# Chapter-1: Introduction

# 1.1 Introduction of Application:

A Job management system is a software application that helps users to organize, track, and complete tasks or projects. It can also provide features such as scheduling, prioritizing, assigning, reporting, and communicating with other users. Job management systems are useful for various domains, such as business, education, health care, and engineering.

The aim of this project is to develop a job management system using Visual Basic.net as the programming language and Microsoft Access as the database. The system will have the following functionalities:

* Create, edit, and delete tasks
* Assign tasks to other users
* View the status, progress, and deadline of each task
* Generate reports on the performance and efficiency of the system
* Send and receive messages with other users

The system will use a Windows Forms application to display the user interface and a data access layer to interact with the database. The system will follow the object-oriented design principles and the Model-View-Controller (MVC) pattern. The system will also implement security features, such as encryption, authentication, and authorization.

# 1.2 Objective of Application:

The objective of a Job Management System in Visual Basic.NET is to streamline and automate the management of tasks within an organization. Here are some key objectives:

* Resource Allocation and Management: The system is developed to assist companies in resource allocation and management1. It helps in efficiently distributing tasks among employees based on their skills and availability.
* Manage and Update Employee Details: The system aims to manage and update the employee details in the company1. This includes tracking employee performance, attendance, and other relevant information.
* Computerization of Recruitment Process: The system can also aid in the computerization of the recruitment process1. This involves automating tasks such as posting job vacancies, screening resumes, and scheduling interviews.
* Employee Record Tracking: The system allows the admin to track their employee’s record as well as see how often an employee takes leave.
* Report Generation: The system can generate reports where the admin can view or monitor the employee’s daily, weekly, or monthly activities.

Remember, the specific objectives can vary based on the exact requirements of the organization and the specific features implemented in the system.

# 1.3 Use of Application:

In order for the user to get an access into the Job Management System, he must go through the log in form first by entering the username and password correctly. Then, after logging in into the system the main menu is displayed. In here the user has the ability to search for any customer by clicking the search box and typing the exact name of the customer. He can even view all the customers that are registered in the system and he can also create, update the user information. While, the Job Ordering process begins once an item is registered for Repair. To do that, the customer has to fill up the necessary customer information. The user can put a Job Estimation such as Labor, Pick-Up and Delivery then, the Total amount will auto generate and the Job order registering of item/s ends when the user click the Save Button. After settling the payment in the cashier, a receipt is given to the customer. In this Job Ordering Management System, the admin has the ability to create, delete or update user accounts.

# Chapter-2: Introduction of Programming

# 2.1 Introduction of Visual Programming:

Any language that uses the graphics or blocks that are already defined with the code and you just need to use those blocks without worrying about the lines of code is known as a visual programming language. In today’s era majority of the programming languages are text-based i.e. we have to write the lines of code to perform the specific task like in C or C++. Programming if you want to print a table of 2 then you have to write the complete text using syntax and functions of that language but in visual programming language this task is replaced by graphics or blocks like components then can be joined logically to perform the task.

Visual Programming language lets the user think in a logical manner unlike in regular programming language the user has to think about that how he/she can explain the program to the computer, to do this let’s take one small analogy like if you have to code multiplication table of 2 than in regular programming language what you will do is you will take the loop and with the help of it you can print the multiplication table but in the visual basic language you just have to add the block which has the inbuilt code in it of loop and you just specify the value and you just have to think logically and your work is done without worrying about the semicolon, syntax, functions, etc.

# 2.2 Introduction of Visual Basic.NET:

Visual Basic.NET is a programming language that is designed for creating applications that run on the .NET Framework. It is an evolution of the classic Visual Basic language, with new features such as object-oriented programming, generics, delegates, and lambda expressions. Visual Basic.NET is also compatible with other .NET languages, such as C# and F#, and can use the same libraries and tools12.

If you want to learn Visual Basic.NET, you can start with the following resources:

Visual Basic docs: This is the official documentation for Visual Basic.NET, where you can find tutorials, reference, and language features.

Visual Studio IDE: This is the integrated development environment (IDE) for Visual Basic.NET, where you can write, debug, and deploy your code. You can download the free Community edition, or the Professional or Enterprise editions for more features.

Get started: This is a section of the documentation that helps you get started with Visual Basic.NET application development. You can learn how to create a console application, a library, and a web app with Visual Basic.NET.

**2.3 Applications of Visual Programming language:**

# VPL can be used in multiple domains like multimedia, educational purpose, video games, automation. Let’s see them in brief:

# Multimedia:- VPL helps users create multimedia without worrying about the real code or other complex features. It narrows down to specific functions and with the help of those functions, multimedia is created.

# Educational Purpose:-Scratch VPL, etc are used to help students in their projects and make them familiar with the coding.

* Videogames:-VPL helps to create the videogames without writing lines of codes Ex- Scratch VPL is used to make videogames.

# 2.4 Advantages and Disadvantages of Visual Programming language:

2.4.1 Advantages of visual programming language:

* Easy to convert ideas into reality for example you don’t know how to code so you can start with VPL(Visual Programming Language). and then switch to actual coding.
* Visuals are easy to grasp i.e. to develop something in visual programming language requires less efforts.
* It includes a variety of built in objects that are required while creating something using VPL.
* It is a beginner-friendly also anyone will be able to derive the logic without worrying about writing lines of code.
* Adding a user-specific code is also available and simple as it allows to create of blocks as per the convenience of the user.

2.4.2 Disadvantages of visual programming language:

* These languages require more memory as they use graphics, as a result, their execution is also slow and a large amount of memory is occupied by them.
* They can only work in an operating system like windows, mac, or any other operating system which supports graphics.
* As the inbuilt functions are not sufficient so you have to add your custom code as a result it is cumbersome.
* Only limited functions are present in these languages.
* Adding our custom code as a block requires coding knowledge or else you have to work with limited functions which are provided with the language.
* As a computer engineer, it is not a good idea to use VPL as most of the tech giants like FAANG or other tech companies work on textual languages like JAVA, HTML, etc, rather than VPL.
* For the long run VPL might not be that much useful as in a regular language you can explore more in it but in VPL at one point you will get bored by using the same it.

# Chapter-3: Analysis

**3.1 Basics Analysis:**

# This system is designed to manage employees within a company. Here are the key features of this system:

# User-Friendly Interface:

# The Employee’s Management System is user-friendly, making it accessible even for new users.

# It contains an admin section that plays a crucial role in running and operating the system.

# Employee Tracking and Record Management:

# The system allows the admin to track employee records.

# Admins can monitor how often an employee takes leave in a month or throughout their tenure in the company.

# 2. Reporting Capabilities:

# The system generates reports for the admin to monitor daily, weekly, or monthly activities of employees.

# Reports include details such as leave of absence, employee lists, and remaining leave.

# 3. Specific Features:

# Manage Employee: Admins can handle employee information.

# Manage Leave of Absence: Track and manage employee leaves.

# Manage Position of Employee: Handle employee positions.

# Manage Department of Employee: Organize employees by department.

# Manage User: User management within the system.

# Search by Department or Position: Efficiently search for employees.

# Generate Monitoring Leave of Absence Report: Detailed leave reports.

# Generate List of Employees: Obtain employee lists.

# Generate Pie Chart for Remaining Leave: Visual representation of remaining leave.

**3.2 Crises Analysis:**

Crisis analysis on a Job Management System in Visual Basic.NET would involve identifying potential problems or crises that could occur with the system and developing strategies to manage these crises. Here are some potential crises and their solutions:

Data Loss or Corruption: This could occur due to hardware failure, software bugs, or cyber-attacks. Regular backups and use of reliable hardware can mitigate this risk.

System Downtime: This could be due to server issues, power outages, or maintenance activities. Having a robust disaster recovery plan and redundant systems can help minimize downtime.

Security Breaches: Unauthorized access to the system could lead to data theft or manipulation. Implementing strong security measures like encryption, two-factor authentication, and regular security audits can help prevent such incidents.

Software Bugs: Errors in the code could lead to system malfunctions. Regular testing and debugging can help identify and fix these issues.

User Errors: Users might make mistakes while using the system, leading to incorrect data entry or operation. Providing comprehensive user training and implementing user-friendly interfaces can help reduce user errors.

Remember, the specific crises can vary based on the exact features implemented in the system and the environment in which it is used. It’s always a good idea to conduct a thorough risk assessment and develop a crisis management plan

# Chapter-4: System Design

**4.1 Data Flow Diagram for Application:**

A data flow diagram (DFD) is a graphical representation of how data flows through a system or process. It shows the sources and destinations of data, the processes that transform or manipulate data, and the data stores that hold data. A DFD can help you understand, analyze, and improve a system or process by identifying its components and interactions.

To draw a data flow diagram for a job management system, you need to follow some steps:

Identify the entities involved in the system, such as job seekers, employers, recruiters, etc. These are the external agents that interact with the system and provide or receive data. You can represent them as rectangles with rounded corners in your DFD.

Identify the main processes that occur in the system, such as posting a job, applying for a job, screening candidates, etc. These are the activities that transform or manipulate data in the system. You can represent them as circles or ovals in your DFD.

Identify the data flows that connect the entities and processes. These are the paths that data takes from one component to another in the system. You can represent them as arrows with labels in your DFD. The labels should describe the content and format of the data being transferred.

Identify the data stores that store data in the system, such as job database, resume database, etc. These are the places where data is kept for later use or reference. You can represent them as open-ended rectangles in your DFD.

You can start with a high-level overview of the system, also known as a context diagram or level 0 DFD. This shows the system as a single process with inputs and outputs from the external entities. Then, you can decompose the system into more detailed levels, such as level 1 DFD, level 2 DFD, etc. Each level shows more sub processes and data flows within the system.

