID.Substring(2, 4);
                        empNo = int.Parse(t) + 1;
                    }
                    finalEmpID = "E" + "/" + empNo.ToString("D4");
                }
                else
                {
                    finalEmpID = "E" + "/0001";
                }
                txtEmpID.Text = finalEmpID;
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
            }
        }
        private void btnUpdateEmpDetails_Click(object sender, EventArgs e)
        {
            if (listBoxEmployeeList.SelectedIndex > -1 ||
listBoxEmpDetailsSearchResults.SelectedIndex > 0)
            {
                btnEmpDetailsSave.Enabled = true;
                empdetails_editable(true);
                flag = "Update";

                btnEmpDetailsCancel.BringToFront();
                ctrls_enabled(true);
            }
        }
        private void btnRemoveEmpDetails_Click(object sender, EventArgs e)
        {
            if (txtEmpID.Text != null) //((listBoxCustomerList.SelectedIndex > -1 ||
listBoxCustDetailsSearchResults.SelectedIndex > 0) && (txtCustID!=null))
            {
                try
                {
                    connectingCsmsEmpDetails();
                    String stm = "SELECT emp_id,name FROM employee_Details where emp_id =
'" + txtEmpID.Text + "'";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
```

```
                        if (MessageBox.Show("Do you want to delete Employee with name " +
rdr.GetString(1) + "?", "Delete Employee?", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
                        {
                            cmd = con.CreateCommand();
                            cmd.CommandText = "delete from employee_details where emp_id='"
+ rdr.GetString(0) + "';";
                            rdr.Close();
                            cmd.ExecuteNonQuery();
                        }
                    }
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                    reloadListBoxEmployeeList();
                    clearAllFields();
                }
            }
        }
        private void btnEmpDetailsSave_Click(object sender, EventArgs e)
        {
            if (dateTimePickerEmpDOB.Value.Date >=
dateTimePickerEmpDateOfJoining.Value.Date || dateTimePickerEmpDOB.Value.Date ==
dateTimePickerEmpDateOfJoining.Value.Date ||
dateTimePickerEmpDOB.Value.Date>=System.DateTime.Now ||
dateTimePickerEmpDateOfJoining.Value.Date>=System.DateTime.Now)
            {
                MessageBox.Show("Please verify the dates! Date of Birth cannot be after
Date of Joining or System Date or Vice-Versa.", "Error");
            }
            else
            {
                //Extracting data from fields
                emp_id = txtEmpID.Text;
                name = txtEmpName.Text;
                if (comboBoxEmpGender.Text == "Male")
                    gender = 'M';
                else if (comboBoxEmpGender.Text == "Female")
                    gender = 'F';
                else
                    gender = ' ';
                phone1 = txtEmpPhone1.Text;
                phone2 = txtEmpPhone2.Text;
                mob1 = txtEmpMobile1.Text;
                mob2 = txtEmpMobile2.Text;
                email1 = txtEmpEmail1.Text;
                email2 = txtEmpEmail2.Text;
                address1 = txtEmpResidenceAddress.Text;
                address2 = txtEmpPermanentAddress.Text;
                dob = dateTimePickerEmpDOB.Value.Date;
                hiredate = dateTimePickerEmpDateOfJoining.Value.Date;
```

```
                designation = txtEmpDesignation.Text;
                remarks = txtEmpRemarks.Text;
                basic_salary = txtEmpBasicSalary.Text;

                //Inserting data
                connectingCsmsEmpDetails();
                try
                {
                    MemoryStream memoryStream = new MemoryStream();
                    byte[] imageBt;
                    if (pictureBoxEmpPhoto.Image != null)
                    {
                        Image image = pictureBoxEmpPhoto.Image;
                        image.Save(memoryStream, ImageFormat.Png);
                    }
                    imageBt = memoryStream.ToArray();

                    cmd = con.CreateCommand();
                    if (flag.Equals("Add"))
                    {
                        cmd.CommandText = "insert into employee_details
values(@emp_id,@name,@gender,@add1,@add2,@ph1,@ph2,@mob1,@mob2,@dob,@hiredate,@designation,
@remarks,@email1,@email2,@basic_salary,@image)";
                    }
                    else if (flag.Equals("Update"))
                    {
                        cmd.CommandText = "UPDATE employee_details SET
name=@name,gender=@gender,address1=@add1,address2=@add2,phone1=@ph1,phone2=@ph2,mob1=@mob1,
mob2=@mob2,dob=@dob,hiredate=@hiredate,designation=@designation,remarks=@remarks,email1=@em
ail1,email2=@email2,basic_salary=@basic_salary, image=@image WHERE emp_id = @emp_id";
                    }
                    cmd.Parameters.AddWithValue("@emp_id", emp_id);
                    cmd.Parameters.AddWithValue("@name", name);
                    cmd.Parameters.AddWithValue("@gender", gender);
                    cmd.Parameters.AddWithValue("@add1", address1);
                    cmd.Parameters.AddWithValue("@add2", address2);
                    cmd.Parameters.AddWithValue("@ph1", phone1);
                    cmd.Parameters.AddWithValue("@ph2", phone2);
                    cmd.Parameters.AddWithValue("@mob1", mob1);
                    cmd.Parameters.AddWithValue("@mob2", mob2);
                    cmd.Parameters.AddWithValue("@dob", dob);
                    cmd.Parameters.AddWithValue("@hiredate", hiredate);
                    cmd.Parameters.AddWithValue("@designation", designation);
                    cmd.Parameters.AddWithValue("@remarks", remarks);
                    cmd.Parameters.AddWithValue("@email1", email1);
                    cmd.Parameters.AddWithValue("@email2", email2);
                    cmd.Parameters.AddWithValue("@basic_salary", basic_salary);
                    cmd.Parameters.AddWithValue("@image", imageBt);
                    cmd.ExecuteNonQuery();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
```

