SE361- Object-Oriented Design
Spring 2013

Just the Job Cleaning Solutions

Red Team

Devin Edmundowicz

Brandon Weber

Eric Sanders

Table Of Contents

Summary

For this project, we were to build a computer program for Just-the-Job Cleaning Solutions that handles all of its day-to-day exercises that would normally be done on paper. In our program, we implemented a function that allows a receptionist or company manager to add, modify, and delete customers, as well as the ability to view a list of the company's current customers. Our program also managed the business side of Just-the-Job by allowing the company to add and keep track of its employees as well as any pending or completed jobs that the company has taken on. Lastly, our program allows the company to record and manage its invoices, as well as the payments made by customers for the jobs completed.

In order to complete this project, we used the object-oriented design process, which is a four step process that starts with understanding what the company managers wanted out of the program and ends with implementation through time-box iterations. During the analysis process, we used a use-case diagram, which maps each of the requirements that the program is to fulfill. In the design process, we created a class diagram, which maps every class and form that needs to be implemented into our code in order for our program to handle all of the use cases. The class diagram is to be set up into a three-tier system with boundary classes at the top that extend to the control classes that handle multiple instances of entities, which then extend to the entity classes. From this diagram, we created CRC cards, which entails a class's responsibility and the collaborators the class uses to complete its responsibilities. Lastly, we derived a sequence diagram for each use case from the use-case diagram. The purpose of these diagrams is to map out the message passing that occurs between classes in completing a function.

Chapter 0- Introduction

Project Description

This project is based on a case study included in the book *Object-Oriented Development*, required for the class SE-361. The case study describes some automation needs for a local business, "Just-the-Job" Cleaning Solutions. The manager of Just-the-Job had recently come into some funding, and decided it was time to upgrade their computer system. This computer system needed to handle most of the paperwork involved in the company's day to day operations.

Functional Requirements

1. **Manage customers** (Receptionist & Manager)
    a. Add a new customer (Receptionist & Manager)
    b. Delete an existing customer (Manager Only)
    c. Modify information stored about an existing customer (Receptionist & Manager)
    d. Display/print information about an existing customer based on his/her id or name (Receptionist & Manager)
    e. Display/print a list of all customers and their information (Receptionist & Manager)

2. **Manage employee** (i.e. cleaner) (Receptionist & Manager)
    a. Add a new employee—name, id, address, pay-rate, weekly schedule, etc. (Receptionist & Manager)
    b. Delete an existing employee (Manager Only)
    c. Display/print information about an existing employee (Receptionist & Manager)
    d. Display/print a list of all employees with their information (Receptionist & Manager)
    e. Modify information stored about an existing employee (Receptionist & Manager)

3. **Manage jobs/appointments** (Receptionist & Manager)
    a. Add/register a new job (Receptionist & Manager)
    b. Delete/cancel an existing job (Manager Only)
    c. Display/print the status of a specific job (e.g. job number; address of property, completion status, by-whom etc.) (Receptionist & Manager)
    d. Display/print a list of all jobs with their completion status (Receptionist & Manager)
    e. Display/print a list of all completed jobs (Receptionist & Manager)
    f. Display/print a list of all pending jobs (Receptionist & Manager)
    g. Modify information stored about an existing job (Receptionist & Manager)

4. **Manage customer invoices for one-time jobs** (Receptionist & Manager)
    a. Create a new invoice (Receptionist & Manager)
    b. Cancel an existing invoice (Manager Only)
    c. Modify an existing invoice (Receptionist & Manager)
    d. Display/print an existing invoice (Receptionist & Manager)
    e. Display/print a list of all invoices and their payment status (Receptionist & Manager)

5. **Manage customer invoices for regular jobs** (Receptionist & Manager)
    a. Create a new invoice (Receptionist & Manager)
    b. Cancel an existing invoice (Manager Only)
    c. Modify an existing invoice (Receptionist & Manager)
    d. Display/print an existing invoice (Receptionist & Manager)
    e. Display/print a list of all invoices for a given week with payment status (Receptionist & Manager)

6. **Manage customer payments** (Receptionist & Manager)
    a. Record a full payment (Receptionist & Manager)
    b. Cancel/credit a payment (Manager Only)
    c. Print receipt (Receptionist & Manager)

7. **Maintain manager's personal weekly schedule** (Receptionist & Manager)
    a. Add a new appointment (Receptionist & Manager)
    b. Cancel an existing appointment (Receptionist & Manager)
    c. Modify an existing appointment (Receptionist & Manager)
    d. Print weekly schedule (Receptionist & Manager)

8. **Save information** (Receptionist & Manager) Upon user request, save all data to disk at any time. (This is in addition to the automatic save to disk which occurs at shutdown)

Software Development Process

This project was designed to be completed over the course of nine weeks, and was separated into three time-boxed iterations. We used the object-oriented approach to our design, which meant we often had to update previously completed diagrams and documentation.

The object-oriented approach begins with the *inception* phase, which includes establishing the business case for the project, incorporating basic risk assessment, and the development of the scope of the system. The next step is *elaboration*, and this step deals with putting the basic architecture of the system in place. The third, and most involved, step of the process is called the *construction* phase, and it is during this stage that the three time-box iterations are implemented. The final phase of the approach is the *transition* phase, which deals with transferring the completed program to the client. We were not required to deal with the transition phase during the project, as we ended with the construction phase.

Team Members and Roles

Devin Edmundowicz – Developer/GUI Designer
Eric Sanders – Developer/Documentation
Brandon Weber – Developer/System Design

Organization of the Report

This report includes all code and diagrams associated with the analysis, requirements, design, and implementation of all three iterations. Included are the use-case diagram, class diagram, sequence diagrams, CRC cards, TOE tables. We have also included screenshots of our GUI and code snippets with feedback from our tester. We have also explained each of our roles in the project, and how our abilities evolved over the course of the semester. There are explanations of our overall experience, and how the skills learned in SE361 will translate very well to our future careers.

Chapter 1- Iteration 1

Analysis

The analysis of our domain model began with reading and understanding the project description. From the project description we created a list of nouns. We found the following nouns:

Red Team's List of Nouns

Just-The-Job- Outside Scope
Company- Too Vague
House cleaning services- Operation/Event
House- Redundant to property
People- Too Vague
Customer
Receptionist- Attribute
Appointment
Office Manager- Attribute
Property- Attribute
Date- Attribute
Price- Attribute
Booking form-Input/Output
Office- Input/Output
Employees
Invoice- Input/Output
Payment- Input/Output
Customer numbers- Attribute
Details- Too vague
New computer system- Whole System
Daily routine- Operation/Event
Weekly Schedule- Input/Output

Final List
Customer
Employee
Appointment

Figure (1 -1) List of Nouns

We began by removing any noun that was out of the scope of the program. We removed:

Removed out of scope

- *Just-The-Job*
- *New computer system*

We then removed any nouns that were too vague. We removed:

Removed too vague

- *Company*
- *People*
- *Details*

We then removed nouns that were attributes of a class. We removed:

Removed attributes

- *Receptionist*
- *Office Manager*
- *Property*
- *Date*
- *Price*
- *Customer Numbers*

We then removed nouns that were an operation or even. We removed:

Removed Operation/Event

- *House Cleaning Services*
- *Daily routine*

We then removed nouns that were Redundant. We removed:

Removed Redundant Nouns

- *House*

We then removed nouns that were Input/Output. We removed

Removed Input/Output Nouns

- Booking Form
- Office
- Invoice
- Payment
- Weekly Schedule

We then had our final nouns.

Final Nouns

- Customer
- Appointment

- Employee

After reviewing our nouns we created six classes

1. clsCustomer
2. clsCustomerListManager
3. clsEmployee
4. clsEmployeeListManager
5. clsJob
6. clsJobListManager

Requirements

Use Case Diagram

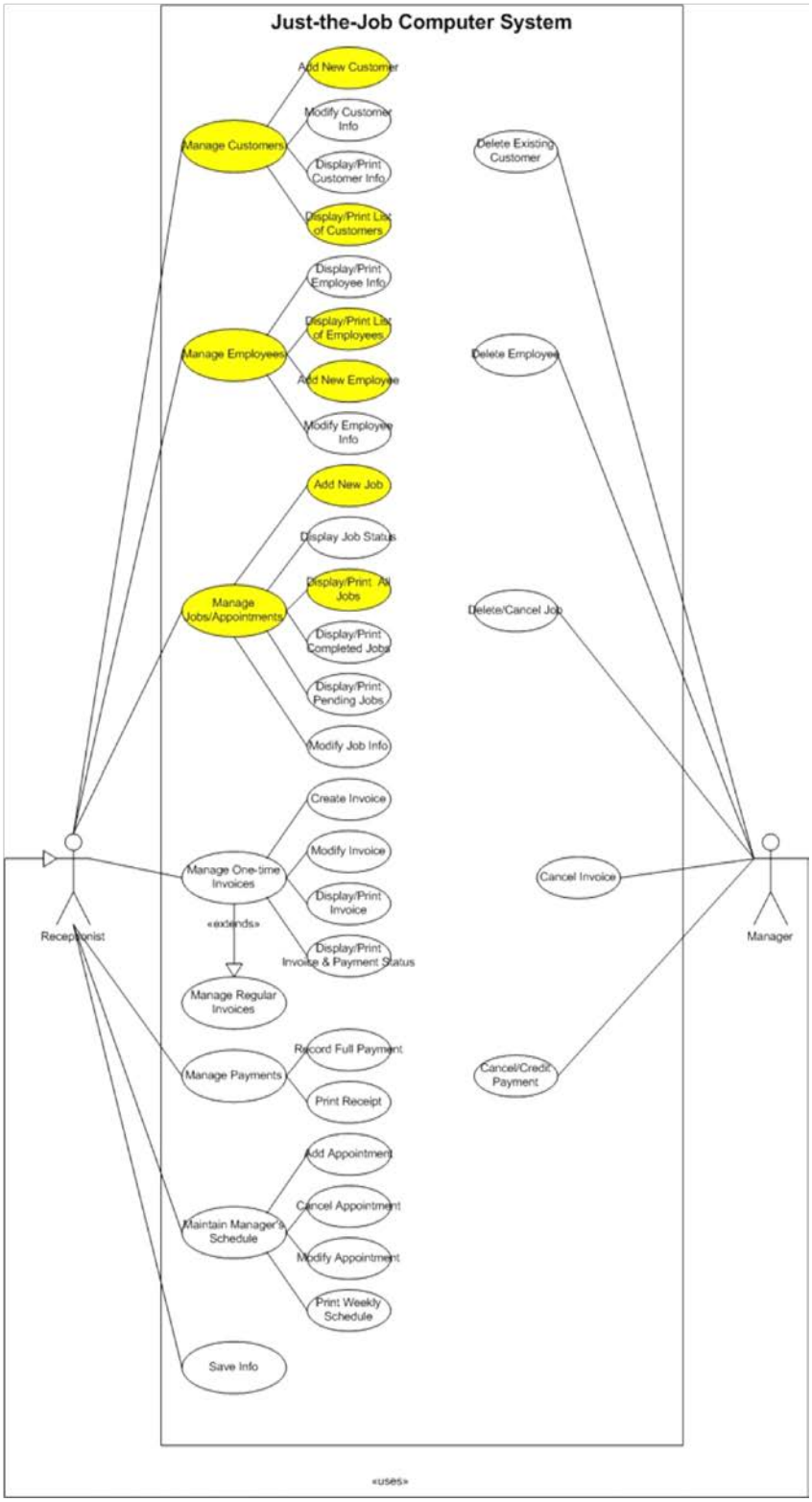


Figure (1-2) Use Case Diagram

For iteration 1, we added the following to our use case diagram and functionality to our program:

- Manager Customers
    - Add New Customer
    - Display/Print List of Customers
- Manage Employees
    - Display/Print List of Employees
    - Add New Employee
- Manage Job/Appointments
    - Add New Job
    - Display/Print All Jobs

Use Case Scenarios

These use case scenarios model typical courses of events.

Use Case: Add a new employee—name, id, address, pay-rate, weekly schedule, etc.
Actors: Receptionist/Manager
Goal: Enter basic information of new employees into the computer system.
Overview: When a new employee is hired on to the staff of Just-the-Job Cleaning Co. The receptionist enters the basic information of the employee into the new computer system. Basic info includes name, id, address, pay-rate, weekly schedule, etc.

Typical Course of Events:

Actor Action:	**System Response:**
1. Logs into system	
	2. Allows access
3. Chooses to add a new employee	
	4. Requests employee information
5. Enters employee information	
	6. Checks employee information and adds new employee
7. Logs out of system	

Alternative Courses:
Step 1 & 2: If the actor enters an incorrect username or password the system will not allow them in.
Step 3-6: The actor may decide to not add a new customer and end the process
Step 5 & 6: The actor enters the information in an incorrect format, or does not fill in all fields. They receive an error message.

Use Case: Display/print a list of all employees with their information
Actors: Receptionist/Manager
Goal: System will display the information of all the employees in the computer system and will be able to print the list.
Overview: The receptionist has the ability to display/print all of the new and existing employee's information that was entered into the computer system.

Typical Course of Events:

Actor Action:	**System Response:**
1. Logs into system	
	2. Allows access
3. Chooses to display list of all employees	
	4. Searches through text file and displays all employees in list boxes
5. Logs out of system	

Alternative Courses:
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.

Use case: Add New Customer
Actors: Manager, Receptionist
Goal: To add a new customer to the system
Description: When there is a new customer that would like cleaning, their name and information must be added to the system.
Typical Course of Events:

Actor Action:	**System Response:**
1. Logs in to system	
	2. Allows access
3. Chooses to add a new customer	
	4. Requests name and customer info
5. Enters the name and information	
	6. Checks employee information and adds new customer
7. Logs out of the system	

Alternate Courses
Step 1 & 2: If the actor enters an incorrect username or password the system will not allow them in.
Step 3-6: The actor may decide to not add a new customer and end the process
Step 5 & 6: The actor enters the information in an incorrect format, or does not fill in all fields. They receive an error message.

Use case: Display/Print list of customers and info
Actors: Manager, Receptionist
Goal: To Display and or Print a list of customers and their information
Description: A list of all customers and their information is displayed. If the actor chooses to, he or she can also print the list off.
Typical Course of Events:

Actor Action:	**System Response:**
1. Logs in to system	
	2. Allows access
3. Chooses to display the list of customers	
	4. Searches through text file and displays all customers in list boxes
7. Logs out of the system	

Alternate Courses
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.

Use Case: Add/register a new job
Actors: Receptionist/Manager
Goal: Add a new job
Description: Allows the Manager or Receptionist to add a new job to the system

Typical Course of Events:

Actor Action:	**System Response:**
1. Logs in to system	
	2. Allows access
3. Chooses to add a new job	
	4. Requests job information
5. Enters the information	
	6. Checks job information and adds new job
7. Logs out of the system	

Alternate Courses:
Step 1 & 2: If the actor enters an incorrect username or password the system will not allow them in.
Step 3-6: The actor may decide to not add a new job and end the process
Step 5 & 6: The actor enters the information in an incorrect format, or does not fill in all fields. They receive an error message.

Use Case: Display/print a list of all jobs with their completion status
Actors: Receptionist/Manager
Goal: Display/print list of all jobs and their status
Description: Allows the Manager or Receptionist to view or print a list of all jobs along with their completion statuses.

Typical Course of Events:

Actor Action:	**System Response:**
1. Logs in to system	
	2. Allows access
3. Chooses to display the list of jobs	
	4. Searches through text file and displays all jobs in list boxes
7. Logs out of the system	

Alternate Courses
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.

Sequence Diagrams

The following diagrams model the message passing of a use-case.