Here is an example of a level 0 DFD for a job management system:

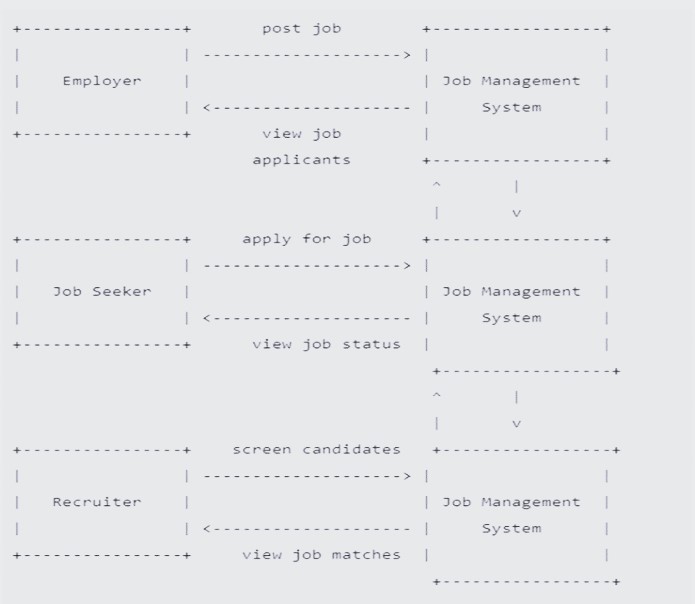


Fig 1 DFD of Job Management System

This diagram represents a job management system involving three key roles: Employer, Job Seeker, and Recruiter. Let’s break down the interactions:

1. Employer:
   * Posts a job on the system.
   * Views applicants who have applied for the job.
2. Job Seeker:
   * Applies for a job listed in the system.
   * Views the status of their job application.
3. Recruiter:
   * Screens candidates based on job requirements.
   * Views job matches for potential candidates.

The system facilitates efficient communication and coordination between these stakeholders, streamlining the job search and recruitment process.

**4.2 E-R Diagram for Application:**

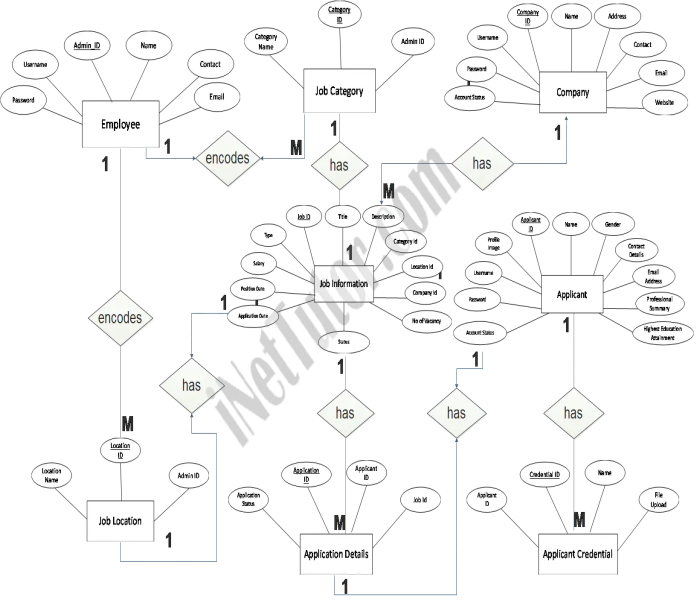
****

Fig 2 E-R Diagram of Job Management System

**4.3 System Design:**

Designing a Job Management System in Visual Basic.NET involves several components. Here’s a high-level overview:

* Database (MySQL or MS Access): This is where all the data related to jobs, employees, departments, and other relevant information is stored.
* User Interface: This is developed using Visual Basic.NET. It should be user-friendly so that even new users can easily operate it1. It includes forms for managing employees, jobs, departments, and generating reports.
* Admin Section: This section is responsible for managing the system. It includes features like managing employees, managing leave of absence, managing positions and departments of employees, and generating various reports.
* Employee Section: This section allows employees to view and update their details, apply for leaves, and view their tasks.
* Authentication System: This system ensures that only authorized users can access the system. It could be as simple as a username and password system.
* Reporting System: This system generates various reports like monitoring leave of absence, list of employees, and others.

Remember, this is a high-level design and the actual implementation can vary based on the specific requirements of the organization. It’s always a good idea to start with a detailed requirements analysis and then proceed with the design and implementation phases. Also, remember to follow best practices for software development, including writing clean and maintainable code, performing regular testing, and implementing robust security measures.

Here’s a simple code snippet in VB.NET for a login form:

Public Class LoginForm

Private Sub btnLogin\_Click(sender As Object, e As EventArgs) Handles btnLogin.Click

' Validate the user credentials

If txtUsername.Text = "admin" And txtPassword.Text = "admin" Then

' If valid, show the main form

MainForm.Show()

Me.Hide()

Else

' If invalid, show an error message

MessageBox.Show("Invalid username or password.")

End If

End Sub

End Class

This is a very basic example and a real-world application would need a more secure way of handling user credentials, such as hashing passwords and storing them securely in a database. It’s also important to handle potential errors and edge cases to ensure the application is robust and reliable.

# Chapter-5: Background Study

A Job Management System in Visual Basic.NET is a software application designed to streamline and automate the management of tasks within an organization. The background study for such a system would involve understanding the needs of the organization, the tasks to be managed, and the users who will be interacting with the system123.

Here are some key points that might be included in a background study:

* Understanding the Organization: This involves understanding the structure of the organization, the roles of different employees, and the tasks they perform. This information can help in designing a system that fits the organization’s needs.
* Understanding the Tasks: This involves understanding the tasks that need to be managed, their complexity, and their interdependencies. This information can help in designing a system that can effectively manage these tasks.
* Understanding the Users: This involves understanding the needs and capabilities of the users who will be interacting with the system. This information can help in designing a system that is user-friendly and meets the needs of its users.
* Technical Considerations: This involves understanding the technical requirements of the system, such as the hardware and software resources available, the programming languages and frameworks to be used (in this case, Visual Basic.NET), and the database system to be used (such as MySQL or MS Access).
* Legal and Ethical Considerations: This involves understanding the legal and ethical requirements related to the system, such as data privacy laws and ethical guidelines for software development.

The background study would also involve a review of existing literature and systems to understand the current state of the art and identify gaps that the new system could fill123. This could include academic papers, technical reports, and existing software applications.

# Chapter-6: Requirement

**6.1 Hardware Requirements**

Computer based application needs hardware as well as software to develop and run project successfully. A good performance of hardware and software makes the project successful.

Minimum hardware requirements for efficient operation of this project are as follows:

|  |  |
| --- | --- |
| Processor | 2.6 GHz or faster processor |
| Memory Module | 1GB or More |
| Hard disk | 80 GB or More |
| Printer | Any |
| Keyboard | Any |
| Mouse | Any |

**6.2 Software Requirements**

Minimum Software requirements for efficient operation of this project are as follows:

|  |  |
| --- | --- |
| Operating System | Windows 8, Windows 10 , Windows 11 + |
| Extra | MS-Word , MS Access , Visual Studio |

The following environment were used for the development of this Project :

OS: Windows 11 x 64

IDE: Visual Studio 2022 & MS Access

# **Setting Up the System**:

Accessing admin accounts

Username: admin  
Password: admin

# Chapter-7: Development

Based on the requirements and system design, the development process for the application was carried out using the VB.Net programming language. The application was developed using an object-oriented approach, which helped in achieving modularity, flexibility and maintainability of the code.

The graphical user interface was designed using Windows Forms, which allowed for a user-friendly and intuitive layout. The buttons and labels were placed in a logical manner, and the color scheme was chosen to be visually appealing and easy on the eyes.

Throughout the development process, a series of tests were performed to ensure the correctness and accuracy of the arithmetic operations. The testing process involved creating test cases for various input scenarios, and verifying that the output matched the expected results.

To ensure that the application was error-free and met the requirements, the development process involved multiple iterations of coding, testing and debugging. This iterative process helped in improving the quality of the code and eliminating any errors or bugs that were identified during the testing process.

Overall, the development process for the application was carried out in a structured and organized manner, following the principles of software development life cycle (SDLC) methodology. The final application was tested thoroughly and met the requirements outlined in the initial system design.

**7.1. Source code:**

1. Main.vb code

Public Class Main

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

txttime.Text = My.Computer.Clock.LocalTime.Date.ToLongDateString

End Sub

Private Sub Main\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Timer1.Start()

JobOrdertool.Visible = False

paymentstool.Visible = False

customertool.Visible = False

ReportsToolStripMenuItem.Visible = False

ManageUserToolStripMenuItem.Visible = False

techToolStrip.Visible = False

End Sub

Private Sub LogInToolStripMenuItem\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles customertool.Click

Customer.Show()

Payment.Close()

order.Close()

End Sub

Private Sub ExitToolStripMenuItem\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ExitToolStripMenuItem.Click

Me.Close()

End Sub

Private Sub GuestToolStripMenuItem\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles logintool.Click

If logintool.Text = "Log in" Then

login.Show()

login.BringToFront()

Else

jokenupdate("UPDATE logs SET lastlogout = #" & DateValue(DateTimePicker1.Value) & "#, userid = '" & userid.Text & "' where ID = (SELECT last(ID) FROM logs)")

Main\_Load(sender, e)

logintool.Text = "Log in"

Guest.Text = "Guest"

End If

End Sub

Private Sub ManageUserToolStripMenuItem\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ManageUserToolStripMenuItem.Click

useraccounts.Show()

useraccounts.BringToFront()

End Sub

Private Sub ReportsToolStripMenuItem\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ReportsToolStripMenuItem.Click

Reports.Show()

End Sub

Private Sub ToolStripMenuItem2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles techToolStrip.Click

technician.Show()

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

If logintool.Text = "Log in" Then

Me.Close()

Else

MsgBox("Please Log out first before Exit program!")