```
                }
                reloadListBoxEmployeeList();
            }
            empdetails_editable(false);

            ctrls_enabled(false);
            btnEmpDetailsCancel.SendToBack();
        }
    }
    private void btnEmpDetailsClear_Click_1(object sender, EventArgs e)
    {
        clearAllFields();
    }
    private void btnEmpDetailsCancel_Click(object sender, EventArgs e)
    {
        clearAllFields();
        empdetails_editable(false);
        ctrls_enabled(false);
        btnEmpDetailsCancel.SendToBack();
    }
    private void btnEmpDetailsClose_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void btnEmpAddPhoto_Click(object sender, EventArgs e)
    {
        openFileDialogEmpPhotoUpload.Filter = "Image Files(*.jpg; *.jpeg;)|*.jpg;
*.jpeg;";

        if (openFileDialogEmpPhotoUpload.ShowDialog() == DialogResult.OK)
        {
            var img = new Bitmap(openFileDialogEmpPhotoUpload.FileName);
            System.IO.FileInfo file = new
System.IO.FileInfo(openFileDialogEmpPhotoUpload.FileName);

            if (img.Width > 500 || img.Height > 500)
            {
                MessageBox.Show("The image should not be more than 500x500 in
resolution","Error");
            }
            else
            {
                pictureBoxEmpPhoto.Image = new
Bitmap(openFileDialogEmpPhotoUpload.FileName);
                pictureBoxEmpPhoto.SizeMode = PictureBoxSizeMode.StretchImage;
            }
        }
    }
    private void btnEmpRemovePhoto_Click(object sender, EventArgs e)
    {
        pictureBoxEmpPhoto.Image =
Test_GUI_for_Inventory_App.Properties.Resources._1__160_;
        pictureBoxEmpPhoto.SizeMode = PictureBoxSizeMode.Zoom;
    }

    private void btnEmpDetailsSearch_Click(object sender, EventArgs e)
    {
        listBoxEmpDetailsSearchResults.Items.Clear();
        if (comboBoxEmpDetailsSearchBy.SelectedIndex > -1)
```

```
            {
                string stm;
                if (comboBoxEmpDetailsSearchBy.SelectedItem.ToString() == "ID")
                {
                    stm = "SELECT emp_id FROM employee_details where emp_id like '%" +
txtEmpDetailsSearchKeywords.Text + "%'";
                    listBoxEmpDetailsSearchResults.Items.Add("Based on ID:");
                }
                else if (comboBoxEmpDetailsSearchBy.SelectedItem.ToString() == "Name")
                {
                    stm = "SELECT name FROM employee_details where name like '%" +
txtEmpDetailsSearchKeywords.Text + "%'";
                    listBoxEmpDetailsSearchResults.Items.Add("Based on Name:");
                }
                else if (comboBoxEmpDetailsSearchBy.SelectedItem.ToString() == "Phone 1")
                {
                    stm = "SELECT phone1 FROM employee_details where phone1 like '%" +
txtEmpDetailsSearchKeywords.Text + "%'";
                    listBoxEmpDetailsSearchResults.Items.Add("Based on Phone 1:");
                }
                else if (comboBoxEmpDetailsSearchBy.SelectedItem.ToString() == "Phone 2")
                {
                    stm = "SELECT phone2 FROM employee_details where phone2 like '%" +
txtEmpDetailsSearchKeywords.Text + "%'";
                    listBoxEmpDetailsSearchResults.Items.Add("Based on Phone 2:");
                }
                else if (comboBoxEmpDetailsSearchBy.SelectedItem.ToString() == "Mobile 1")
                {
                    stm = "SELECT mob1 FROM employee_details where mob1 like '%" +
txtEmpDetailsSearchKeywords.Text + "%'";
                    listBoxEmpDetailsSearchResults.Items.Add("Based on Mobile 1:");
                }
                else if (comboBoxEmpDetailsSearchBy.SelectedItem.ToString() == "Mobile 2")
                {
                    stm = "SELECT mob2 FROM employee_details where mob2 like '%" +
txtEmpDetailsSearchKeywords.Text + "%'";
                    listBoxEmpDetailsSearchResults.Items.Add("Based on Mobile 2:");
                }
                else if (comboBoxEmpDetailsSearchBy.SelectedItem.ToString() == "Email 1")
                {
                    stm = "SELECT email1 FROM employee_details where email1 like '%" +
txtEmpDetailsSearchKeywords.Text + "%'";
                    listBoxEmpDetailsSearchResults.Items.Add("Based on Email 1:");
                }
                else // if (comboBoxCustDetailsSearchBy.SelectedItem == "Email 2")
                {
                    stm = "SELECT email2 FROM employee_details where email2 like '%" +
txtEmpDetailsSearchKeywords.Text + "%'";
                    listBoxEmpDetailsSearchResults.Items.Add("Based on Email 2:");
                }
                try
                {
                    connectingCsmsEmpDetails();
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        listBoxEmpDetailsSearchResults.Items.Add(rdr.GetString(0));
                    }
```

```
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void listBoxEmpDetailsSearchResults_SelectedIndexChanged_1(object sender,
EventArgs e)
        {
            empdetails_editable(false);
            if (listBoxEmpDetailsSearchResults.SelectedIndex > 0)
            {
                String selectedValue =
listBoxEmpDetailsSearchResults.SelectedItem.ToString();
                String stm;
                connectingCsmsEmpDetails();
                if (listBoxEmpDetailsSearchResults.Items[0].ToString() == "Based on ID:")
                {
                    stm = "SELECT * FROM employee_details where emp_id = '" + selectedValue
+ "'";
                }
                else if (listBoxEmpDetailsSearchResults.Items[0].ToString() == "Based on
Name:")
                {
                    stm = "SELECT * FROM employee_details where name = '" + selectedValue +
"'";
                }
                else if (listBoxEmpDetailsSearchResults.Items[0].ToString() == "Based on
Phone 1:")
                {
                    stm = "SELECT * FROM employee_details where phone1 = '" + selectedValue
+ "'";
                }
                else if (listBoxEmpDetailsSearchResults.Items[0].ToString() == "Based on
Phone 2:")
                {
                    stm = "SELECT * FROM employee_details where phone2 = '" + selectedValue
+ "'";
                }
                else if (listBoxEmpDetailsSearchResults.Items[0].ToString() == "Based on
Mobile 1:")
                {
                    stm = "SELECT * FROM employee_details where mob1 = '" + selectedValue +
"'";
                }
                else if (listBoxEmpDetailsSearchResults.Items[0].ToString() == "Based on
Mobile 2:")
                {
                    stm = "SELECT * FROM employee_details where mob2 = '" + selectedValue +
"'";
                }
```

```
                else if (listBoxEmpDetailsSearchResults.Items[0].ToString() == "Based on
Email 1:")
                {
                    stm = "SELECT * FROM employee_details where email1 = '" + selectedValue
+ "'";
                }
                else //if (listBoxCustDetailsSearchResults.Items[0].ToString() == "Based on
Email 2:")
                {
                    stm = "SELECT * FROM employee_details where email2 = '" + selectedValue
+ "'";
                }
                try
                {
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    populatingForm();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        public void reloadListBoxEmployeeList()
        {
            listBoxEmployeeList.Items.Clear();
            listBoxAttendanceEmp.Items.Clear();
            listBoxPayrollEmployeeList.Items.Clear();
            try
            {
                connectingCsmsEmpDetails();
                string stm = "SELECT name FROM employee_details";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();

                while (rdr.Read())
                {
                    listBoxEmployeeList.Items.Add(rdr.GetString(0));
                    listBoxAttendanceEmp.Items.Add(rdr.GetString(0));
                    listBoxPayrollEmployeeList.Items.Add(rdr.GetString(0));
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
```