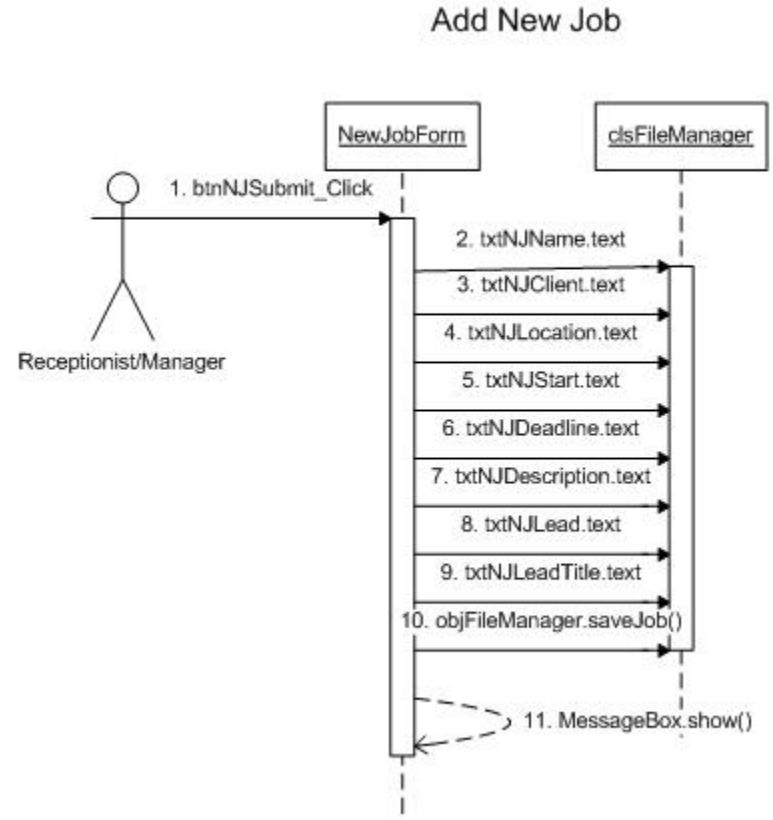


Figure 1.3 Add New Job Sequence Diagram

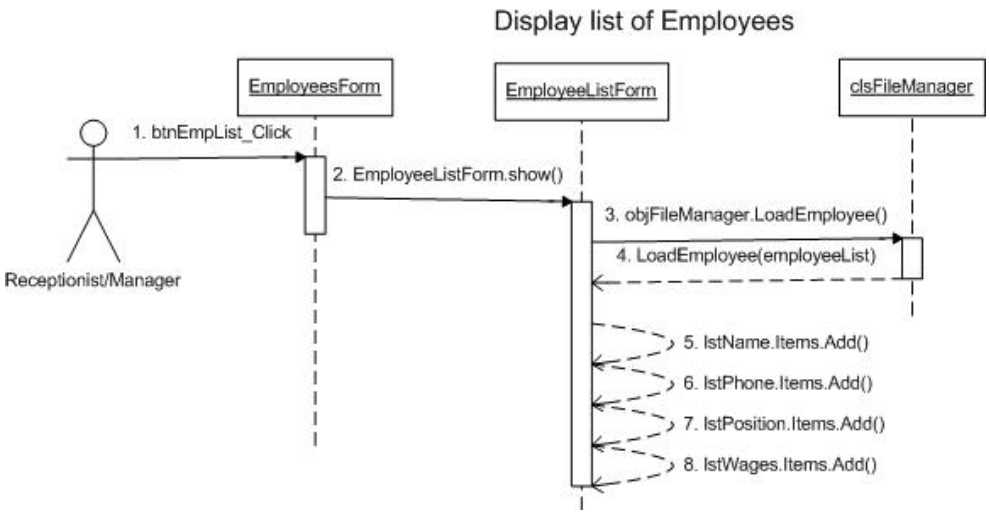


Figure 1.4 Display list of Employees Sequence Diagram

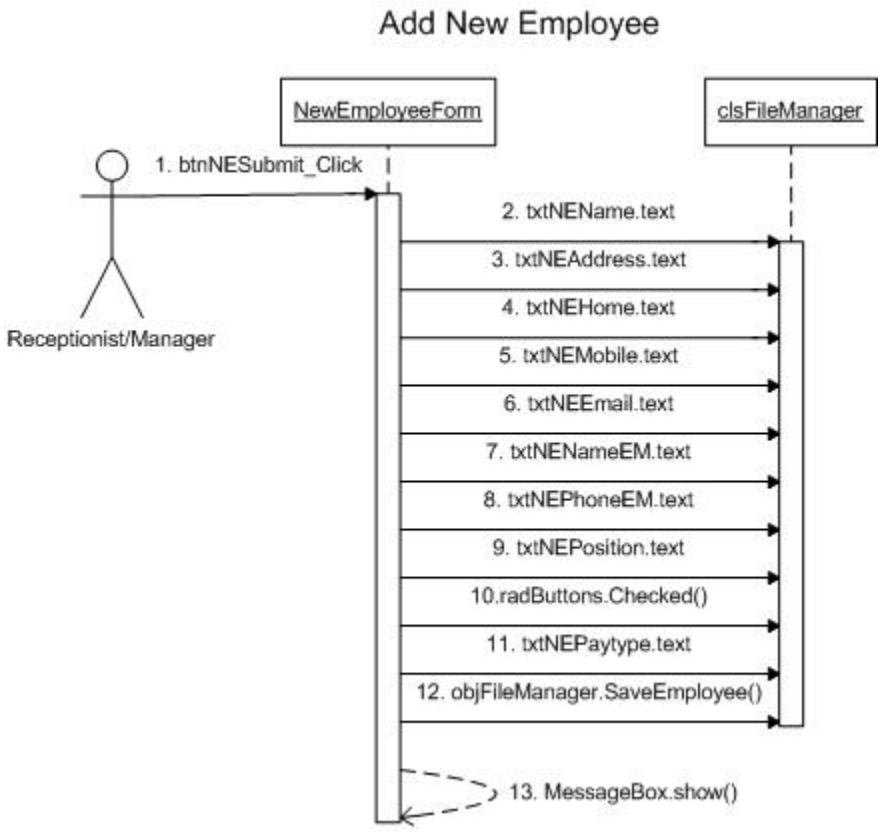


Figure 1.5 Add New Employee Sequence Diagram

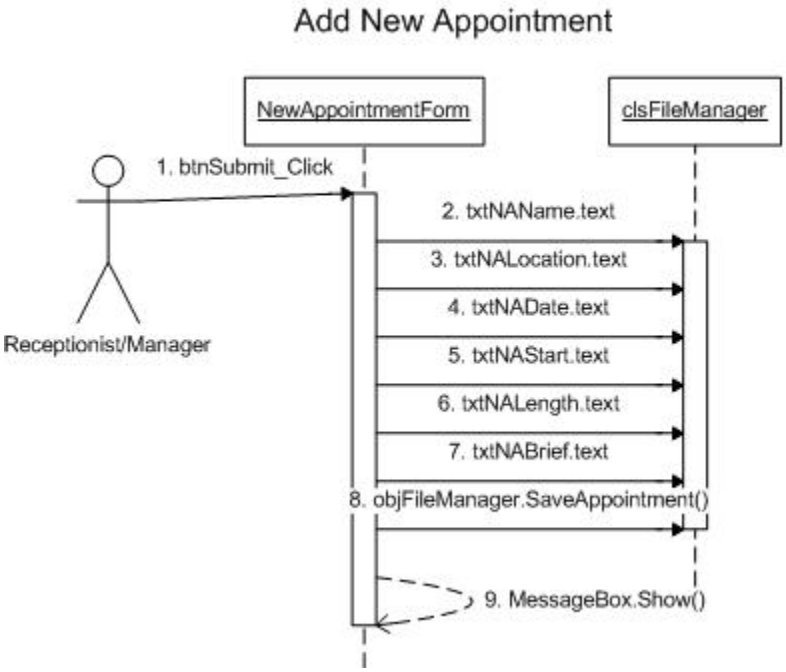


Figure 1.6 Add New Appointment Sequence Diagram

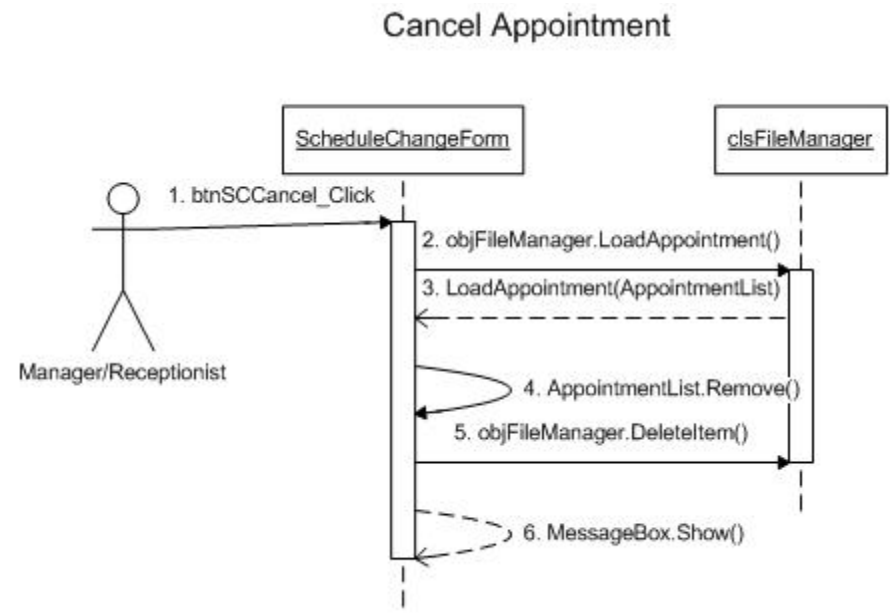


Figure 1.7 Cancel Appointment Sequence Diagram

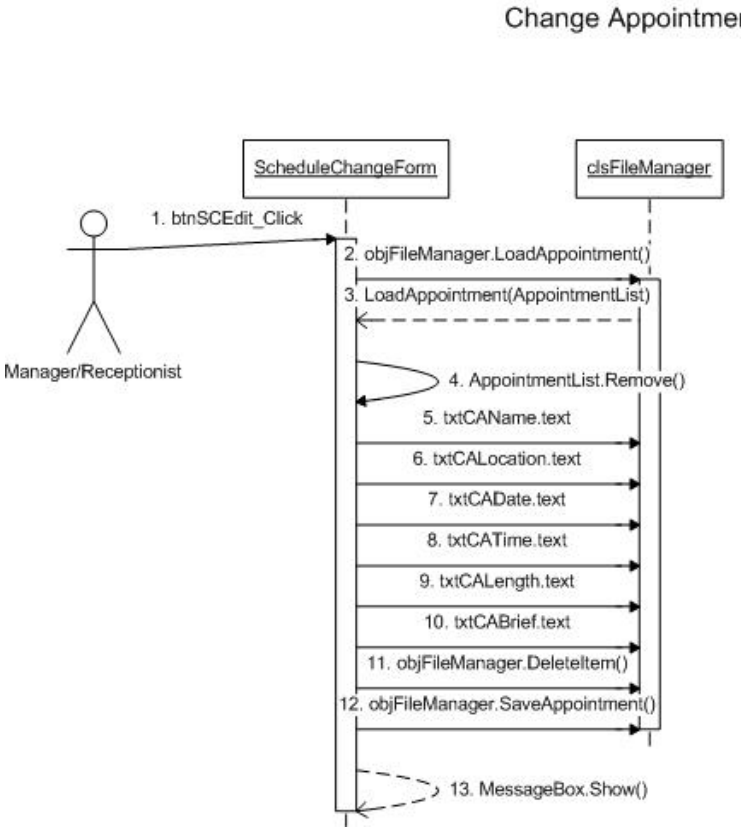


Figure 1.8 Change Appointment Sequence Diagram

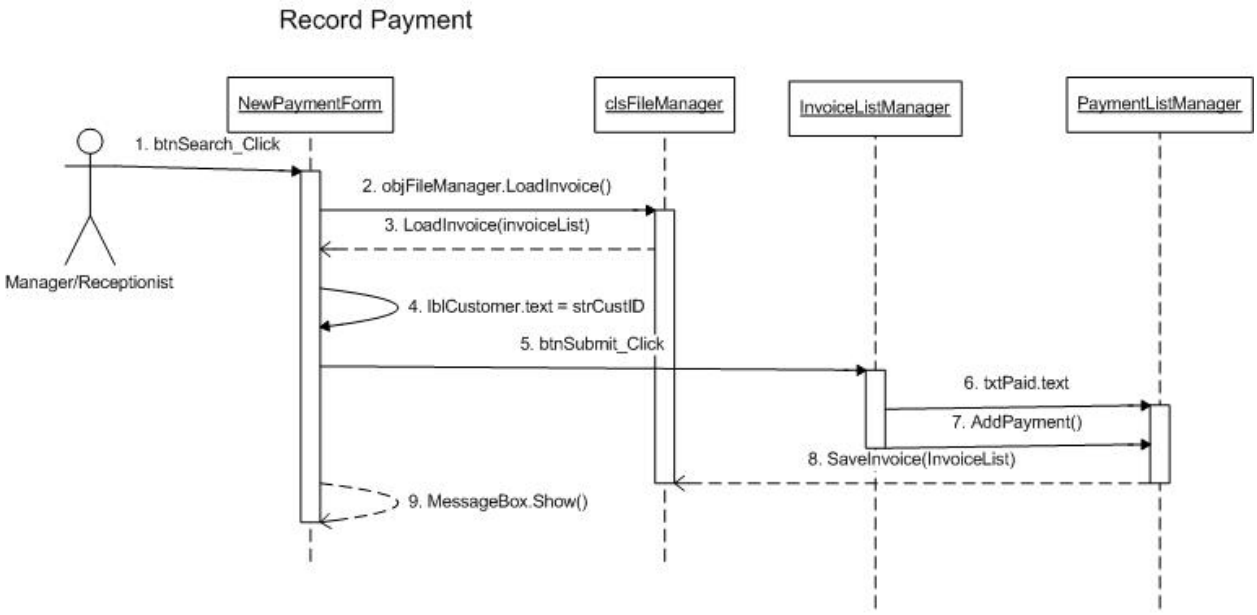


Figure 1.9 Record Payment Sequence Diagram

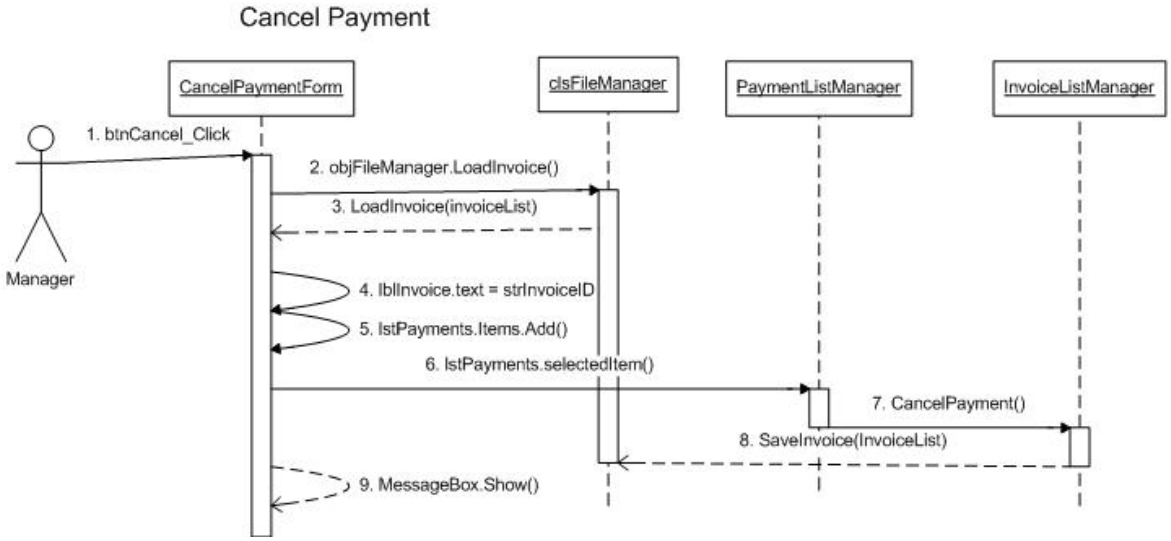


Figure 1.10 Cancel Payment Sequence Diagram

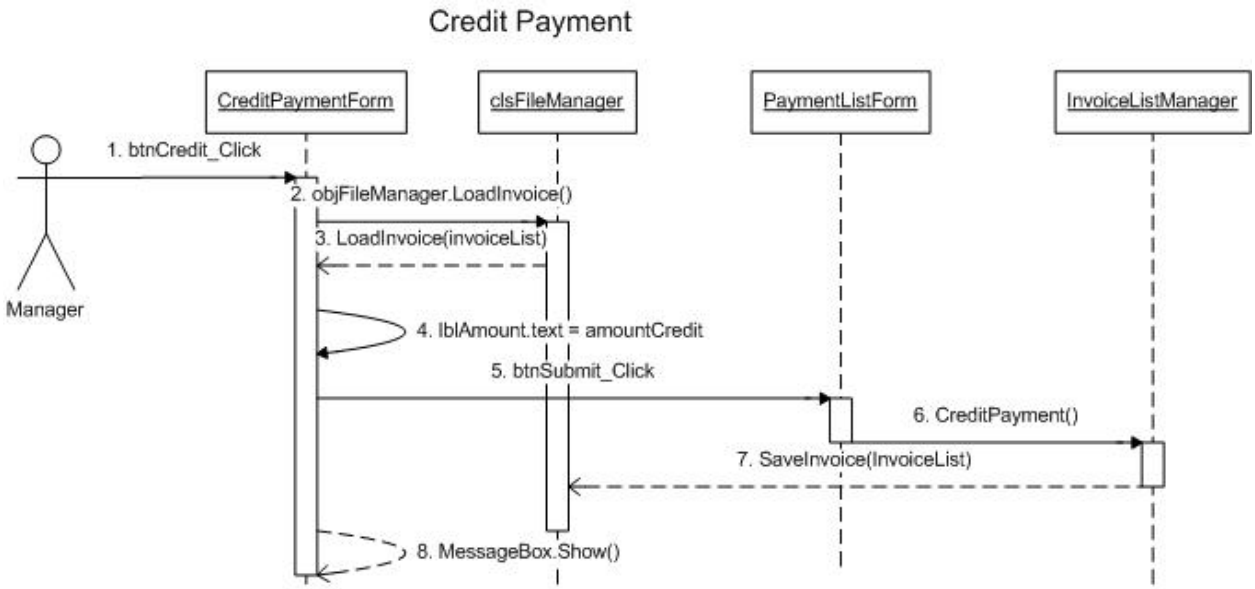


Figure 1.11 Credit Payment Sequence Diagram

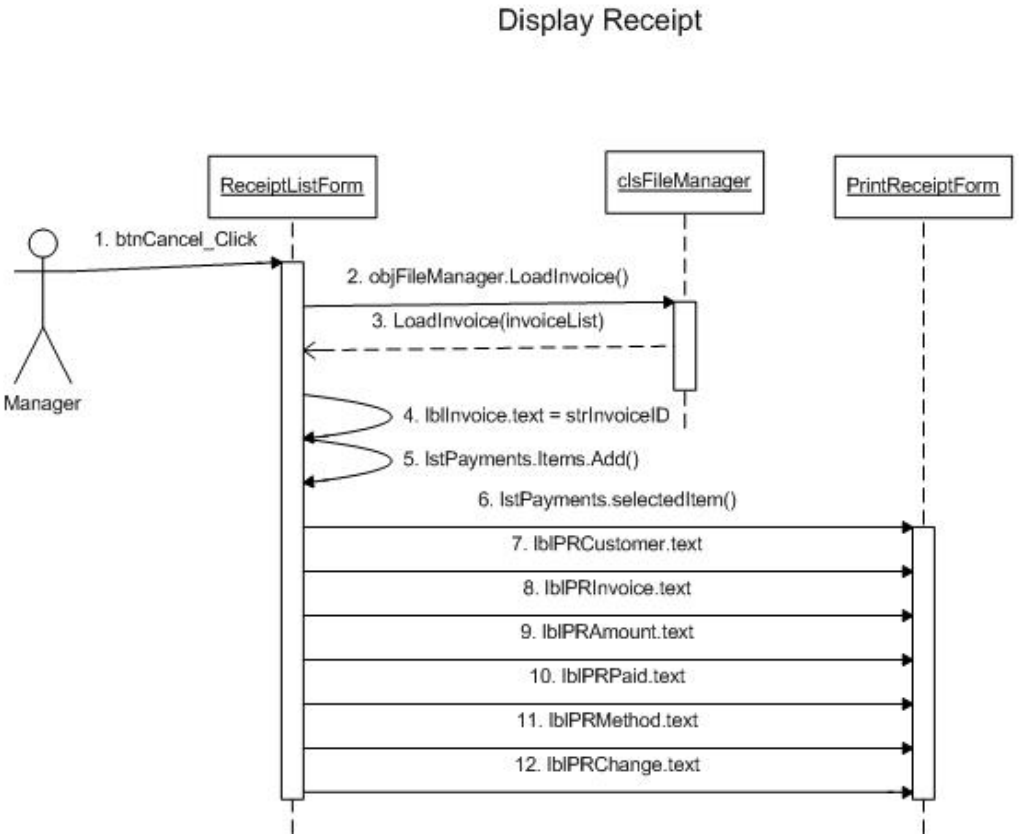


Figure 1.12 Display Receipt Sequence Diagram

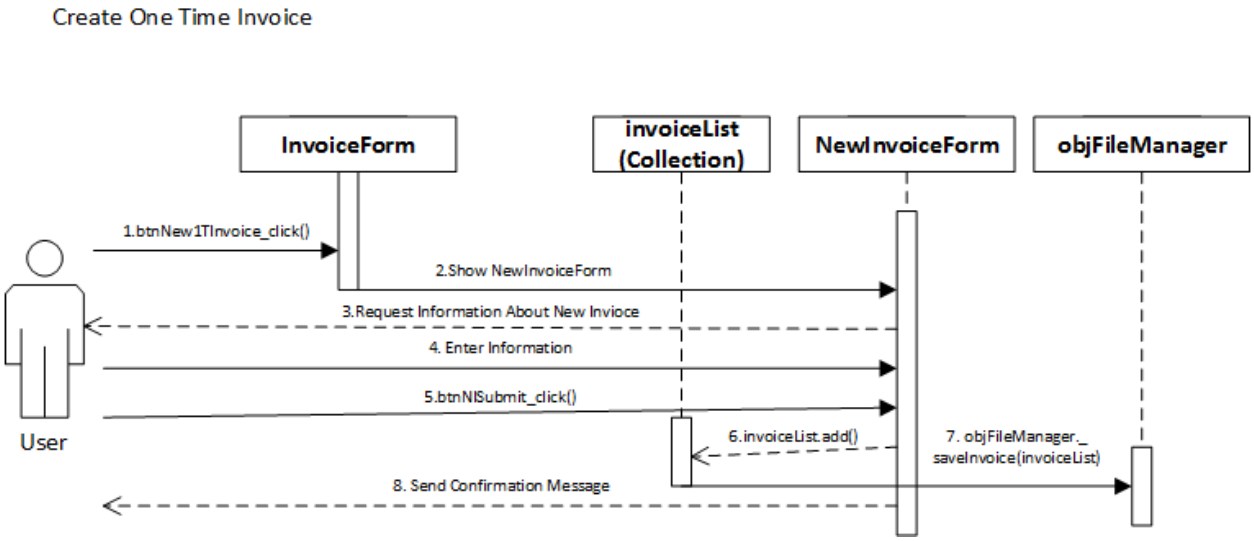


Figure 1.13 Create One Time Invoice Sequence Diagram

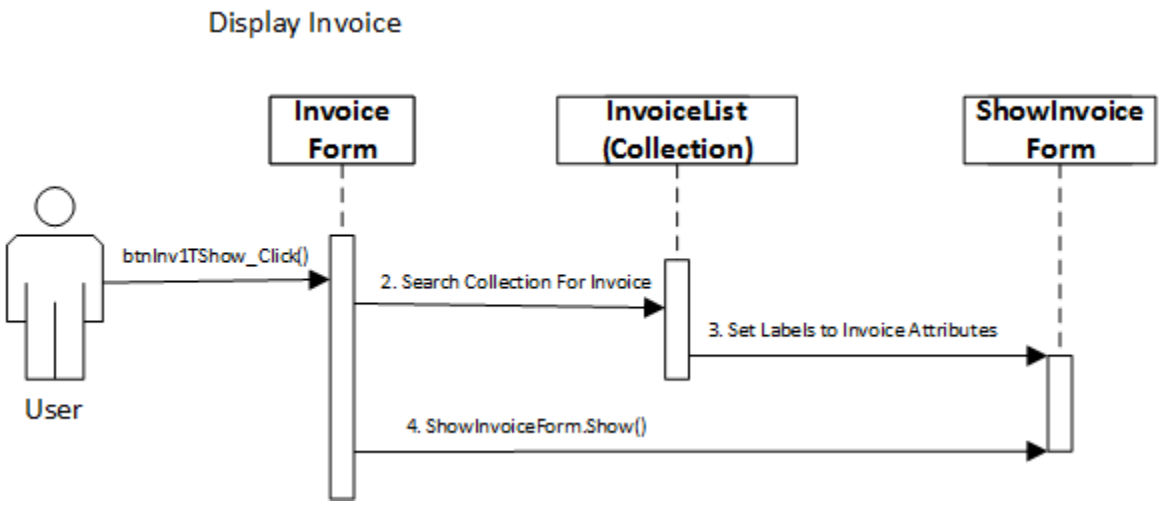


Figure 1.14 Display Invoice Sequence Diagram

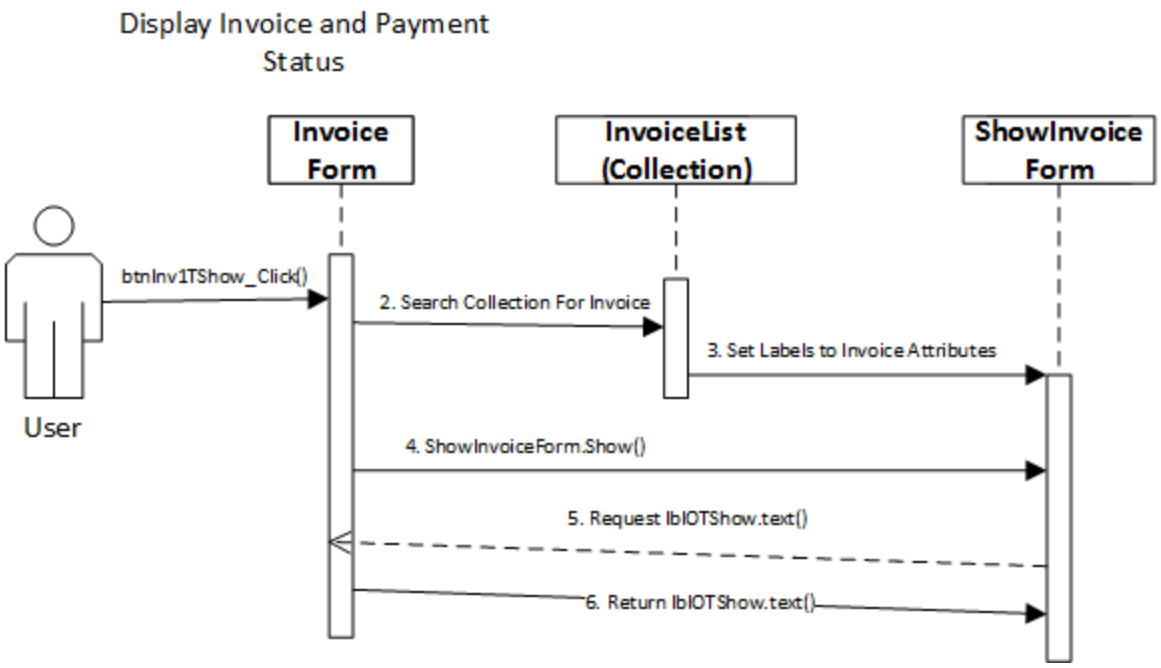


Figure 1.15 Display Invoice and Payment Status Sequence Diagram