End If

End Sub

Private Sub Panel1\_Paint(ByVal sender As System.Object, ByVal e As System.Windows.Forms.PaintEventArgs)

End Sub

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

Dim f As Customer = New Customer

'f.ShowDialog()

showform(f)

End Sub

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

Dim f As technician = New technician

'f.ShowDialog()

showform(f)

End Sub

Private Sub Button7\_Click(sender As Object, e As EventArgs) Handles Button7.Click

Dim f As Reports = New Reports

'f.ShowDialog()

showform(f)

End Sub

Private Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

Dim f As order = New order

showform(f)

'f.ShowDialog()

End Sub

Private Sub Button5\_Click\_1(sender As Object, e As EventArgs) Handles Button5.Click

Dim f As Payment = New Payment

showform(f)

'f.ShowDialog()

End Sub

Private Sub Button6\_Click(sender As Object, e As EventArgs) Handles Button6.Click

Dim f As useraccounts = New useraccounts

showform(f)

'f.ShowDialog()

End Sub

Private Sub Button8\_Click(sender As Object, e As EventArgs) Handles Button8.Click

Me.Close()

login.Show()

End Sub

Private Sub Panel1\_Paint\_1(sender As Object, e As PaintEventArgs) Handles Panel1.Paint

End Sub

End Class

1. Login page

Public Class login

Private Sub btnok\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnok.Click

Dim sql As String

If txtuser.Text = "" And txtpass.Text = "" Then

MsgBox("Password or Username Incorrect!")

Else

sql = "select \* from tbluseraccounts where userusername ='" & txtuser.Text & "' and userpassword = '" & txtpass.Text & "'"

jokenfindthis(sql)

checkresult()

End If

End Sub

Private Sub btncancel\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btncancel.Click

Me.Close()

End Sub

Private Sub login\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

End Sub

End Class

1. Customer.vb

Public Class Customer

Private Sub Customer\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Timer1.Start()

GroupBox1.Enabled = False

Dim custcode As String

Dim da As New OleDb.OleDbDataAdapter

Dim con As OleDb.OleDbConnection = jokenconn()

custcode = "Select appendchar & '-' & autoend + incrementvalue from tblauto where ID = 2"

Try

con.Open()

da = New OleDb.OleDbDataAdapter(custcode, con)

Dim ds As New DataSet

da.Fill(ds, "jobdb")

txtcustcode.Text = ds.Tables("jobdb").Rows(0).Item(0)

Catch ex As Exception

MsgBox(ex.Message, MsgBoxStyle.Information)

End Try

con.Close()

End Sub

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

lbldatetime.Text = Date.Today & " " & TimeOfDay

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btncreate.Click

jokeninsert("Insert Into tblcustomer(custcode,custname,custcontact,Address,custtel) " & \_

" Values('" & txtcustcode.Text & "','" & txtcustname.Text & "', " & \_

" '" & txtcontact.Text & "','" & txtaddress.Text & "','" & txttell.Text & "')")

jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 2")

Customer\_Load(sender, e)

jokenselect("Select ID, custcode as [Customer Code],custname As [Customer],custcontact as [Contact Person],Address,custtel as [Telephone No] from tblcustomer")

fillitemtable(DataGridView1)

End Sub

Private Sub btnloadall\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnloadall.Click

jokenselect("Select ID, custcode as [Customer Code],custname As [Customer],custcontact as [Contact Person],Address,custtel as [Telephone No] from tblcustomer")

fillitemtable(DataGridView1)

End Sub

Private Sub TextBox3\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtcustsearch.TextChanged

jokenselect("Select ID, custcode as [Customer Code],custname As [Customer], " & \_

" custcontact as [Contact Person],Address,custtel as [Telephone No] " & \_

" from tblcustomer where custcode like'%" & txtcustsearch.Text & "%' or custname like '%" & txtcustsearch.Text & "%'")

fillitemtable(DataGridView1)

End Sub

Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click

Me.Close()

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

GroupBox1.Enabled = True

End Sub

Private Sub Button4\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnuse.Click

Try

With order

.txtcustcode.Text = DataGridView1.CurrentRow.Cells(1).Value.ToString

.txtcustname.Text = DataGridView1.CurrentRow.Cells(2).Value.ToString

.txtcontact.Text = DataGridView1.CurrentRow.Cells(3).Value.ToString

.txtaddress.Text = DataGridView1.CurrentRow.Cells(4).Value.ToString

.txtcontact.Text = DataGridView1.CurrentRow.Cells(5).Value.ToString

End With

showform(order)

Me.Close()

Catch ex As Exception

MsgBox("Please select a customer!", MsgBoxStyle.Exclamation)

End Try

End Sub

Private Sub DataGridView1\_CellContentClick(sender As Object, e As DataGridViewCellEventArgs) Handles DataGridView1.CellContentClick

End Sub

End Class

1. Order.vb

Public Class order

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

lbldate.Text = Date.Today

lbltime.Text = TimeOfDay

End Sub

Private Sub order\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

GroupBox1.Enabled = False

txtremarks.Enabled = False

Timer1.Start()

Dim jborder, itemcode As String

Dim da, da1 As New OleDb.OleDbDataAdapter

Dim con As OleDb.OleDbConnection = jokenconn()

jborder = "Select appendchar & '-' & autoend + incrementvalue from tblauto where ID = 1"

itemcode = "Select appendchar & '-' & autoend + incrementvalue from tblauto where ID = 3"

Try

con.Open()

da = New OleDb.OleDbDataAdapter(jborder, con)

Dim ds As New DataSet

da.Fill(ds, "jobdb")

lblorderno.Text = ds.Tables("jobdb").Rows(0).Item(0)

da1 = New OleDb.OleDbDataAdapter(itemcode, con)

Dim ds1 As New DataSet

da1.Fill(ds1, "jobdb")

lblitemcode.Text = ds1.Tables("jobdb").Rows(0).Item(0)

Catch ex As Exception

MsgBox(ex.Message, MsgBoxStyle.Information)

End Try

con.Close()

jokenselect("SELECT i.ID,qty as [QTY],description as [Description],serialno as [Serial No],problem as [Problem],actiontaken as [Action Taken],totalamount as [Total Amount] " & \_

" FROM tblcustomer c, tbliteminfo i, tbljobestimate j where c.custcode =i.custcode " & \_

" and j.itemcode = i.itemcode and i.custcode ='" & txtcustcode.Text & "' ")

fillitemtable(DataGridView1)

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

'If GroupBox1.Enabled = False Then

'MsgBox("Please Provide Job Estimate")

'Else

If txtcustcode.Text = "" Then

MsgBox("Please provide Customer Information!", MsgBoxStyle.Exclamation)

Else

jokeninsert("INSERT INTO tbliteminfo (joborderno,itemcode, custcode, qty, description, serialno, " & \_

" problem, actiontaken,dateIn, recievedby, tech\_incharge,transactiontype) " & \_

" VALUES('" & lblorderno.Text & "','" & lblitemcode.Text & "','" & txtcustcode.Text & "', " & \_

" " & Val(txtqty.Text) & ",'" & txtdesc.Text & "','" & txtserial.Text & "', " & \_

" '" & txtprob.Text & "','" & txtaction.Text & "',#" & lbldate.Text & "#,'" & Main.Guest.Text & "', " & \_

" '" & txtincharge.Text & "','" & lblchckresult.Text & "')")

jokeninsert("INSERT INTO tbljobestimate(jobordercode,itemcode,parts,labor,pickup,delivery,downpayment,balanceamount,totalamount,Remarks) " & \_

" VALUES('" & lblorderno.Text & "','" & lblitemcode.Text & "'," & Val(txtparts.Text) & "," & Val(txtlabor.Text) & ", " & \_

" " & Val(txtpickup.Text) & "," & Val(txtdeliver.Text) & "," & 0 & "," & Val(txttotalamnt.Text) & "," & Val(txttotalamnt.Text) & ",'" & txtremarks.Text & "')")

jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 3")

jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 1")

order\_Load(sender, e)

jokenselect("SELECT i.ID,qty as [QTY],description as [Description],serialno as [Serial No],problem as [Problem],actiontaken as [Action Taken] " & \_

" FROM tblcustomer c, tbliteminfo i where c.custcode =i.custcode and i.custcode ='" & txtcustcode.Text & "' and i.joborderno='" & lblorderno.Text & "'")

fillitemtable(DataGridView1)

End If

'End If

PrintJB.Show()

End Sub

Private Sub btnsavetrans\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnsavetrans.Click

If chckUW.Checked = True Then

lblchckresult.Text = chckUW.Text

ElseIf chckOUW.Checked = True Then

lblchckresult.Text = chckOUW.Text

ElseIf chckOS.Checked = True Then

lblchckresult.Text = chckOS.Text

ElseIf chckR.Checked = True Then

lblchckresult.Text = chckR.Text

ElseIf chckO.Checked = True Then

lblchckresult.Text = chckO.Text

Else

lblchckresult.Text = chckR.Text

End If

If txtcustcode.Text = "" Then

MsgBox("Please provide Customer Information!", MsgBoxStyle.Exclamation)

Else

jokeninsert("INSERT INTO tbliteminfo (joborderno,itemcode, custcode, qty, description, serialno, " & \_

" problem, actiontaken,dateIn, recievedby, tech\_incharge,transactiontype) " & \_

" VALUES('" & lblorderno.Text & "','" & lblitemcode.Text & "','" & txtcustcode.Text & "', " & \_

" " & Val(txtqty.Text) & ",'" & txtdesc.Text & "','" & txtserial.Text & "', " & \_

" '" & txtprob.Text & "','" & txtaction.Text & "',#" & lbldate.Text & "#,'" & Main.Guest.Text & "', " & \_

" '" & txtincharge.Text & "','" & lblchckresult.Text & "')")

'jokeninsert("INSERT INTO tbljobestimate(itemcode,parts,labor,pickup,delivery,totalamount) " & \_

' " VALUES('" & lblitemcode.Text & "'," & Val(txtparts.Text) & "," & Val(txtlabor.Text) & ", " & \_