```
        }
    }
}
private void populatingForm()
{
    while (rdr.Read())
    {
        txtEmpID.Text = rdr.GetString(0);
        txtEmpName.Text = rdr.GetString(1);
        if (rdr.GetChar(2) == 'M')
            comboBoxEmpGender.Text = "Male";
        else if (rdr.GetChar(2) == 'F')
            comboBoxEmpGender.Text = "Female";
        else
            comboBoxEmpGender.Text = "";
        txtEmpResidenceAddress.Text = rdr.GetString(3);
        txtEmpPermanentAddress.Text = rdr.GetString(4);
        txtEmpPhone1.Text = rdr.GetString(5);
        txtEmpPhone2.Text = rdr.GetString(6);
        txtEmpMobile1.Text = rdr.GetString(7);
        txtEmpMobile2.Text = rdr.GetString(8);
        dateTimePickerEmpDOB.Value = rdr.GetDateTime(9);
        dateTimePickerEmpDateOfJoining.Value = rdr.GetDateTime(10);
        txtEmpDesignation.Text = rdr.GetString(11);
        txtEmpRemarks.Text = rdr.GetString(12);
        txtEmpEmail1.Text = rdr.GetString(13);
        txtEmpEmail2.Text = rdr.GetString(14);
        txtEmpBasicSalary.Text = rdr.GetString(15);
        byte[] blob = (byte[])rdr[16];
        if (blob.Length > 0)
        {
            Bitmap bm = ByteToImage(blob);
            pictureBoxEmpPhoto.Image = bm;
        }
        else
        {
            pictureBoxEmpPhoto.Image = null;
        }
    }
}
public Bitmap ByteToImage(byte[] blob)
{
    MemoryStream mStream = new MemoryStream();
    byte[] pData = blob;
    mStream.Write(pData, 0, Convert.ToInt32(pData.Length));
    Bitmap bm = new Bitmap(mStream, false);
    mStream.Dispose();
    return bm;
}
private void empdetails_editable(bool choice)
{
    choice = !choice;
    //txtEmpID.ReadOnly = choice;
    txtEmpName.ReadOnly = choice;
    comboBoxEmpGender.Enabled = !choice;
    txtEmpPhone1.ReadOnly = choice;
    txtEmpPhone2.ReadOnly = choice;
    txtEmpMobile1.ReadOnly = choice;
    txtEmpMobile2.ReadOnly = choice;
```

```
    txtEmpEmail1.ReadOnly = choice;
    txtEmpEmail2.ReadOnly = choice;
    txtEmpResidenceAddress.ReadOnly = choice;
    txtEmpPermanentAddress.ReadOnly = choice;
    dateTimePickerEmpDOB.Enabled = !choice;
    txtEmpBasicSalary.ReadOnly = choice;
    dateTimePickerEmpDateOfJoining.Enabled = !choice;
    txtEmpDesignation.ReadOnly = choice;
    txtEmpRemarks.ReadOnly = choice;
}
private void clearAllFields()
{
    txtEmpID.Text = "";
    txtEmpName.Text = "";
    txtEmpResidenceAddress.Text = "";
    txtEmpPermanentAddress.Text = "";
    txtEmpPhone1.Text = "";
    txtEmpPhone2.Text = "";
    txtEmpMobile1.Text = "";
    txtEmpMobile2.Text = "";
    comboBoxEmpGender.SelectedIndex = 0;
    txtEmpEmail1.Text = "";
    txtEmpEmail2.Text = "";
    dateTimePickerEmpDOB.Value = DateTime.Now;
    dateTimePickerEmpDateOfJoining.Value = DateTime.Now;
    txtEmpDesignation.Text = "";
    txtEmpRemarks.Text = "";
}
private void ctrls_enabled(bool choice)
{
    tableLayoutPanelEmpDisplay.Enabled = !choice;
    tableLayoutPanelEmpPhotoCtrls.Enabled = choice;
    tableLayoutPanelSearchCtrls.Enabled = !choice;

    btnEmpDetailsCancel.Enabled = choice;
    btnEmpDetailsClose.Enabled = !choice;
    btnEmpDetailsClear.Enabled = choice;
    btnEmpDetailsSave.Enabled = choice;
}
private void txtEmpName_KeyPress(object sender, KeyPressEventArgs e)
{
    Test_GUI_for_Inventory_App.Validations.OnlyTextAllowed(e);
}
private void txtEmpPhone1_KeyPress(object sender, KeyPressEventArgs e)
{
    Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
}
private void txtEmpPhone2_KeyPress(object sender, KeyPressEventArgs e)
{
    Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
}
private void txtEmpMobile1_KeyPress(object sender, KeyPressEventArgs e)
{
    Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
}
private void txtEmpMobile2_KeyPress(object sender, KeyPressEventArgs e)
{
    Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
}
```

```
        private void txtEmpEmail1_Leave(object sender, EventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.ValidateEmail(e, txtEmpEmail1);
        }
        private void txtEmpEmail2_Leave(object sender, EventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.ValidateEmail(e, txtEmpEmail2);
        }
        private void txtEmpBasicSalary_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }
        public void connectingCsmsEmpDetails()
        {
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }

        //Muster Codes

        private void listBoxAttendanceEmp_DoubleClick(object sender, EventArgs e)
        {
            if (listBoxAttendanceEmp.SelectedIndex > -1)
            {
                try
                {
                    connectingCsmsEmpDetails();
                    String stm = "SELECT emp_id from employee_details where name = '" +
listBoxAttendanceEmp.SelectedItem.ToString() + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        txtAttendanceEmpID.Text = rdr.GetString(0);
                    }
                    rdr.Close();
                }
                catch(Exception)
                {
                    throw;
                }
                finally
                {
                    if(con.State==ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void btnMarkAttendance_Click(object sender, EventArgs e)
        {
            if (dateTimePickerAttendanceDate.Value.Date >= System.DateTime.Now)
            {
                MessageBox.Show("Date ahead of System Date", "Error");
            }
            else
```