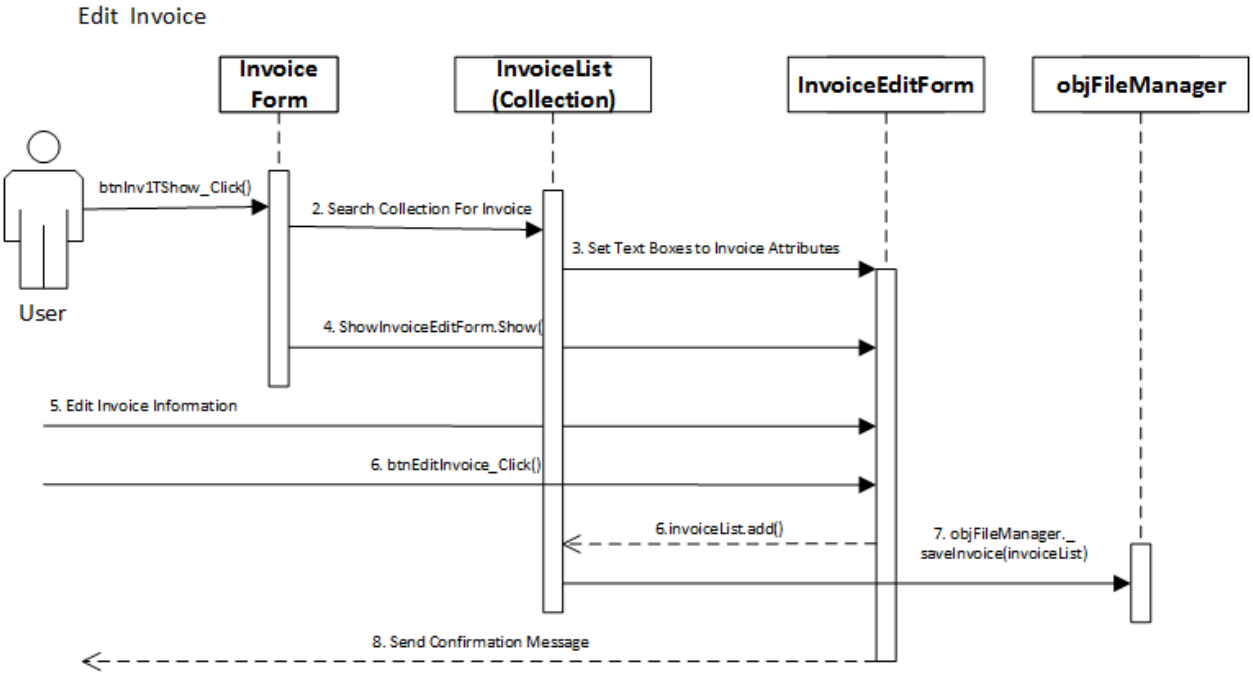


Figure 1.16 Edit Invoice Sequence Diagram

Display Job Status

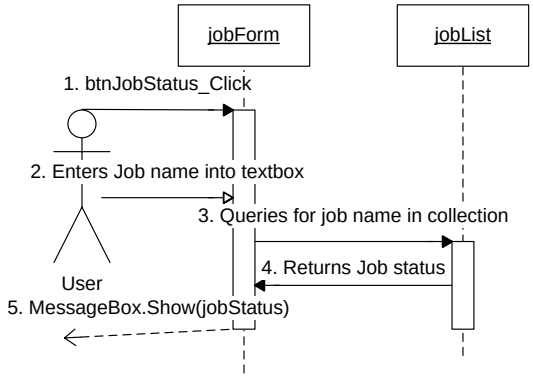


Figure 1.17 Display Job Status Sequence Diagram

Delete Invoice

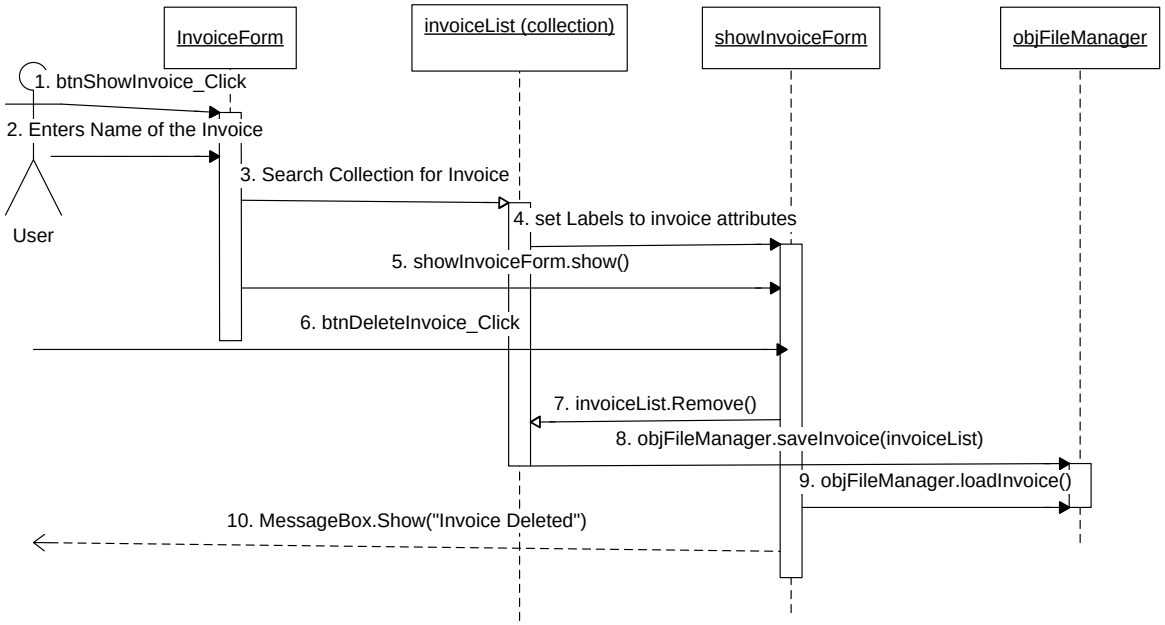


Figure 1.18 Delete Invoice Sequence Diagram

Display Pending Jobs

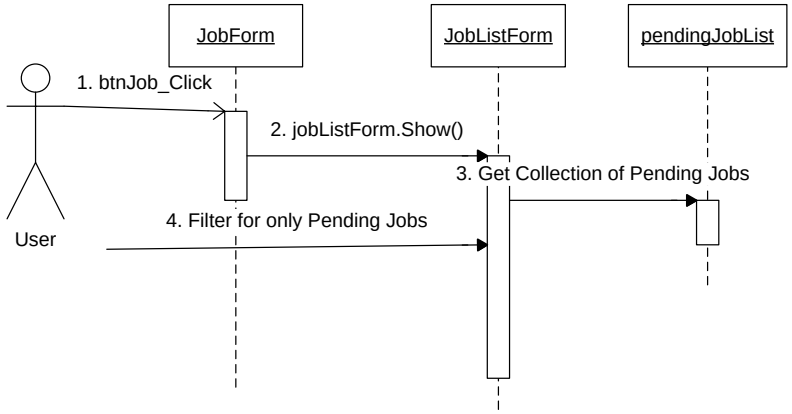


Figure 1.19 Display Pending Jobs Sequence Diagrams

Display Completed Jobs

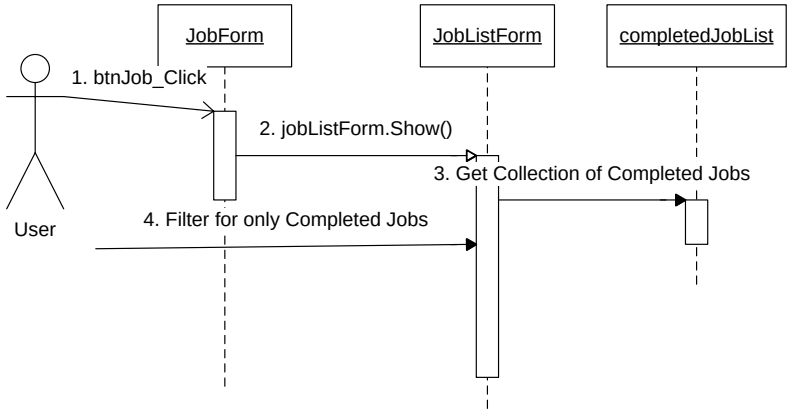


Figure 1.20 Display Completed Jobs Sequence Diagrams

Delete Job

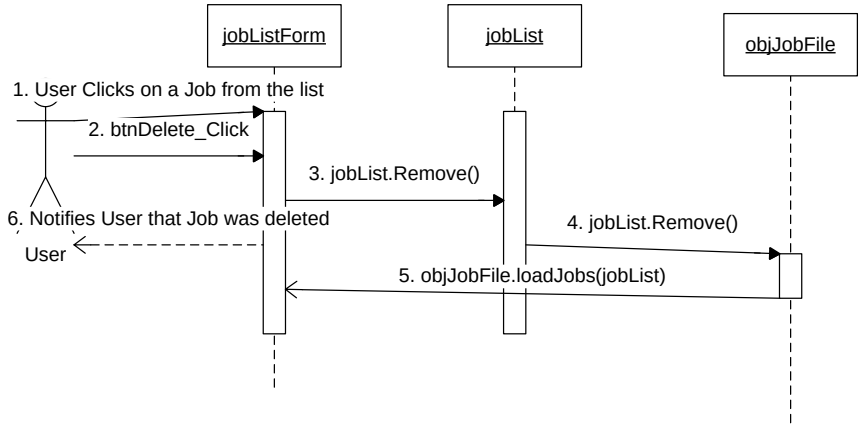


Figure 1.21 Delete Job Sequence Diagram

Display/Print Customer Info

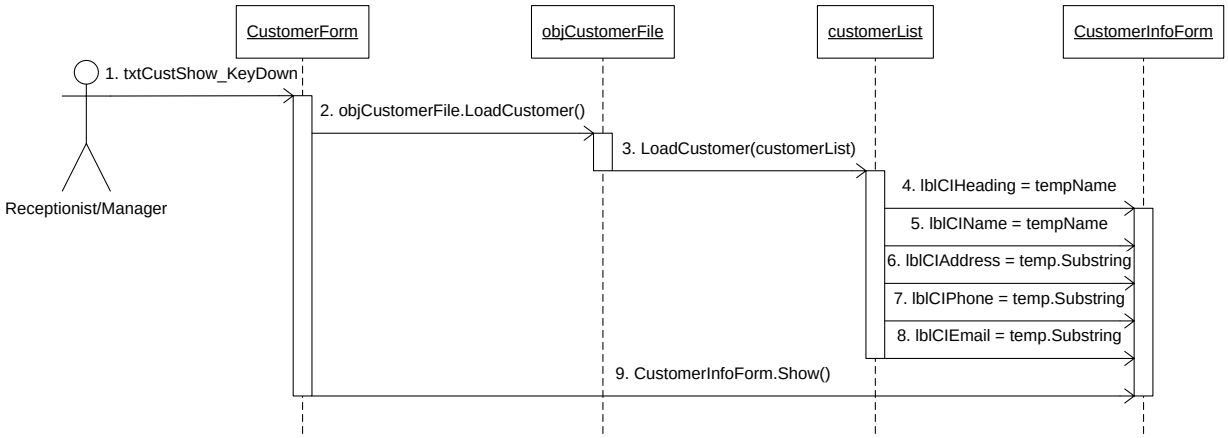


Figure 1.22 Display/Print Customer Info

Edit Customer

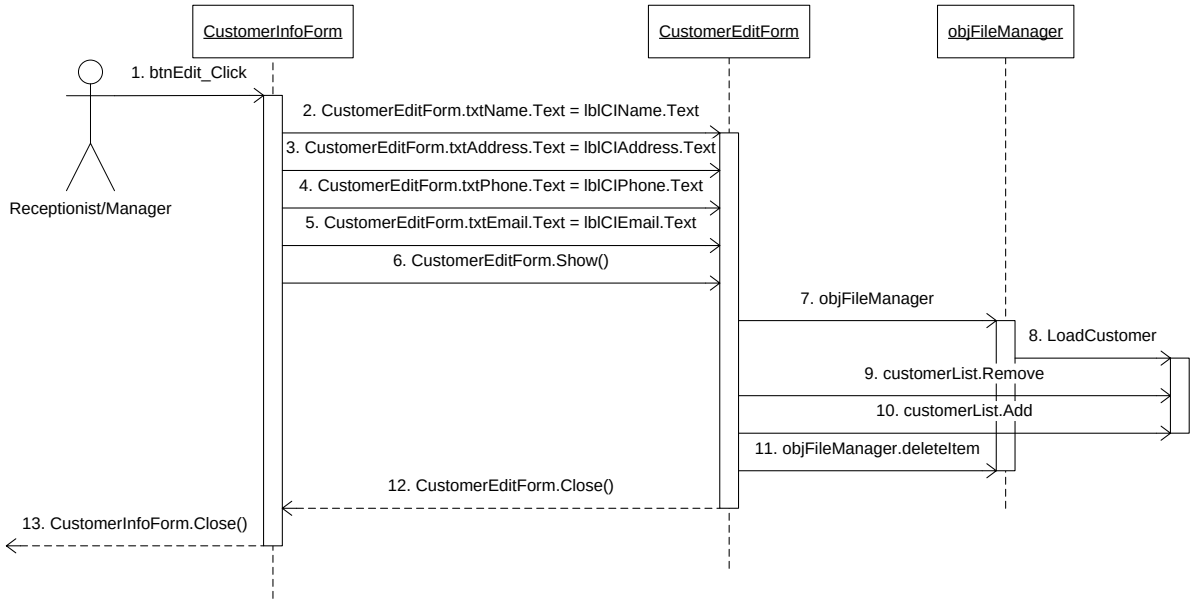


Figure 1.23 Edit Customer Sequence Diagram

Add New Customer

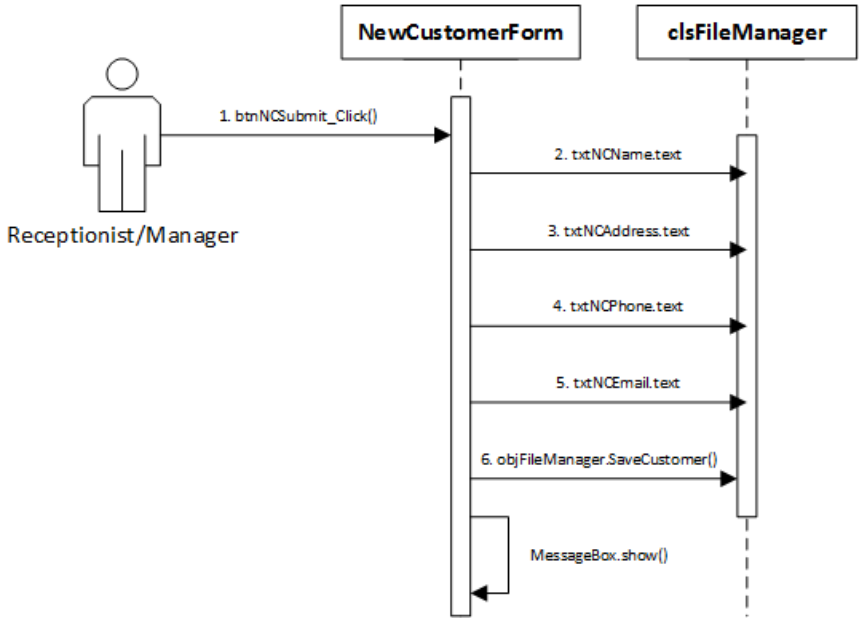


Figure 1.24 Add New Customer Sequence Diagram

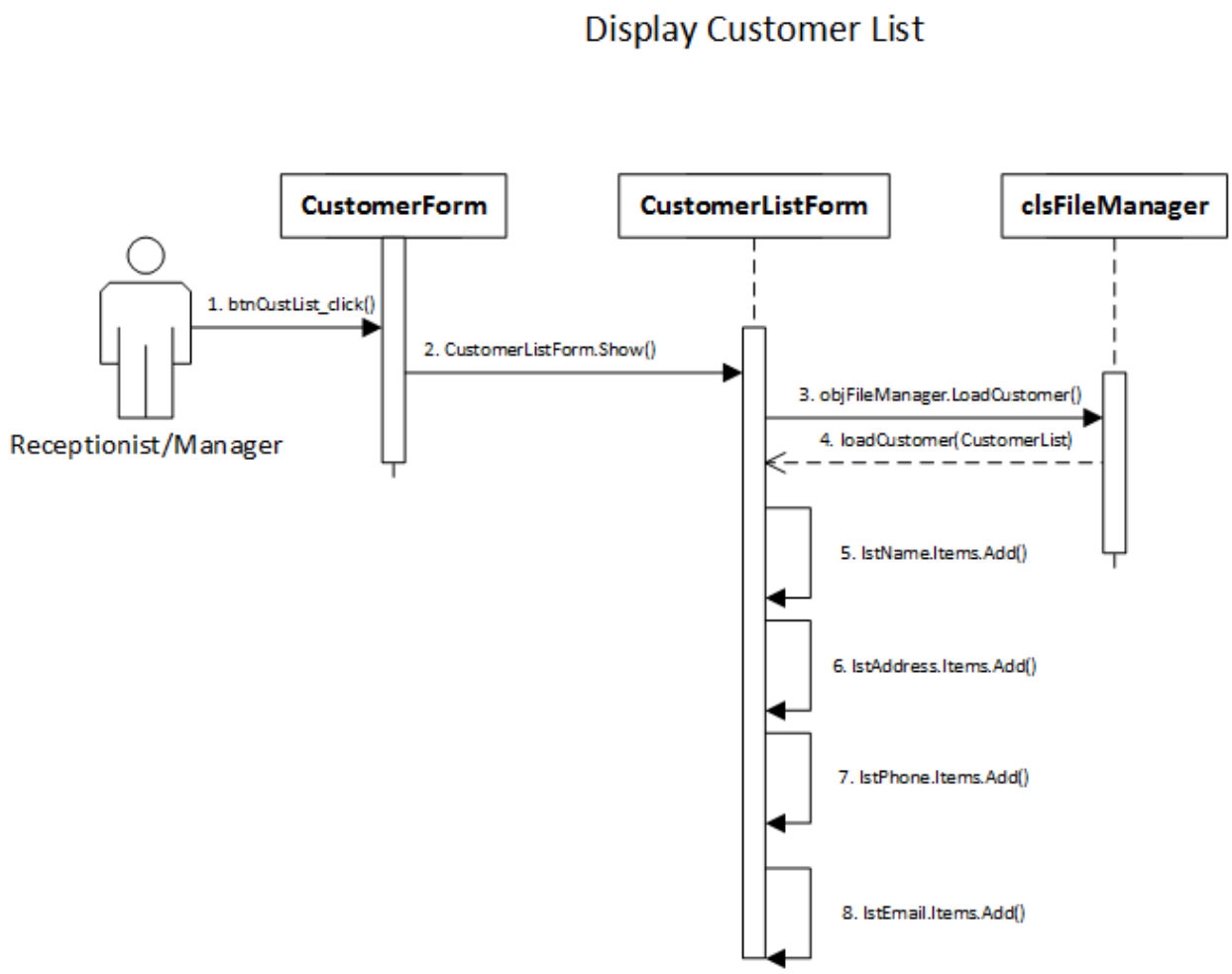


Figure 1.25 Display Customer List Sequence Diagram

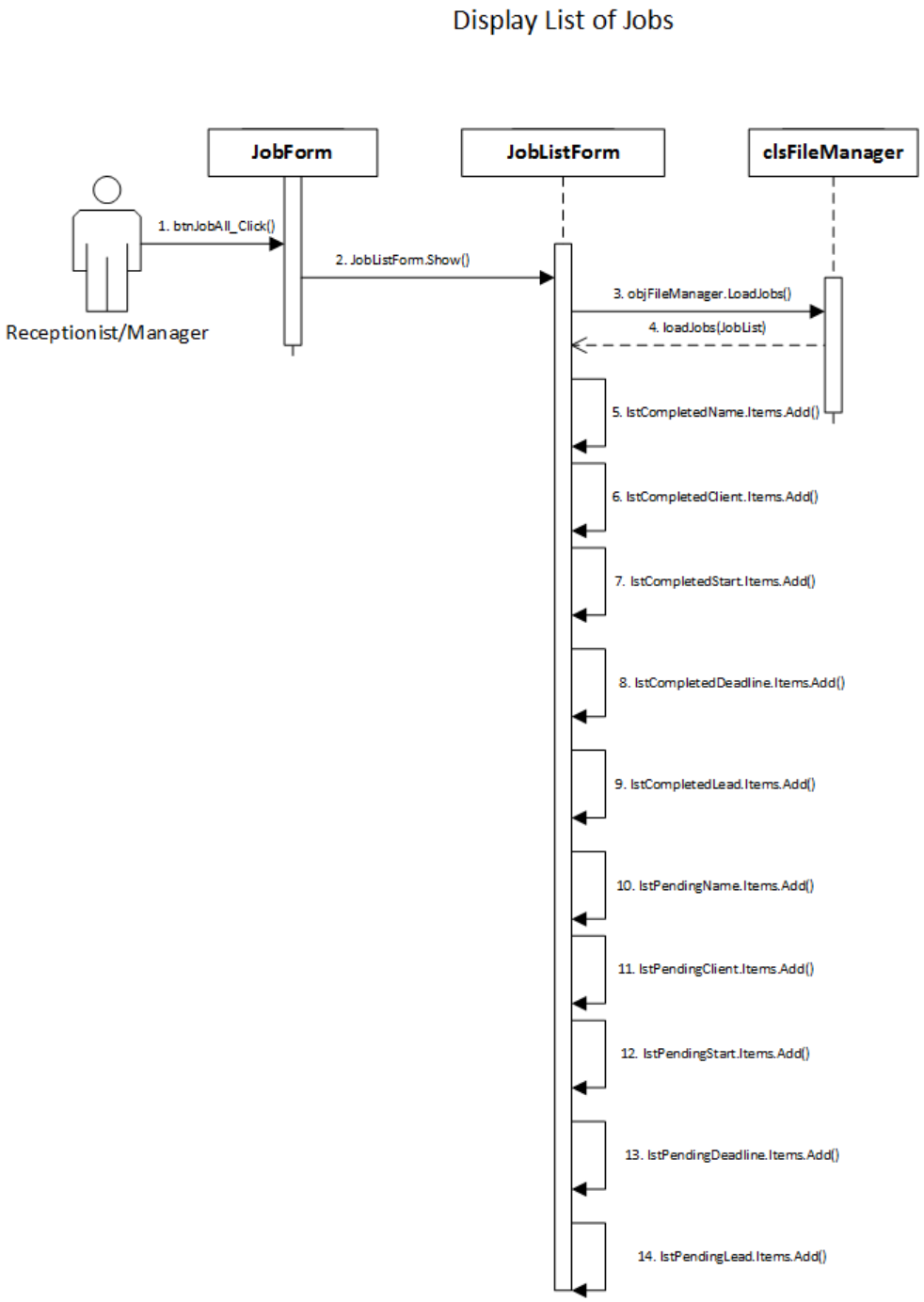


Figure 1.26 Display List of Jobs Sequence Diagram

CRC Cards

The following is our CRC Cards for iteration 1.