' " " & Val(txtpickup.Text) & "," & Val(txtdeliver.Text) & "," & Val(txttotalamnt.Text) & ")")

jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 3")

'jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 1")

'Form1\_Load(sender, e)

jokenselect("SELECT i.ID,qty as [QTY],description as [Description],serialno as [Serial No],problem as [Problem],actiontaken as [Action Taken] " & \_

" FROM tblcustomer c, tbliteminfo i where c.custcode =i.custcode and i.custcode ='" & txtcustcode.Text & "' and i.joborderno='" & lblorderno.Text & "'")

fillitemtable(DataGridView1)

End If

End Sub

Private Sub btnnew\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnnew.Click

cleartext(GroupBox2)

cleartext(GroupBox1)

End Sub

Private Sub jobest\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles jobest.Click

GroupBox1.Enabled = True

txtremarks.Enabled = True

End Sub

Private Sub txtcustcode\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtcustcode.TextChanged

'jokenselect("SELECT i.ID,qty as [QTY],description as [Description],serialno as [Serial No],problem as [Problem],actiontaken as [Action Taken] " & \_

' " FROM tblcustomer c, tbliteminfo i where c.custcode =i.custcode and i.custcode ='" & txtcustcode.Text & "' and i.joborderno='" & lblorderno.Text & "'")

'fillitemtable(DataGridView1)

End Sub

Private Sub btnopenlist\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnopenlist.Click

'Dim f As Customer = New Customer

With Customer

.Show()

.Focus()

.btnuse.Visible = True

End With

End Sub

Private Sub txtparts\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtparts.TextChanged, txtlabor.TextChanged, txtpickup.TextChanged, txtdeliver.TextChanged

If Not IsNumeric(txtparts.Text) Then

txtparts.Text = Nothing

End If

If Not IsNumeric(txtlabor.Text) Then

txtlabor.Text = Nothing

End If

If Not IsNumeric(txtpickup.Text) Then

txtpickup.Text = Nothing

End If

If Not IsNumeric(txtdeliver.Text) Then

txtdeliver .Text = Nothing

End If

compute()

End Sub

Public Sub compute()

txttotalamnt.Text = Val(txtdeliver.Text) + Val(txtparts.Text) + Val(txtlabor.Text) + Val(txtpickup.Text)

txttotalamnt.Text = FormatNumber(txttotalamnt.Text, 2)

End Sub

Private Sub txtqty\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtqty.TextChanged

If Not IsNumeric(txtqty.Text) Then

txtqty.Text = ""

End If

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

technician.Show()

End Sub

Private Sub btncancel\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btncancel.Click

Me.Close()

End Sub

Private Sub btnsavejoborder\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnsavejoborder.Click

If GroupBox1.Enabled = False Then

MsgBox("Please Provide Job Estimate")

Else

If txtcustcode.Text = "" Then

MsgBox("Please provide Customer Information!", MsgBoxStyle.Exclamation)

Else

jokeninsert("INSERT INTO tbliteminfo (joborderno,itemcode, custcode, qty, description, serialno, " & \_

" problem, actiontaken,dateIn, recievedby, tech\_incharge,transactiontype) " & \_

" VALUES('" & lblorderno.Text & "','" & lblitemcode.Text & "','" & txtcustcode.Text & "', " & \_

" " & Val(txtqty.Text) & ",'" & txtdesc.Text & "','" & txtserial.Text & "', " & \_

" '" & txtprob.Text & "','" & txtaction.Text & "',#" & lbldate.Text & "#,'" & Main.Guest.Text & "', " & \_

" '" & txtincharge.Text & "','" & lblchckresult.Text & "')")

jokeninsert("INSERT INTO tbljobestimate(jobordercode,itemcode,parts,labor,pickup,delivery,downpayment,balanceamount,totalamount,Remarks) " & \_

" VALUES('" & lblorderno.Text & "','" & lblitemcode.Text & "'," & Val(txtparts.Text) & "," & Val(txtlabor.Text) & ", " & \_

" " & Val(txtpickup.Text) & "," & Val(txtdeliver.Text) & "," & 0 & "," & Val(txttotalamnt.Text) & "," & Val(txttotalamnt.Text) & ",'" & txtremarks.Text & "')")

jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 3")

jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 1")

order\_Load(sender, e)

jokenselect("SELECT i.ID,qty as [QTY],description as [Description],serialno as [Serial No],problem as [Problem],actiontaken as [Action Taken] " & \_

" FROM tblcustomer c, tbliteminfo i where c.custcode =i.custcode and i.custcode ='" & txtcustcode.Text & "' and i.joborderno='" & lblorderno.Text & "'")

fillitemtable(DataGridView1)

End If

End If

End Sub

Private Sub GroupBox2\_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles GroupBox2.Enter

End Sub

End Class

1. Payment.vb

Public Class Payment

Dim payments As Decimal

Private Sub Timer1\_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick

txtdate.Text = Date.Today

End Sub

Private Sub Payment\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Timer1.Start()

Dim dt, dt1 As New DataTable

Dim da, da1 As New OleDb.OleDbDataAdapter

Dim con As OleDb.OleDbConnection = jokenconn()

Dim orno As String = "Select autoend + incrementvalue from tblauto where ID = 4"

Try

da1 = New OleDb.OleDbDataAdapter(orno, con)

da1.Fill(dt1)

lblor.Text = dt1.Rows(0).Item(0)

Catch ex As Exception

MsgBox(ex.Message, MsgBoxStyle.Information)

End Try

con.Close()

jokenselect("Select ID,jobordercode as [Job Order],Orno as [Or No],curdate as [Date],Cashier,Amount from tblpayments")

filltotaltable (DataGridView2)

Try

Me.DataGridView2.Columns(5).DefaultCellStyle.Alignment = DataGridViewContentAlignment.BottomRight

DataGridView2.Rows(Me.DataGridView2.Rows.Count - 1).DefaultCellStyle.Font = New Font(DataGridView2.Font, FontStyle.Bold)

DataGridView2.Columns(5).DefaultCellStyle.Format = "n"

Catch ex As Exception

MsgBox(ex.Message, MsgBoxStyle.Information)

End Try

btnsaveonly.Enabled = False

btnprint.Enabled = False

End Sub

Private Sub btnopenlist\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Customer.Show()

Customer.btnuse.Visible = False

End Sub

Private Sub DataGridView1\_CellClick(ByVal sender As Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles DataGridView1.CellClick

Try

lblid.Text = DataGridView1.CurrentRow.Cells(0).Value.ToString

txttotal.Text = FormatNumber(DataGridView1.CurrentRow.Cells(6).Value.ToString, 2)

txtbal.Text = DataGridView1.CurrentRow.Cells(8).Value.ToString

Txtdownpayment.Text = DataGridView1.CurrentRow.Cells(7).Value.ToString

' txtremainingbal.Text = FormatNumber(DataGridView1.CurrentRow.Cells(8).Value.ToString, 2)

txtcustcode.Text = DataGridView1.CurrentRow.Cells(11).Value.ToString

txtaddress.Text = DataGridView1.CurrentRow.Cells(13).Value.ToString

txttell.Text = DataGridView1.CurrentRow.Cells(14).Value.ToString

txtjborder.Text = DataGridView1.CurrentRow.Cells(1).Value.ToString

btnsaveonly.Enabled = True

btnprint.Enabled = True

txtchange.Text = "0.00"

txtrecieve.Text = Nothing

'txttotal.Text = FormatNumber(txttotal.Text, 2)

txtnetbal.Text = txtbal.Text

txtnetbal.Text = FormatNumber(txtnetbal.Text, 2)

Catch ex As Exception

MsgBox("Column Header is no allowed", MsgBoxStyle.Information)

End Try

End Sub

Private Sub txtrecieve\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtrecieve.TextChanged

Dim str, str1 As String

txtremainingbal.Text = Val(txtbal.Text) - Val(txtrecieve.Text)

txtrpayments.Text = Val(txtbal.Text) - Val(txtremainingbal.Text)

txtgrandtotal.Text = Val(Txtdownpayment.Text) + Val(txtrpayments.Text)

txtchange.Text = Val(txtrecieve.Text) - Val(txtbal.Text)

txtchange.Text = FormatNumber(txtchange.Text, 2)

txtremainingbal.Text = FormatNumber(txtremainingbal.Text, 2)

txtgrandtotal.Text = FormatNumber(txtgrandtotal.Text, 2)

txtrpayments.Text = FormatNumber(txtrpayments.Text, 2)

If Val(txtrecieve.Text) > Val(txtbal.Text) Then

txtgrandtotal.Text = txttotal.Text

txtrpayments.Text = Val(txtbal.Text)

txtgrandtotal.Text = FormatNumber(txtgrandtotal.Text, 2)

txtrpayments.Text = FormatNumber(txtrpayments.Text, 2)

End If

If txtrecieve.Text = "" Then

txtrecieve.Text = Nothing

txtchange.Text = Nothing

txtchange.Text = "0.00"

txtgrandtotal.Text = Txtdownpayment.Text

txtremainingbal.Text = txtbal.Text

txtrpayments.Text = txtrecieve.Text

End If

If Not IsNumeric(txtrecieve.Text) Then

txtrecieve.Text = Nothing

txtchange.Text = Nothing

txtchange.Text = "0.00"

txtgrandtotal.Text = Txtdownpayment.Text

txtremainingbal.Text = txtbal.Text

txtrpayments.Text = "0.00"

End If

str = Val(txtchange.Text)

If str.Contains("-") = True Then

txtchange.Text = "0.00"

End If

str1 = Val(txtremainingbal.Text)

If str1.Contains("-") = True Then

txtremainingbal.Text = "0.00"

End If

End Sub

Private Sub btnprint\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnprint.Click

savereciept()

cleartext(GroupBox1)

Payment\_Load(sender, e)

receipt.Show()