```
            {
                if (txtAttendanceEmpID.Text != "")
                {
                    try
                    {
                        String da = dateTimePickerAttendanceDate.Value.Date.ToString("yyyy-
MM-dd");
                        connectingCsmsEmpDetails();
                        String stm = "SELECT emp_id,date,attendance_id from attendance
where emp_id = '" + txtAttendanceEmpID.Text + "';";
                        MySqlCommand cmd = new MySqlCommand(stm, con);
                        rdr = cmd.ExecuteReader();
                        while (rdr.Read())
                        {
                            if (da.Equals(rdr.GetString(1)))
                            {
                                String attId = rdr.GetString(2);
                                rdr.Close();
                                DialogResult dialogResult = MessageBox.Show("Attendance
already marked. Do you want to update?", "Already Marked", MessageBoxButtons.YesNo);
                                if (dialogResult == DialogResult.Yes)
                                {
                                    cmd = con.CreateCommand();
                                    cmd.CommandText = "UPDATE attendance SET
reporting_time=@reporting_time, present_absent=@present_absent,leaving_time=@leaving_time
WHERE attendance_id=@attID;";
                                    cmd.Parameters.AddWithValue("@attID", attId);
                                    cmd.Parameters.AddWithValue("@reporting_time",
txtAttendanceReportingTime.Text);
                                    cmd.Parameters.AddWithValue("@present_absent",
comboBoxAttendance.SelectedItem.ToString());
                                    cmd.Parameters.AddWithValue("@leaving_time",
txtAttendanceLeavingTime.Text);
                                    cmd.ExecuteNonQuery();
                                    return;
                                }
                                else if (dialogResult == DialogResult.No)
                                {
                                    return;
                                }
                            }
                        }
                        rdr.Close();
                        cmd = con.CreateCommand();
                        cmd.CommandText = "INSERT into
attendance(emp_id,date,reporting_time,present_absent,leaving_time)
values(@emp_id,@date,@reporting_time,@present_absent,@leaving_time);";
                        cmd.Parameters.AddWithValue("@emp_id", txtAttendanceEmpID.Text);
                        cmd.Parameters.AddWithValue("@date", da);
                        cmd.Parameters.AddWithValue("@reporting_time",
txtAttendanceReportingTime.Text);
                        cmd.Parameters.AddWithValue("@present_absent",
comboBoxAttendance.SelectedItem.ToString());
                        cmd.Parameters.AddWithValue("@leaving_time",
txtAttendanceLeavingTime.Text);
                        cmd.ExecuteNonQuery();
                    }

                    catch (Exception)
```

```
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
    }
    private void btnAnalyzeAttendance_Click(object sender, EventArgs e)
    {
        try
        {
            dataGridViewAttendanceShow.Rows.Clear();
            dataGridViewAttendanceShow.Rows.Add();
            connectingCsmsEmpDetails();
            String stm = "SELECT * from attendance where date = '" +
dateTimePickerAttendanceDate.Value.Date.ToString("yyyy-MM-dd") + "';";
            MySqlCommand cmd = new MySqlCommand(stm, con);
            cmd = new MySqlCommand(stm, con);
            rdr = cmd.ExecuteReader();

            while (rdr.Read())
            {
                DataGridViewRow row =
(DataGridViewRow)dataGridViewAttendanceShow.Rows[0].Clone();
                row.Cells[0].Value = rdr.GetString(1);
                row.Cells[1].Value = rdr.GetString(2);
                row.Cells[2].Value = rdr.GetString(3);
                row.Cells[3].Value = rdr.GetString(5);
                row.Cells[4].Value = rdr.GetString(4);
                dataGridViewAttendanceShow.Rows.Add(row);
            }
        }
        catch (Exception)
        {
            throw;
        }
        finally
        {
            if (con.State == ConnectionState.Open)
            {
                con.Close();
            }
        }
    }

    //Payroll

    private void listBoxPayrollEmployeeList_DoubleClick(object sender, EventArgs e)
    {
        if (listBoxPayrollEmployeeList.SelectedIndex > -1)
        {
            emptyPayrollFields();
            try
```

```
                {
                    comboBoxPayrollYear.Items.Clear();
                    DateTime payrollHireDate;
                    connectingCsmsEmpDetails();
                    String stm = "SELECT emp_id, name, gender, mob1, designation,
basic_salary, hiredate from employee_details where name = '" +
listBoxPayrollEmployeeList.SelectedItem.ToString() + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        txtPayrollEmpID.Text = rdr.GetString(0);
                        txtPayrollEmpName.Text = rdr.GetString(1);
                        txtPayrollGender.Text = rdr.GetString(2);
                        txtPayrollMob.Text = rdr.GetString(3);
                        txtPayrollDesignation.Text = rdr.GetString(4);
                        txtPayrollBasicSalary.Text = rdr.GetString(5);
                        payrollHireDate = rdr.GetDateTime(6);
                        int f = payrollHireDate.Year;
                        while(f<=DateTime.Today.Year)
                        {
                            comboBoxPayrollYear.Items.Add(f);
                            f++;
                        }
                    }
                    rdr.Close();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void btnPayrollCalculate_Click(object sender, EventArgs e)
        {
            if(txtPayrollBasicSalary.Text!="")
            {
                txtPayrollTotalSalary.Text=txtPayrollBasicSalary.Text;
                if(txtPayrollDeduction.Text!="")
                {
                    txtPayrollTotalSalary.Text =
Convert.ToString(Int32.Parse(txtPayrollTotalSalary.Text) -
Int32.Parse(txtPayrollDeduction.Text));
                }
                if (txtPayrollCommission.Text != "")
                {
                    txtPayrollTotalSalary.Text =
Convert.ToString(Int32.Parse(txtPayrollTotalSalary.Text) +
Int32.Parse(txtPayrollCommission.Text));
                }
            }
        }
```

```
        private void btnPayrollSave_Click(object sender, EventArgs e)
        {
            if(txtPayrollEmpID.Text!="")
            {
                connectingCsmsEmpDetails();
                try
                {
                    cmd = con.CreateCommand();
                    cmd.CommandText = "INSERT into
payroll(emp_id,year,month,leaves_taken,leaves_allowed,commission,deduction,total_salary,pay
ment_Status)
values(@emp_id,@year,@month,@leaves_taken,@leaves_allowed,@commission,@deduction,@total_sal
ary,@payment_Status)";
                    cmd.Parameters.AddWithValue("@emp_id", txtPayrollEmpID.Text);
                    cmd.Parameters.AddWithValue("@year",
comboBoxPayrollYear.SelectedItem.ToString());
                    cmd.Parameters.AddWithValue("@month",
comboBoxPayrollMonth.SelectedItem.ToString());
                    cmd.Parameters.AddWithValue("@leaves_taken",
txtPayrollLeavesTaken.Text);
                    cmd.Parameters.AddWithValue("@leaves_allowed",
comboBoxPayrollLeavesAllowed.SelectedItem.ToString());
                    cmd.Parameters.AddWithValue("@commission", txtPayrollCommission.Text);
                    cmd.Parameters.AddWithValue("@deduction", txtPayrollDeduction.Text);
                    cmd.Parameters.AddWithValue("@total_salary",
txtPayrollTotalSalary.Text);
                    cmd.Parameters.AddWithValue("@payment_Status",
comboBoxPaymentStatus.SelectedItem.ToString());
                    cmd.ExecuteNonQuery();
                    MessageBox.Show("Updated");
                    emptyPayrollFields();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void emptyPayrollFields()
        {
            txtPayrollEmpID.Text = "";
            txtPayrollEmpName.Text = "";
            txtPayrollGender.Text = "";
            txtPayrollMob.Text = "";
            txtPayrollDesignation.Text = "";
            comboBoxPayrollYear.SelectedIndex = -1;
            comboBoxPayrollMonth.SelectedIndex = -1;
            txtPayrollLeavesTaken.Text = "";
            comboBoxPayrollLeavesAllowed.SelectedIndex = 0;
            txtPayrollCommission.Text = "";
            txtPayrollDeduction.Text = "";
```