Customer	
Responsibility	*Collaborator*
Store information about a customer.	Assignment

Assignment	
Responsibility	*Collaborator*
Store the cost, date, and address for each job.	Customer
Store which cleaner will be completing the job.	Employee

Employee	
Responsibility	*Collaborator*
Store information about an employee.	Assignment
View and print a weekly schedule.	

Figure 1.27 CRC Cards

High-Level Class Diagram

The following diagram is our class diagram that models our classes with their attributes and their interactions with each other.

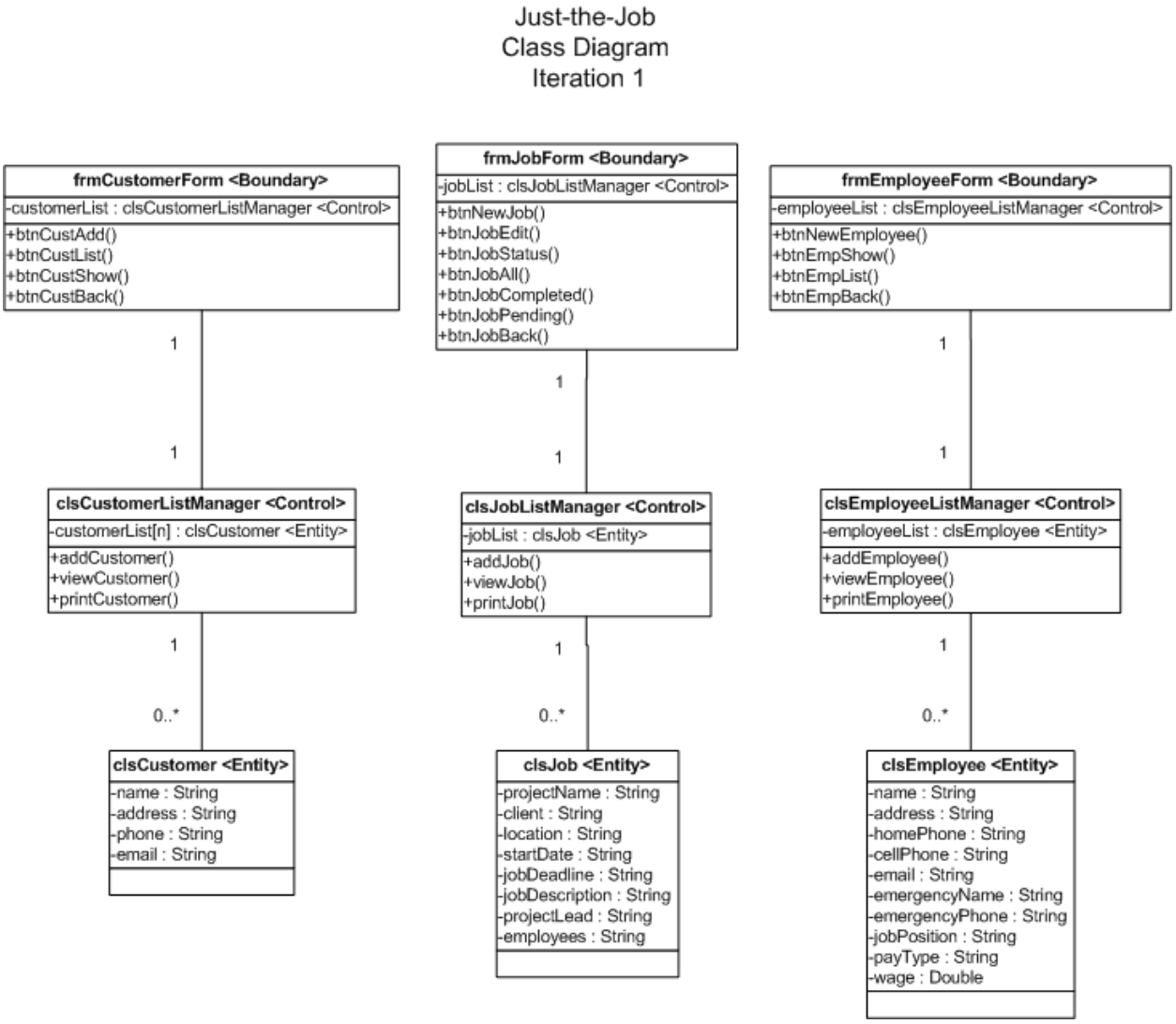
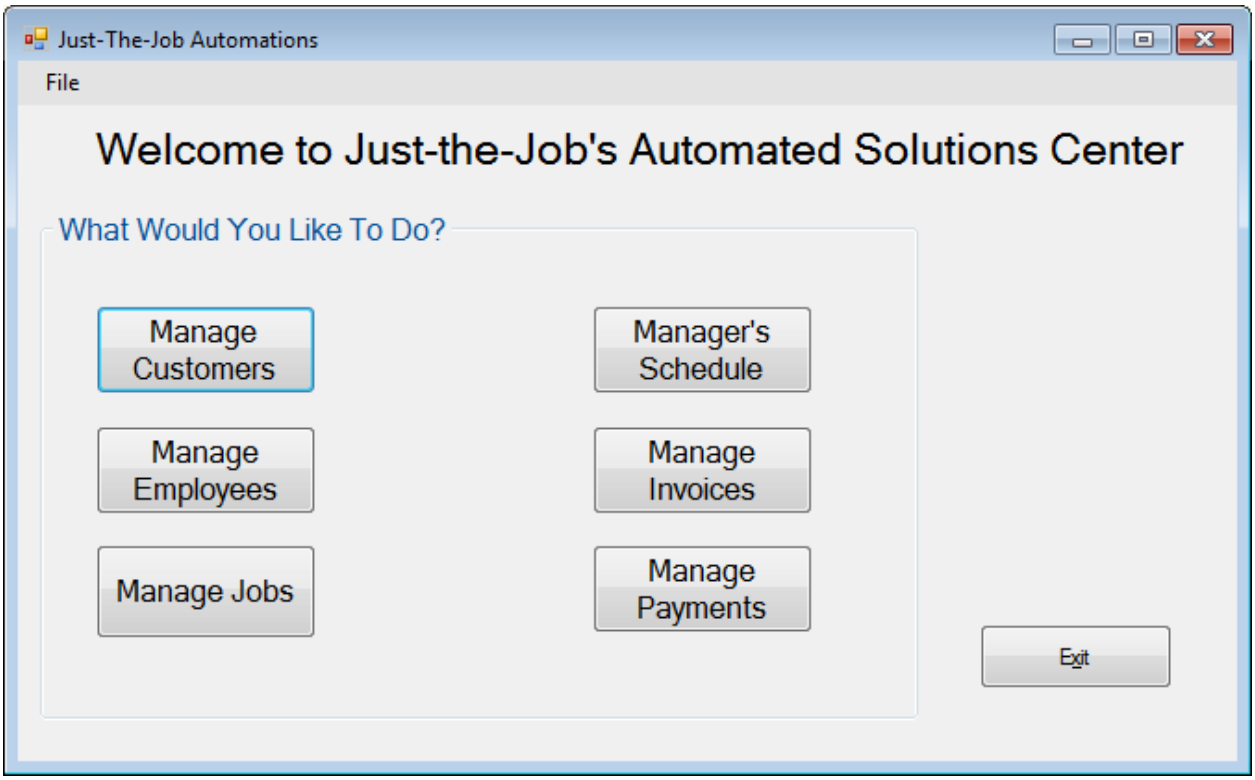


Figure 1.28 High-Level Class Diagram

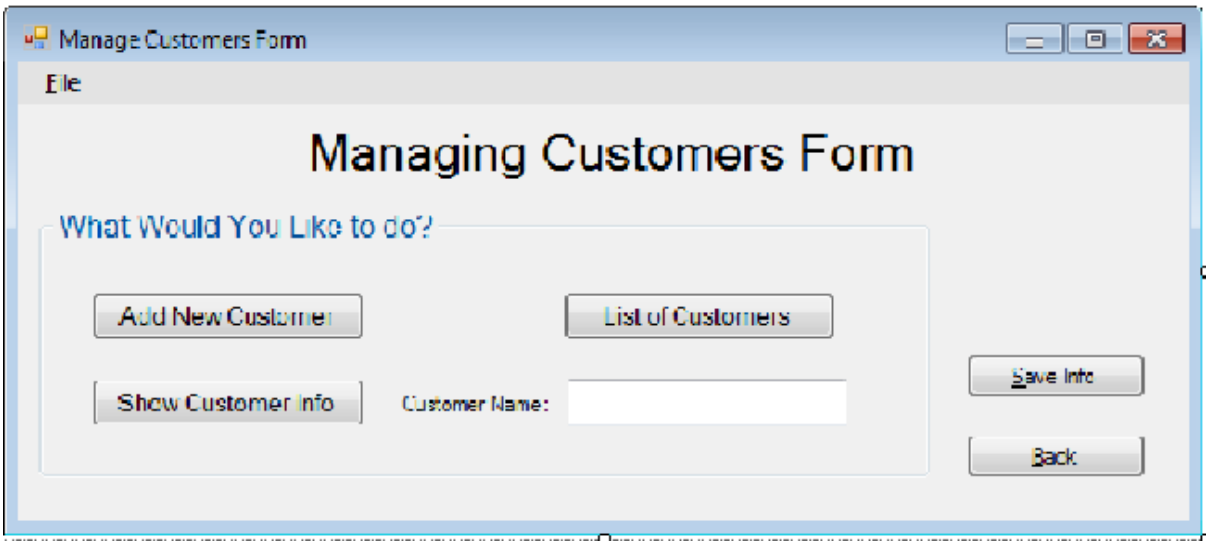
GUI

The following images are screen shots of the program interface based on varied user input. Shown are the cases for both correct and incorrect cases for most of the events.

