****A Project Work Report On

**Fruit Inventory Management System (FIMS)**

Masters of Science in Information Technology (MSc IT)

6th Semester

Of

SIKKIM MANIPAL UNIVERSITY- DDE

By

Tilrupa Timsina

Registration No: 1308004727

Course: MSc IT

Semester: 6th Semester

LC Code: 03290

Submitted To: Niva Management & IT College

**VIVA-VOCE SHEET**

We have conducted the viva-voce examination of the report

**SUBMITTED BY**

Tilrupa Timsina

MSc IT 6th Semester

Roll No: 1308004727

**ENTITLED**

**Fruit Inventory Management System (FIMS)**

And found that the thesis to be the original work of the student and written accordingly to the prescribed format. We recommended the report to be accepted as partial fulfillment of the requirement for the degree of

**Masters of Science in Information Technology (BSc IT)**

VIVA-VOCE COMMITTEE

Head, Research Department: ………………………….

Member (Thesis Supervisor): …………………………

Member (External Expert): ……………………………

Date: - …………………

DECLARATION

I hereby declare that the project entitled “**Fruit Inventory Management System (FIMS)**” submitted to Niva Management & IT College is a record of my original work and this project work is submitted in the partial fulfillment of the requirement for the award of the degree of Masters of Science in Information Technology (MSc IT). The results embodied in this thesis has not been submitted to any other university or Institute or other degree, association ship, fellowship or any other similar titles for the award of any degree or diploma.

------------------------------

Signature of the Student

**Tilrupa Timsina**

Roll No: 1308004727

Course: MSc IT 6th Semester

Place: Birtamode-7, Jhapa

CERTICICATE FROM SUPERVISORS

This is to certify that the Project entitled “**Fruit Inventory Management System (FIMS)**” submitted by **Tilrupa Timsina** to the Department of Computer Application, School of Science and Technology, Niva Management & IT College towards partial fulfillment of the requirements for the award of the degree of Bachelor of Science and Information Technology (MSc IT 6th Semester) is an original work carried out by him/her and meets all the requirements defined by the university to award the degree.

------------------------------

Signature

Name: Er. Prakash Gurung

Designation: Lecturer

(Project Supervisor)

------------------------------

Signature

Name:

Designation:

(External Examiner)

------------------------------

Signature

Name: Mr. Sandesh Malla

Designation: Principal

(Principal and Co- coordinator of Niva College)

Place: Birtamode-7, Jhapa

Address: Niva Management & IT College, Birtamode, Jhapa.

Date:

ACKNOWLEDGEMENT

According to the course of study of MSc IT final semester determined by SMU a computer project is to be carried out for the partial fulfillment of the requirements for degree of MSc IT. Therefore as a student of this course I have developed an application “Fruit Inventory Management System (FIMS)”.

I feel very glad for getting such an opportunity to accomplish the MSc IT final semester project. This project gave me insight knowledge about the practical aspect of the various stages and procedures of software development project.

This Project Report consist of explanation on the requirements and design of the new system. The requirements are started in different forms such as natural languages and diagrams to provide clear picture of each requirement.

I would like to acknowledge our sincere thanks towards our study center “Niva Management & IT College” for facilitating the completion of entire Project work.

First of all I would like to thank our supervisor, Er. Prakash Gurung sir for his valuable guidance and encouragement throughout the project. I also thank him for helping us out in difficulties by guiding us to the right direction and for the valuable suggestions which are very crucial for the success of our project and department of Computer Application for their valuable guidance and suggestions that have resulted in the successful completion of the project.

I am grateful to all those who have directly or indirectly helped me in completion of the project.

**Tilrupa Timsina**

INTRODUCTION

Inventory Management System (IMS) software is a computer-based system for tracking inventory levels, orders, sales and deliveries. It can also be used in the manufacturing industry to create a work orders, bill materials and other production-related documents. Companies use inventory management software to avoid product overstock and outages. It is a tool for organizing inventory data that before was generally stored in hard-copy form or in spreadsheets.

Fruit Inventory Management System (FIMS) is all about managing the fruit products. Inventory Management System has become all that important in today’s knowledge economy as it deals with commodity requires systematic mode of arrangement of inventory. A business without a proper management of inventories lacks upto date stock levels and the requirements of business. The lack of information of the stocks slows the business transaction on time, non availability of stocks delays the production and operation which leads to dissatisfation of customers. This software helps in proper functioning of the business and upto date inventory details.With C# it gives us the platform to create a software which help organization in managing its fruit product.

The working mechanism of FIMS is similar to that of coding various C# programs. Understanding how FIMS works in different companies gives ideas about how we can code our program in C#. Using different syntax, functions, Windows Forms, and so on we can code Fruit Inventory Management System (FIMS). With a local database can be saved and kept. When any Organization uses this software they would have knowledge to keep information of fruits and various other things.

**Brief History of Inventory Management System (FIMS)**

The Universal Product Code (UPC) was adopted by the grocery industry in April 1973 as the standard barcode for all grocers, though it was not introduced at retailing locations until 1974.This helped drive down costs for inventory management because retailers in the United States and Canada didn’t have to purchase multiple barcode readers to scan competing barcodes. There was now one primary barcode for grocers and other retailers to buy one type of reader for.

In the early 1980s personal computers began to be popular. This further pushed down the cost of barcodes and readers. It also allowed the first versions of inventory management software to be put into place. One of the biggest hurdles in selling readers and barcodes to retailers was the fact that they didn’t have a place to store the information they scanned. As computers became more common and affordable, this hurdle was overcome. Once barcodes and inventory management programs started spreading through Grocery stores. Inventory management by hand became less practical. Writing inventory data by hand on paper was replaced by scanning products and inputting information into a computer by hand.

Starting in the early 2000s, inventory management software progressed to the point where businesspeople no longer needed to input data by hand but could instantly update their database with barcode readers.

Also, the existence of cloud based business software and their increasing adoption by businesses mark a new era for inventory management software. Now they usually allow integrations with other business backend processes, like accounting and online sales.

**ABSTARCT**

Proposed **“Fruit Inventory Management System** (FIMS)**”** is developed for to provide information of fruits in c#. It includes modules required to successfully operate Fruit Informations without any difficulties. It has Admin functionalities to add modify and delete Stock information, sales information. This project has included all reports of stock and sales and all other types of information regarding fruit. It includes validation for forms to make forms distinct and avoid any difficulties.

Admin can modify, delete and updateForm from front end and change in all forms and change reflect in all Database tables immediately.

Therefore proposed “**Fruit Inventory Management System (FIMS)**” has been designed to control Fruit information and activities. This System can make the daily activities efficient and providing the fast response.

It included inbuilt login form where when we give proper login information we can be authenticated to use the application.

## Fruit Inventory Management System (FIMS)Benefits

* Complete information of Fruits for all the activities in the future use.
* Supports addition, update, cancellation and deletion of Fruit Information to Main forms.
* It is a scalable system.
* Has a clearly arranged and user-friendly interface for Admins and anyone who wishes to use it.
* Validation of forms in order to make data clean.
* Calculator, notepad, wordpad, taskmanager and msword for admins to make their job Easier.
* All important details can be updated by front end master module only.
* Password are protected since it is linked to database in login table.
* Strong Security features

**Propose of thesis**

The thesis aims to theoretically study the use of Fruit Inventory Management System (FIMS) and use in practical field using C#. Development of Inventory Management System is essential for any organization that would like to be dynamic and growth-oriented. The objective of the thesis is to plan/design, and theoretically and practically implement use of FIMS in fruit shop that would serve as boon in the fruit shop by which the admin of an organization is helped, in a continuous and planned way to:

1. Acquire or sharpen capabilities required to perform various functions and have future roles;
2. Develop the information of various kinds of fruits which makes the work of both admins and customers easy and
3. Develop fruit shop to extend its business in a good and futuristic ways.

## Scope

## The scope of Fruit Inventory Management System is very wide. : 1. Personnel aspect- This is concerned with manpower planning, and development, layoff and retrenchment, remuneration, incentives, productivity etc. 2. Welfare aspect- It deals with working conditions and amenities such as canteens, crèches, rest and lunch rooms, housing, transport, medical assistance, education, health and safety, recreation facilities, etc. when modified to use in each aspect of business. 3. Industrial relations aspect- This covers union-management relations, joint consultation, collective bargaining, grievance and disciplinary procedures, settlement of disputes, etc.

## Goals

### The fruit inventory management system plays an important role to the success of that organization. It helps admin or owner of that shop to do their work smoothly and fast and have their transaction faster.

**Objective**

* To help the organization reach its goals.
* To improve the quality of life of admins.
* To give the work reponse fast.
* To provide fast business transaction.
* To be able to use information regarding products for future use.

**Requirements**

There are two requirements to be fulfilled for building this project:-

Hardware Requirements:

Processor -1 GHz, RAM- 512

Disk Space (Minimum) - 32 bit (850MB) and 64 bit (1GB).

Software Requirements:

Operating Systems – Windows (for better performance higher versions of windows are used)

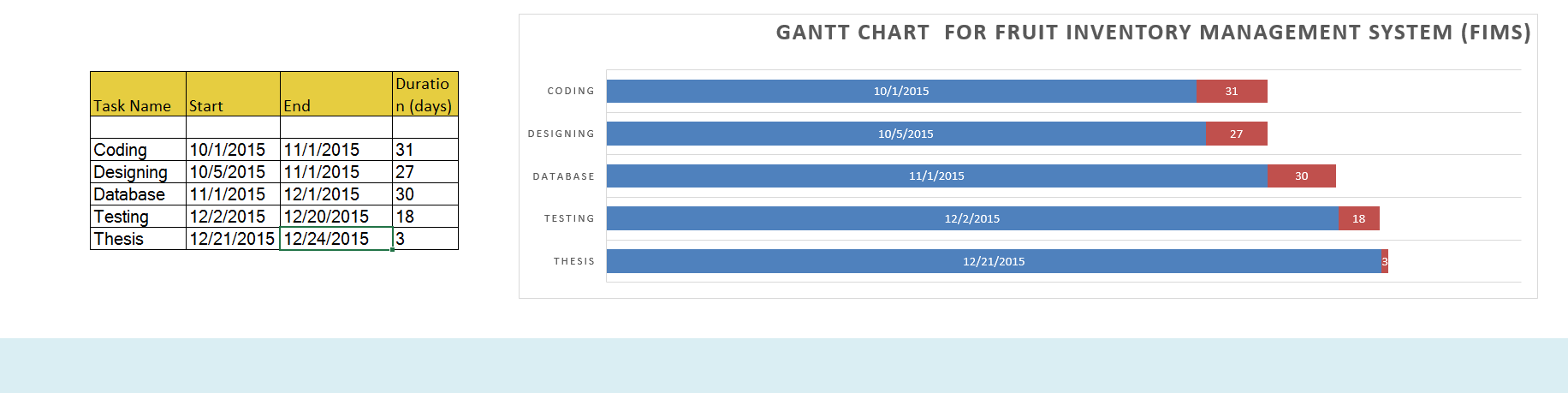
Software **–** Visual Studio 2012 and MySQL workbench.