```
            txtPayrollTotalSalary.Text = "";
            comboBoxPaymentStatus.SelectedIndex = -1;
        }
        private void button1_Click(object sender, EventArgs e)
        {
            PrintDetails pd = new PrintDetails();
            Test_GUI_for_Inventory_App.Validations.flagReport = "Emp";
            pd.pass_values_EmpID = txtEmpID.Text;
            pd.Show();
        }
        private void txtPayrollCommission_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }
        private void txtPayrollDeduction_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }
    }
}
```

UserAccounts.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace Test_GUI_for_Inventory_App
{
    public partial class UserAccounts : Form
    {
        String MyConnectionString;
        String username, password, email;
        char privilege;
        String flag;
        int f;
        MySqlConnection con;
        MySqlCommand cmd;
        MySqlDataReader rdr;

        private static UserAccounts _instance;

        public UserAccounts()
        {
            InitializeComponent();
        }
        private void UserAccounts_Load(object sender, EventArgs e)
        {
            reloadListBoxUserAcc();
            useraccdetails_editable(false);
            flag = "";
            f = 0;
        }
```

```
        public UserAccounts instance
        {
            get
            {
                if (UserAccounts._instance == null)
                {
                    UserAccounts._instance = new UserAccounts();
                }
                return UserAccounts._instance;
            }
        }
        private void UserAccounts_FormClosed(object sender, FormClosedEventArgs e)
        {
            UserAccounts._instance = null;
        }
        private void listBoxUserAccList_SelectedIndexChanged(object sender, EventArgs e)
        {
            useraccdetails_editable(false);
            if (listBoxUserAccList.SelectedIndex > -1)
            {
                String selectedValue = listBoxUserAccList.SelectedItem.ToString();
                connectingCsmsUserAccDetails();
                try
                {
                    string stm = "SELECT * FROM login where username='" + selectedValue +
"'";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    populatingForm();
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
        }
        private void btnAddUser_Click(object sender, EventArgs e)
        {
            btnUserAccSave.Enabled = true;
            clearAllFieldsUserAccDetails();
            useraccdetails_editable(true);
            flag = "Add";

            ctrls_enabled(true);
        }
        private void btnUpdateUserDetails_Click(object sender, EventArgs e)
        {
            if (listBoxUserAccList.SelectedIndex > -1)
            {
                btnUserAccSave.Enabled = true;
                useraccdetails_editable(true);
```

```
                flag = "Update";

                ctrls_enabled(true);
            }
        }
        private void btnRemoveUser_Click(object sender, EventArgs e)
        {
            if (txtUserName.Text != null) //((listBoxCustomerList.SelectedIndex > -1 ||
listBoxCustDetailsSearchResults.SelectedIndex > 0) && (txtCustID!=null))
            {
                try
                {
                    connectingCsmsUserAccDetails();
                    String stm = "SELECT privilege FROM login where username != '" +
txtUserName.Text + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        if (rdr.GetString(0).Equals("Y"))
                        {
                            f++;
                        }
                    }
                    if (f == 0)
                    {
                        MessageBox.Show("No other account with all the privilege. Please
allow atleast one account with all the privilege");
                        checkBoxUserPrivilegeLtd.Checked = false;
                        reloadListBoxUserAcc();
                        useraccdetails_editable(false);
                        btnUserAccSave.Enabled = false;
                        ctrls_enabled(false);
                        return;
                    }
                    rdr.Close();

                    stm = "SELECT username FROM login where username = '" +
txtUserName.Text + "'";
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
                        if (MessageBox.Show("Do you want to delete User with name " +
rdr.GetString(0) + "?", "Delete User?", MessageBoxButtons.YesNo, MessageBoxIcon.Question)
== DialogResult.Yes)
                        {
                            if (listBoxUserAccList.Items.Count == 1)
                            {
                                MessageBox.Show("Cannot delete the only account present!",
"Error");
                            }
                            else
                            {
                                cmd = con.CreateCommand();
                                cmd.CommandText = "delete from login where username='" +
rdr.GetString(0) + "';";
                                rdr.Close();
                                cmd.ExecuteNonQuery();
```

```
                        }
                    }
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
                reloadListBoxUserAcc();
                clearAllFieldsUserAccDetails();
            }
        }

    }
    private void btnUserAccSave_Click(object sender, EventArgs e)
    {
        //Extracting data from fields

        username = txtUserName.Text;
        password = txtUserPassword.Text;
        email = txtRecoveryEmail.Text;

        if (checkBoxUserPrivilegeLtd.Checked == true)
            privilege = 'N';
        else
            privilege = 'Y';

        //Inserting data
        connectingCsmsUserAccDetails();
        try
        {
            String stm = "SELECT privilege FROM login where username != '"+
txtUserName.Text +"';";
            MySqlCommand cmd = new MySqlCommand(stm, con);
            rdr = cmd.ExecuteReader();
            while(rdr.Read())
            {
                if(rdr.GetString(0).Equals("Y"))
                {
                    f++;
                }
            }
            if(f==0 && checkBoxUserPrivilegeLtd.Checked==true)
            {
                MessageBox.Show("No other account with all the privilege. Please allow
atleast one account with all the privilege");
                checkBoxUserPrivilegeLtd.Checked = false;
                reloadListBoxUserAcc();
                useraccdetails_editable(false);
                btnUserAccSave.Enabled = false;
                ctrls_enabled(false);
                return;
            }
```

```
                rdr.Close();
                cmd = con.CreateCommand();
                if (flag.Equals("Add"))
                {
                    cmd.CommandText = "insert into login values(@username, @password,
@email,@privilege)";
                }
                else if (flag.Equals("Update"))
                {
                    cmd.CommandText = "UPDATE login SET username=@username,
password=@password, email=@email, privilege=@privilege WHERE username = @username";
                }
                cmd.Parameters.AddWithValue("@username", username);
                cmd.Parameters.AddWithValue("@password", password);
                cmd.Parameters.AddWithValue("@email", email);
                cmd.Parameters.AddWithValue("@privilege", privilege);
                cmd.ExecuteNonQuery();
                f = 0;
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    con.Close();
                }
                reloadListBoxUserAcc();
            }
            useraccdetails_editable(false);
            btnUserAccSave.Enabled = false;

            ctrls_enabled(false);
        }
        private void btnUserAccClear_Click(object sender, EventArgs e)
        {
            clearAllFieldsUserAccDetails();
        }
        private void btnUserAccCancel_Click(object sender, EventArgs e)
        {
            clearAllFieldsUserAccDetails();
            useraccdetails_editable(false);
            ctrls_enabled(false);
        }
        private void btnUserAccClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        public void reloadListBoxUserAcc()
        {
            listBoxUserAccList.Items.Clear();
            try
            {
                connectingCsmsUserAccDetails();
                string stm = "SELECT username FROM login";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
```

```
        while (rdr.Read())
        {
            listBoxUserAccList.Items.Add(rdr.GetString(0));
        }
    }
    catch (Exception)
    {
        throw;
    }
    finally
    {
        if (con.State == ConnectionState.Open)
        {
            con.Close();
        }
    }
}
private void populatingForm()
{
    while (rdr.Read())
    {
        txtUserName.Text = rdr.GetString(0);
        txtUserPassword.Text = rdr.GetString(1);
        txtRecoveryEmail.Text = rdr.GetString(2);
        if (rdr.GetChar(3) == 'N')
            checkBoxUserPrivilegeLtd.Checked = true;
        else
            checkBoxUserPrivilegeLtd.Checked = false;
    }
}
private void useraccdetails_editable(bool choice)
{
    txtUserName.ReadOnly = !choice;
    txtUserPassword.ReadOnly = !choice;
    txtRecoveryEmail.ReadOnly = !choice;
    checkBoxUserPrivilegeLtd.Enabled = choice;
}
private void clearAllFieldsUserAccDetails()
{
    txtUserName.Text = "";
    txtUserPassword.Text = "";
    txtRecoveryEmail.Text = "";
}
private void ctrls_enabled(bool choice)
{
    tableLayoutPanelUserAccDisplay.Enabled = !choice;

    btnUserAccCancel.Enabled = choice;
    btnUserAccClose.Enabled = !choice;
    btnUserAccClear.Enabled = choice;
    btnUserAccSave.Enabled = choice;
}
private void txtRecoveryEmail_Leave(object sender, EventArgs e)
{
    Test_GUI_for_Inventory_App.Validations.ValidateEmail(e, txtRecoveryEmail);
}
public void connectingCsmsUserAccDetails()
{
```

```
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }
    }
}
```