End Sub

Private Sub btnsaveonly\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnsaveonly.Click

savereciept()

cleartext(GroupBox1)

Payment\_Load(sender, e)

End Sub

Private Sub txtcustname\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtcustname.TextChanged

Dim sql As String = "SELECT j.ID,j.jobordercode as [JobOrder],Parts, Labor,pickup as [Pick up],Delivery,Totalamount as [Total],downpayment as [Payment], " & \_

" balanceamount as [Balance],Settled,c.custcode,custname,custcontact, Address,custtel " & \_

" FROM tbljobestimate AS j,tblcustomer AS c, tbliteminfo AS i " & \_

" WHERE j.itemcode=i.itemcode And c.custcode=i.custcode And c.custname like'%" & txtcustname.Text & "%'"

jokenselect(sql)

fillpaymentdata(DataGridView1)

With DataGridView1

.Columns(2).DefaultCellStyle.Format = "n"

.Columns(3).DefaultCellStyle.Format = "n"

.Columns(4).DefaultCellStyle.Format = "n"

.Columns(5).DefaultCellStyle.Format = "n"

.Columns(6).DefaultCellStyle.Format = "n"

.Columns(7).DefaultCellStyle.Format = "n"

.Columns(8).DefaultCellStyle.Format = "n"

End With

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click, txtrecieve.TextChanged

'If balance.Text = "0.00" Then

' txtamount.Text = txttotal.Text

' txtrpayments.Text = txtbal.Text

' txtrpayments.Text = FormatNumber(txtrpayments.Text, 2)

' 'ElseIf balance.Text = "0" Then

' ' balance.Text = "0.00"

'Else

' txtrpayments.Text = Val(txtbal.Text) - Val(balance.Text)

' ' txtrpayments.Text = FormatNumber(txtrpayments.Text, 2)

' txtamount.Text = Val(Txtpayments.Text) + Val(txtrpayments.Text)

' ' txtamount.Text = FormatNumber(txtamount.Text, 2)

'End If

End Sub

Public Sub savereciept()

If txtremainingbal.Text = "0.00" Then

jokenupdate("Update tbljobestimate set downpayment = '" & txttotal.Text & "',balanceamount = 0,settled = 1 where ID = " & Val(lblid.Text) & "")

jokeninsert("Insert INTO tblpayments (jobordercode,curdate,amount,cashier,orno,custcode)VALUES('" & txtjborder.Text & "',#" & txtdate.Text & "#,'" & txtrpayments.Text & "','" & Main.Guest.Text & "'," & lblor.Text & ",'" & txtcustcode.Text & "')")

jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 4")

jokenupdate("Update tbliteminfo set datereleased = #" & txtdate.Text & "# where joborderno = '" & txtjborder.Text & "'")

Else

'MsgBox(txttotal.Text)

'MsgBox(txtremainingbal.Text)

jokenupdate("Update tbljobestimate set downpayment = '" & txtgrandtotal.Text & "',balanceamount = '" & txtremainingbal.Text & "',settled = 0 where ID = " & Val(lblid.Text) & "")

jokeninsert("Insert INTO tblpayments (jobordercode,curdate,amount,cashier,orno,custcode)VALUES('" & txtjborder.Text & "',#" & txtdate.Text & "#,'" & txtrpayments.Text & "','" & Main.Guest.Text & "'," & lblor.Text & ",'" & txtcustcode.Text & "')")

jokenupdate("Update tblauto set autoend = autoend + incrementvalue where id = 4")

End If

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

MsgBox(txtgrandtotal.Text)

End Sub

Private Sub GroupBox1\_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles GroupBox1.Enter

End Sub

Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click

payables.Label2.Text = txtjborder.Text

payables.Show()

End Sub

Private Sub DataGridView1\_CellContentClick(ByVal sender As System.Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles DataGridView1.CellContentClick

End Sub

End Class

1. reports.vb

Public Class Reports

Private Sub btnshow\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnshow.Click

DataGridView2.DataSource = Nothing

If RadioButton1.Checked = True Then

jokenselect("SELECT ID,jobordercode as [Job Order],itemcode as [Item Code],Parts,Labor, " & \_

" pickup as [Pick up],Delivery,totalamount as [Total Amount],Downpayment, " & \_

" balanceamount as [Balance],Settled,Remarks FROM tbljobestimate")

fillitemtable(DataGridView1)

With DataGridView1

.Columns(3).DefaultCellStyle.Format = "n"

.Columns(4).DefaultCellStyle.Format = "n"

.Columns(5).DefaultCellStyle.Format = "n"

.Columns(6).DefaultCellStyle.Format = "n"

.Columns(7).DefaultCellStyle.Format = "n"

.Columns(8).DefaultCellStyle.Format = "n"

.Columns(9).DefaultCellStyle.Format = "n"

End With

ElseIf RadioButton2.Checked = True Then

jokenselect("SELECT ID,jobordercode as [Job Order],Orno as [Or No],curdate as [Date],Cashier,Amount FROM tblpayments")

'jokenselect("SELECT ID,jobordercode as [Job Order],curdate as [Date],Cashier,Orno as [Or No],Amount, " & \_

' " custcode as [Customercode] FROM tblpayments")

filltotaltable(DataGridView1)

With DataGridView1

.Columns(5).DefaultCellStyle.Format = "n"

End With

ElseIf RadioButton3.Checked = True Then

jokenselect("SELECT ID,jobordercode as [Job Order],itemcode as [Item Code],Parts,Labor,pickup as [Pick up], " & \_

" Delivery,totalamount as [Total Amount],Downpayment,balanceamount as [Balance],Settled, " & \_

" Remarks FROM tbljobestimate where settled = yes")

fillitemtable(DataGridView1)

With DataGridView1

.Columns(3).DefaultCellStyle.Format = "n"

.Columns(4).DefaultCellStyle.Format = "n"

.Columns(5).DefaultCellStyle.Format = "n"

.Columns(6).DefaultCellStyle.Format = "n"

.Columns(7).DefaultCellStyle.Format = "n"

.Columns(8).DefaultCellStyle.Format = "n"

.Columns(9).DefaultCellStyle.Format = "n"

End With

ElseIf RadioButton4.Checked = True Then

jokenselect("SELECT ID,jobordercode as [Job Order],itemcode as [Item Code],Parts,Labor,pickup as [Pick up], " & \_

" Delivery,totalamount as [Total Amount],Downpayment,balanceamount as [Balance],Settled, " & \_

" Remarks FROM tbljobestimate where settled = no")

fillitemtable(DataGridView1)

With DataGridView1

.Columns(3).DefaultCellStyle.Format = "n"

.Columns(4).DefaultCellStyle.Format = "n"

.Columns(5).DefaultCellStyle.Format = "n"

.Columns(6).DefaultCellStyle.Format = "n"

.Columns(7).DefaultCellStyle.Format = "n"

.Columns(8).DefaultCellStyle.Format = "n"

.Columns(9).DefaultCellStyle.Format = "n"

End With

ElseIf RadioButton5.Checked = True Then

jokenselect("SELECT ID, joborderno as [Job Order],itemcode as [Item Code],custcode as [Customer Code],Qty, " & \_

" Description,serialno as [Serial No],Problem,actiontaken as [Action Taken],dateIn as [Date In],datereleased as [Date Out], " & \_

" recievedby as [Recieved By],tech\_incharge as [Incharge],transactiontype as [Transaction Type]FROM tbliteminfo")

fillitemtable(DataGridView1)

End If

End Sub

Private Sub btnpreview\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnpreview.Click

AllReport.Show()

End Sub

Private Sub Reports\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

RadioButton1.Checked = True

End Sub

Private Sub btngo\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btngo.Click

'jokenselect("SELECT ID,jobordercode as [Job Order],curdate as [Date],Amount,Cashier,Orno as [Or No], " & \_

' " custcode as [Customercode] FROM tblpayments where curdate between #" & DateValue(DateTimePicker1.Value) & "# and #" & DateValue(DateTimePicker2.Value) & "#")

jokenselect("SELECT ID,jobordercode as [Job Order],Orno as [Or No],curdate as [Date],Cashier,Amount FROM tblpayments where curdate between #" & DateValue(DateTimePicker1.Value) & "# and #" & DateValue(DateTimePicker2.Value) & "#")

filltotaltable(DataGridView1)

With DataGridView1

.Columns(5).DefaultCellStyle.Format = "n"

End With

'kjjndnnjsjfs

End Sub

Private masterBindingSource As New BindingSource()

Private detailsBindingSource As New BindingSource()

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

If Radiocustpay.Checked = True Then

DataGridView1.DataSource = masterBindingSource

DataGridView2.DataSource = detailsBindingSource

Try

'Connection obj to database

Dim conn As OleDb.OleDbConnection = jokenconn()

conn.Open()

Dim da1, da2 As New OleDb.OleDbDataAdapter

Dim sql1 As String = "Select custcode,custname,custcontact,Address,custtel from tblcustomer"

Dim sql2 As String = "Select jobordercode as [Job Order],custcode as [Customer-Code],Orno as [Or No],curdate as [Date],Cashier,Amount from tblpayments"

Dim data As New DataSet()

data.Locale = System.Globalization.CultureInfo.InvariantCulture

da1 = New OleDb.OleDbDataAdapter(sql1, conn)

da1.Fill(data, "tblcustomer")

da2 = New OleDb.OleDbDataAdapter(sql2, conn)

da2.Fill(data, "tblpayments")

Dim relation As New DataRelation("ustorders", \_

data.Tables("tblcustomer").Columns(0), \_

data.Tables("tblpayments").Columns(1))

data.Relations.Add(relation)

masterBindingSource.DataSource = data

masterBindingSource.DataMember = "tblcustomer"

detailsBindingSource.DataSource = masterBindingSource

detailsBindingSource.DataMember = "ustorders"

Catch ex As Exception

MessageBox.Show(ex.Message)