## Solution Concept

The Fruit Inventory Management System consists of:

* **Login Form**:   
  Here Admins are logged in for use of this Software. First login form is opened and with correct Username and password are given the main form opens and are opened from database for better security reasons.
* **Main Form**: It includes all administrative Privileges like saving, updating, deleting, Viewing data which are connected to database. Here we can export report in Excel, Stock Report and Sales Report for viewing too and modifying them.
* **Validation Class**– A powerful validation is given on forms on both Admins and Employee Forms.
* **Calculator and other accessories**– Calculator and other accessories is Provided to admins for their better Performance.

**Gnatt Chart for Fruit Inventory Management System (FIMS)**

****

## Main Form Process Flow Chart

**Main Form: Admin**

Start

## 

Manage stock and its details Forms and Reports

Manage Sales and its Details and Reports

Sales

Stock

Process, as per Admin rights

Main screen

Saved in Database

End

Save, Update, Delete Employee Forms

Login Success

Login

Not Success

## 

## User Profiles

The following user types are expected for the **Fruit Inventory Management System (FIMS)**:

|  |  |
| --- | --- |
| **User** | **Brief Description of Use Actions** |
| Administrator  (Admin) | Create new Employees, set and alter the role and privilege to the system users for accessing the system resource. Can delete the data. Also responsible to database backup, backend performance. And overall the system performance. |

## User Activity Diagram

Human Resource Management with C# is given following users’ activity diagram scenarios, as represented in the following diagram:

# 

# Development Tools and Technologies

## Front End: C#

**Overview of C#**

C# is a modern, general-purpose, object-oriented programming language developed by Microsoft and approved by European Computer Manufacturers Association (ECMA) and International Standards Organization (ISO).

C# was developed by Anders Hejlsberg and his team during the development of .Net Framework.

C# is designed for Common Language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architectures.

The following reasons make C# a widely used professional language:

* It is a modern, general-purpose programming language
* It is object oriented.
* It is component oriented.
* It is easy to learn.
* It is a structured language.
* It produces efficient programs.
* It can be compiled on a variety of computer platforms.
* It is a part of .Net Framework.

**Strong Programming Features of C#**

Although C# constructs closely follow traditional high-level languages, C and C++ and being an object-oriented programming language. It has strong resemblance with Java, it has numerous strong programming features that make it endearing to a number of programmers worldwide.

Following is the list of few important features of C#:

* Boolean Conditions
* Automatic Garbage Collection
* Standard Library
* Assembly Versioning
* Properties and Events
* Delegates and Events Management
* Easy-to-use Generics
* Indexers
* Conditional Compilation
* Simple Multithreading
* LINQ and Lambda Expressions
* Integration with Windows
* Conditional compilation.
* C# is pure object-oriented.
* Formalized concept of get-set methods, so the code becomes more legible.
* More clean events management (using delegates)’

**Namespaces in C#**

Namespace is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.Defining a Namespace: A namespace definition begins with the keyword namespace followed by the namespace name as follows:

namespace namespace\_name

{

// code declarations

}

**Interfaces in C#**

An interface is defined as a syntactical contract that all the classes inheriting the interface should follow. The interface defines the 'what' part of the syntactical contract and the deriving classes define the 'how' part of the syntactical contract. Interfaces define properties, methods, and events, which are the members of the interface. Interfaces contain only the declaration of the members. It is the responsibility of the deriving class to define the members. It often helps in providing a standard structure that the deriving classes would follow. Abstract classes to some extent serve the same purpose, however, they are mostly used when only few methods are to be declared by the base class and the deriving class implements the functionalities.Declaring Interfaces: Interfaces are declared using the interface keyword. It is similar to class declaration. Interface statements are public by default. Following is an example of an interface declaration:

public interface ITransactions

{

// interface members

void showTransaction();

double getAmount();

}

**Class in C#**

When we define a class, you define a blueprint for a data type. This does not actually define any data, but it does define what the class name means. That is, what an object of the class consists of and what operations can be performed on that object. Objects are instances of a class. The methods and variables that constitute a class are called members of the class. Defining a Class: A class definition starts with the keyword class followed by the class name; and the class body enclosed by a pair of curly braces. Following is the general form of a class definition:

<access specifier> class class\_name

{

// member variables

<access specifier> <data type> variable1;

<access specifier> <data type> variable2;

...

<access specifier> <data type> variableN;

// member methods

<access specifier> <return type> method1(parameter\_list)

{

// method body

}

<access specifier> <return type> method2(parameter\_list)

{

// method body

}

...

<access specifier> <return type> methodN(parameter\_list)

{

// method body

}

}

## Back End

MySQL workbench  
  
**Why MySQL Server?**

Ans: We use My SQL Server for the following reasons: -  
High performance and scalability**:** In many situations, MySQL Server offers better performance than an Access database. MySQL Server also provides support for very large databases, up to one terabyte, which is much larger than the current limit for an Access database of two gigabytes. Finally, MySQL Server works very efficiently on Microsoft Windows and various linux and OS Servers by processing queries in parallel (using multiple native threads within a single process to handle user requests) and minimizing additional memory requirements when more users are added.

**Increased availability**

MySQL Server databases, can be backed, either incremental or complete, while the database is in use. Consequently, you do not have to force users to exit the database to back up data. This means your database can be running up to 24 hours a day, seven days a week.

**Improved security**

MySQL Server can integrate with any operating system security to provide a single log on to the network and the database. This makes it much easier for you to administer complex security schemes. An MySQL Server database on a server is also better protected because unauthorized users can't get to the database file directly but must access the server first. Inbuilt Access security has been removed in later versions. Relying on application security features in Access is never as secure as in MySQL Server.

**Immediate recoverability**

In case of system failure (such as an operating system crash or power outage), MySQL Server has an automatic recovery mechanism that recovers a database to the last state of consistency in a matter of minutes, with no database administrator intervention. Critical applications can be up and running again right away.

**Reliable distributed data and transactions**

Transaction processing is a vital requirement for a system that is designed to support critical applications, such as banking and online order entry. MySQL Server supports atomic transactions with transaction logging, which guarantees that all changes performed within a transaction are either committed or rolled back.

Consistency and recoverability of a database transaction are guaranteed even in the case of system failure and in the middle of complex updates by more than one user. MySQL Server treats all database changes inside a transaction as a single unit of work. By definition, either an entire transaction is completed safely and all resulting changes are reflected in the database, or the transaction is rolled back—and all changes to the database are undone.

Using a two-phase commit protocol, MySQL Server can even support synchronized transactions that span more than one server—ensuring that all servers on the network are maintained in a consistent state.

**Server-based processing**

MySQL Server from the beginning as a client/server database. Data and indexes reside on a single server computer that is often accessed over the network by many client computers. MySQL Server reduces network traffic by processing database queries on the server before sending results to the client. Thus client/server application can do processing where it's done best - on the server.

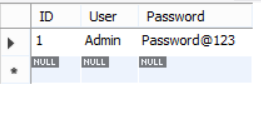
Applications can also use stored procedures and triggers to centralize and share application logic, business rules and policies, complex queries, and data validation and referential integrity code on the server, rather than on the client.

**Development Environment**

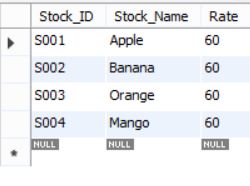
1. Visual Studio 2013.
2. MySQL Workbench.

# Fruit Inventory Management System (FIMS) Data Dictionary

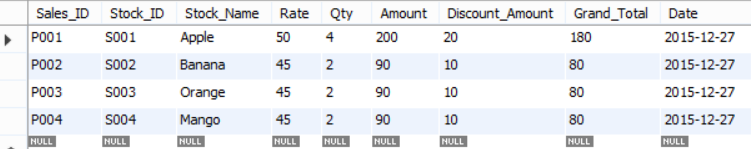
**Table: login**

****

**Table: Stock**

****

**Table: Sales**

****

# 

# Fruit Inventory Management System (FIMS) – Data Schema ER Diagram

# C:\Users\Hacker-PC\Desktop\ER1.PNG

# Fruit Inventory Management (FIMS):DFD (Data Flow Diagram)

## Top Level DFD

Stock

Report

Sales

Accounts

## 1st Level DFD

Admins

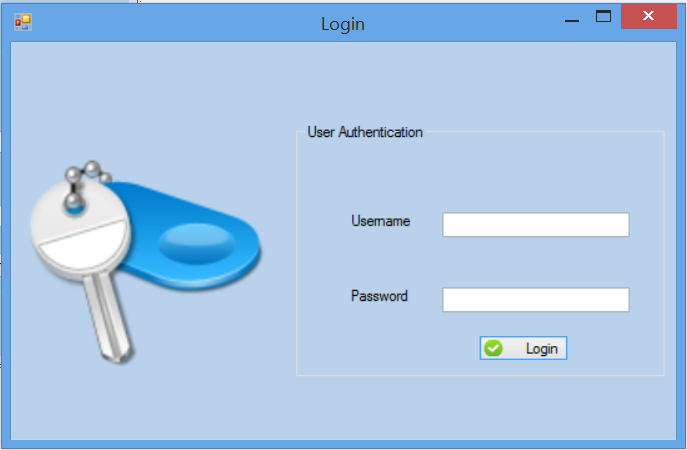
Stock Details

Sales View All

Stock View All

# Coding and Interface

## Fruit Inventory Management System: Login



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Fruit\_Inventory\_Management\_System