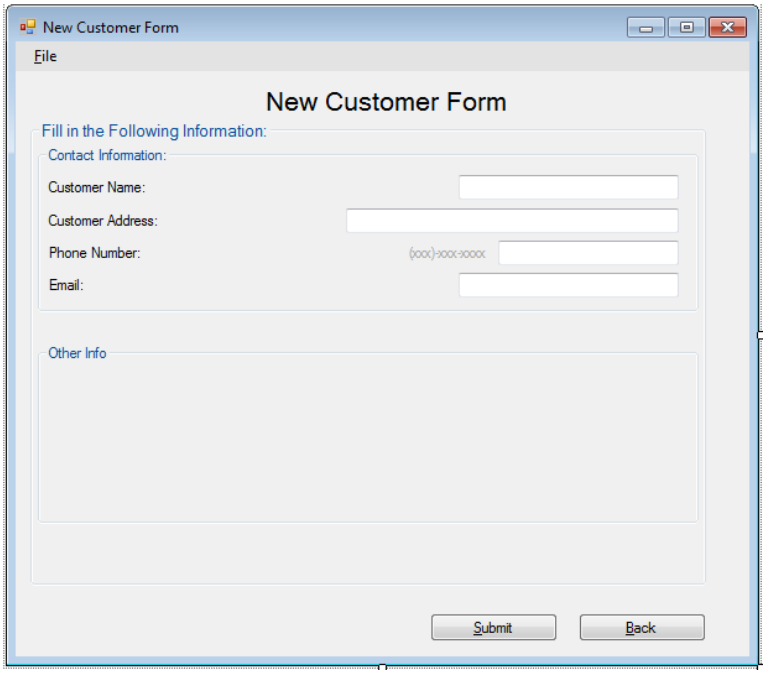
Main Page



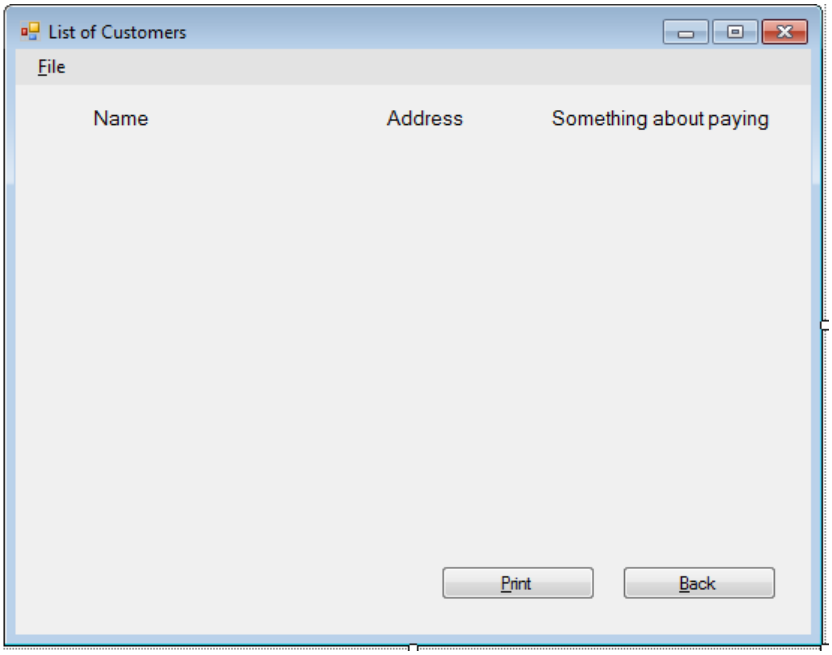
Manage Customer Form



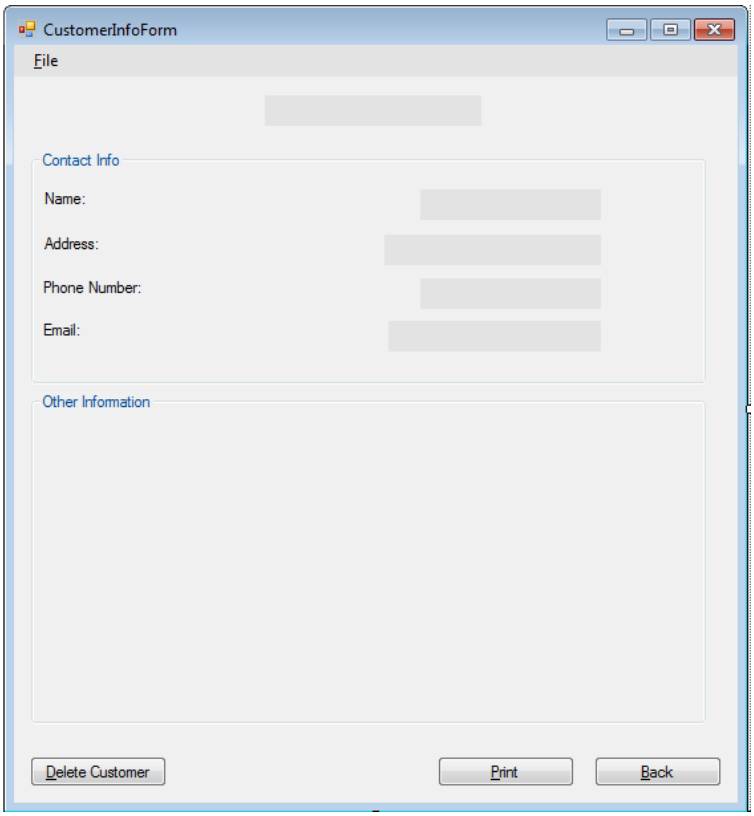
New Customer Form



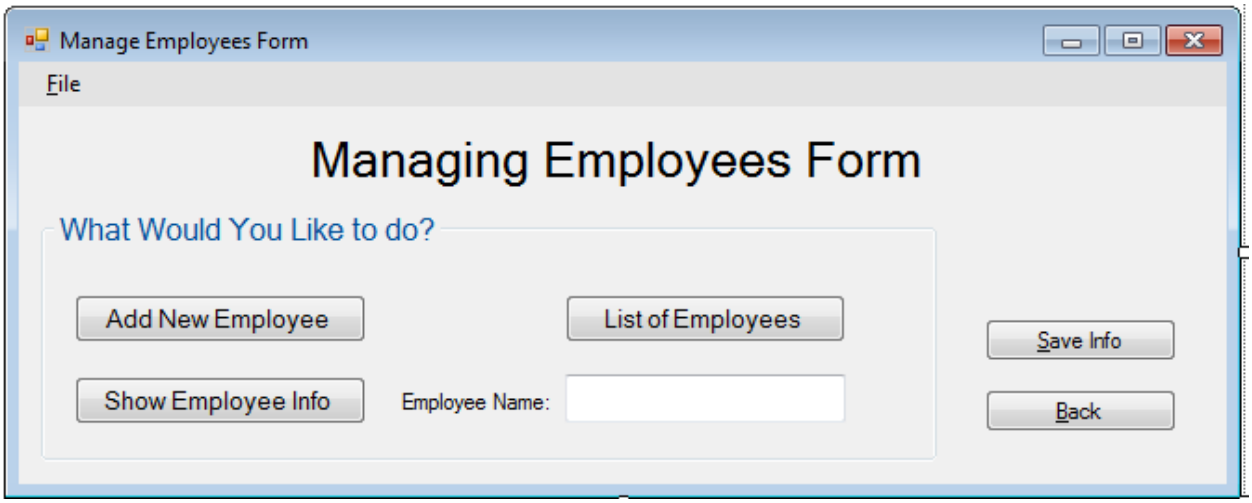
Customer List Form



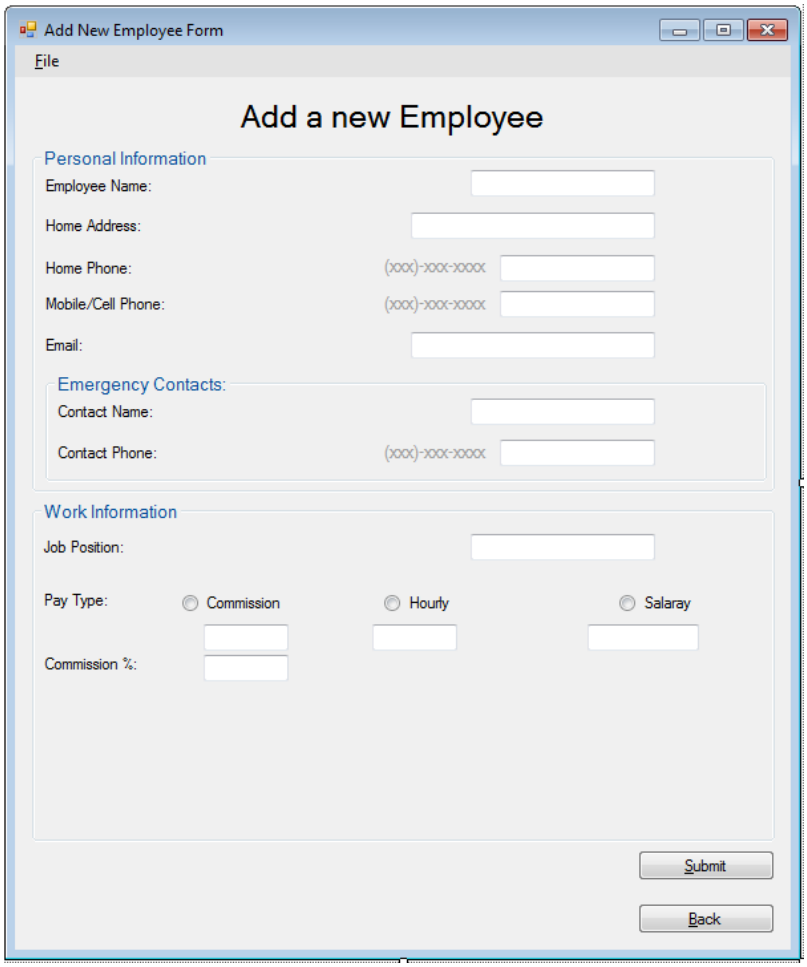
Customer Info Form



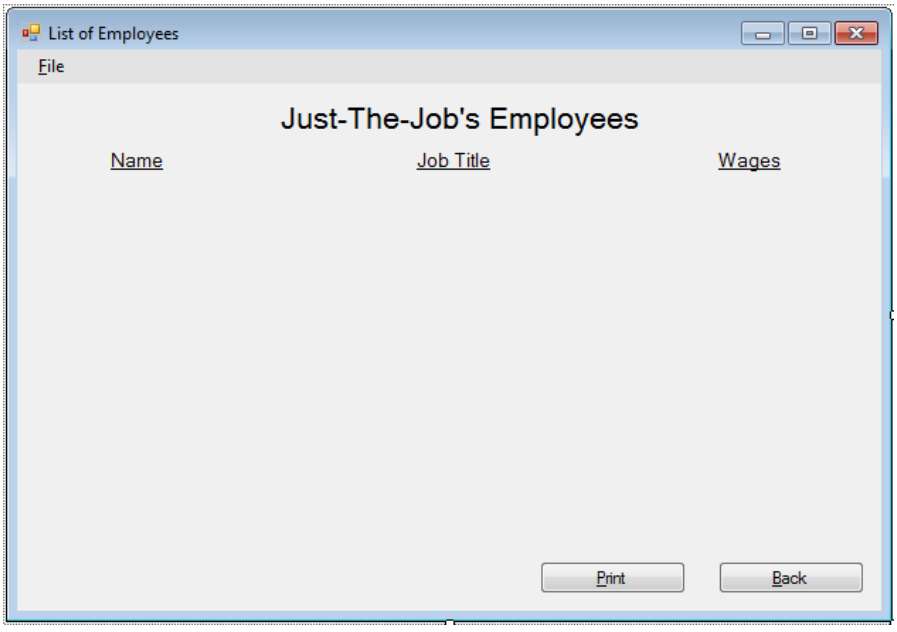
Manage Employee Form



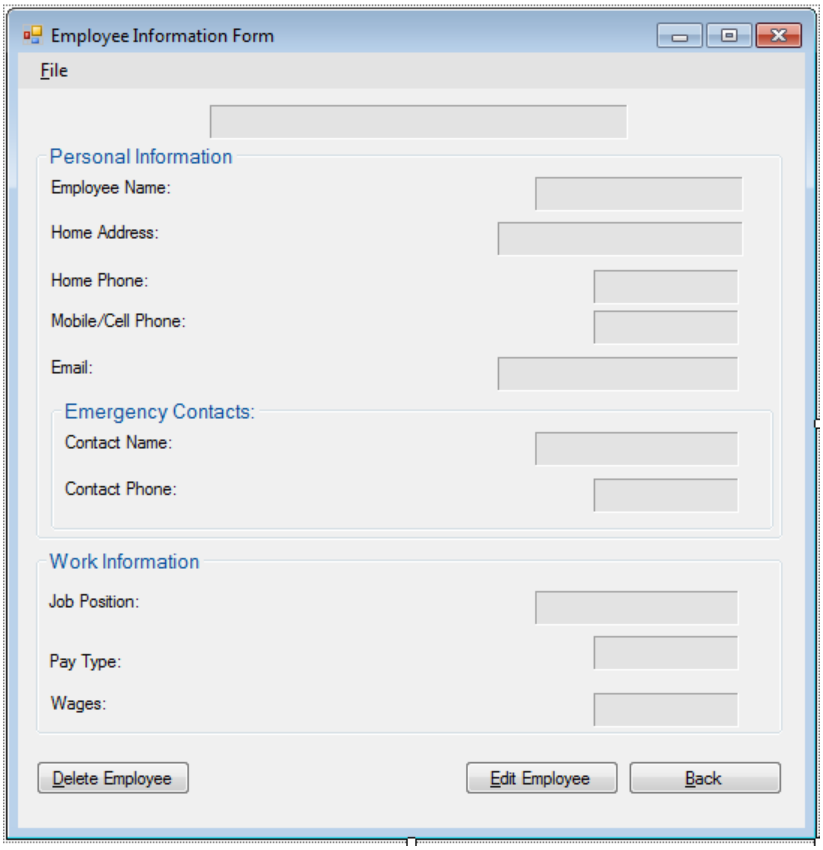
New Employee Form



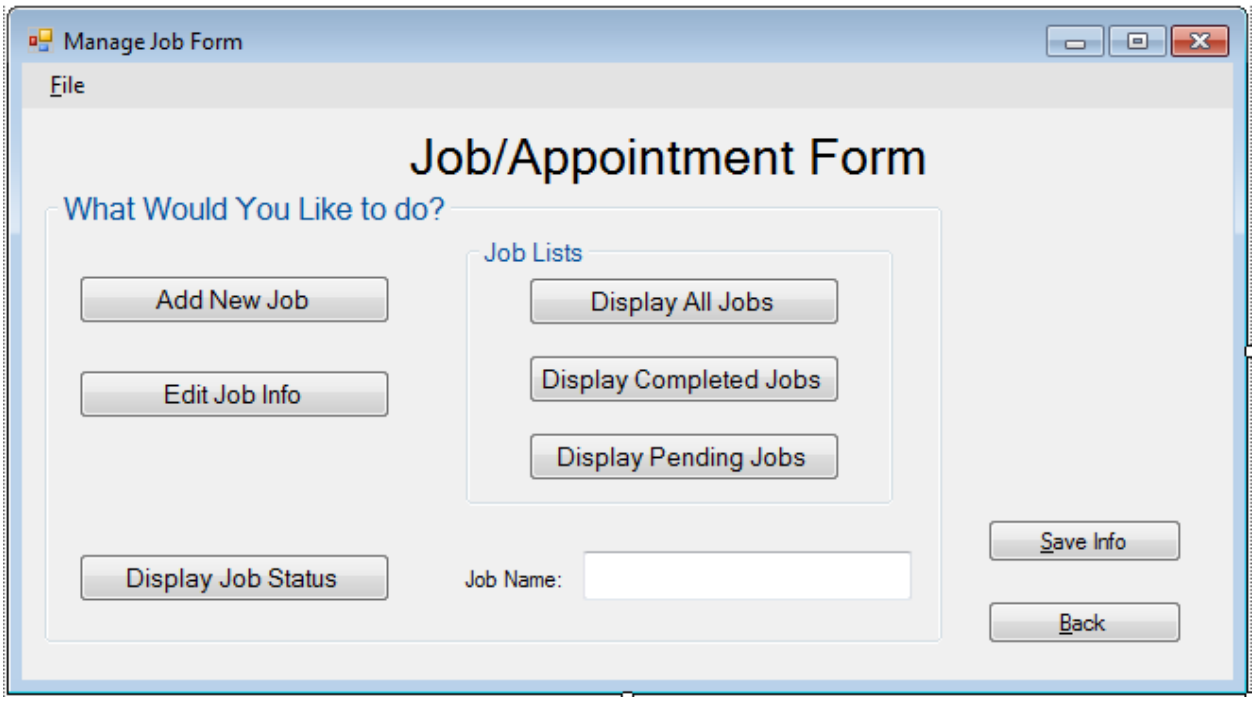
List of Employees Form



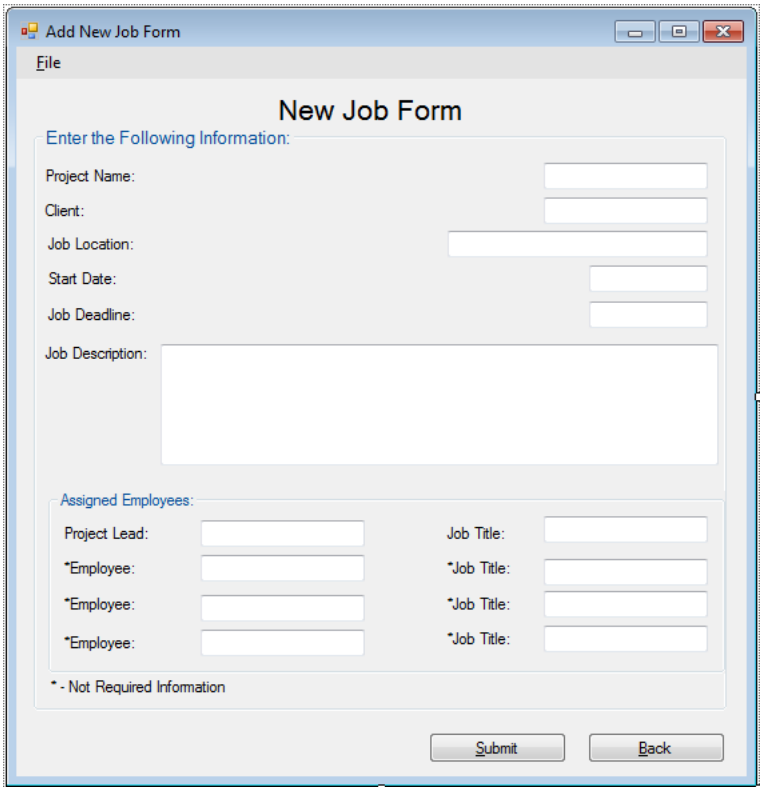
Employee Information Form



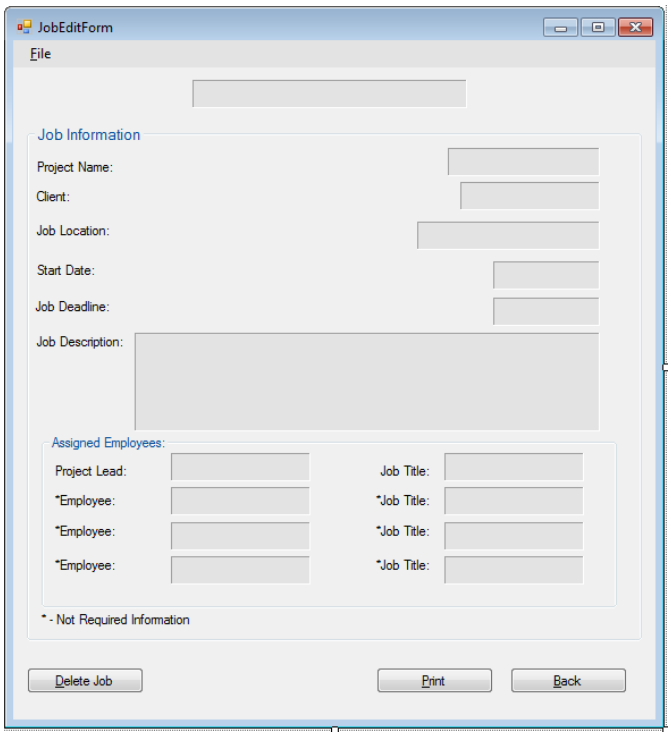
Jobs Form



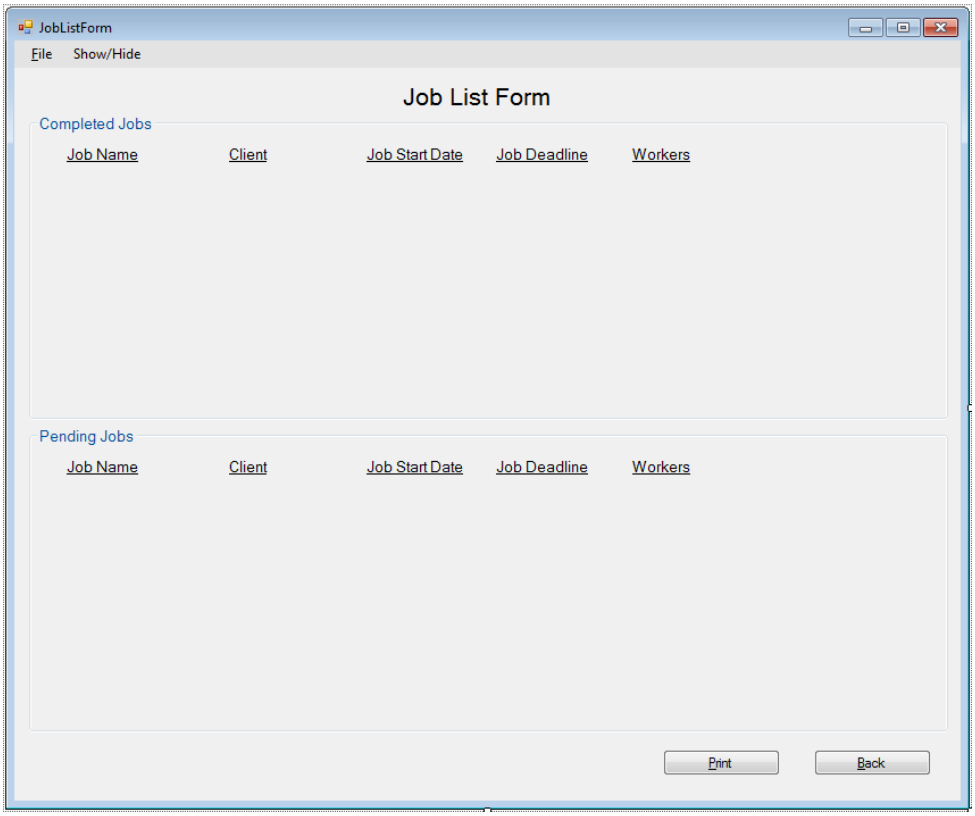
New Job Form



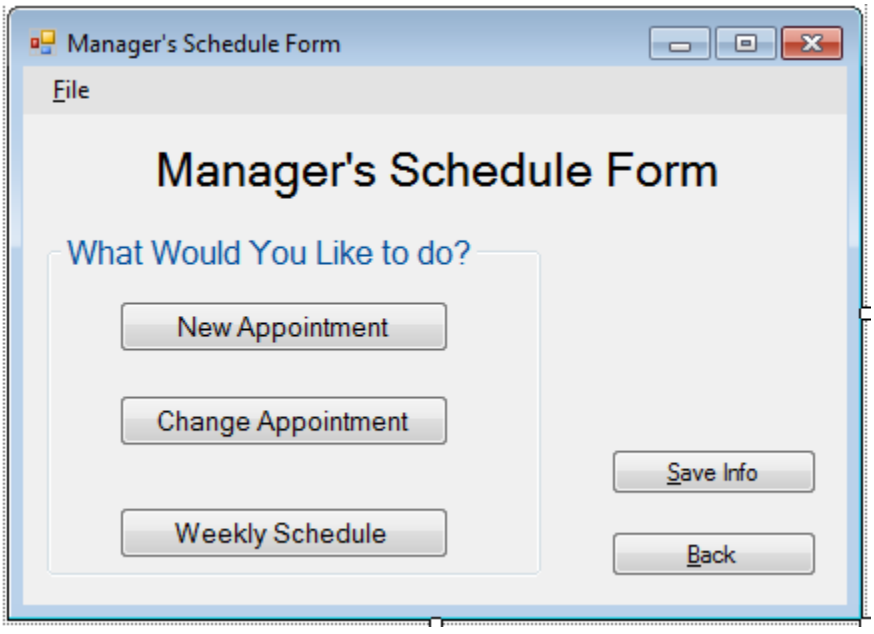
Edit Job Info Form



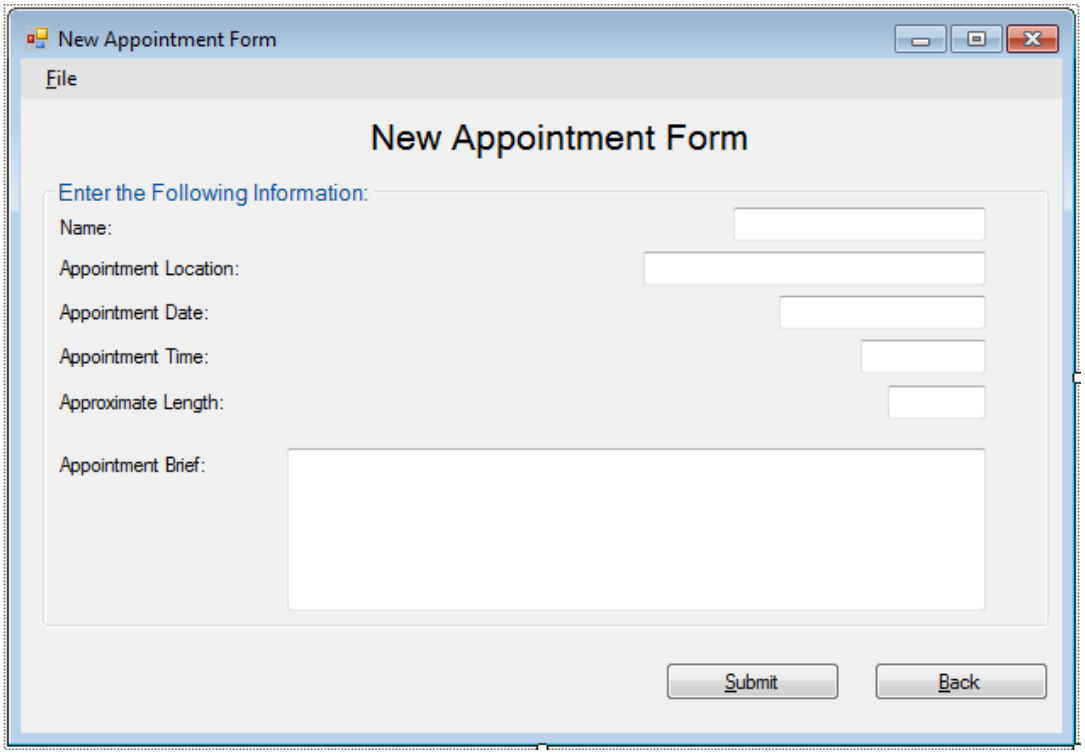
Job List Form



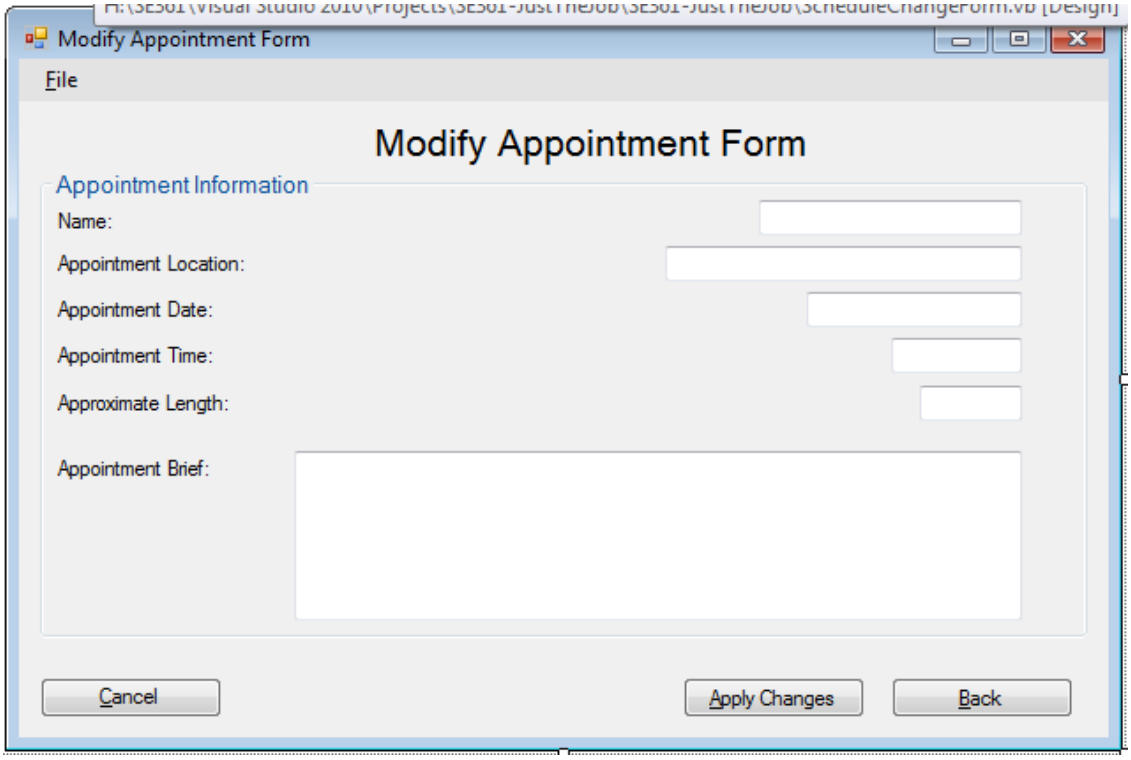
Manage Schedule Form



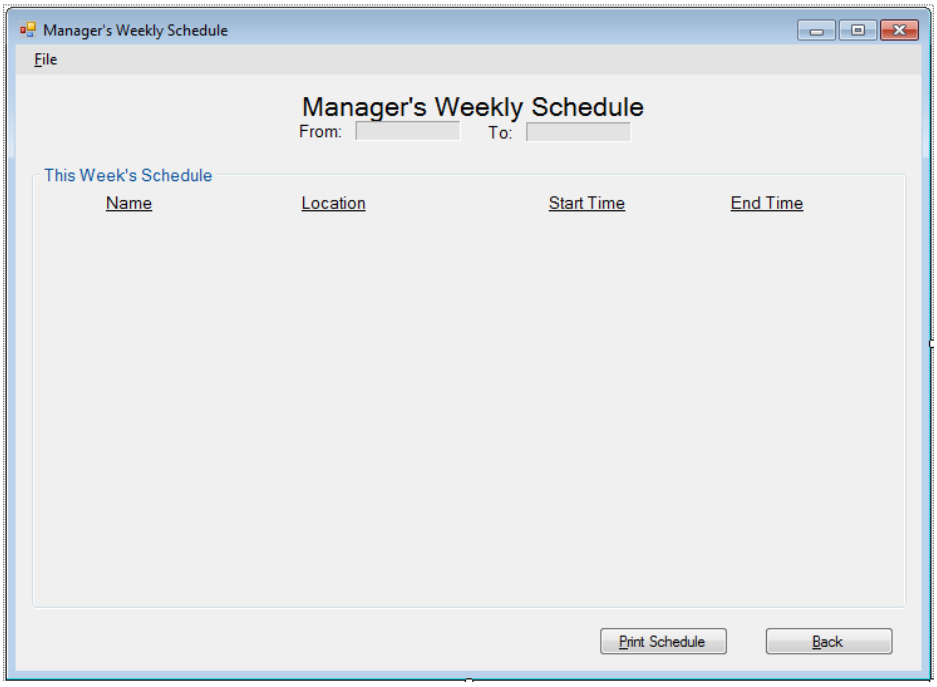
New Appointment Form



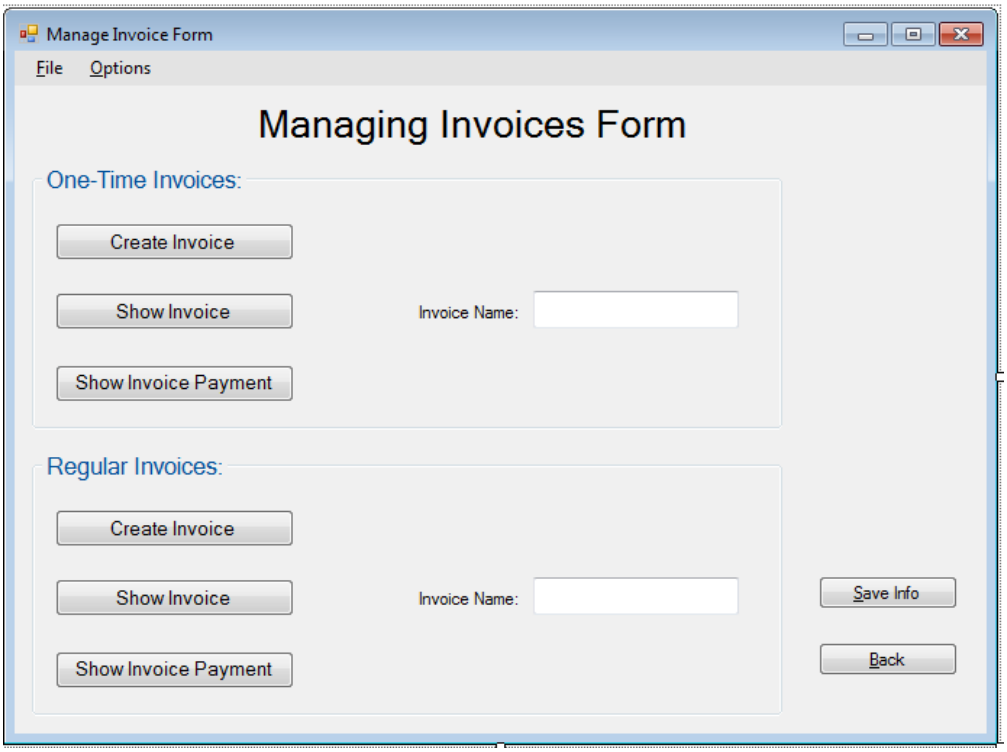
Modify Appointment Form



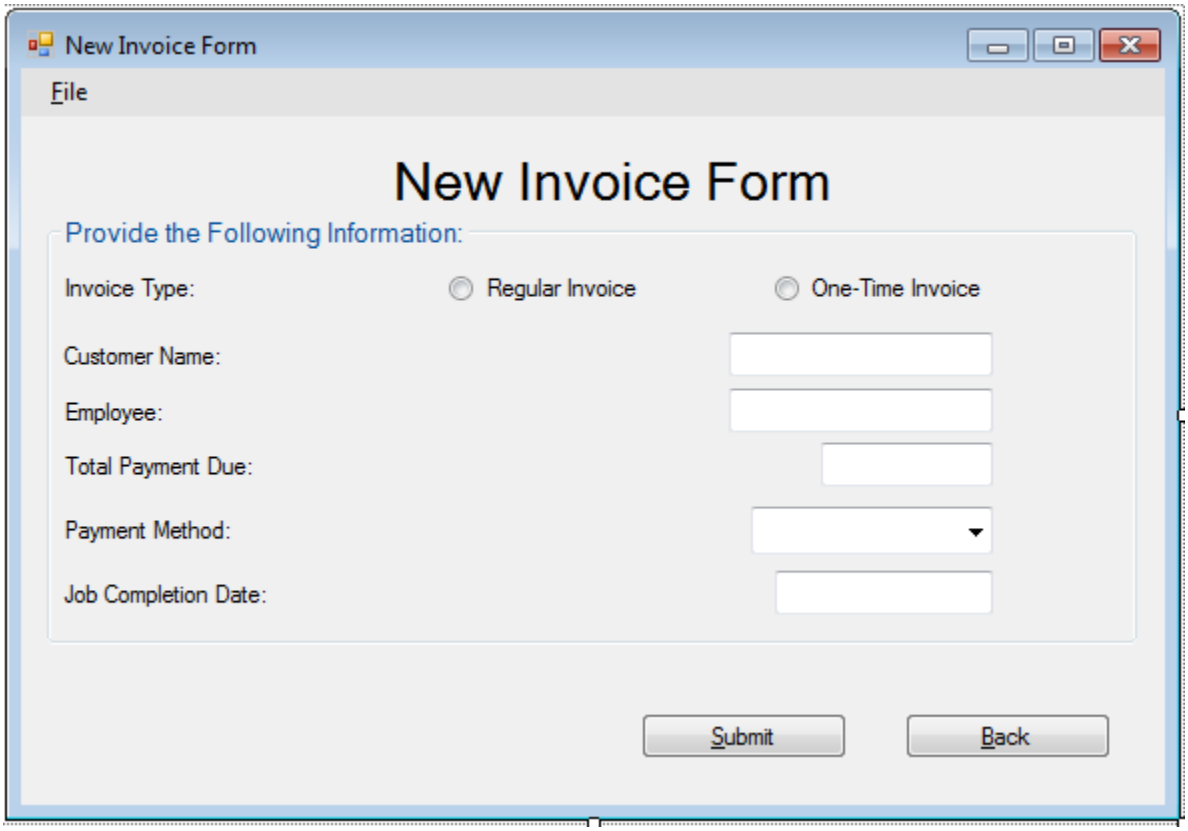
Weekly Schedule Form



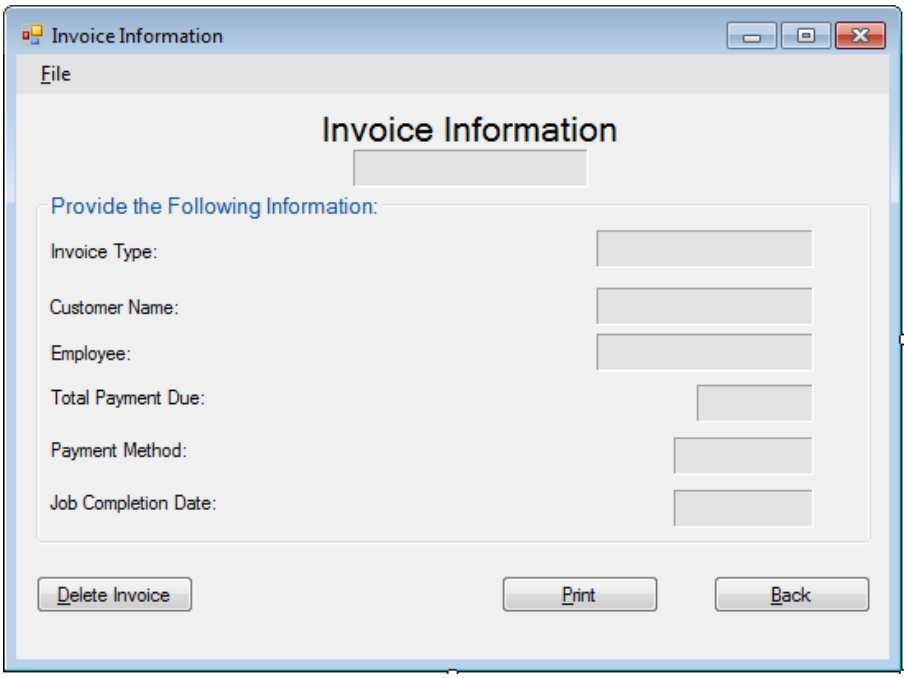
Manage Invoices Form (Will direct both regular & one-Time invoices)



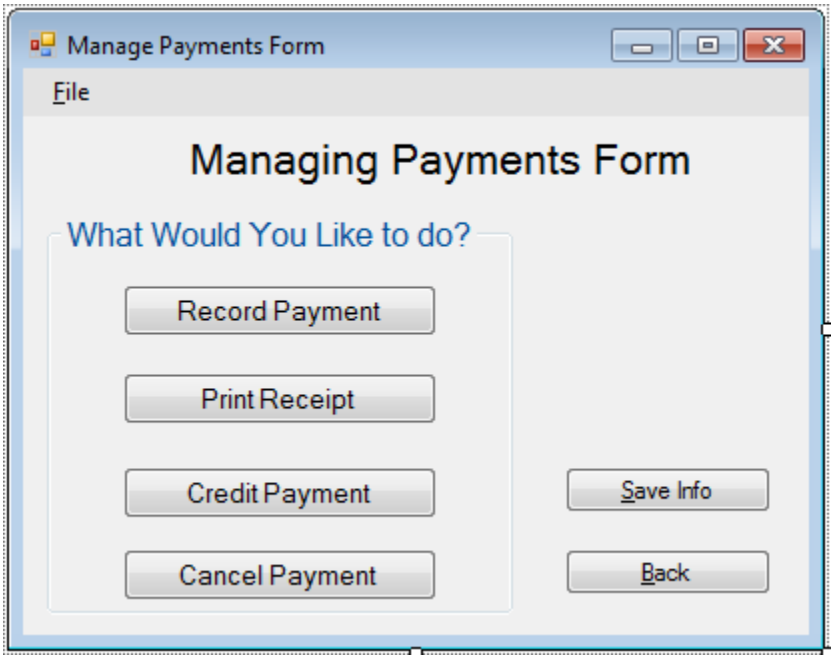
New Invoice Form (Handles Creation of One-Time & Regular Invoices)



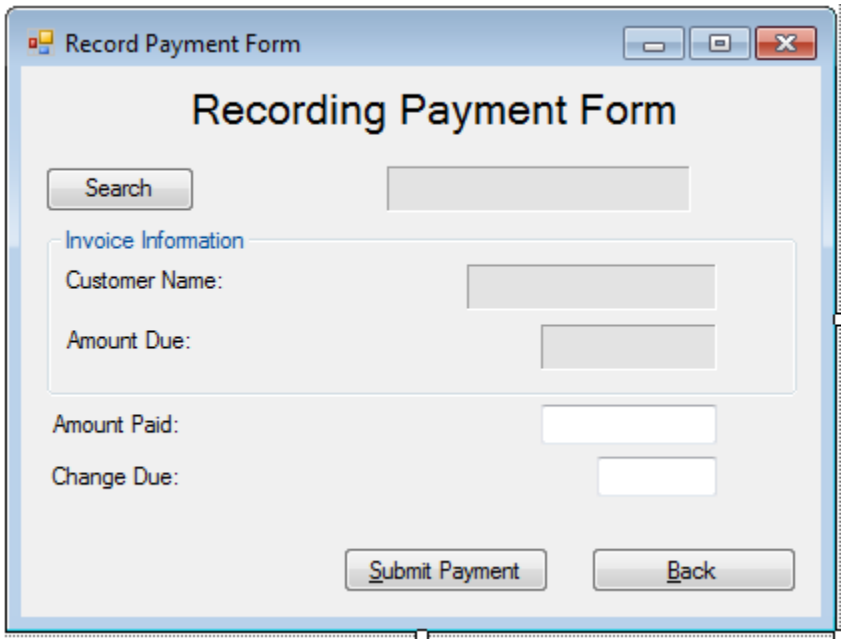
Show Invoice Form (handles both One-Time & Regular Invoices)



Manage Payments Form



Record Payment Form



Print Receipt Form

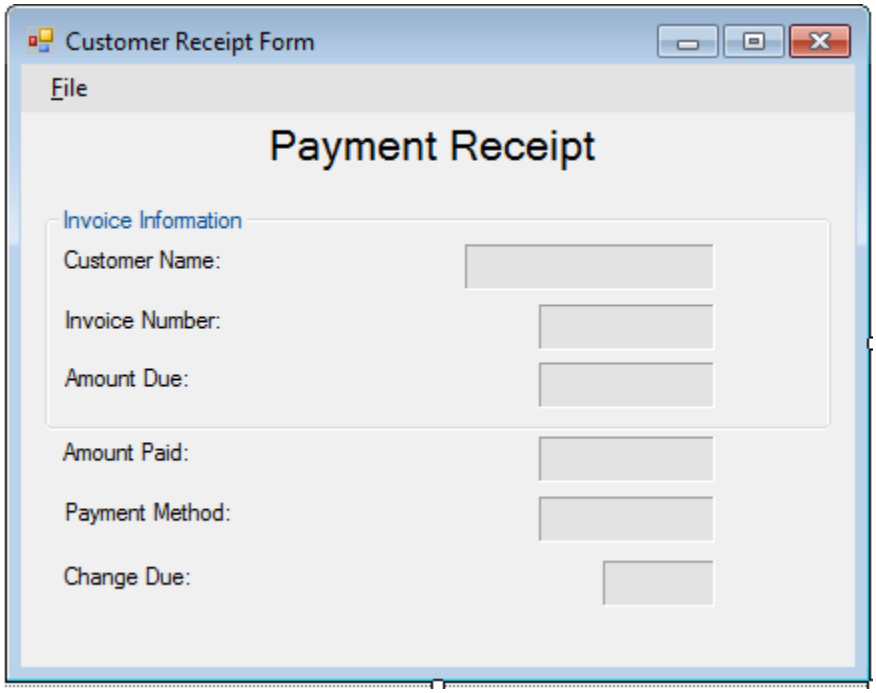


Figure 1.29 GUI

Iteration 1 code

The following contains the all of the code that was implemented for iteration 1. This includes the code from the home page, as well as all the forms that pertain to adding, and displaying a list of all customers, employees, and jobs. The code for the object creating and list managing classes are also included below.

clsCustomer

```
Public Class clsCustomer
    Private strName As String
    Private strAddress As String
    Private strPhone As String
    Private strEmail As String

    Public Sub New()
        strName = ""
        strAddress = ""
        strPhone = ""
        strEmail = ""
    End Sub

    Public Sub New(ByVal name, ByVal address, ByVal phone, ByVal email)
        strName = name
        strAddress = address
        strPhone = phone
        strEmail = email
    End Sub

    Public Property name As String
        Get
            Return strName
        End Get
        Set(ByVal value As String)

            strName = value

        End Set
    End Property

    Public Property address As String
        Get
            Return strAddress
        End Get
        Set(ByVal value As String)

            strAddress = value
        End Set
    End Property

    Public Property phone As String
        Get
            Return strPhone
        End Get
        Set(ByVal value As String)
                strPhone = value
```

```
        End Set
    End Property

    Public Property email As String
        Get
            Return strEmail
        End Get
        Set(ByVal value As String)

            strEmail = value
        End Set
    End Property
End Class
```

clsCustomerListManager

```
Public Class clsCustomerListManager
    Private colCustomers As New Collection

    Public Sub addCustomer(ByVal name As String, ByVal address As String, ByVal phone As
String, ByVal email As String)
        Dim objCustomer As New clsCustomer

        objCustomer.name = name
        objCustomer.address = address
        objCustomer.phone = phone
        objCustomer.email = email

        colCustomers.Add(objCustomer)
    End Sub

    Public Sub viewCustomerName(ByRef guiDisplay As ListBox)
        Dim cust As clsCustomer

        For Each cust In colCustomers
            guiDisplay.Items.Add(cust.name)
        Next
    End Sub

    Public Sub viewCustomerAddress(ByRef guiDisplay As ListBox)
        Dim cust As clsCustomer

        For Each cust In colCustomers
            guiDisplay.Items.Add(cust.address)
        Next
    End Sub

    Public Sub viewCustomerPhone(ByRef guiDisplay As ListBox)
        Dim cust As clsCustomer

        For Each cust In colCustomers
            guiDisplay.Items.Add(cust.phone)
        Next
    End Sub

    Public Sub viewCustomerEmail(ByRef guiDisplay As ListBox)
        Dim cust As clsCustomer
```

```
        For Each cust In colCustomers
            guiDisplay.Items.Add(cust.email)
        Next
    End Sub
End Class
```

clsEmployee

```
Public Class clsEmployee
    Private strName As String
    Private strAddress As String
    Private strHomePhone As String
    Private strCellPhone As String
    Private strEmail As String
    Private strEmergencyName As String
    Private strEmergencyPhone As String
    Private strPosition As String
    Private strPaytype As String
    Private dblWages As Double

    Public Sub New(ByVal name, ByVal address, ByVal homePhone, ByVal cellPhone, ByVal
email,
                   ByVal emergencyName, ByVal emergencyPhone, ByVal position, ByVal
paytype, ByVal wages)
        strName = name
        strAddress = address
        strHomePhone = homePhone
        strCellPhone = cellPhone
        strEmail = email
        strEmergencyName = emergencyName
        strEmergencyPhone = emergencyPhone
        strPosition = position
        strPaytype = paytype
        dblWages = wages

    End Sub

    Sub New()
        ' TODO: Complete member initialization
    End Sub

    Public Property name As String
        Get
            Return strName
        End Get
        Set(ByVal value As String)
            strName = value
        End Set
    End Property

    Public Property address As String
        Get
            Return strAddress
        End Get
        Set(ByVal value As String)
            strAddress = value
        End Set
    End Property
```

```
Public Property homePhone As String
    Get
        Return strHomePhone
    End Get
    Set(ByVal value As String)
        strHomePhone = value
    End Set
End Property

Public Property cellPhone As String
    Get
        Return strCellPhone
    End Get
    Set(ByVal value As String)
        strCellPhone = value
    End Set
End Property

Public Property email As String
    Get
        Return strEmail
    End Get
    Set(ByVal value As String)
        strEmail = value
    End Set
End Property

Public Property emergencyName As String
    Get
        Return strEmergencyName
    End Get
    Set(ByVal value As String)
        strEmergencyName = value
    End Set
End Property

Public Property emergencyPhone As String
    Get
        Return strEmergencyPhone
    End Get
    Set(ByVal value As String)
        strEmergencyPhone = value
    End Set
End Property

Public Property position As String
    Get
        Return strPosition
    End Get
    Set(ByVal value As String)
        strPosition = value
    End Set
End Property

Public Property paytype As String
    Get
        Return strPaytype
```

```
        End Get
        Set(ByVal value As String)
            strPaytype = value
        End Set
    End Property

    Public Property wages As Double