Add Model.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Test_GUI_for_Inventory_App
{
    public partial class AddModel : Form
    {
        String MyConnectionString;
        MySqlConnection con;
        MySqlCommand cmd;
        MySqlDataReader rdr;

        private static AddModel _instance;

        public AddModel()
        {
            InitializeComponent();
        }
        public AddModel instance
        {
            get
            {
                if (AddModel._instance == null)
                {
                    AddModel._instance = new AddModel();
                }
                return AddModel._instance;
            }
        }
        private void btnAddModel_Click(object sender, EventArgs e)
        {
            if (txtModelName.Text != "")
            {
                String ModelName = txtModelName.Text;
                String Brand = txtModelBrand.Text;
                String Type = (String)comboModelType.SelectedItem;
                String Category = (String)comboModelCategory.SelectedItem;
                String Price = txtModelPrice.Text;
                String Warranty = txtModelWarranty.Text;
                String Remarks = txtModelRemarks.Text;

                try
                {
```

```
                    connectingCsmsCustDetails();
                    String stm = "SELECT model_id FROM stock where model_id = '" +
ModelName + "'";
                    cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
                        MessageBox.Show("This Model already exists in database.");
                        rdr.Close();
                    }
                    else
                    {
                        rdr.Close();
                        cmd = con.CreateCommand();
                        cmd.CommandText = "insert into stock
values(@model_id,@brand,@type,@category,@warr,@price,@quantity,@remarks)";
                        cmd.Parameters.AddWithValue("@model_id", ModelName);
                        cmd.Parameters.AddWithValue("@brand", Brand);
                        cmd.Parameters.AddWithValue("@type", Type);
                        cmd.Parameters.AddWithValue("@category", Category);
                        cmd.Parameters.AddWithValue("@warr", Warranty);
                        cmd.Parameters.AddWithValue("@price", Price);
                        cmd.Parameters.AddWithValue("@quantity", '0');
                        cmd.Parameters.AddWithValue("@remarks", Remarks);
                        cmd.ExecuteNonQuery();
                        MessageBox.Show("Added Successfully!");
                        clearAll();
                    }
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
            else
            {
                MessageBox.Show("Add Model Name");
            }
        }
        public void connectingCsmsCustDetails()
        {
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }
        public void clearAll()
        {
            txtModelName.Text = "";
            txtModelBrand.Text = "";
            comboModelType.SelectedIndex = 0;
            comboModelCategory.SelectedIndex = 0;
            txtModelPrice.Text = "";
```