{

public partial class login : Form

{

public login()

{

InitializeComponent();

}

private void btnLogin\_Click(object sender, EventArgs e)

{

try

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand= new MySqlCommand("select \* from fims.login where User='" + txtUsername.Text + "' and Password='" + txtPassword.Text+"';", myConn);

MySqlDataReader myReader;

myConn.Open();

myReader = SelectCommand.ExecuteReader();

int count = 0;

while (myReader.Read())

{

count = count + 1;

}

if (count == 1)

{

MessageBox.Show("Username and Password are correct");

new Main\_Form().Show();

this.Hide();

}

else if (count > 1)

{

MessageBox.Show("Duplicate username and password, Accesss Denied");

}

else

{

MessageBox.Show("Username and Password are incorrect");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

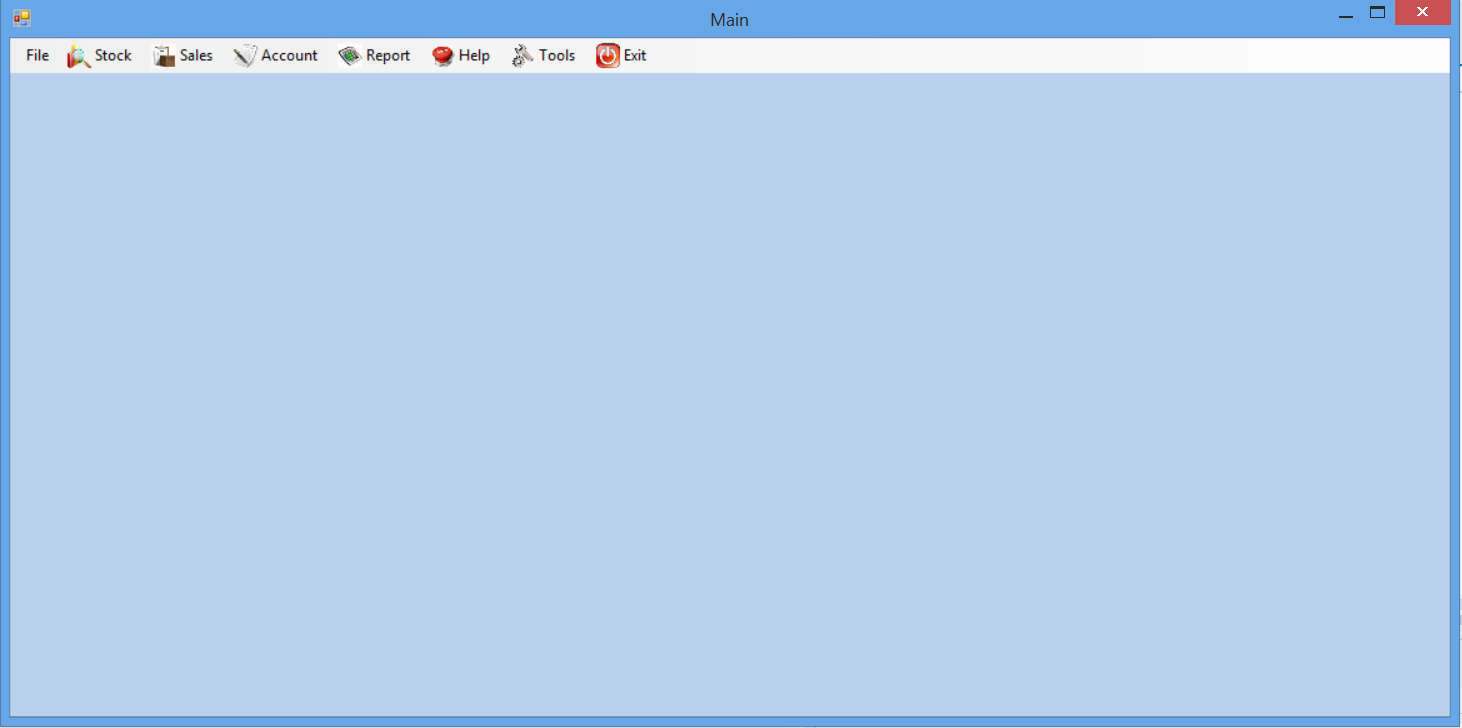
}

}

}

}

## Fruit Inventory Management System: Main



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Fruit\_Inventory\_Management\_System

{

public partial class Main\_Form : Form

{

public Main\_Form()

{

InitializeComponent();

}

private void exitToolStripMenuItem1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void newStockToolStripMenuItem\_Click(object sender, EventArgs e)

{

new Stock.NewStock().Show();

}

private void viewStockToolStripMenuItem\_Click(object sender, EventArgs e)

{

new Stock.StockViewAll().Show();

}

private void newSalesToolStripMenuItem\_Click(object sender, EventArgs e)

{

new Sales.NewSales().Show();

}

private void viewSalesToolStripMenuItem\_Click(object sender, EventArgs e)

{

new Sales.SalesViewAll().Show();

}

private void salesToolStripMenuItem1\_Click(object sender, EventArgs e)

{

new Report.DailySalesReport().Show();

}

private void stockToolStripMenuItem1\_Click(object sender, EventArgs e)

{

new Report.DailyStockReport().Show();

}

private void salesToolStripMenuItem2\_Click(object sender, EventArgs e)

{

new Report.MonthlySalesReport().Show();

}

private void stockToolStripMenuItem2\_Click(object sender, EventArgs e)

{

new Report.MonthlyStockReport().Show();

}

private void calculatorToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("calc.exe");

}

private void notepadToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("Notepad.exe");

}

private void wordpadToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("Wordpad.exe");

}

private void taskManagerToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("TaskMgr.exe");

}

private void msWordToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("Winword.exe");

}

private void billingToolStripMenuItem\_Click(object sender, EventArgs e)

{

new Accounts.Billing().ShowDialog();

}

private void aboutUsToolStripMenuItem1\_Click(object sender, EventArgs e)

{

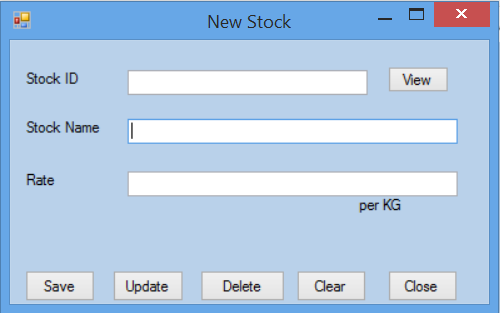
new AboutUs().ShowDialog();

}

}

}

## Fruit Inventory Management System: New Stock



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Fruit\_Inventory\_Management\_System.Stock

{

public partial class NewStock : Form

{

public NewStock()

{

InitializeComponent();

}

private void txtName\_KeyPress(object sender, KeyPressEventArgs e)

{

validation.IsAlpha(e);

}

private void txtRate\_KeyPress(object sender, KeyPressEventArgs e)

{

validation.IsInteger(e);

}

private void btnSave\_Click(object sender, EventArgs e)

{

if (this.txtStockID.Text != "")

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand = new MySqlCommand("insert into fims.stock(Stock\_ID,Stock\_Name,Rate)values('" + txtStockID.Text + "','" + txtName.Text + "','" + txtRate.Text + "')", myConn);

try

{

myConn.Open();

SelectCommand.ExecuteNonQuery();

}

catch (MySqlException ex) { throw ex; }

finally { myConn.Close(); }

MessageBox.Show("Congratulations!!!, Your Data is Saved");

}

else

{

MessageBox.Show("Enter All the Fields");

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnClear\_Click(object sender, EventArgs e)

{

foreach (Control b in Controls)

{

txtStockID.Text = "";

txtName.Text = "";

txtRate.Text = "";

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand = new MySqlCommand("delete from fims.stock WHERE (Sales\_ID = '" + this.txtStockID.Text + "')", myConn);

myConn.Open();

SelectCommand.ExecuteNonQuery();

myConn.Close();

MessageBox.Show("your data is Deleted");

this.Refresh();

}

private void btnUpdate\_Click(object sender, EventArgs e)

{

if (txtStockID.Text != "")

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand = new MySqlCommand("update fims.stock set Stock\_Name='" + this.txtName.Text + "',Rate='" + this.txtRate.Text + "'", myConn);

try

{

myConn.Open();

SelectCommand.ExecuteNonQuery();

}

catch (MySqlException ex) { throw ex; }

finally { myConn.Close(); }

MessageBox.Show("Data is Updated");

this.Refresh();

}

else

{

MessageBox.Show("Enter All the Fields");

}

}

private void btnView\_Click(object sender, EventArgs e)

{

if (txtStockID.Text != "")

{

try

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand = new MySqlCommand("select \* from fims.stock where Stock\_ID ='" + this.txtStockID.Text + "'", myConn);

MySqlDataAdapter adp = new MySqlDataAdapter();

adp.SelectCommand = SelectCommand;

DataTable dt = new DataTable();

adp.Fill(dt);

// now get the values

this.txtName.Text = dt.Rows[0][1].ToString();

this.txtRate.Text = dt.Rows[0][2].ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