End Try

ElseIf Radiojobest.Checked = True Then

Try

'Connection obj to database

Dim conn As OleDb.OleDbConnection = jokenconn()

conn.Open()

Dim da1, da2 As New OleDb.OleDbDataAdapter

Dim sql1 As String = " SELECT jobordercode as [Job Order],itemcode as [Item Code],Parts,Labor,pickup as [Pick up], " & \_

" Delivery,totalamount as [Total Amount],Downpayment,balanceamount as [Balance],Settled, " & \_

" Remarks FROM tbljobestimate"

Dim sql2 As String = "SELECT joborderno AS [Job Order], itemcode AS [Item Code], custcode AS [Customer Code], " & \_

" Qty, Description, serialno AS [Serial No], Problem, actiontaken AS [Action Taken], dateIn AS [Date In], " & \_

" datereleased AS [Date Out], recievedby AS [Recieved By], tech\_incharge AS Incharge, " & \_

" transactiontype AS [Transaction Type]FROM tbliteminfo"

Dim data As New DataSet()

data.Locale = System.Globalization.CultureInfo.InvariantCulture

da1 = New OleDb.OleDbDataAdapter(sql1, conn)

da1.Fill(data, "tbljobestimate")

da2 = New OleDb.OleDbDataAdapter(sql2, conn)

da2.Fill(data, "tbliteminfo")

Dim relation As New DataRelation("ustorders", \_

data.Tables("tbljobestimate").Columns(0), \_

data.Tables("tbliteminfo").Columns(0))

data.Relations.Add(relation)

masterBindingSource.DataSource = data

masterBindingSource.DataMember = "tbljobestimate"

detailsBindingSource.DataSource = masterBindingSource

detailsBindingSource.DataMember = "ustorders"

Catch ex As Exception

MessageBox.Show(ex.Message)

End Try

End If

End Sub

Private Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

Try

jokenselect("SELECT tblcustomer.ID,custname,custcontact,Address,custtel , " & \_

" jobordercode as [Job Order],Orno as [Or No],curdate as [Date],Cashier, " & \_

" Amount FROM tblcustomer LEFT JOIN tblpayments ON tblcustomer.custcode =tblpayments.custcode where tblcustomer.custname like'%" & TextBox1.Text & "%'")

'jokenselect("SELECT ID,jobordercode as [Job Order],curdate as [Date],Cashier,Orno as [Or No],Amount, " & \_

' " custcode as [Customercode] FROM tblpayments")

filltotalcustmerpayments(DataGridView1)

Catch ex As Exception

MessageBox.Show(ex.Message)

End Try

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

If TextBox1.Text.Length = 0 Then

MsgBox("Please provide customer name before Clicking preview!", MsgBoxStyle.Information)

Else

RadioButton1.Checked = False

RadioButton2.Checked = False

RadioButton3.Checked = False

RadioButton4.Checked = False

RadioButton5.Checked = False

AllReport.Show()

End If

End Sub

End Class

1. technician.vb

Public Class technician

Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click

TextBox1.Enabled = True

Button1.Enabled = True

End Sub

Private Sub technician\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

Button1.Enabled = False

TextBox1.Enabled = False

jokenselect("SELECT ID,techname as [Technician Name] FROM tbltch")

fillitemtable(DataGridView1)

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

jokeninsert("INSERT INTO tbltch(techname)VALUES('" & TextBox1.Text & "')")

jokenselect("SELECT ID,techname as [Technician Name] FROM tbltch")

fillitemtable(DataGridView1)

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Try

With order

.txtincharge.Text = DataGridView1.CurrentRow.Cells(1).Value

End With

Me.Close()

Catch ex As Exception

MsgBox("Please select a customer!", MsgBoxStyle.Exclamation)

End Try

End Sub

Private Sub GroupBox1\_Enter(sender As Object, e As EventArgs) Handles GroupBox1.Enter

End Sub

End Class

1. useraccount.vb

Public Class useraccounts

Public Sub displaymember()

jokenselect("Select userID,username as [Name], userusername as [Username],usertype as [Type] From tbluseraccounts")

filltable(DataGridView1)

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

If txtpass.Text <> txtconfirm.Text Then

MsgBox("Password Confirmation did not match!", MsgBoxStyle.Information)

Else

jokeninsert("INSERT INTO tbluseraccounts ( username, userusername, userpassword, usertype ) " & \_

" VALUES('" & txtname.Text & "','" & txtuname.Text & "','" & txtpass.Text & "','" & cbtype.SelectedItem & "')")

displaymember()

End If

cleartext(GroupBox1)

End Sub

Private Sub btndel\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btndel.Click

jokendelete("Delete \* from tbluseraccounts where userID= " & lblid.Text & "")

displaymember()

cleartext(GroupBox1)

End Sub

Private Sub btnedit\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnedit.Click

If txtpass.Text <> txtconfirm.Text Then

MsgBox("Password Confirmation did not match!", MsgBoxStyle.Information)

Else

jokenupdate("UPDATE tbluseraccounts set username ='" & txtname.Text & "' , userusername = '" & txtuname.Text & "', userpassword = '" & txtpass.Text & "', usertype= '" & cbtype.SelectedItem & "' where userID = " & lblid.Text & "")

displaymember()

End If

cleartext(GroupBox1)

End Sub

Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click

clearall(GroupBox1, DataGridView1)

End Sub

Private Sub itemdatagrid\_CellClick(ByVal sender As Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles DataGridView1.CellClick

lblid.Text = DataGridView1.CurrentRow.Cells(0).Value.ToString

txtname.Text = DataGridView1.CurrentRow.Cells(1).Value.ToString

txtuname.Text = DataGridView1.CurrentRow.Cells(2).Value.ToString

cbtype.SelectedItem = DataGridView1.CurrentRow.Cells(3).Value.ToString

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Me.Close()

End Sub

Private Sub useraccounts\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

displaymember()

End Sub

End Class

1. payable.vb

Public Class payables

Private Sub txtparts\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtparts.TextChanged

compute()

End Sub

Public Sub compute()

If Not IsNumeric(txtparts.Text) Then

txtparts.Text = Nothing

End If

If Not IsNumeric(txtlabor.Text) Then

txtlabor.Text = Nothing

End If

If Not IsNumeric(txtpickup.Text) Then

txtpickup.Text = Nothing

End If

If Not IsNumeric(txtdeliver.Text) Then

txtdeliver.Text = Nothing

End If

txttotalamnt.Text = Val(txtdeliver.Text) + Val(txtparts.Text) + Val(txtlabor.Text) + Val(txtpickup.Text)

txttotalamnt.Text = FormatNumber(txttotalamnt.Text, 2)

End Sub

Private Sub txtlabor\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtlabor.TextChanged

compute()

End Sub

Private Sub txtpickup\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtpickup.TextChanged

compute()

End Sub

Private Sub txtdeliver\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles txtdeliver.TextChanged

compute()

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

jokenupdate("UPDATE tbljobestimate set parts = '" & txtparts.Text & "',labor = '" & txtlabor.Text & "', " & \_

" pickup= '" & txtpickup.Text & "',delivery = '" & txtdeliver.Text & "',balanceamount='" & txttotalamnt.Text & "', " & \_

" totalamount='" & txttotalamnt.Text & "' where jobordercode= '" & Label2.Text & "'")

Me.Close()

Payment.Show()

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Me.Close()

End Sub

Private Sub payables\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

End Sub

End Class

1. welcome.vb

Public NotInheritable Class welcome

'TODO: This form can easily be set as the splash screen for the application by going to the "Application" tab

' of the Project Designer ("Properties" under the "Project" menu).

Private Sub welcome\_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

'Set up the dialog text at runtime according to the application's assembly information.

'TODO: Customize the application's assembly information in the "Application" pane of the project

' properties dialog (under the "Project" menu).

'Application title

If My.Application.Info.Title <> "" Then

ApplicationTitle.Text = My.Application.Info.Title

Else

'If the application title is missing, use the application name, without the extension

ApplicationTitle.Text = System.IO.Path.GetFileNameWithoutExtension(My.Application.Info.AssemblyName)

End If

'Format the version information using the text set into the Version control at design time as the

' formatting string. This allows for effective localization if desired.

' Build and revision information could be included by using the following code and changing the

' Version control's designtime text to "Version {0}.{1:00}.{2}.{3}" or something similar. See

' String.Format() in Help for more information.

'

' Version.Text = System.String.Format(Version.Text, My.Application.Info.Version.Major, My.Application.Info.Version.Minor, My.Application.Info.Version.Build, My.Application.Info.Version.Revision)

Version.Text = System.String.Format(Version.Text, My.Application.Info.Version.Major, My.Application.Info.Version.Minor)

'Copyright info

Copyright.Text = My.Application.Info.Copyright

End Sub

Private Sub ApplicationTitle\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ApplicationTitle.Click

End Sub

End Class

**Chapter-8: Testing**

In the development of our Job Management System in Visual Basic.NET, we have conducted rigorous testing to ensure the system’s reliability and effectiveness. Our testing process was comprehensive, covering all possible scenarios to validate the system’s functionality. We employed a variety of testing methodologies, including unit tests, integration tests, and load tests. Each component was individually tested to ensure its proper operation, and then the components were tested together to verify their seamless integration. Load tests were performed to confirm the system’s ability to handle the expected user load. The results of all these tests were positive, demonstrating that the system is robust, reliable, and ready for deployment. This thorough testing gives us confidence in the system’s ability to manage jobs effectively and meet the needs of the organization. We are proud to report that our Job Management System has passed all tests with flying colors and is ready to make job management easier and more efficient. Overall, testing was critical to ensuring that program worked as expected and met the needs of the users. Any issues or bugs discovered during testing were fixed before the final version of the program was released.