```
            txtModelWarranty.Text = "";
            txtModelRemarks.Text = "";
        }
        private void txtModelPrice_KeyPress(object sender, KeyPressEventArgs e)
        {
            Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
        }
    }
}
```

PrintDetails.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using CrystalDecisions.CrystalReports.Engine;
using CrystalDecisions.ReportSource;

namespace Test_GUI_for_Inventory_App
{
    public partial class PrintDetails : Form
    {
        private string payroll_emp_id;

        public string pass_values_EmpID
        {
            get { return payroll_emp_id; }
            set { payroll_emp_id = value; }
        }
        private string customer_cust_id;
        public string pass_values_CustID
        {
            get { return customer_cust_id; }
            set { customer_cust_id = value; }
        }
        private string supplier_supp_id;
        public string pass_values_SuppID
        {
            get { return supplier_supp_id; }
            set { supplier_supp_id = value; }
        }
        private string bill_order_id;
        public string pass_values_OrderID
        {
            get { return bill_order_id; }
            set { bill_order_id = value; }
        }
        public PrintDetails()
        {
            InitializeComponent();
        }
        private void PrintDetails_Load(object sender, EventArgs e)
        {
```

```
            Payroll pr = new Payroll();
            Customer cr = new Customer();

            string MyConnectionString;
            MySqlConnection con;
            MySqlDataReader rdr;

            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();


            if (Test_GUI_for_Inventory_App.Validations.flagReport=="Emp")
            {
                string stm = "SELECT * from employee_details where emp_id='" +
payroll_emp_id + "'";
                MySqlCommand cmd = new MySqlCommand(stm, con);

                DataSet ds = new DataSet();

                MySqlDataAdapter myDataAdapter = new MySqlDataAdapter();

                myDataAdapter.SelectCommand = cmd;

                myDataAdapter.Fill(ds, "employee_details");

                EmpDetailsPrint objRpt = new EmpDetailsPrint();
                objRpt.SetDataSource(ds);
                crystalReportViewer1.ReportSource = objRpt;
                crystalReportViewer1.Refresh();
            }
            else if (Test_GUI_for_Inventory_App.Validations.flagReport == "Cust")
            {
                string stm = "SELECT * from cust_details where cust_id='" +
customer_cust_id + "'";
                MySqlCommand cmd = new MySqlCommand(stm, con);

                DataSet ds = new DataSet();

                MySqlDataAdapter myDataAdapterCust = new MySqlDataAdapter();

                myDataAdapterCust.SelectCommand = cmd;

                myDataAdapterCust.Fill(ds, "cust_details");

                CustDetailsPrint objRpt = new CustDetailsPrint();
                objRpt.SetDataSource(ds);
                crystalReportViewer1.ReportSource = objRpt;
                crystalReportViewer1.Refresh();
            }
            else if (Test_GUI_for_Inventory_App.Validations.flagReport == "Supp")
            {
                string stm = "SELECT * from dealer_details where dealer_id='" +
supplier_supp_id + "'";
                MySqlCommand cmd = new MySqlCommand(stm, con);

                DataSet ds = new DataSet();

                MySqlDataAdapter myDataAdapterCust = new MySqlDataAdapter();
```

```
                myDataAdapterCust.SelectCommand = cmd;

                myDataAdapterCust.Fill(ds, "dealer_details");

                SuppDetailsPrint objRpt = new SuppDetailsPrint();
                objRpt.SetDataSource(ds);
                crystalReportViewer1.ReportSource = objRpt;
                crystalReportViewer1.Refresh();
            }

            else if (Test_GUI_for_Inventory_App.Validations.flagReport == "Bill")
            {
                //string stm = "SELECT a.model_id, a.amount, a.quantity, b.total_amount
from sale_item a, cust_bill b where a.sale_order_id=b.sale_order_id and a.sale_order_id='"
+ bill_order_id + "'";
                string stm = "SELECT model_id, amount, quantity, sale_order_id from
sale_item where sale_order_id='" + bill_order_id + "'";
                MySqlCommand cmd = new MySqlCommand(stm, con);

                DataSet ds = new DataSet();

                MySqlDataAdapter myDataAdapterCust = new MySqlDataAdapter();

                myDataAdapterCust.SelectCommand = cmd;

                myDataAdapterCust.Fill(ds, "sale_item");

                Invoice objRpt = new Invoice();
                objRpt.SetDataSource(ds);
                crystalReportViewer1.ReportSource = objRpt;
                crystalReportViewer1.Refresh();
            }
        }
    }
}
```

PaymentCustomer.cs

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Test_GUI_for_Inventory_App
{
    public partial class PaymentCustomer : Form
    {
        String MyConnectionString, stm;
        MySqlConnection con;
        MySqlCommand cmd;
        MySqlDataReader rdr;

        public PaymentCustomer()
```

```
        {
            InitializeComponent();
            formLoad();
        }
        private void formLoad()
        {
            txtBillID.Text = Test_GUI_for_Inventory_App.Customer_Billing.BillId;
            invoiceGeneration();
            totalRetrieval();
        }
        private void invoiceGeneration()
        {
            String invoice = "", finalInvoice = "";
            int ordNo = 1;
            if (txtBillID.Text != "")
            {
                connectingCust_Payment();
                try
                {
                    String stm = "SELECT cust_invoice_no from payment_cust where
cust_bill_id = '" + txtBillID.Text + "';";
                    MySqlCommand cmd = new MySqlCommand(stm, con);
                    rdr = cmd.ExecuteReader();
                    if (rdr.Read())
                    {
                        rdr.Close();
                        rdr = cmd.ExecuteReader();
                        while (rdr.Read())
                        {
                            invoice = rdr.GetString(0);
                            int initialIndex = txtBillID.Text.Length + 1;
                            string t = invoice.Substring(initialIndex, 4);
                            ordNo = int.Parse(t) + 1;
                        }
                        finalInvoice = txtBillID.Text + "/" + ordNo.ToString("D4");
                    }
                    else
                    {
                        finalInvoice = txtBillID.Text + "/0001";
                    }
                    txtCustInvoiceNo.Text = finalInvoice;
                }
                catch (Exception)
                {
                    throw;
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        rdr.Close();
                        con.Close();
                    }
                }
            }
        }
        private void totalRetrieval()
        {
            connectingCust_Payment();
```

```
            try
            {
                String stm = "SELECT total_amount,paid_amount from cust_bill where
cust_bill_id = '" + txtBillID.Text + "';";
                MySqlCommand cmd = new MySqlCommand(stm, con);
                rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    txtTotalAmount.Text = rdr.GetString(0);
                    txtPaidAmount.Text = rdr.GetString(1);
                }
            }
            catch (Exception)
            {
                throw;
            }
            finally
            {
                if (con.State == ConnectionState.Open)
                {
                    rdr.Close();
                    con.Close();
                }
            }
        }
        private void btnAddPayment_Click(object sender, EventArgs e)
        {
            try
            {
                connectingCust_Payment();
                String pay_type = comboPaymentType.SelectedItem.ToString();
                int paid_Amount = Int32.Parse(txtPaidAmount.Text);
                int pay_now = Int32.Parse(txtPayNow.Text);
                int total_amount = Int32.Parse(txtTotalAmount.Text);
                if ((paid_Amount + pay_now) <= total_amount)
                {
                    cmd = con.CreateCommand();
                    cmd.CommandText = "INSERT into payment_cust
values(@c_i_n,@date,@c_b_i,@payment_type,@paid_amount);";
                    cmd.Parameters.AddWithValue("@c_i_n", txtCustInvoiceNo.Text);
                    cmd.Parameters.AddWithValue("@date", dateTimePicker1.Value.Date);
                    cmd.Parameters.AddWithValue("@c_b_i", txtBillID.Text);
                    cmd.Parameters.AddWithValue("@payment_type", pay_type);
                    cmd.Parameters.AddWithValue("@paid_amount", pay_now);
                    cmd.ExecuteNonQuery();

                    cmd = con.CreateCommand();
                    cmd.CommandText = "UPDATE cust_bill SET paid_amount=@pa where
cust_bill_id = @c_b_i;";
                    cmd.Parameters.AddWithValue("@pa", (paid_Amount + pay_now));
                    cmd.Parameters.AddWithValue("@c_b_i", txtBillID.Text);
                    cmd.ExecuteNonQuery();

                    MessageBox.Show("Added");
                    this.Close();
                }
                else
                {
                    MessageBox.Show("Pay less then total payment");
```