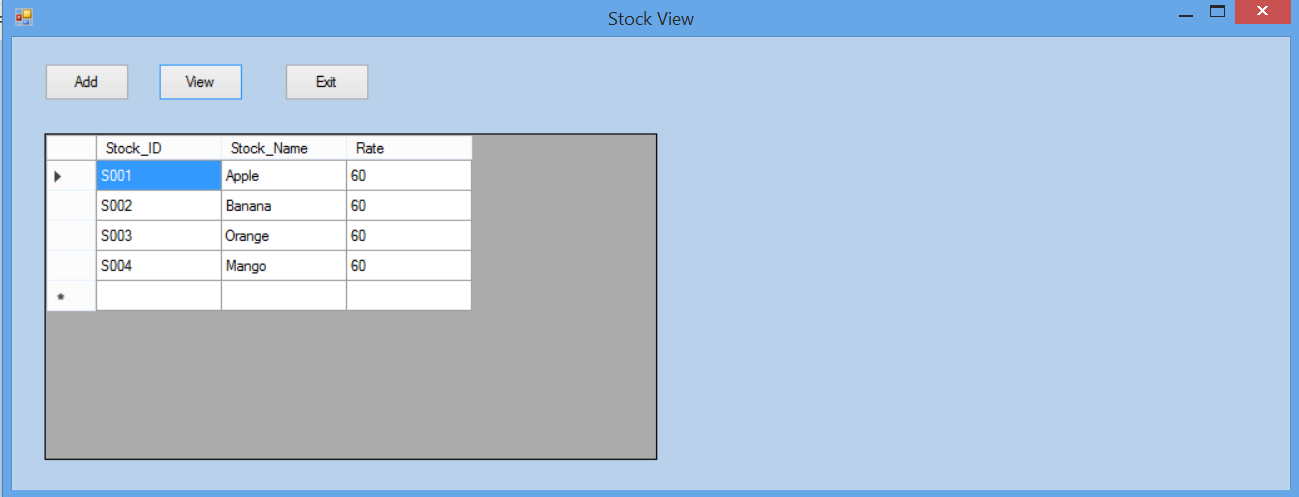
}

}

}

}

## Fruit Inventory Management System: Stock ViewAll



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Fruit\_Inventory\_Management\_System.Stock

{

public partial class StockViewAll : Form

{

public StockViewAll()

{

InitializeComponent();

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

new NewStock().Show();

}

private void btnView\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand command = new MySqlCommand();

command.Connection = myConn;

command.CommandText = "select \* from fims.stock";

DataTable data = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.Fill(data);

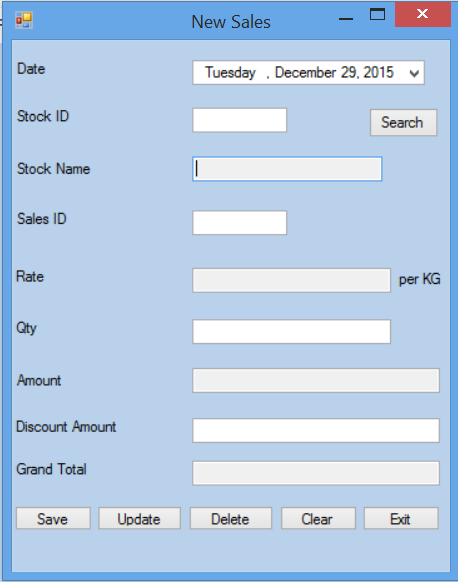
dataGridView1.DataSource = data;

}

}

}

## Fruit Inventory Management System: Sales



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Fruit\_Inventory\_Management\_System.Sales

{

public partial class NewSales : Form

{

public NewSales()

{

InitializeComponent();

}

private void txtQty\_TextChanged(object sender, EventArgs e)

{

try

{

double val1 = 0;

int val2 = 0;

double.TryParse(txtRate.Text, out val1);

int.TryParse(txtQty.Text, out val2);

double I = (val1 \* val2);

txtAmount.Text = I.ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void txtDiscountAmt\_TextChanged(object sender, EventArgs e)

{

try

{

double val3 = 0;

int val4 = 0;

double.TryParse(txtAmount.Text, out val3);

int.TryParse(txtDiscountAmt.Text, out val4);

double I = (val3 - val4);

txtGrandTotal.Text = I.ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void txtQty\_KeyPress(object sender, KeyPressEventArgs e)

{

validation.IsQuantity(e);

}

private void txtDiscountAmt\_KeyPress(object sender, KeyPressEventArgs e)

{

validation.IsInteger(e);

}

private void btnSearch\_Click(object sender, EventArgs e)

{

if (txtStockID.Text != "")

{

try

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand = new MySqlCommand("select \* from fims.stock where Stock\_ID ='" + this.txtStockID.Text + "'", myConn);

MySqlDataAdapter adp = new MySqlDataAdapter();

adp.SelectCommand = SelectCommand;

DataTable dt = new DataTable();

adp.Fill(dt);

// now get the values

this.txtStockName.Text = dt.Rows[0][1].ToString();

this.txtRate.Text = dt.Rows[0][2].ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnSave\_Click(object sender, EventArgs e)

{

if (this.txtSalesID.Text != "")

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

string dt = dateTime\_Date.Text;

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand = new MySqlCommand("insert into fims.sales(Sales\_ID,Stock\_ID,Stock\_Name,Rate,Qty,Amount,Discount\_Amount,Grand\_Total,Date)values('" + txtSalesID.Text + "','" + txtStockID.Text + "','" + txtStockName.Text + "','" + txtRate.Text + "','" + txtQty.Text + "','" + txtAmount.Text + "','" + txtDiscountAmt.Text + "','" + txtGrandTotal.Text + "','" + dateTime\_Date.Text + "')", myConn);

try

{

myConn.Open();

SelectCommand.ExecuteNonQuery();

}

catch (MySqlException ex) { throw ex; }

finally { myConn.Close(); }

MessageBox.Show("Congratulations!!!, Your Data is Saved");

}

else

{

MessageBox.Show("Enter All the Fields");

}

}

private void btnClear\_Click(object sender, EventArgs e)

{

foreach (Control b in Controls)

{

txtStockID.Text = "";

txtStockName.Text = "";

txtRate.Text = "";

txtSalesID.Text = "";

txtQty.Text = "";

txtAmount.Text="";

txtDiscountAmt.Text = "";

txtGrandTotal.Text = "";

}

}

private void btnUpdate\_Click(object sender, EventArgs e)

{

if (txtSalesID.Text != "")

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

string dt = dateTime\_Date.Text;

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand = new MySqlCommand("update fims.sales set Qty='" + this.txtQty.Text + "',Discount\_Amount='" + this.txtDiscountAmt.Text + "',Date='" + this.dateTime\_Date.Text + "'", myConn);

try

{

myConn.Open();

SelectCommand.ExecuteNonQuery();

}

catch (MySqlException ex) { throw ex; }

finally { myConn.Close(); }

MessageBox.Show("Data is Updated");

this.Refresh();

}

else

{

MessageBox.Show("Enter All the Fields");

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand SelectCommand = new MySqlCommand("delete from fims.sales WHERE (Sales\_ID = '" + this.txtSalesID.Text + "')", myConn);

myConn.Open();

SelectCommand.ExecuteNonQuery();

myConn.Close();

MessageBox.Show("your data is Deleted");

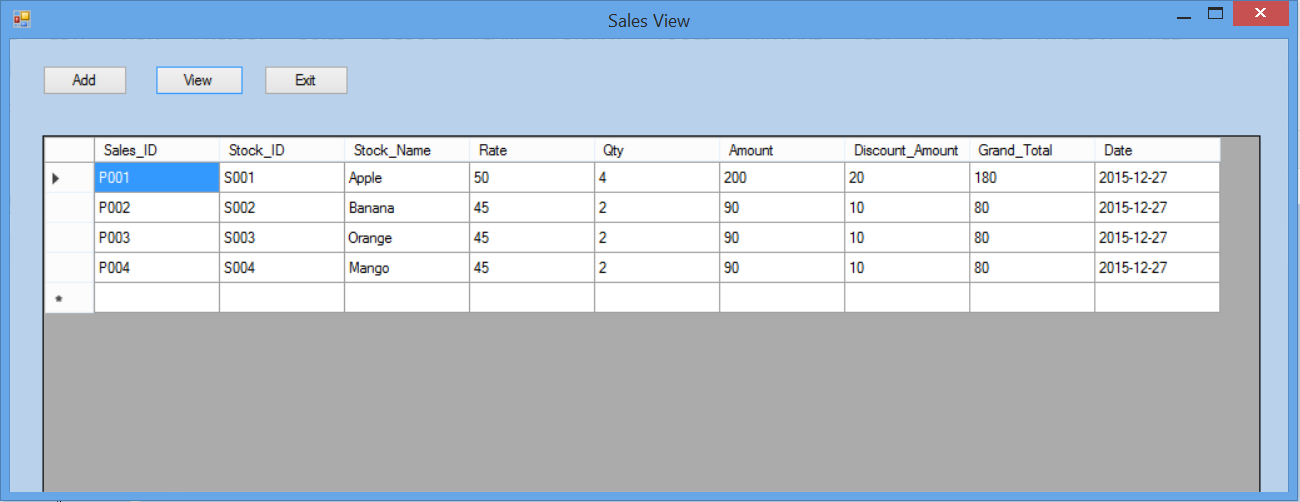
this.Refresh();

}

}

}

## Fruit Inventory Management System: Sales View



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Fruit\_Inventory\_Management\_System.Sales

{

public partial class SalesViewAll : Form

{

public SalesViewAll()

{

InitializeComponent();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

new NewSales().Show();

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnSearch\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand command = new MySqlCommand();

command.Connection = myConn;

command.CommandText = "select \* from fims.sales";

DataTable data = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.Fill(data);