        Get
            Return dblWages
        End Get
        Set(ByVal value As Double)
            dblWages = value
        End Set
    End Property
End Class
```

clsEmployeeListManager

```
Public Class clsEmployeeListManager
    Private colEmployees As New Collection

    Public Sub addEmployee(ByVal name, ByVal address, ByVal homePhone, ByVal cellPhone,
ByVal email,
                           ByVal emergencyName, ByVal emergencyPhone, ByVal position,
ByVal paytype, ByVal wages)
        Dim objEmployee As New clsEmployee

        objEmployee.name = name
        objEmployee.address = address
        objEmployee.homePhone = homePhone
        objEmployee.cellPhone = cellPhone
        objEmployee.email = email
        objEmployee.emergencyName = emergencyName
        objEmployee.emergencyPhone = emergencyPhone
        objEmployee.position = position
        objEmployee.paytype = paytype
        objEmployee.wages = wages

        colEmployees.Add(objEmployee)
    End Sub

    Public Sub viewEmployeeName(ByRef guiDisplay As ListBox)
        Dim empl As clsEmployee

        For Each empl In colEmployees
            guiDisplay.Items.Add(empl.name)
        Next
    End Sub

    Public Sub viewEmployeecellPhone(ByRef guiDisplay As ListBox)
        Dim empl As clsEmployee

        For Each empl In colEmployees
            guiDisplay.Items.Add(empl.cellPhone)
        Next
    End Sub

    Public Sub viewEmployeeposition(ByRef guiDisplay As ListBox)
```

```
        Dim empl As clsEmployee

        For Each empl In colEmployees
            guiDisplay.Items.Add(empl.position)
        Next
    End Sub

    Public Sub viewEmployeewages(ByRef guiDisplay As ListBox)
        Dim empl As clsEmployee

        For Each empl In colEmployees
            guiDisplay.Items.Add(empl.wages)
        Next
    End Sub

End Class
```

clsJob

```
Public Class clsJob
    Private strProjectName As String
    Private strClient As String
    Private strLocation As String
    Private strStartDate As String
    Private strDeadline As String
    Private strDescription As String
    Private strProjectLead As String
    Private strEmployee1 As String = ""
    Private strEmployee2 As String = ""
    Private strEmployee3 As String = ""
    Private strLeadTitle As String
    Private strTitle1 As String = ""
    Private strTitle2 As String = ""
    Private strTitle3 As String = ""



    Public Sub New(ByVal projectName, ByVal client, ByVal location, ByVal startDate,
                   ByVal Deadline, ByVal Description, ByVal projectLead, ByVal employee1,
                   ByVal employee2, ByVal employee3, ByVal leadTitle, ByVal title1,
ByVal title2, ByVal title3)
        strProjectName = projectName
        strClient = client
        strLocation = location
        strStartDate = startDate
        strDeadline = Deadline
        strDescription = Description
        strProjectLead = projectLead
        strEmployee1 = employee1
        strEmployee2 = employee2
        strEmployee3 = employee3
        strLeadTitle = leadTitle
        strTitle1 = title1
        strTitle2 = title2
        strTitle3 = title3
    End Sub

    Sub New()
```

```
        ' Needed this here because I was getting an error in clsJobListManager and visual
studio needed this created. - Eric
    End Sub

    Public Property projectName As String
        Get
            Return strProjectName
        End Get
        Set(ByVal value As String)
            strProjectName = value
        End Set
    End Property

    Public Property client As String
        Get
            Return strClient
        End Get
        Set(ByVal value As String)
            strClient = value
        End Set
    End Property

    Public Property location As String
        Get
            Return strLocation
        End Get
        Set(ByVal value As String)
            strLocation = value
        End Set
    End Property

    Public Property startDate As String
        Get
            Return strStartDate
        End Get
        Set(ByVal value As String)
            strStartDate = value
        End Set
    End Property

    Public Property Deadline As String
        Get
            Return strDeadline
        End Get
        Set(ByVal value As String)
            strDeadline = value
        End Set
    End Property

    Public Property Description As String
        Get
            Return strDescription
        End Get
        Set(ByVal value As String)
            strDescription = value
        End Set
    End Property
```

```
Public Property projectLead As String
    Get
        Return strProjectLead
    End Get
    Set(ByVal value As String)
        strProjectLead = value
    End Set
End Property

Public Property employee1 As String
    Get
        Return strEmployee1
    End Get
    Set(ByVal value As String)
        strEmployee1 = value
    End Set
End Property
Public Property employee2 As String
    Get
        Return strEmployee2
    End Get
    Set(ByVal value As String)
        strEmployee2 = value
    End Set
End Property
Public Property employee3 As String
    Get
        Return strEmployee3
    End Get
    Set(ByVal value As String)
        strEmployee3 = value
    End Set
End Property
Public Property leadTitle As String
    Get
        Return strLeadTitle
    End Get
    Set(ByVal value As String)
        strLeadTitle = value
    End Set
End Property
Public Property title1 As String
    Get
        Return strTitle1
    End Get
    Set(ByVal value As String)
        strTitle1 = value
    End Set
End Property
Public Property title2 As String
    Get
        Return strTitle2
    End Get
    Set(ByVal value As String)
        strTitle2 = value
    End Set
End Property
Public Property title3 As String
```

```
        Get
            Return strTitle3
        End Get
        Set(ByVal value As String)
            strTitle3 = value
        End Set
    End Property
End Class
```

clsJobListManager

```
Public Class clsJobListManager
    Private colJobs As New Collection

    Public Sub addJob(ByVal projectName As String, ByVal client As String, ByVal location
As String, ByVal startDate As String,
                      ByVal Deadline As String, ByVal Description As String, ByVal
projectLead As String, ByVal employee1 As String,
                      ByVal employee2 As String, ByVal employee3 As String, ByVal leadTitle
As String, ByVal title1 As String,
                      ByVal title2 As String, ByVal title3 As String)
        Dim objJob As New clsJob

        objJob.projectName = projectName
        objJob.client = client
        objJob.location = location
        objJob.startDate = startDate
        objJob.Deadline = Deadline
        objJob.Description = Description
        objJob.projectLead = projectLead
        objJob.employee1 = employee1
        objJob.employee2 = employee2
        objJob.employee3 = employee3
        objJob.leadTitle = leadTitle
        objJob.title1 = title1
        objJob.title2 = title2
        objJob.title3 = title3



        colJobs.Add(objJob)
    End Sub

End Class
```