```
                txtPayNow.Text = "";
            }
        }
        catch (Exception)
        {
            throw;
        }
        finally
        {
            if (con.State == ConnectionState.Open)
            {
                rdr.Close();
                con.Close();
            }
        }
    }
    public void connectingCust_Payment()
    {
        MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
        con = new MySqlConnection(MyConnectionString);
        con.Open();
    }
    private void txtPayNow_KeyPress(object sender, KeyPressEventArgs e)
    {
        Test_GUI_for_Inventory_App.Validations.OnlyNosAllowed(e);
    }
}
}
```

PasswordRecovery.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using System.Net.Mail;

namespace Test_GUI_for_Inventory_App
{
    public partial class PasswordRecoveryForm : Form
    {
        private string login_combo_username;

        public string pass_values
        {
            get { return login_combo_username; }
            set { login_combo_username = value; }
        }
        string MyConnectionString;
        public String uid="", email, pwd;
        MySqlConnection con;
        MySqlDataReader rdr;

        public PasswordRecoveryForm()
```

```
        {
            InitializeComponent();
        }
        public void connectingCsmsPasswordRecovery()
        {
            MyConnectionString = "Server=localhost;Database=csms;Uid=corpinfo;Pwd=kvvk;";
            con = new MySqlConnection(MyConnectionString);
            con.Open();
        }
        private void btnSendPassword_Click(object sender, EventArgs e)
        {
            connectingCsmsPasswordRecovery();

            string stm = "SELECT email from login where username='" + login_combo_username
+ "'";
            MySqlCommand cmd = new MySqlCommand(stm, con);
            rdr = cmd.ExecuteReader();

            while (rdr.Read())
            {
                try
                {
                    if(rdr.IsDBNull(0))
                    {
                        MessageBox.Show("Error");
                    }
                    else
                    {
                        email = rdr.GetString(0);
                    }

                }
                catch (Exception)
                {
                    MessageBox.Show("Error");
                }
            }
            if(txtRecoveryEmail.Text==email)
            {
                stm = "SELECT password from login where username='" + login_combo_username
+ "'";
                MySqlCommand cmd_pass = new MySqlCommand(stm, con);
                rdr.Close();
                rdr = cmd_pass.ExecuteReader();

                while (rdr.Read())
                {
                    pwd = rdr.GetString(0);
                }

                try
                {
                    MailMessage mail = new MailMessage("username@gmail.com", email);
                    SmtpClient client = new SmtpClient();
                    client.Port = 25;
                    client.DeliveryMethod = SmtpDeliveryMethod.Network;
                    client.UseDefaultCredentials = false;
                    client.Credentials = new
System.Net.NetworkCredential("username@gmail.com", "password");
```

```
                    client.Host = "smtp.gmail.com";
                    mail.Subject = "Password Recovery";
                    mail.IsBodyHtml = true;
                    mail.Body = "Your Password is: <b>" + pwd + "</b> <br><br> NOTE: This
is a system generated mail. Please do not 'Reply To' this mail.";
                    client.EnableSsl = true;
                    client.Send(mail);
                }
                catch(Exception)
                {
                    MessageBox.Show("The mailbox you requested could not be
reached.\nPlease check the id or try again later.","Sending Failure");
                }
                finally
                {
                    if (con.State == ConnectionState.Open)
                    {
                        con.Close();
                    }
                }
            }
            else
            {
                MessageBox.Show("The recovery email id specified does not correspond to the
one specified for the account.\nPlease check the id and try again");
            }
        }
    }
}
```

2) Testing Phase

2.1 Testing and Methodologies Adopted for Testing

Testing

Testing is a process of examining a product to determine what defect it contains. Hence we have tested this software by reviewing the construction, by exercising the functions and examining the result.

Test Methods:

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases:

1) White-box testing
   White-box testing (by seeing the source code) tests internal structures or workings of a program. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. It is a detailed investigation of internal logic and structure of the code.
2) Black-box testing
   Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it.

Test Levels:

1) Unit Testing
   Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently examined for proper operation.
2) Integration Testing
   Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Software components may be integrated in an iterative way or all together, Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design.
3) System Testing
   System testing tests a completely integrated system to verify that the system meets its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.
4) Acceptance Testing
   Acceptance testing is a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users.

Unit testing

Login form:

Field- 'Username'	Field- 'Password'	Action	Expected Output	Test Result
(Not Chosen)	(Empty)	"Login button clicked"	Message displayed – "Select a username!"	Successful
admin	(Empty)	"Login button clicked"	Message displayed – "Select a password!"	Successful
admin	*** (Wrong password)	"Login button clicked"	Message displayed – "Invalid Username or Password"	Successful
admin	***** (Right password)	"Login button clicked"	Control goes to the home page.	Successful

Employee Details Form:

Textbox	Input	Action	Expected Output	Test Result
Name	Ron1	On key press	Message displayed – "Only text allowed"	Successful
Phone 1	54a	On key press	Message displayed – "Only numbers allowed"	Successful
Phone 2	4132g	On key press	Message displayed – "Only numbers allowed"	Successful
Mob 1	6333d	On key press	Message displayed – "Only numbers allowed"	Successful
Mob 2	369543r	On key press	Message displayed – "Only numbers allowed"	Successful
Email 1	as@1	On leave	Message displayed – "Invalid email"	Successful
Email 2	gafs@coe.a	On leave	Message displayed – "Invalid email"	Successful

Integration testing:

Login form:

Field- 'Username'	Field- 'Password'	Action	Expected Output	Test Result
(Not Chosen or Wrong Password)	(Not Chosen or Wrong Password)	"Login button clicked"	Application program not invoked and appropriate message is displayed.	Successful
Admin (Valid Username)	***** (Right password)	"Login button clicked"	Application program invoked and values accepted.	Successful

Customer form:

All Fields State:	Action	Expected output	Test Result
Read Only	"Add Customer button clicked"	Fields state changed to editable	Successful

System Testing:

Serial No.	Action	Expected Output	Test Result
1.	Executable File Clicked	Application Starts	Successful
2.	On Modifying Details from Different Forms	Database Updated	Successful
3.	Close	System Closed	Successful

Acceptance Testing:

Serial No.	Requirements	Expected Output	Test Results
1.	Login form	Available and executes.	Successful
2.	Inventory form	Available and executes.	Successful
3.	Supplier Details form	Available and executes.	Successful
4.	Purchase Order form	Available and executes.	Successful
5.	Billing form	Available and executes.	Successful
6.	Customer Details form	Available and executes.	Successful
7.	Muster form	Available and executes.	Successful
8.	Payment form	Available and executes.	Successful
9.	Employee Details	Available and executes.	Successful
10.	Accounts	Available and executes.	Successful

Conclusion

An attempt is made in all its earnest towards the successful completion of the project. The system was verified with valid as well as invalid data.

This system is user friendly since it has been developed in Microsoft Visual Studio 2013, a successful GUI environment.
Since the connection connection can be extended to database, the control will be more powerful.

Any suggestions for future development of the system are welcome.

Upgrading the system can be done without affecting the proper functioning of system.

References

Website

1) www.google.com
2) www.youtube.com
3) www.stackoverflow.com