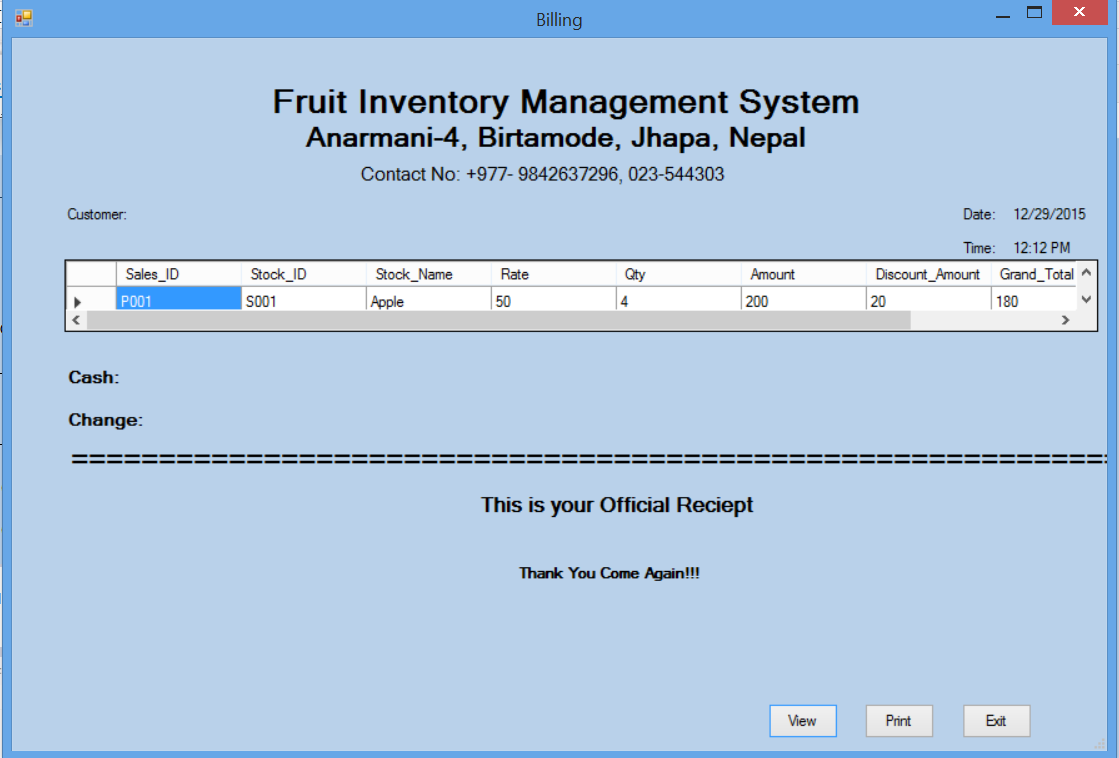
dataGridView1.DataSource = data;

}

}

}

## Fruit Inventory Management System: Billing



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing.Printing;

using MySql.Data.MySqlClient;

namespace Fruit\_Inventory\_Management\_System.Accounts

{

public partial class Billing : Form

{

public Billing()

{

InitializeComponent();

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnView\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand command = new MySqlCommand();

command.Connection = myConn;

command.CommandText = "select \* from fims.sales";

DataTable data = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.Fill(data);

dataGridView1.DataSource = data;

}

private void myPrintDocument1\_PrintPage(System.Object sender, System.Drawing.Printing.PrintPageEventArgs e)

{

Bitmap myBitmap1 = new Bitmap(panel1.Width, panel1.Height);

panel1.DrawToBitmap(myBitmap1, new Rectangle(0, 0, panel1.Width, panel1.Height));

e.Graphics.DrawImage(myBitmap1, 0, 0);

myBitmap1.Dispose();

}

private void btnPrint\_Click(object sender, EventArgs e)

{

System.Drawing.Printing.PrintDocument myPrintDocument1 = new System.Drawing.Printing.PrintDocument();

PrintDialog myPrinDialog1 = new PrintDialog();

myPrintDocument1.PrintPage += new System.Drawing.Printing.PrintPageEventHandler(myPrintDocument1\_PrintPage);

myPrinDialog1.Document = myPrintDocument1;

if (myPrinDialog1.ShowDialog() == DialogResult.OK)

{

myPrintDocument1.Print();

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

lblTime.Text = DateTime.Now.ToShortTimeString();

}

private void Billing\_Load(object sender, EventArgs e)

{

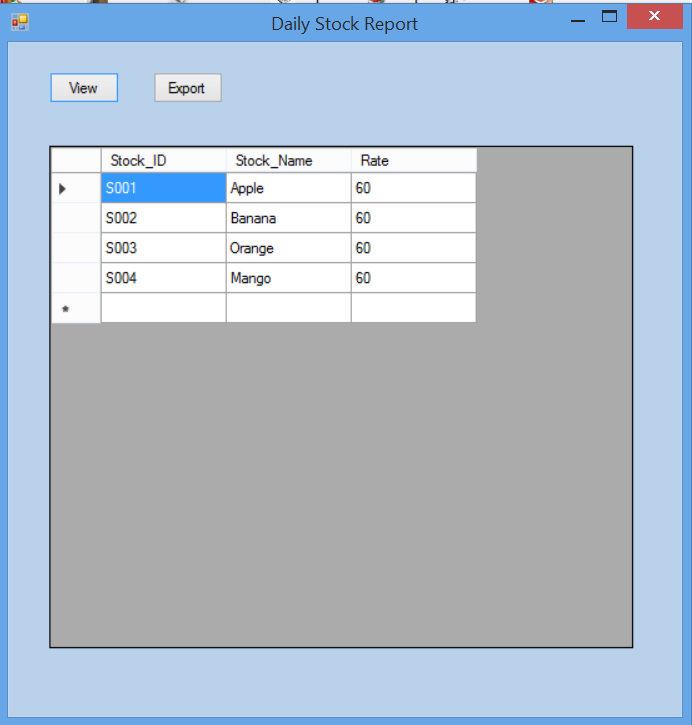
lblDate.Text = DateTime.Now.ToShortDateString();

}

}

}

## Fruit Inventory Management System: Daily Stock Report



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using Microsoft.Office.Interop.Excel;

namespace Fruit\_Inventory\_Management\_System.Report

{

public partial class DailyStockReport : Form

{

public DailyStockReport()

{

InitializeComponent();

}

private void DailyStockReport\_Load(object sender, EventArgs e)

{

}

private void btnView\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand command = new MySqlCommand();

command.Connection = myConn;

command.CommandText = "select \* from fims.stock";

System.Data.DataTable data = new System.Data.DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.Fill(data);

dataGridView1.DataSource = data;

}

private void btnExport\_Click(object sender, EventArgs e)

{

// Creating a Excel object.

Microsoft.Office.Interop.Excel.\_Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.\_Workbook workbook = excel.Workbooks.Add(Type.Missing);

Microsoft.Office.Interop.Excel.\_Worksheet worksheet = null;

try

{

worksheet = workbook.ActiveSheet;

worksheet.Name = "ExportedFromDatGrid";

int cellRowIndex = 1;

int cellColumnIndex = 1;

//Loop through each row and read value from each column.

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

for (int j = 0; j < dataGridView1.Columns.Count; j++)

{

// Excel index starts from 1,1. As first Row would have the Column headers, adding a condition check.

if (cellRowIndex == 1)

{

worksheet.Cells[cellRowIndex, cellColumnIndex] = dataGridView1.Columns[j].HeaderText;

}

else

{

worksheet.Cells[cellRowIndex, cellColumnIndex] = dataGridView1.Rows[i].Cells[j].Value.ToString();

}

cellColumnIndex++;

}

cellColumnIndex = 1;

cellRowIndex++;

}

//Getting the location and file name of the excel to save from user.

SaveFileDialog saveDialog = new SaveFileDialog();

saveDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveDialog.FilterIndex = 2;

if (saveDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

workbook.SaveAs(saveDialog.FileName);

MessageBox.Show("Export Successful");

}

}

catch (System.Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

excel.Quit();

workbook = null;

excel = null;

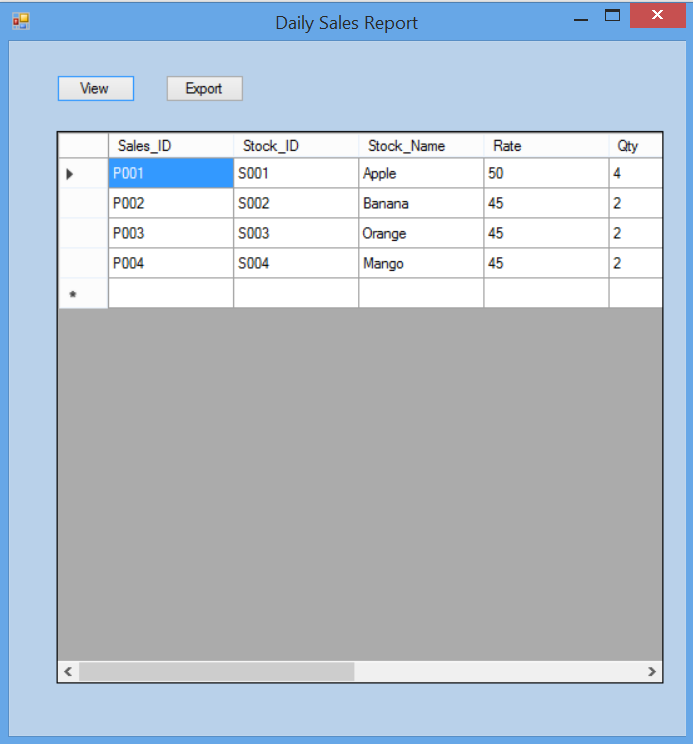
}

}

}

}

## Fruit Inventory Management System: Daily Sales Report



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using Microsoft.Office.Interop.Excel;

namespace Fruit\_Inventory\_Management\_System.Report

{

public partial class DailySalesReport : Form

{

public DailySalesReport()

{

InitializeComponent();

}

private void btnView\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand command = new MySqlCommand();

command.Connection = myConn;

command.CommandText = "select \* from fims.sales";

System.Data.DataTable data = new System.Data.DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.Fill(data);

dataGridView1.DataSource = data;

}

private void btnExport\_Click(object sender, EventArgs e)

{

// Creating a Excel object.

Microsoft.Office.Interop.Excel.\_Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.\_Workbook workbook = excel.Workbooks.Add(Type.Missing);

Microsoft.Office.Interop.Excel.\_Worksheet worksheet = null;

try

{

worksheet = workbook.ActiveSheet;

worksheet.Name = "ExportedFromDatGrid";

int cellRowIndex = 1;

int cellColumnIndex = 1;

//Loop through each row and read value from each column.