Mainform

```
'USERNAME= "Manager" or "Receptionist"
'PASSWORD = "mpass" or "rpass" respectively

Public Class Mainform
    'Closes Program
    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnExit.Click
        Me.Close()
    End Sub
    'Opens CustomerForm
    Private Sub btnCustomer_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCustomer.Click
```

```
        CustomerForm.Show()
    End Sub
    'Opens ScheduleForm
    Private Sub btnSchedule_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSchedule.Click
        ScheduleForm.Show()
    End Sub
    'Opens EmployeeForm
    Private Sub btnEmployees_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEmployees.Click
        EmployeesForm.Show()
    End Sub
    'Opens InvoiceForm
    Private Sub btnInvoices_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnInvoices.Click
        InvoiceForm.Show()
    End Sub
    'Opens JobForm
    Private Sub btnJobs_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnJobs.Click
        JobForm.Show()
    End Sub
    'Opens PaymentForm
    Private Sub btnPayments_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPayments.Click
        PaymentForm.Show()
    End Sub
    'Closes Program
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub Mainform_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        grpMainForm.Enabled = False
        grpLogin.Focus()
    End Sub

    Private Sub btnLogin_Click(sender As Object, e As EventArgs) Handles btnLogin.Click
        If (txtUsername.Text = "Manager" And txtPassword.Text = "mpass") Then
            grpMainForm.Enabled = True
            grpLogin.Visible = False
        ElseIf (txtUsername.Text = "Receptionist" And txtPassword.Text = "rpass") Then
            grpMainForm.Enabled = True
            grpLogin.Visible = False
            CustomerInfoForm.btnCIDelete.Visible = False
            EmployeeInfo.btnRemove.Visible = False
            JobViewForm.btnJEDelete.Visible = False
            InvoiceInfoForm.btnIIDelete.Visible = False
            PaymentForm.btnPayCredit.Visible = False
            PaymentForm.btnPayCancel.Visible = False
        Else
            MessageBox.Show("Username/Password combination is invalid. Please try again.",
"Invalid Credentials", MessageBoxButtons.OK)
            txtUsername.Clear()
            txtPassword.Clear()
        End If
    End Sub
```

```
    Private Sub txtPassword_KeyDown(sender As Object, e As KeyEventArgs) Handles
txtPassword.KeyDown
        'if user presses enter, it will automatically login without having to press
button

        If (e.KeyCode = Keys.Enter) Then
            If (txtUsername.Text = "Manager" And txtPassword.Text = "mpass") Then
                grpMainForm.Enabled = True
                grpLogin.Visible = False
            ElseIf (txtUsername.Text = "Receptionist" And txtPassword.Text = "rpass")
Then
                grpMainForm.Enabled = True
                grpLogin.Visible = False
                CustomerInfoForm.btnCIDelete.Visible = False
                EmployeeInfo.btnRemove.Visible = False
                JobViewForm.btnJEDelete.Visible = False
                InvoiceInfoForm.btnIIDelete.Visible = False
                PaymentForm.btnPayCredit.Visible = False
                PaymentForm.btnPayCancel.Visible = False
            Else
                MessageBox.Show("Username/Password combination is invalid. Please try
again.", "Invalid Credentials", MessageBoxButtons.OK)
                txtUsername.Clear()
                txtPassword.Clear()
            End If
        End If
    End Sub
End Class
```

CustomerForm

```
Imports System.IO
Public Class CustomerForm

    'Display form containing list of Customers. Inside form give option to print list
    Private Sub btnCustList_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCustList.Click
        CustomerListForm.Show()
    End Sub

    'Open Add New Customer Form
    Private Sub btnCustAdd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCustAdd.Click
        NewCustomerForm.Show()
        NewCustomerForm.txtNCName.Focus()
    End Sub

    'create input box for customer name to search for
    Private Sub btnCustShow_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCustShow.Click
        Dim strCustomerName As String

        Dim objCustomerFile As New clsFileManager
        Dim customerList As New Collection
        Dim temp As String          'temporary string
        Dim tempName As String      'Hold the testing customer name for search function
        objCustomerFile.LoadCustomer(customerList)
```

```
        strCustomerName = InputBox("Enter the Customer Name you want to search for.",
"Customer Name")

            For Each cust In customerList
                temp = cust
                temp = temp.Substring(6)                    'Removes beginning <name>
tags
                tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempName.ToLower().Trim = strCustomerName.ToLower().Trim) Then
                'Set contents
                CustomerInfoForm.lblCIHeading.Text = tempName
                CustomerInfoForm.lblCIName.Text = tempName

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(9)

                CustomerInfoForm.lblCIAddress.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(7)

                CustomerInfoForm.lblCIPhone.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(7)

                CustomerInfoForm.lblCIEmail.Text = temp.Substring(0)
                CustomerInfoForm.Show()
                Exit Sub
            End If
            Next
            MessageBox.Show("There is no customer by that name in the database.",
"Customer Not Found", MessageBoxButtons.OK, MessageBoxIcon.Error)

    End Sub

    'Closes Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub btnCustBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCustBack.Click
        Me.Close()
    End Sub
End Class
```

NewCustomerForm

```
Public Class NewCustomerForm
    Dim customerList As New clsCustomerListManager
    Dim objFileManager As New clsFileManager
    'Closes NewCustomerForm
    Private Sub btnNCBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNCBack.Click
```

```
        Me.Close()
    End Sub
    'Closes Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    'Submits customer into database
    Private Sub btnNCSubmit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNCSubmit.Click
        Dim phoneCorrect As Boolean = False
        Dim nameCorrect As Boolean = False
        Dim fieldsFilled As Boolean = False
        Dim phoneNum As String = txtNCPhone.Text

        Dim objCheck As New clsCheck

        phoneNum = objCheck.editPhone(phoneNum)
        phoneCorrect = objCheck.checkPhone(phoneNum)
        nameCorrect = objCheck.checkName(txtNCName.Text)
        fieldsFilled = checkFieldsFilled()

        If phoneCorrect = True And nameCorrect = True And fieldsFilled = True Then
            customerList.addCustomer(txtNCName.Text, txtNCAddress.Text, phoneNum,
txtNCEmail.Text)

            objFileManager.SaveCustomer(txtNCName.Text, txtNCAddress.Text, phoneNum,
txtNCEmail.Text)

            MessageBox.Show("A new customer has been added to the database.",
"Confirmation", MessageBoxButtons.OK)
            txtNCName.Clear()
            txtNCAddress.Clear()
            txtNCPhone.Clear()
            txtNCEmail.Clear()
        End If
    End Sub


    'Checks to see if all the fields are filled out
    Private Function checkFieldsFilled() As Boolean

        If txtNCName.Text.Length = 0 Or txtNCAddress.Text.Length = 0 Or
txtNCPhone.Text.Length = 0 Or txtNCEmail.Text.Length = 0 Then
            MessageBox.Show("Please fill in all fields", "Missing information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function


End Class
```

CustomerListForm

```
'Contains list of Customers with info(don't know what). Don't know how we will implement
it
Public Class CustomerListForm
    Dim objFileManager As New clsFileManager
    Dim customerList As New Collection
    'Closes ListForm
    Private Sub btnCLBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCLBack.Click
        Me.Close()
    End Sub
    'Closes ListForm
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub CustomerListForm_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        objFileManager.LoadCustomer(customerList)

        Dim i As Integer
        Dim counter As Integer
        Dim temp As String
        i = 1
        counter = customerList.Count
        While counter > 0
            temp = customerList.Item(i)
            temp = temp.Substring(6)
            lstName.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(9)
            lstAddress.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(7)
            lstPhone.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(7)
            lstEmail.Items.Add(temp.Substring(0))

            counter = counter - 1
            i = i + 1
        End While

    End Sub
End Class
```

EmployeesForm

```
Public Class EmployeesForm
    'Opens Add new Employee Form
    Private Sub btnNewEmployee_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNewEmployee.Click
        NewEmployee.Show()
        NewEmployee.txtNEName.Focus()
    End Sub
    'Displays a form containing a list of employees
```

```
    Private Sub btnEmpList_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEmpList.Click
        EmployeeListForm.Show()
    End Sub
    'create input box for employee name to search for
    Private Sub btnEmpShow_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEmpShow.Click
        Dim objEmployeeFile As New clsFileManager
        Dim employeeList As New Collection
        Dim temp As String          'temporary string
        Dim tempName As String      'Hold the testing customer name for search function
        Dim strEmployeeName As String
        objEmployeeFile.LoadEmployee(employeeList)
        strEmployeeName = InputBox("Enter the Employee Name you want to search for.",
"EmployeeName")
            For Each empl In employeeList
                temp = empl
                temp = temp.Substring(6)                    'Removes beginning <name>
tags
                tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempName.ToLower().Trim = strEmployeeName.ToLower().Trim) Then
                'Set contents
                EmployeeInfo.lblEIHeading.Text = tempName
                EmployeeInfo.lblEIName.Text = tempName

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(9)

                EmployeeInfo.lblEIAddress.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)

                EmployeeInfo.lblEIHome.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(8)

                EmployeeInfo.lblEIMobile.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(7)

                EmployeeInfo.lblEIEmail.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(8)

                EmployeeInfo.lblEINameEM.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(7)

                EmployeeInfo.lblEIPhoneEM.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)
```

```
                EmployeeInfo.lblEIPosition.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(9)

                EmployeeInfo.lblEIPayType.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(5)

                EmployeeInfo.lblEIWages.Text = temp

                EmployeeInfo.Show()
                Exit Sub
            End If
            Next
            MessageBox.Show("There is no employee by that name in the database.",
"Employee Not Found", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Sub
    'Opens form containing employee information. This form will contain option for
editing & printing info, as well as giving option to delete
    'the employee if the user is a Manager


    'Closes EmployeeForm
    Private Sub btnEmpBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEmpBack.Click
        Me.Close()
    End Sub

    'Closes Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

End Class
```

NewEmployee

```
'Author: Devin Edmundowicz
Public Class NewEmployee
    Dim employeeList As New clsEmployeeListManager
    Dim objFileManager As New clsFileManager
    'Closes New Employee Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub
    'set lblNEPay to "Hourly rate:" & set txtNECommWage, lblCommission, and
txtNECommission to visible
    Private Sub radNECommission_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radNECommission.CheckedChanged
        If radNECommission.Checked = True Then
            txtNECommWages.Visible = True
            lblNEPay.Text = "Commission:"
            lblCommWages.Visible = True
```

```
            lblSalaryWages.Visible = False
            lblHourlyWages.Visible = False
            txtNEHourly.Visible = False
            txtNESalary.Visible = False
        End If
    End Sub
    'set lblNEPay to "Hourly rate:" & set txtNEHourly to visible
    Private Sub radNEHourly_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radNEHourly.CheckedChanged
        If radNEHourly.Checked = True Then
            txtNECommWages.Visible = False
            txtNEHourly.Visible = True
            lblHourlyWages.Visible = True
            lblCommWages.Visible = False
            lblSalaryWages.Visible = False
            txtNEHourly.Focus()
            txtNESalary.Visible = False
            lblNEPay.Text = "Hourly rate:"
        End If
    End Sub

    Private Sub radNESalary_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radNESalary.CheckedChanged
        If radNESalary.Checked = True Then
            txtNECommWages.Visible = False
            lblCommWages.Visible = False
            lblHourlyWages.Visible = False
            lblSalaryWages.Visible = True
            txtNEHourly.Visible = False
            txtNESalary.Visible = True
            txtNESalary.Focus()
            lblNEPay.Text = "Salary:"
        End If

    End Sub

    Private Sub btnNEBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNEBack.Click
        Me.Close()
    End Sub

    Private Sub btnNESubmit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNESubmit.Click
        'we also need to calculate commission, as it's using two data fields
        Dim payType As String = ""
        Dim pay As String = ""
        Dim employeeNameCorrect As Boolean = False 'Employee Name
        Dim homePhoneCorrect As Boolean = False 'Home Phone
        Dim cellPhoneCorrect As Boolean = False 'Cell Phone
        Dim emPhoneCorrect As Boolean = False 'Emergency Contact Phone Number
        Dim emNameCorrect As Boolean = False 'Emergency Contact Name
        Dim fieldsFilled As Boolean = False
        'Phone Numbers
        Dim emPhone As String
        Dim homePhone As String
        Dim cellPhone As String

        Dim objCheck As New clsCheck
```

```
        employeeNameCorrect = objCheck.checkName(txtNEName.Text)
        emNameCorrect = objCheck.checkName(txtNENameEM.Text)

        emPhone = objCheck.editPhone(txtNEPhoneEM.Text)
        emPhoneCorrect = objCheck.checkPhone(emPhone)
        homePhone = objCheck.editPhone(txtNEPhone.Text)
        homePhoneCorrect = objCheck.checkPhone(homePhone)
        cellPhone = objCheck.editPhone(txtNECell.Text)
        cellPhoneCorrect = objCheck.checkPhone(cellPhone)
        fieldsFilled = checkFieldsFilled()

        If employeeNameCorrect = True And homePhoneCorrect = True And cellPhoneCorrect =
True And emPhoneCorrect = True And
            emNameCorrect = True And fieldsFilled = True Then
            If radNEHourly.Checked = True Then
                employeeList.addEmployee(txtNEName.Text, txtNEAddress.Text, homePhone,
cellPhone, txtNEEmail.Text, txtNENameEM.Text,
                                         emPhone, txtNEPosition.Text, "Hourly",
txtNEHourly.Text)
                payType = "Hourly"
                pay = txtNEHourly.Text
            ElseIf radNECommission.Checked = True Then
                employeeList.addEmployee(txtNEName.Text, txtNEAddress.Text, homePhone,
cellPhone, txtNEEmail.Text, txtNENameEM.Text,
                                        emPhone, txtNEPosition.Text, "Commission",
txtNECommWages.Text)
                payType = "Commission"


            ElseIf radNESalary.Checked = True Then
                employeeList.addEmployee(txtNEName.Text, txtNEAddress.Text, homePhone,
cellPhone, txtNEEmail.Text, txtNENameEM.Text,
                                        emPhone, txtNEPosition.Text, "Salary",
txtNESalary.Text)
                payType = "Salary"
                pay = txtNESalary.Text
            End If

            objFileManager.SaveEmployee(txtNEName.Text, txtNEAddress.Text, homePhone,
cellPhone, txtNEEmail.Text, txtNENameEM.Text,
                                         emPhone, txtNEPosition.Text, payType, "$" + pay)

            MessageBox.Show("A new employee has been added to the database.",
"Confirmation", MessageBoxButtons.OK)

            'Clear Fields
            txtNEAddress.Clear()
            txtNECommWages.Clear()
            txtNEEmail.Clear()
            txtNEPhone.Clear()
            txtNEHourly.Clear()
            txtNECell.Clear()
            txtNEName.Clear()
            txtNENameEM.Clear()
            txtNEPhoneEM.Clear()
            txtNEPosition.Clear()
            txtNESalary.Clear()
```

```
            lblCommWages.Hide()
            lblSalaryWages.Hide()
            lblHourlyWages.Hide()
            radNECommission.Checked = False
            radNEHourly.Checked = False
            radNESalary.Checked = False

            txtNECommWages.Visible = False
            txtNEHourly.Visible = False
            txtNESalary.Visible = False
            lblNEPay.Text = ""
        End If
    End Sub
    Private Function checkFieldsFilled() As Boolean
        If txtNEAddress.Text = "" Or txtNEName.Text = "" Or txtNEEmail.Text = "" Or
txtNEPhone.Text = "" Or txtNECell.Text = "" Or txtNECell.Text = "" Or txtNENameEM.Text =
"" Or txtNEPhoneEM.Text = "" Or txtNEPosition.Text = "" Then
            MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False

        ElseIf radNECommission.Checked = True Then
            If txtNECommWages.Text = "" Then
                MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                Return False
            End If
        ElseIf radNEHourly.Checked = True Then
            If txtNEHourly.Text = "" Then
                MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                Return False
            End If
        ElseIf radNESalary.Checked = True Then
            If txtNESalary.Text = "" Then
                MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                Return False
            End If
        ElseIf radNECommission.Checked = False And radNEHourly.Checked = False And
radNESalary.Checked = False Then
            MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function
End Class
```

EmployeeListForm

```
Public Class EmployeeListForm
    Dim objFileManager As New clsFileManager
    Dim employeeList As New Collection
    'Close Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
```

```
    End Sub

    'Close Form
    Private Sub btnNLBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNLBack.Click
        Me.Close()
    End Sub

    Private Sub EmployeeListForm_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        objFileManager.LoadEmployee(employeeList)

        Dim i As Integer
        Dim counter As Integer
        Dim temp As String
        i = 1
        counter = employeeList.Count
        While counter > 0
            temp = employeeList.Item(i)
            'pulls name
            temp = temp.Substring(6)
            lstName.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'trims unnecessary info
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(9)
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)
            temp = temp.Substring(temp.IndexOf("<"))
            'pulls phone
            temp = temp.Substring(8)
            lstPhone.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'trims unnecessary info
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(7)
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(8)
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(7)
            temp = temp.Substring(temp.IndexOf("<"))
            'pulls position
            temp = temp.Substring(10)
            lstPosition.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'trims unnecessary info
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(9)
            temp = temp.Substring(temp.IndexOf("<"))
            'pulls wages
            temp = temp.Substring(5)
            lstWages.Items.Add(temp.Substring(0))

            counter = counter - 1
            i = i + 1
        End While
    End Sub
End Class
```

JobForm

```
Public Class JobForm
    'Opens add new Job Form
    Private Sub btnNewJob_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNewJob.Click
        NewJobForm.Show()
        NewJobForm.txtNJName.Focus()
    End Sub
    'Opens form containing list of all jobs
    Private Sub btnJobAll_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnJobAll.Click
        JobListForm.Show()
        JobListForm.gbJLCompleted.Visible = True
        JobListForm.gbJLPending.Visible = True
    End Sub

    'create input box to search for Job
    Private Sub btnJobStatus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnJobStatus.Click
        Dim objJobFile As New clsFileManager
        Dim jobPendingList As New Collection
        Dim jobCompleteList As New Collection
        Dim temp As String          'temporary string
        Dim tempName As String      'Hold the testing customer name for search function
        Dim strJobName As String

        objJobFile.LoadJob(jobPendingList, jobCompleteList)

        strJobName = InputBox("Enter the Job Name you want to search for.", "Job Name")

            For Each job In jobPendingList
                temp = job
                temp = temp.Substring(6)                    'Removes beginning <name>
tags
                tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempName.ToLower().Trim = strJobName.ToLower().Trim) Then
                'Set Contents
                JobViewForm.lblHeading.Text = tempName
                JobViewForm.lblName.Text = tempName

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(8)
                JobViewForm.lblClient.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)
                JobViewForm.lblLocation.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(7)
                JobViewForm.lblStart.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)
                JobViewForm.lblDeadline.Text = temp.Substring(0, temp.IndexOf("<"))
```

```
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(13)
            JobViewForm.lblDescription.Text = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)
            JobViewForm.lblLeadName.Text = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(11)
            JobViewForm.lblEmployee1.Text = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(11)
            JobViewForm.lblEmployee2.Text = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(11)
            JobViewForm.lblEmployee3.Text = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(11)
            JobViewForm.lblLeadTitle.Text = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(8)
            JobViewForm.lblTitle1.Text = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(8)
            JobViewForm.lblTitle2.Text = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(8)
            JobViewForm.lblTitle3.Text = temp.Substring(0)

            JobViewForm.lblStatus.Text = "Pending"

            'open form
            JobViewForm.Show()
            Exit Sub                'end sub, we found the match
        End If
    Next

    For Each job In jobCompleteList
        temp = job
        temp = temp.Substring(6)                    'Removes beginning <name> tags
        tempName = temp.Substring(0, temp.IndexOf("<"))

        If (tempName.ToLower().Trim = strJobName.ToLower().Trim) Then
            'Set Contents
            JobViewForm.lblHeading.Text = tempName
            JobViewForm.lblName.Text = tempName

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(8)
            JobViewForm.lblClient.Text = temp.Substring(0, temp.IndexOf("<"))
```

```
                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)
                JobViewForm.lblLocation.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(7)
                JobViewForm.lblStart.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)
                JobViewForm.lblDeadline.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(13)
                JobViewForm.lblDescription.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)
                JobViewForm.lblLeadName.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(11)
                JobViewForm.lblEmployee1.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(11)
                JobViewForm.lblEmployee2.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(11)
                JobViewForm.lblEmployee3.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(11)
                JobViewForm.lblLeadTitle.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(8)
                JobViewForm.lblTitle1.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(8)
                JobViewForm.lblTitle2.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(8)
                JobViewForm.lblTitle3.Text = temp.Substring(0)

                JobViewForm.lblStatus.Text = "Completed"

                'open form
                JobViewForm.Show()
                Exit Sub                'end sub, we found the match
            End If
        Next
            MessageBox.Show("There is no job by that name in the database.", "Job Not
Found", MessageBoxButtons.OK, MessageBoxIcon.Error)
```

```
    End Sub

    'Closes JobForm
    Private Sub btnJobBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnJobBack.Click
        Me.Close()
    End Sub

    'Closes Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub FileToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
FileToolStripMenuItem.Click

    End Sub
End Class
```