**Chapter-9: Project Results:**

**SCREENSHOTS OF THE PROJECT**

**1 Login Page**

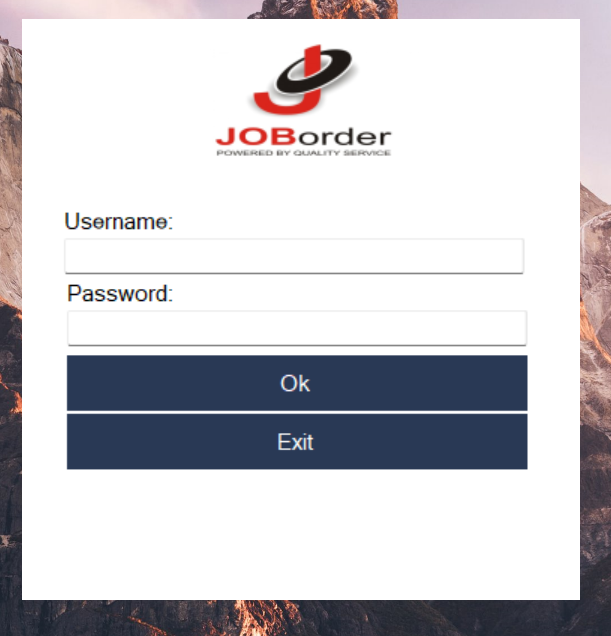


Fig 3 Login Page

**2 After Login Page**



Fig 4 welcome page

**3 Main Page**



Fig 5 Main Page

**4 After Clicking Manage Customer Page**

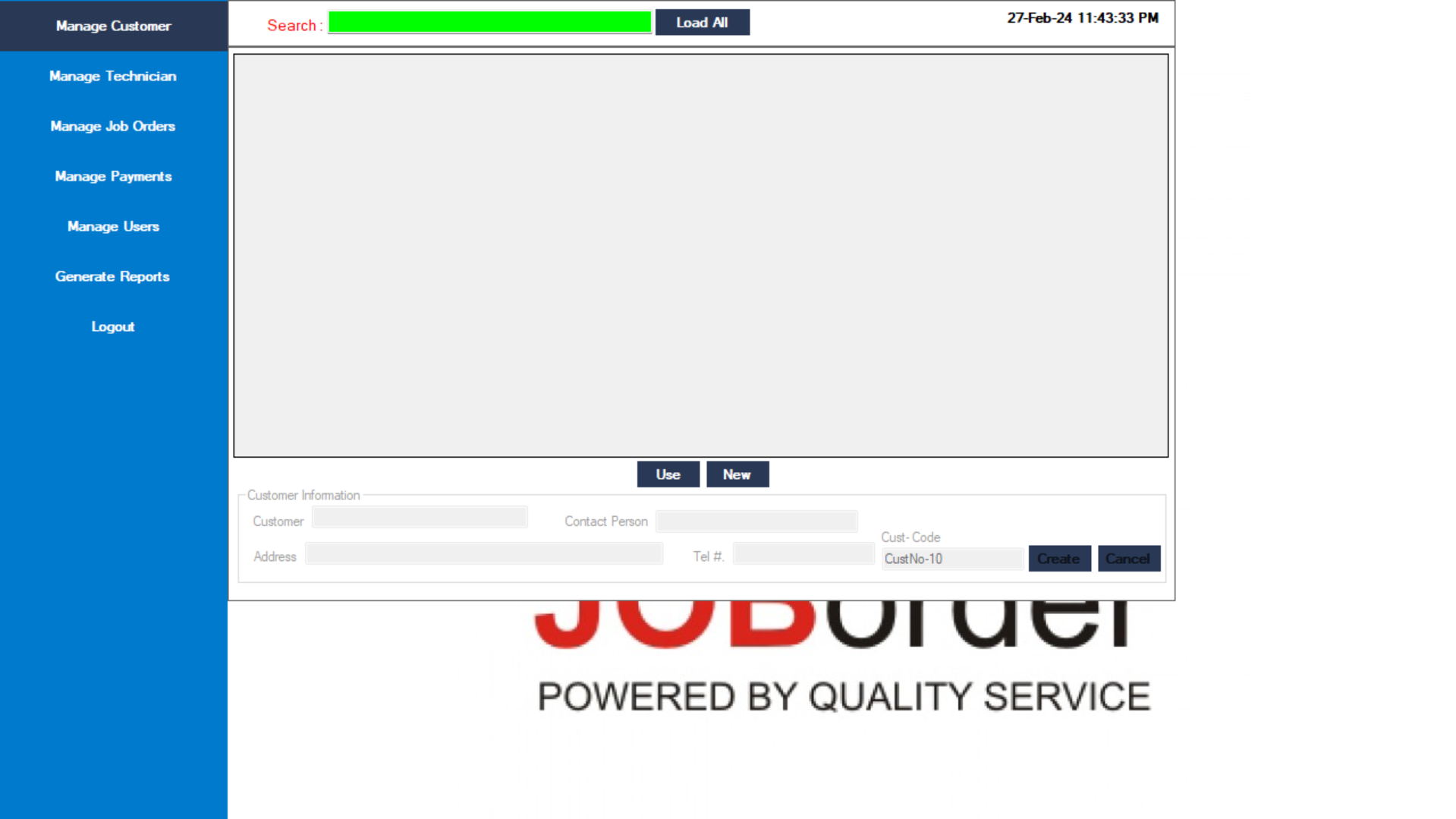


Fig 6 Manage custom

**5 After Clicking Manage Technician Page**



Fig 7 Manage Technician

**6 After Clicking management Payment Page**

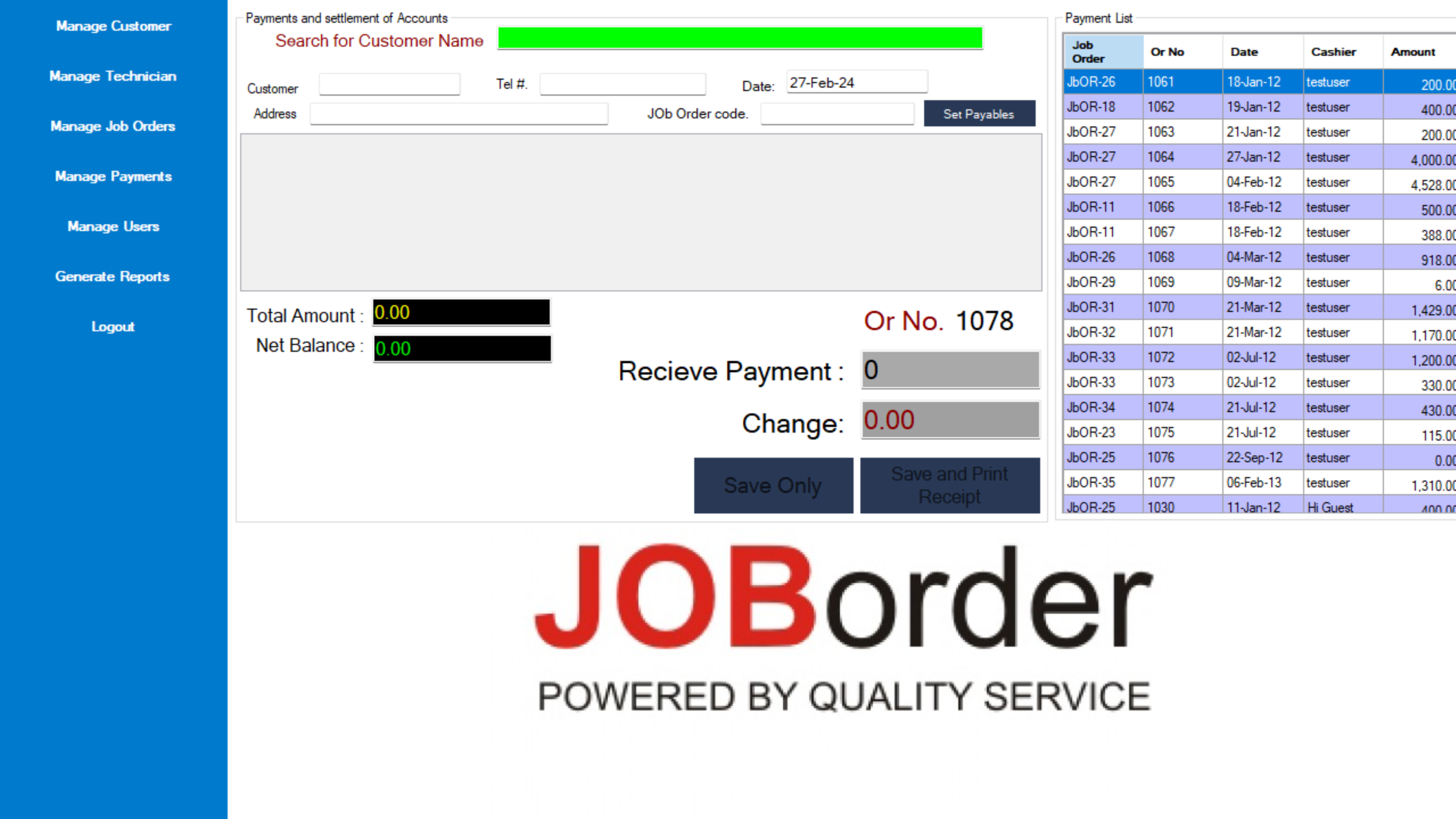


Fig 8 Manage Payment

**7 Manage user Page**

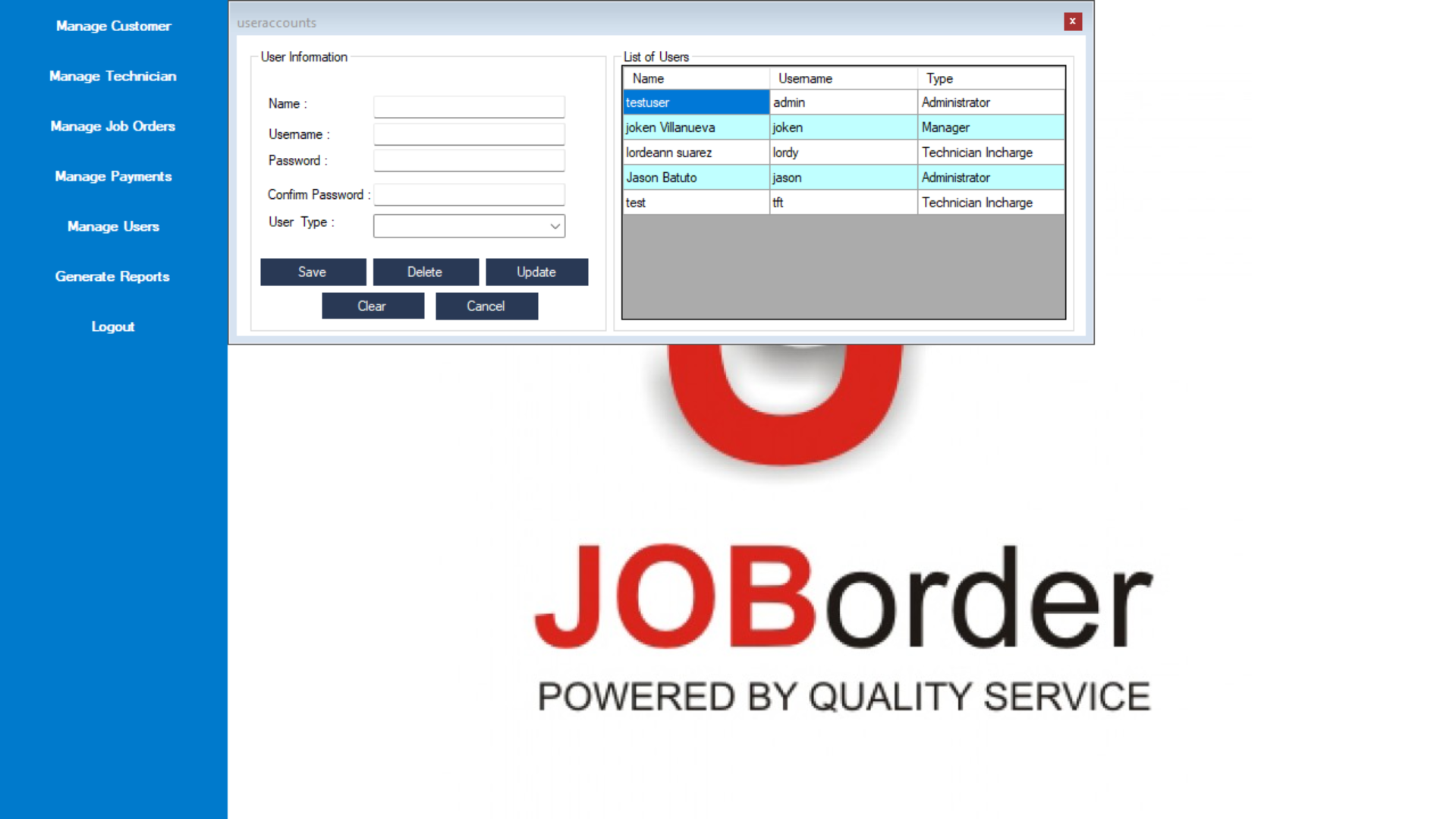


Fig 9 Manage user

**8 Generate Report Page**

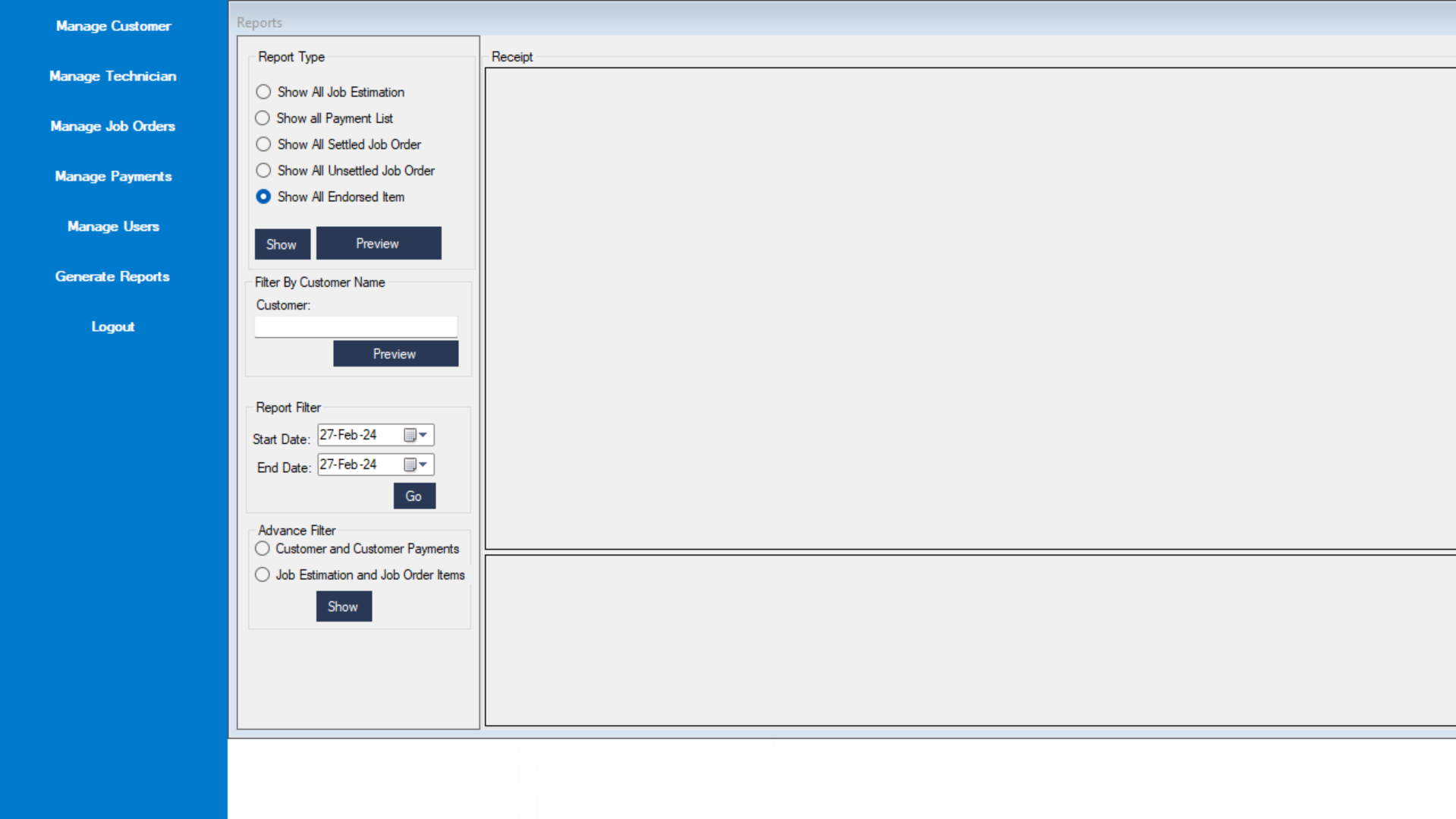


Fig 10 Final Report

**9 After click Log Out & Return login page**

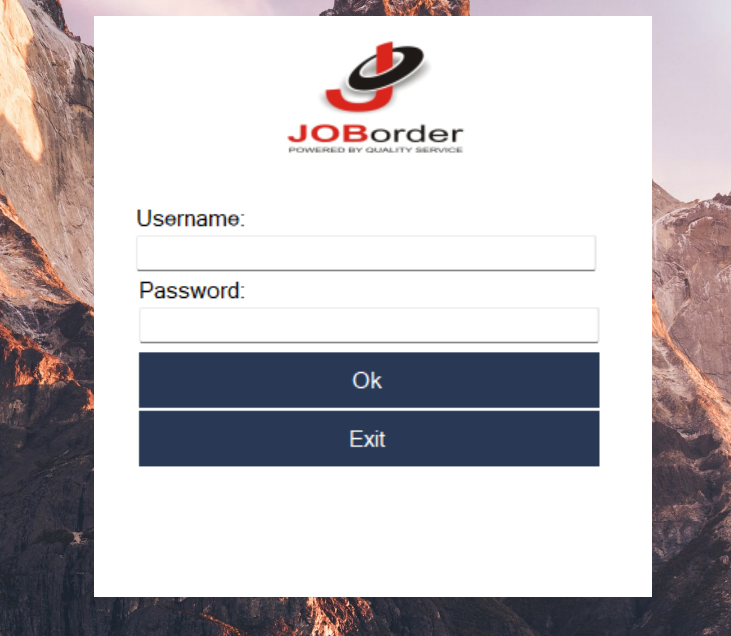


Fig 11 Exit

# Chapter-10: Conclusion

After completing the project, we have gained valuable experience in VB.NET programming and its features. Although our application may not be as feature-rich or advanced as other commercial options available, we are proud of what we have accomplished as beginners in programming. This project has provided us with the opportunity to develop a deeper understanding of VB.NET's various aspects and techniques, and we believe it has helped us improve our programming skills overall.

We welcome any feedback and suggestions for improving our calculator application. We understand that no program is perfect, and there is always room for improvement. We hope that this project will serve as a foundation for future endeavors in programming.

We would like to express our gratitude to our professor for his guidance and support throughout this project. We would also like to thank our friends and colleagues for their help and encouragement. Thank you for taking the time to review our work.

Thank you for taking the time to read about our Job Management System.

* **Raju Gupta**
* **Laxmi Roy**
* **Kritika Adhikari**

# REFERENCES

* <https://learn.microsoft.com/en-us/dotnet/visual-basic/>
* [kaushal878 (Kaushal Raj Trital) (github.com)](https://github.com/kaushal878)
* [W3Schools Online Web Tutorials](https://www.w3schools.com/)