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

for (int j = 0; j < dataGridView1.Columns.Count; j++)

{

// Excel index starts from 1,1. As first Row would have the Column headers, adding a condition check.

if (cellRowIndex == 1)

{

worksheet.Cells[cellRowIndex, cellColumnIndex] = dataGridView1.Columns[j].HeaderText;

}

else

{

worksheet.Cells[cellRowIndex, cellColumnIndex] = dataGridView1.Rows[i].Cells[j].Value.ToString();

}

cellColumnIndex++;

}

cellColumnIndex = 1;

cellRowIndex++;

}

//Getting the location and file name of the excel to save from user.

SaveFileDialog saveDialog = new SaveFileDialog();

saveDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveDialog.FilterIndex = 2;

if (saveDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

workbook.SaveAs(saveDialog.FileName);

MessageBox.Show("Export Successful");

}

}

catch (System.Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

excel.Quit();

workbook = null;

excel = null;

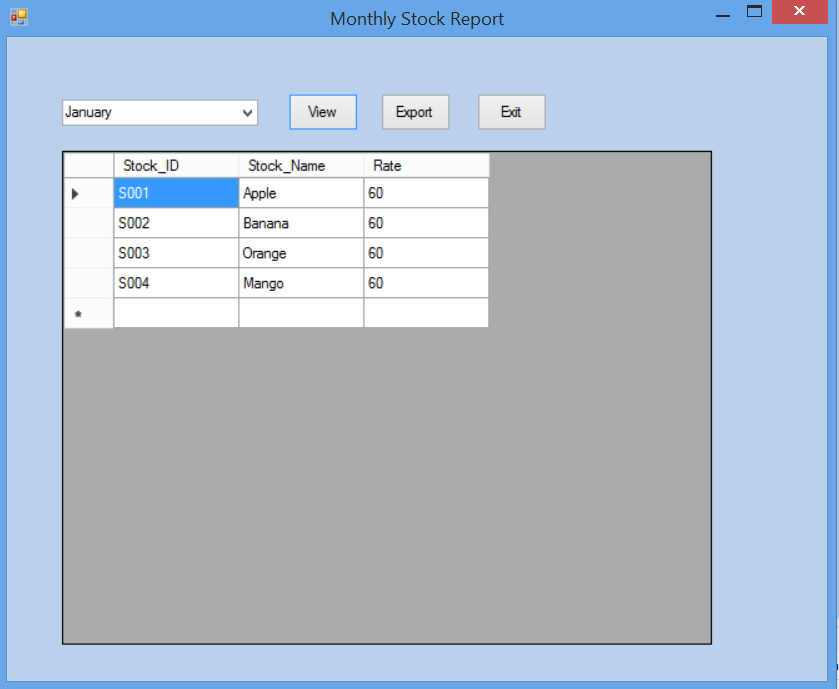
}

}

}

}

## Fruit Inventory Management System: Monthly Stock Report



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using Microsoft.Office.Interop.Excel;

namespace Fruit\_Inventory\_Management\_System.Report

{

public partial class MonthlyStockReport : Form

{

public MonthlyStockReport()

{

InitializeComponent();

}

private void btnExport\_Click(object sender, EventArgs e)

{

// Creating a Excel object.

Microsoft.Office.Interop.Excel.\_Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.\_Workbook workbook = excel.Workbooks.Add(Type.Missing);

Microsoft.Office.Interop.Excel.\_Worksheet worksheet = null;

try

{

worksheet = workbook.ActiveSheet;

worksheet.Name = "ExportedFromDatGrid";

int cellRowIndex = 1;

int cellColumnIndex = 1;

//Loop through each row and read value from each column.

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

for (int j = 0; j < dataGridView1.Columns.Count; j++)

{

// Excel index starts from 1,1. As first Row would have the Column headers, adding a condition check.

if (cellRowIndex == 1)

{

worksheet.Cells[cellRowIndex, cellColumnIndex] = dataGridView1.Columns[j].HeaderText;

}

else

{

worksheet.Cells[cellRowIndex, cellColumnIndex] = dataGridView1.Rows[i].Cells[j].Value.ToString();

}

cellColumnIndex++;

}

cellColumnIndex = 1;

cellRowIndex++;

}

//Getting the location and file name of the excel to save from user.

SaveFileDialog saveDialog = new SaveFileDialog();

saveDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveDialog.FilterIndex = 2;

if (saveDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

workbook.SaveAs(saveDialog.FileName);

MessageBox.Show("Export Successful");

}

}

catch (System.Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

excel.Quit();

workbook = null;

excel = null;

}

}

private void btnView\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand command = new MySqlCommand();

command.Connection = myConn;

command.CommandText = "select \* from fims.stock";

System.Data.DataTable data = new System.Data.DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.Fill(data);

dataGridView1.DataSource = data;

}

private void btnExit\_Click(object sender, EventArgs e)

{

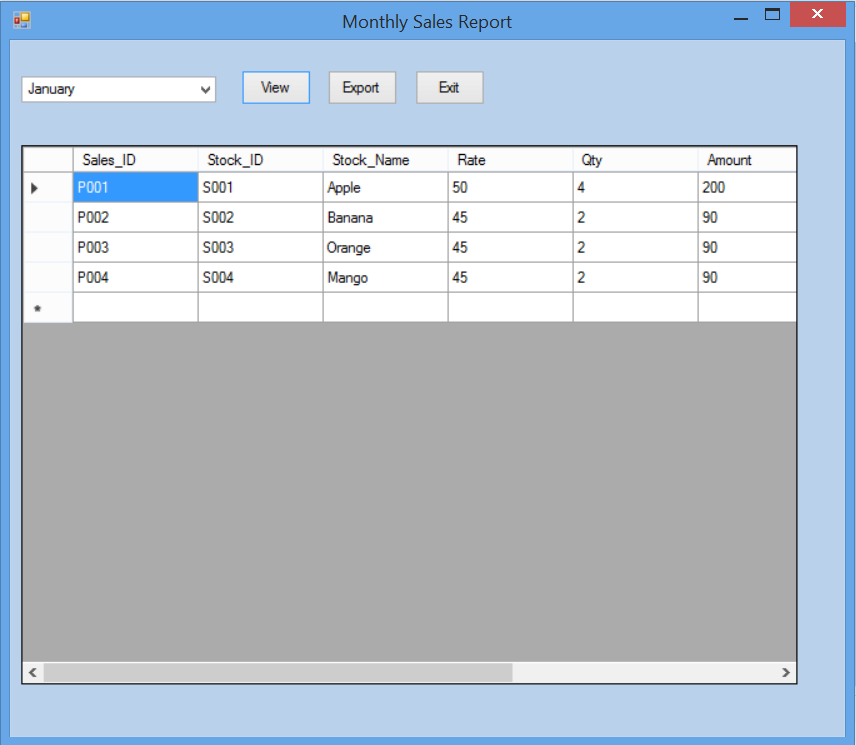
this.Close();

}

}

}

## Fruit Inventory Management System: Monthly Sales Report



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using Microsoft.Office.Interop.Excel;

namespace Fruit\_Inventory\_Management\_System.Report

{

public partial class MonthlySalesReport : Form

{

public MonthlySalesReport()

{

InitializeComponent();

}

private void btnView\_Click(object sender, EventArgs e)

{

string myConnection = "datasource=localhost;port=3306;username=root;password=root";

MySqlConnection myConn = new MySqlConnection(myConnection);

MySqlCommand command = new MySqlCommand();

command.Connection = myConn;

command.CommandText = "select \* from fims.sales";

System.Data.DataTable data = new System.Data.DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.Fill(data);

dataGridView1.DataSource = data;

}

private void btnExit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void btnExport\_Click(object sender, EventArgs e)

{

// Creating a Excel object.

Microsoft.Office.Interop.Excel.\_Application excel = new Microsoft.Office.Interop.Excel.Application();

Microsoft.Office.Interop.Excel.\_Workbook workbook = excel.Workbooks.Add(Type.Missing);

Microsoft.Office.Interop.Excel.\_Worksheet worksheet = null;

try

{

worksheet = workbook.ActiveSheet;

worksheet.Name = "ExportedFromDatGrid";

int cellRowIndex = 1;

int cellColumnIndex = 1;

//Loop through each row and read value from each column.

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

for (int j = 0; j < dataGridView1.Columns.Count; j++)

{

// Excel index starts from 1,1. As first Row would have the Column headers, adding a condition check.

if (cellRowIndex == 1)

{

worksheet.Cells[cellRowIndex, cellColumnIndex] = dataGridView1.Columns[j].HeaderText;

}

else

{

worksheet.Cells[cellRowIndex, cellColumnIndex] = dataGridView1.Rows[i].Cells[j].Value.ToString();

}

cellColumnIndex++;

}

cellColumnIndex = 1;

cellRowIndex++;

}

//Getting the location and file name of the excel to save from user.

SaveFileDialog saveDialog = new SaveFileDialog();

saveDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveDialog.FilterIndex = 2;

if (saveDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

workbook.SaveAs(saveDialog.FileName);

MessageBox.Show("Export Successful");

}

}

catch (System.Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

excel.Quit();

workbook = null;

excel = null;

}

}

}

}

## Fruit Inventory Management System: About Us



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Fruit\_Inventory\_Management\_System

{

public partial class AboutUs : Form

{

public AboutUs()

{

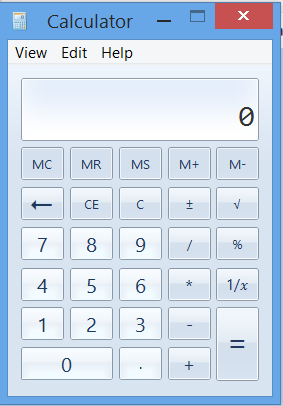
InitializeComponent();

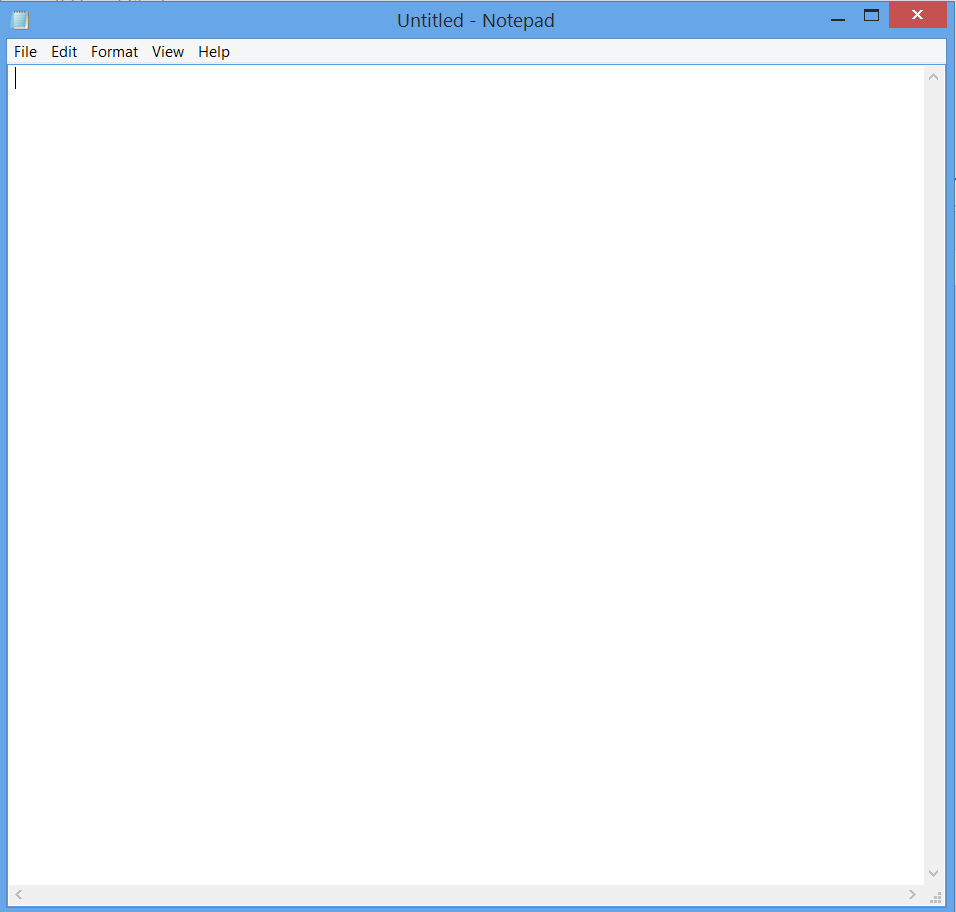
}

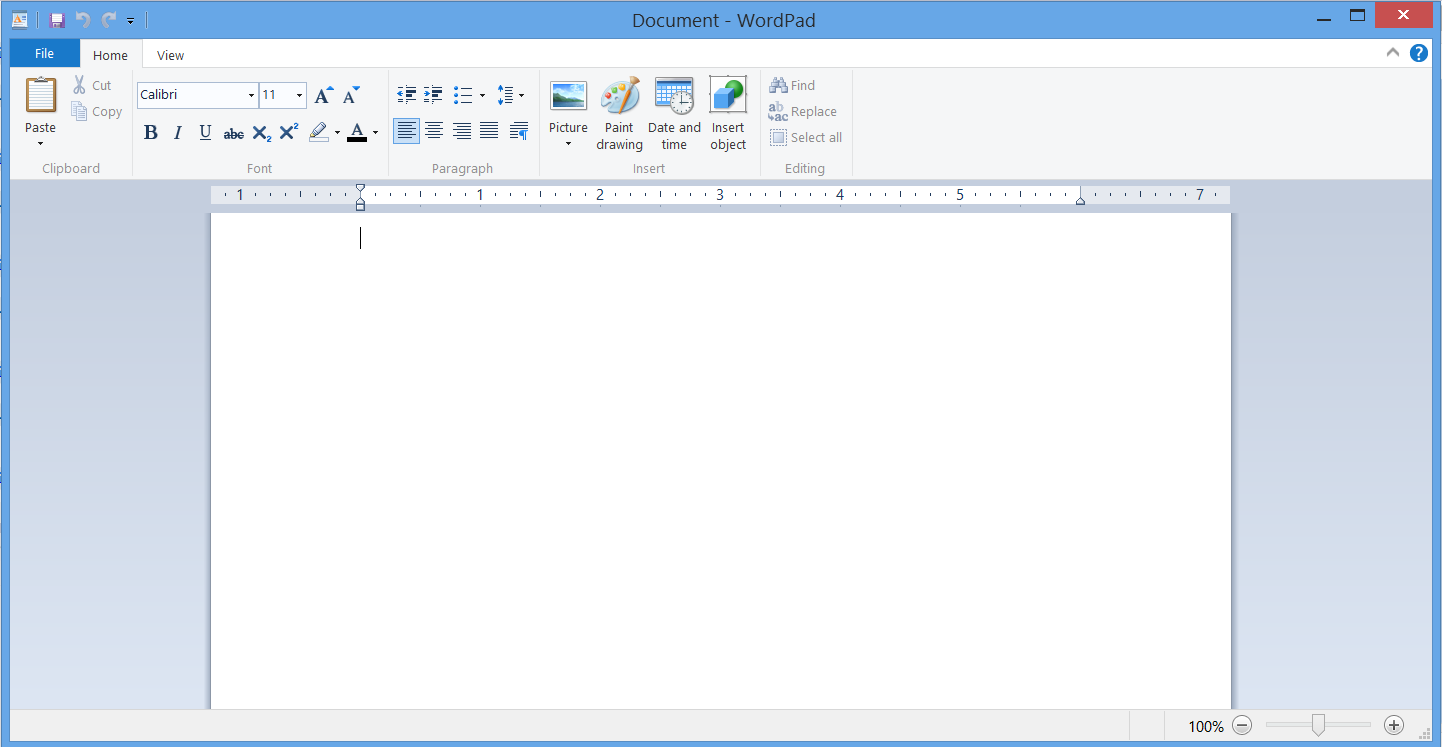
}

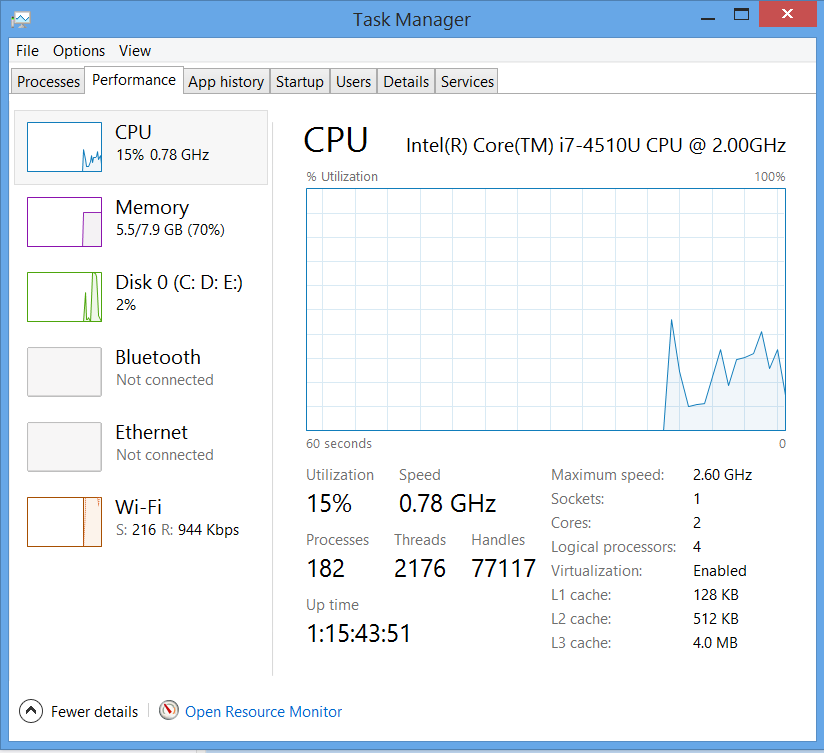
}

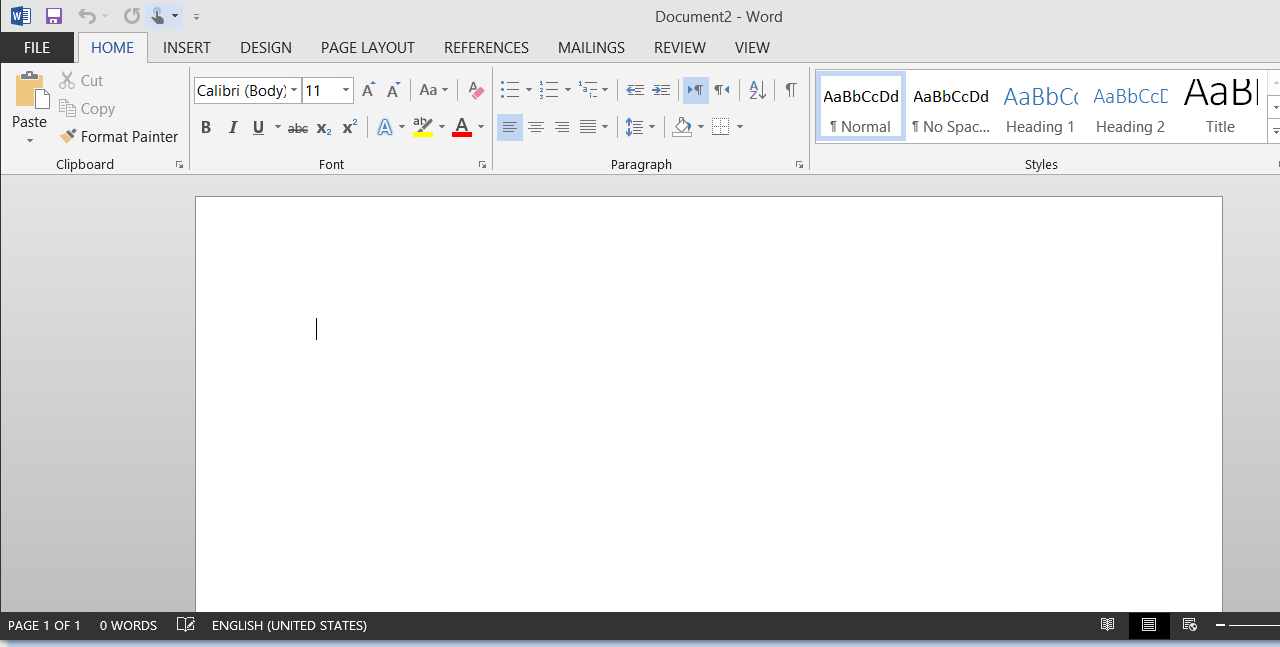
## Fruit Inventory Management System: Tools (System Components)











private void calculatorToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("calc.exe");

}

private void notepadToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("Notepad.exe");

}

private void wordpadToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("Wordpad.exe");

}

private void taskManagerToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("TaskMgr.exe");

}

private void msWordToolStripMenuItem\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("Winword.exe");

}

## Fruit Inventory Management System: Validation Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Fruit\_Inventory\_Management\_System

{

class validation

{

public static void IsAlpha(KeyPressEventArgs e)

{

string strToCheck = Convert.ToString(e.KeyChar);

Regex ObjAlphaPattern = new Regex("^[a-zA-z]|[\b]|[ \t ]|[.]");

if (!ObjAlphaPattern.IsMatch(strToCheck))

{

MessageBox.Show("Please enter only Alphabets");

e.Handled = true;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*(EMAIL VALIDATION)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void IsEmailId(KeyPressEventArgs e)

{

string strToCheck = Convert.ToString(e.KeyChar);

Regex ObjAlphaPattern = new Regex("^[a-zA-Z0-9]|[\b]|[\t]|[.]|[@]");

if (!ObjAlphaPattern.IsMatch(strToCheck))

{

MessageBox.Show("Invalid Email Id");

e.Handled = true;

}

}

public static void checkEmail(string email, KeyPressEventArgs e)

{

// check email

}

public static void IsInteger(KeyPressEventArgs e)

{

string strToCheck = Convert.ToString(e.KeyChar);

Regex ObjIntPattern = new Regex("^[0-9]|[\b]|[ \t]|[+]");

if (!ObjIntPattern.IsMatch(strToCheck))

{

MessageBox.Show("Please enter only number");

e.Handled = true;

}

}

public static void IsAlphaNumeric(KeyPressEventArgs e)

{

string strToCheck = Convert.ToString(e.KeyChar);

Regex ObjAlphaPattern = new Regex("^[a-zA-z0-9]|[\b]|[ \t]");

if (!ObjAlphaPattern.IsMatch(strToCheck))

{

MessageBox.Show("Please write in correct format");

e.Handled = true;

}

}

public static void IsQuantity(KeyPressEventArgs e)

{

string strToCheck = Convert.ToString(e.KeyChar);

Regex ObjAlphaPattern = new Regex("^[0-9]|[.]|[\b]");

if (!ObjAlphaPattern.IsMatch(strToCheck))

{

MessageBox.Show("Please write in correct format");

e.Handled = true;

}

}

}

}

# Human Resource Management with C# Test Plan

## Introduction

This document describes the user acceptance test plan for the Human Resource Management with C#. The complete test strategy for the Human Resource Management with C# is to perform the following kinds of tests, in sequence:

1. **Component testing** of each component that makes up the Human Resource Management with C#.
2. **Integration testing** of the Human Resource Management with C#, to ensure the correct interworking of its components.
3. **Validation testing** of the Human Resource Management with C#, to ensure that it works correctly in a pseudo-live environment.
4. **User acceptance testing** of the Human Resource Management with C#, to ensure that its function is acceptable to its users.

Acceptance testing is the last set of tests to be performed before the application goes officially live.

## Test Strategy

The basis of user acceptance testing is that other tests were completed successfully, so the application and its required infrastructure are considered to be stable and reliable. Acceptance testing concentrates on the application from the user’s perspective, that is, how the application is used and whether it meets the necessary quality criteria.

Change requests will be sent to the development team as the actionable documentation. Change criteria will be determined by the Test team and the Development team prior to the beginning of testing. For instance, criteria may include impact to desired functionality, amount of code impacted by proposed change, and design required by proposed change. The tester will evaluate the criteria. The test lead will determine Change Required or not. Once a bug has been determined as Change Required, the bug report will be translated into a Change Request and passed on to development.

The progress of the acceptance testing will be reported to the Administrator, together with any issues that are discovered and their planned resolutions. Log Out of the tests, and therefore the acceptance of the application, will be performed by the Administrator or a selected representative.

## Preconditions

The following items are required before testing can take place:

* A complete and coherent functional specification of the Human Resource Management with C# expressed as usage scenarios.
* A complete and validation-tested release of the Human Resource Management with C#, delivered according to the delivery plan.
* An agreed-upon procedure for dealing with any anomalies that are discovered during the testing process.
* A set of test specifications describing how each functional area of the the Human Resource Management with C# is to be acceptance tested.
* An implemented test environment for the testing.
* Sufficient, suitable resources to carry out the testing.
* Available standards for the acceptance testing.

## Test Priorities

During testing of the Railway Reservation System, the following qualities will be tested in order of priority:

* Functionality—whether the required functions are available and working as expected.
* Usability—how user-friendly and intuitive the Human Resource Management with C# is.
* Security—how well-protected and guaranteed Admin and Employee data is.
* Performance—whether the response times are within acceptable limits
* Customization—how straightforward it is to use the application in new, unpredicted ways

## Test Techniques

The following techniques will be applied:

* Scripted tests—sequences of user interactions (based on the usage scenarios) using predefined data sets against predicted results
* Unscripted tests—based on scripted tests, the tester tries to modify the scenarios to explore what-if possibilities
* Penetration tests—scripted tests to attempt unauthorized entry into the system
* Usability checklists—tests to determine the complexity of interactions
* Performance statistics—generation of performance information to check against desired performance criteria

## Test Organization

## Roles and Responsibilities

The following roles are defined:

* QA lead/test manager—responsible for planning and ensuring the smooth running of the test process.
* Tester—carries out the tests according to the test plan, and then reports the results.
* Product manager—ensures that the tests are carried out successfully from a user perspective.
* Project sponsor/client—acts as main stakeholder, and ensures that the needs of the customer community as a whole are considered.
* Test support—provides technical assistance, such as test environment configuration, and non-technical assistance, such as methodological support.

Weekly team meetings will be held involving the test manager, testers, and product managers. At these meetings, the progress of the testing process will be reported, any issues will be discussed, and actions will be agreed upon.

## Deliverables

The following deliverables will be expected from the user acceptance testing process:

* Test plan—this document, together with any updates that have occurred during the testing process
* Change requests—any bugs, defects, or other changes required to the Railway Reservation System as a result of the testing process
* Weekly reports—progress reports to enable the status of the testing process to be determined
* Completion report—a report to be signed off by the customer, to signify the successful completion of the user acceptance testing

## Test Environment

## Hardware and Software

The test environment will consist of:

**Hardware Requirements:** Processor -1 GHz, RAM- 512, Disk Space (Minimum) - 32 bit (850MB) and 64 bit (1GB).

**Software Requirements** :Operating Systems – Windows (for better performance higher versions of windows are used)

Software **–** Visual Studio 2013 and Database &Microsoft SQL server 2012 and above.

## Testing Automation Software

No testing automation software packages are selected at present.

## Application Configuration

The following user accounts will be configured on the Form Itself.

## Test Management

Tests shall be managed according to the corporate test management standards, which cover:

* Conduct of tests
* Reporting of test results
* Defect tracking and resolution
* Configuration management of the test environment
* Configuration control of test deliverables.

## Testing Schedules

The user acceptance testing schedules are shown in the project structure document and resulting Gantt charts.

## Threats to Testing

Potential threats to the testing process are as follows:

* **Insufficient resources available for testing.** Testing resources have been seconded from the development departments, whose time is at a premium. Mitigation: ensure department heads apply a high priority to the testing of Human Resource Management with C#.
* **Availability of sales personnel for testing.** The test team should be overseen by at least one sales representative. Mitigation: gain prior agreement from the vice president of Sales for two sales representatives to be assigned to test the application.

# Conclusion

This is my first step towards my experience in developing this project and I personally feel that University’s decision over this project is a good step in making students aware of the environment they are living and also the responsibilities they will be taking in the near future.

As matter of fact I learnt many new things that are not included in the syllabus like the way to do customer requirements analysis, designing an information system, steps that should be followed in software development etc. After doing this project I realize that before designing any information system, it is essential to understand user’s requirements. One of the most crucial parts of doing any project is requirement analysis and system designing. It is because wrong analysis leads to wrong designing which ultimately results in wrong product. Testing is another step in software development that we cannot escape and it is very important to choose a good testing methods and procedures in order to develop a quality software package.

I also wanted to add more features but because of less time span I could not do so I would like to have a plan in doing in future. Last I would like to once again thank all my College, Teachers, Family and Friends for supporting and contributing in completing my project.

**Tilrupa Timsina**

# Bibliography

## Websites

* <http://www.google.com>
* <http://www.youtube.com>
* <http://www.microsoft.com>
* <http://www.programmer2programmer.net>
* <http://www.codeproject.com>
* http://www.msdn.com
* <http://www.tutorialspoint.com/csharp/csharp_overview.htm>
* <http://www.sqltuner.com>
* <http://www.stackoverflow.com>
* <http://thielj.github.io/MetroFramework>

## Books

* Microsoft Press - Microsoft Visual C# 2010 Step by Step
* Wrox - Beginning Visual C# 2010
* O'Reilly - SQL in a Nutshell, Second Edition
* Database Development in Visual C#
* Sams - Microsoft SQL Server High Availability

## Training Videos

* Lynda.com C# Essential Training
* Lynda.com - Visual Studio 2010 Essential Training