NewJobForm

```
Public Class NewJobForm
    Dim jobList As New clsJobListManager
    Dim objFileManager As New clsFileManager

    'Closes Form
    Private Sub btnNJBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNJBack.Click
        Me.Close()
    End Sub
    'Closes Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub
    'Add the information to the database
    Private Sub btnNJSubmit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNJSubmit.Click
        Dim objCheck As New clsCheck
        Dim dateStart As String
        Dim dateEnd As String

        Dim dateStartCorrect As Boolean = False
        Dim dateEndCorrect As Boolean = False
        Dim fieldsFilled As Boolean = False

        dateStart = objCheck.editDate(txtNJStart.Text)
        dateStartCorrect = objCheck.checkDate(dateStart)
        dateEnd = objCheck.editDate(txtNJDeadline.Text)
        dateEndCorrect = objCheck.checkDate(dateEnd)
        fieldsFilled = checkFieldsFilled()



        If dateStartCorrect = True And dateEndCorrect = True And fieldsFilled = True Then
            jobList.addJob(txtNJName.Text, txtNJClient.Text, txtNJLocation.Text,
dateStart, dateEnd, txtNJDescription.Text,
```

```
                            txtNJLead.Text, txtNJEmployee1.Text, txtNJEmployee2.Text,
txtNJEmployee3.Text, txtNJLeadTitle.Text, txtNJTitle1.Text,
                            txtNJTitle2.Text, txtNJTitle3.Text)

            objFileManager.SaveJob(txtNJName.Text, txtNJClient.Text, txtNJLocation.Text,
dateStart, dateEnd, txtNJDescription.Text,
                                   txtNJLead.Text, txtNJEmployee1.Text,
txtNJEmployee2.Text, txtNJEmployee3.Text, txtNJLeadTitle.Text, txtNJTitle1.Text,
                                   txtNJTitle2.Text, txtNJTitle3.Text)

            MessageBox.Show("A new job has been added to the database.", "Confirmation",
MessageBoxButtons.OK)
            txtNJClient.Clear()
            txtNJDeadline.Clear()
            txtNJDescription.Clear()
            txtNJEmployee1.Clear()
            txtNJEmployee2.Clear()
            txtNJEmployee3.Clear()
            txtNJLead.Clear()
            txtNJLeadTitle.Clear()
            txtNJLocation.Clear()
            txtNJName.Clear()
            txtNJStart.Clear()
            txtNJTitle1.Clear()
            txtNJTitle2.Clear()
            txtNJTitle3.Clear()
        End If
    End Sub

    Private Function checkFieldsFilled() As Boolean
        If txtNJClient.Text = "" Or txtNJDeadline.Text = "" Or txtNJDescription.Text = ""
Or txtNJLead.Text = "" Or
            txtNJLeadTitle.Text = "" Or txtNJLocation.Text = "" Or txtNJName.Text = "" Or
txtNJStart.Text = "" Then
            MessageBox.Show("Please fill in all fields", "Missing information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function
End Class
```

JobListForm

```
Public Class JobListForm
    Dim objJobManager As New clsFileManager
    Dim pendingJobList As New Collection
    Dim completedJobList As New Collection
    'Prints list of jobs

    'Closes JobListForm
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Me.Close()
    End Sub
    'Toggle Completed Jobs Visibility
    Private Sub CompletedJobsToolStripMenuItem_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles CompletedJobsToolStripMenuItem.Click
```

```
        gbJLCompleted.Visible = True
        gbJLPending.Visible = False
        Me.Height = 427
        btnJLBack.SetBounds(704, 355, 103, 23)

    End Sub
    'Toggle Pending Jobs Visibility
    Private Sub PendingJobsToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles PendingJobsToolStripMenuItem.Click
        gbJLCompleted.Visible = False
        gbJLPending.Visible = True
        gbJLPending.SetBounds(13, 65, 809, 269)
        Me.Height = 427
        btnJLBack.SetBounds(704, 355, 103, 23)
    End Sub
    'Close Form
    Private Sub btnJLBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnJLBack.Click
        Me.Close()
    End Sub

    Private Sub BackToolStripMenuItem_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub JobListForm_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        objJobManager.LoadJob(pendingJobList, completedJobList)

        Dim i As Integer
        Dim counter As Integer
        Dim temp As String
        i = 1
        counter = pendingJobList.Count
        While counter > 0
            temp = pendingJobList.Item(i)
            'pulls name
            temp = temp.Substring(6)
            lstPendingName.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'pulls client
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(8)
            lstPendingClient.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'trash
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(10)
            temp = temp.Substring(temp.IndexOf("<"))
            'pulls start date
            temp = temp.Substring(7)
            lstPendingStart.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'pulls deadline
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(10)
            lstPendingDeadline.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'trash
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(13)
            temp = temp.Substring(temp.IndexOf("<"))
```

```
            'pulls lead
            temp = temp.Substring(6)
            lstPendingLead.Items.Add(temp.Substring(0, temp.IndexOf("<")))

            counter = counter - 1
            i = i + 1
        End While

        i = 1
        counter = completedJobList.Count
        While counter > 0
            temp = completedJobList.Item(i)
            'pulls name
            temp = temp.Substring(6)
            lstCompletedName.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'pulls client
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(8)
            lstCompletedClient.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'trash
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(10)
            temp = temp.Substring(temp.IndexOf("<"))
            'pulls start date
            temp = temp.Substring(7)
            lstCompletedStart.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'pulls deadline
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(10)
            lstCompletedDeadline.Items.Add(temp.Substring(0, temp.IndexOf("<")))
            'trash
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(13)
            temp = temp.Substring(temp.IndexOf("<"))
            'pulls lead
            temp = temp.Substring(6)
            lstCompletedLead.Items.Add(temp.Substring(0, temp.IndexOf("<")))

            counter = counter - 1
            i = i + 1
        End While
    End Sub

    Private Sub AllJobsToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
AllJobsToolStripMenuItem.Click
        gbJLCompleted.Visible = True
        gbJLPending.Visible = True
        gbJLPending.SetBounds(13, 340, 809, 269)
        Me.Height = 707
        btnJLBack.SetBounds(704, 624, 103, 23)
    End Sub
End Class
```

Chapter 2- Iteration 2

Analysis

For iteration two, we had to create some new classes that would handle the use cases that were assigned for iteration 2. Included in this iteration was creating and modifying an invoice, and to do this, we implemented *clsInvoice*, and *clsInvoiceListManager*. The clsInvoice class was responsible for creating an invoice object that had certain attributes such as an invoice ID number, the total amount due, the amount paid by the customer, and the due date of the invoice payments. Even though there were two types of invoices (one-time and regular), there wasn't much difference in the way each type of invoice interacted. So, we added in an invoiceType attribute that would store the type of the invoice. The clsInvoiceListManager was a control class that was responsible of maintaining a collection of invoice objects. Its primary function is to add a new invoice to the collection.

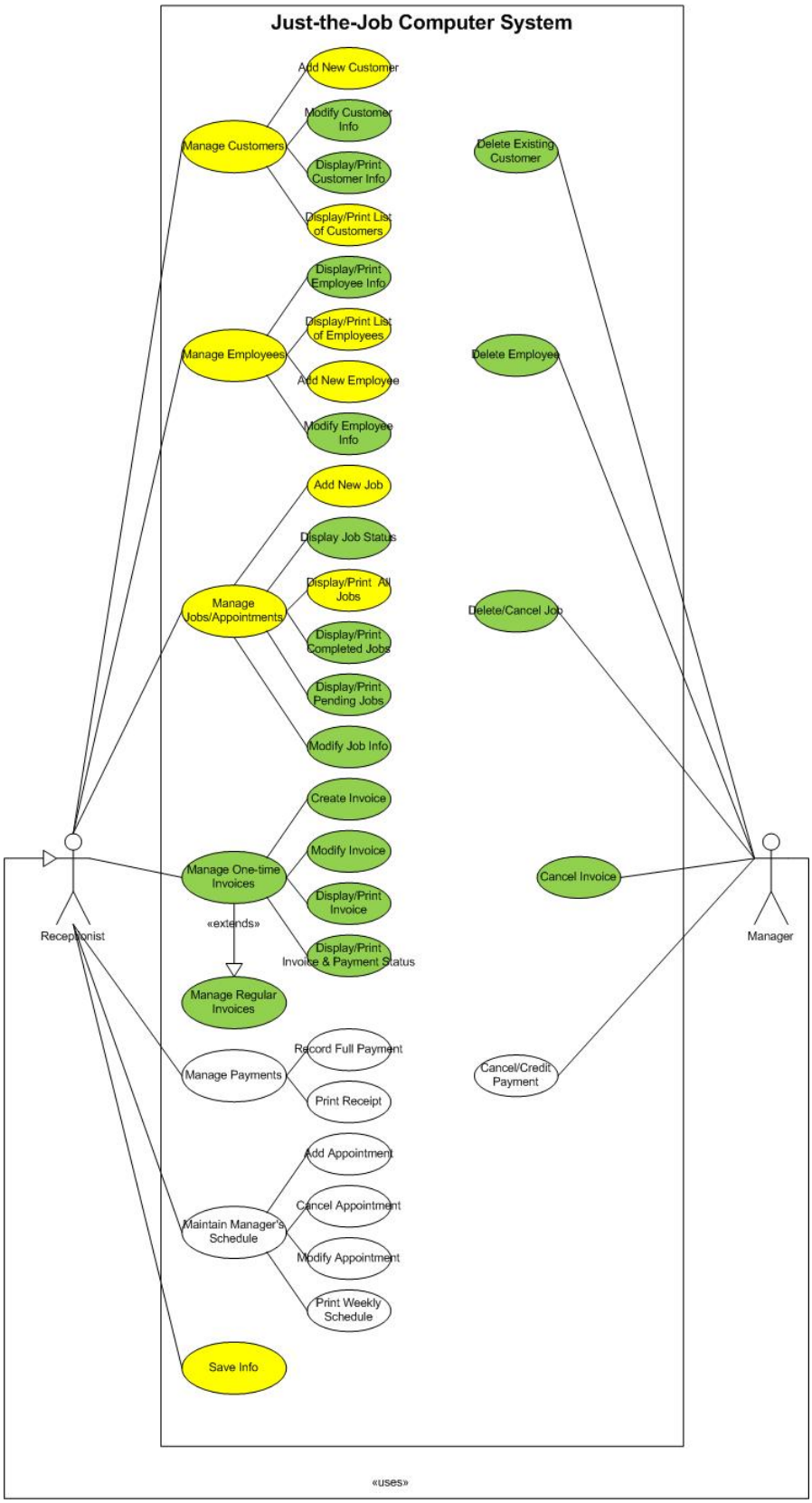
Lastly, we had to create a file manager class called *clsFileManager* that allows the program to save and load customers, employees, jobs, and invoices into a text file database system. These functions, when called, would either read from or write to a designated text file the attributes of each object. When a save function is called, the class will write into the file tags stating each of the attributes, followed by the value being stored. This is critical for being able to pulling the string apart and pulling the correct information. Likewise, a load function will simply pull the string, which includes both the attribute values as well as their preceding tags into a collection.

Requirements

For iteration two, we had to continue to implement our use case diagram. We were to implement the entire functionality for invoices, which includes adding invoices and viewing them in a list, as well as viewing an invoice's payment status. We also had to work on editing information and removing objects from the collections for customers, employees, jobs, and invoices.

Figure 2.1 below shows our use case diagram during the iteration 2 stage. Use cases in yellow are use cases that were assigned for iteration 1 and Use cases in green are the ones assigned for the second iteration. It is also important to note that even though the *Save Info* use case was assigned for the first iteration but wasn't completed until iteration 2 due to our team descoping the first iteration.

Figure 2.1- Use Case Diagram for iteration 2



Use Case Description

The following list contains the use case descriptions. These descriptions map out the basic information for each use-case and displays the sequence of events that occur during the process of the use case, as well as possible alternative events that could occur under specified conditions.

Use Case: Delete/cancel an Existing Job
Actors: Manager
Goal: Cancel and Delete an existing job that is in the system
Overview: If a job is no longer required, it is cancelled and removed from the list of jobs. It must also be removed from the manager's schedule.

Typical Course of Events:

Actor Action:	**System Response:**
1. User logs in	
	2. System accepts or rejects log in
3. User chooses the job to delete	
	4. System deletes the job from the list
	5. System removes the job from the manager's schedule
6. User logs out of system	

Alternative Courses:
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.
Step 3-5: The user may decide to not delete the schedule and exit the system.

Use Case: Display/Print the Status of a Specific Job
Actors: Receptionist/Manager
Goal: Display and print the status of a specific job
Overview: The user can display and print the status of a specific job. This will show the information such as the job name, the client, the start date, the deadline, and the completion status.

Typical Course of Events:

Actor Action:	**System Response:**
1. User logs in	
	2. System accepts or rejects log in
3. User chooses the job to print and display	
	4. System displays and prints the job
5. User logs out of system	

Alternative Courses:
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.
Step 3-4: The user may decide to display the information.
Step 5: There may be an issue with the printer such as no ink or paper.

Use Case: Display/Print a List of All Completed Jobs

Actors: Receptionist/Manager
Goal: Display and print a list of all completed jobs
Overview: The user can display and print all of the completed jobs. This will show the information such as the job name, the client, the start date, the deadline, and the completion status.

Typical Course of Events:

Actor Action:	**System Response:**
1. User logs in	
	2. System accepts or rejects log in
3. User chooses to display and print all of the completed jobs	
	4. System displays and prints all of the completed jobs
5. User logs out of system	

Alternative Courses:
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.
Step 3-4: The user may decide to display the information.
Step 5: There may be an issue with the printer such as no ink or paper.

Use Case: Display/Print a List of All Pending Jobs
Actors: Receptionist/Manager
Goal: Display and print a list of all pending jobs
Overview: The user can display and print all of the pending jobs. This will show the information such as the job name, the client, the start date, the deadline, and the completion status.

Typical Course of Events:

Actor Action:	**System Response:**
1. User logs in	
	2. System accepts or rejects log in
3. User chooses to display and print all of the pending jobs	
	4. System displays and prints all of the pending jobs
5. User logs out of system	

Alternative Courses:
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.
Step 3-4: The user may decide to display the information.
Step 5: There may be an issue with the printer such as no ink or paper.

Use Case: Display/Print customer information
Actors: Receptionist/Manager
Goal: Display and print information of a specific customer
Overview: The user can display and print customer information. This will show the information such as the name, address, phone number, and email.

Typical Course of Events:

Actor Action:	**System Response:**
1. User logs in	
	2. System accepts or rejects log in
3. User searches a customer to print/display	
	4. System displays and prints the customer
5. User logs out of system	

Alternative Courses:
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.
Step 3-4: The user may decide to display the information.
Step 5: There may be an issue with the printer such as no ink or paper.

Use Case: Delete existing customer
Actors: Manager
Goal: Delete an existing customer that is in the system
Overview: If a customer is not going to continue service, they are removed from the list of current customers.

Typical Course of Events:

Actor Action:	**System Response:**
1. User logs in	
	2. System accepts or rejects log in
3. User chooses the customer to delete	
	4. System deletes the customer from the list
5. User logs out of system	

Alternative Courses:
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.

Use Case: Modify customer information
Actors: Receptionist/Manager
Goal: Edit an existing customers information
Overview: If a customer need to change their information, likely address or phone number, the old data must be copied, removed, and overwritten.

Typical Course of Events:

Actor Action:	**System Response:**
1. User logs in	
	2. System accepts or rejects log in
3. User chooses the customer to edit	
	4. System overwrites the old information
	5. System updates any invoices for the customer
	6. System updates any jobs for the customer
6. User logs out of system	

Alternative Courses:
Step 1 & 2: If the user enters an incorrect username or password the system will not allow them in.
Step 3-5: The user may decide to not edit, and instead exit the system.

Sequence Diagrams:

The following images below show all of the sequence diagrams created for iteration 2. These diagrams are used to help track the message sending that occurs between the different classes of each use-case. These are done in depth and have been created by the actual code in the program.

Figure 2.2

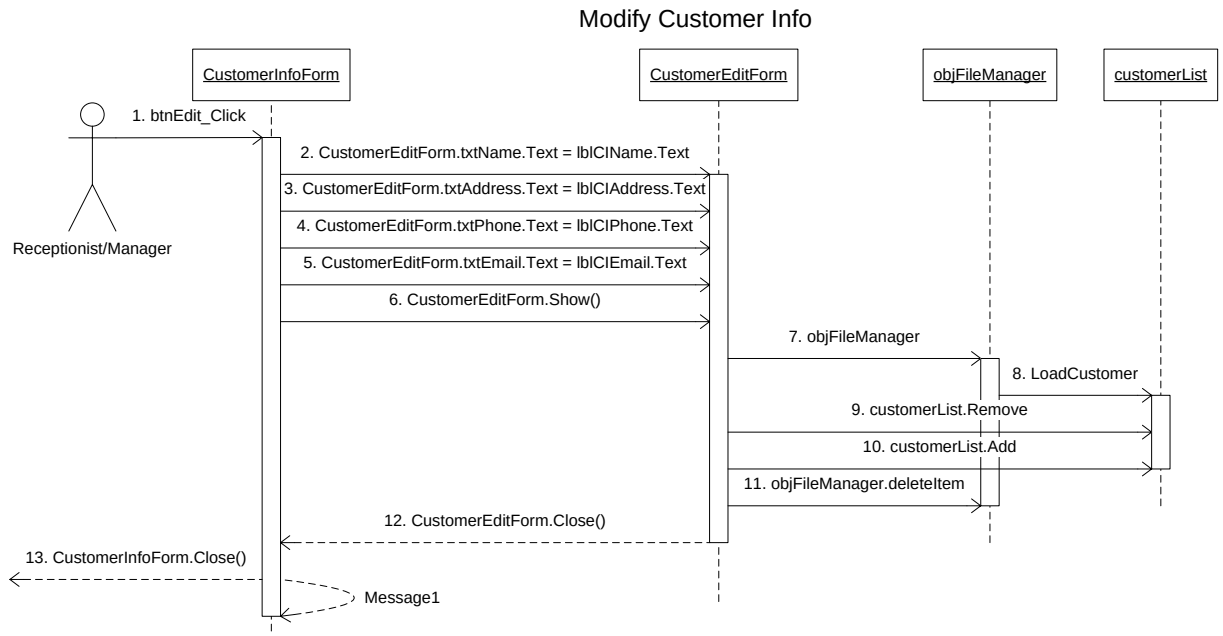


Figure 2.3

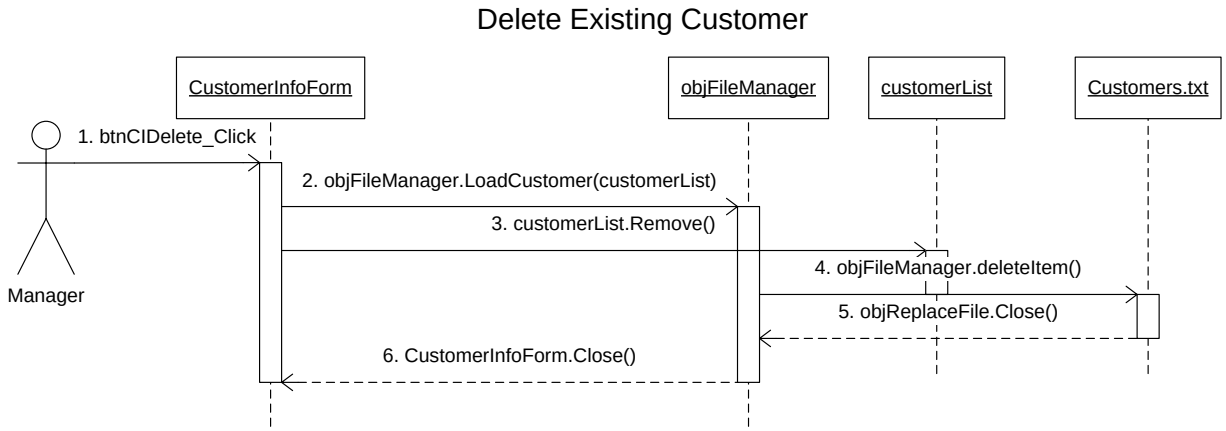


Figure 2.4

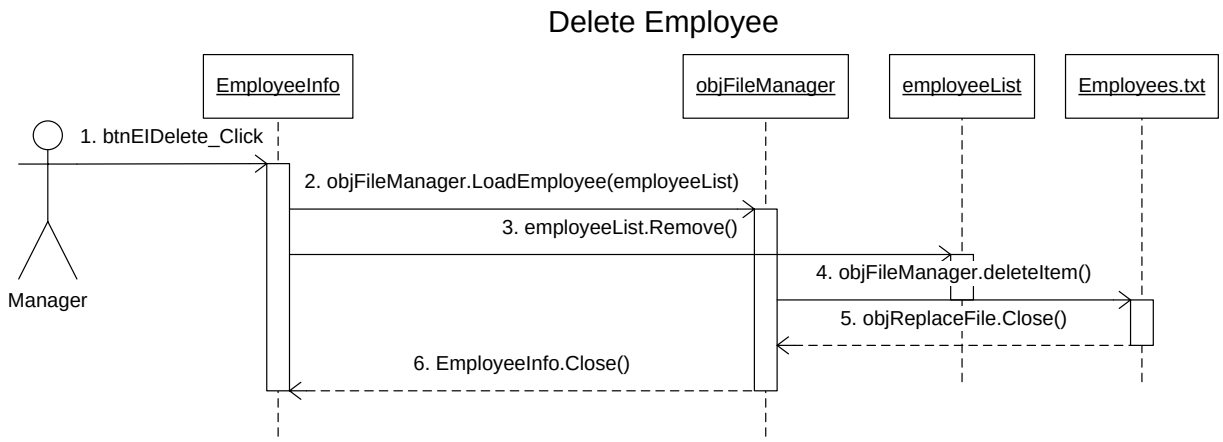


Figure 2.5

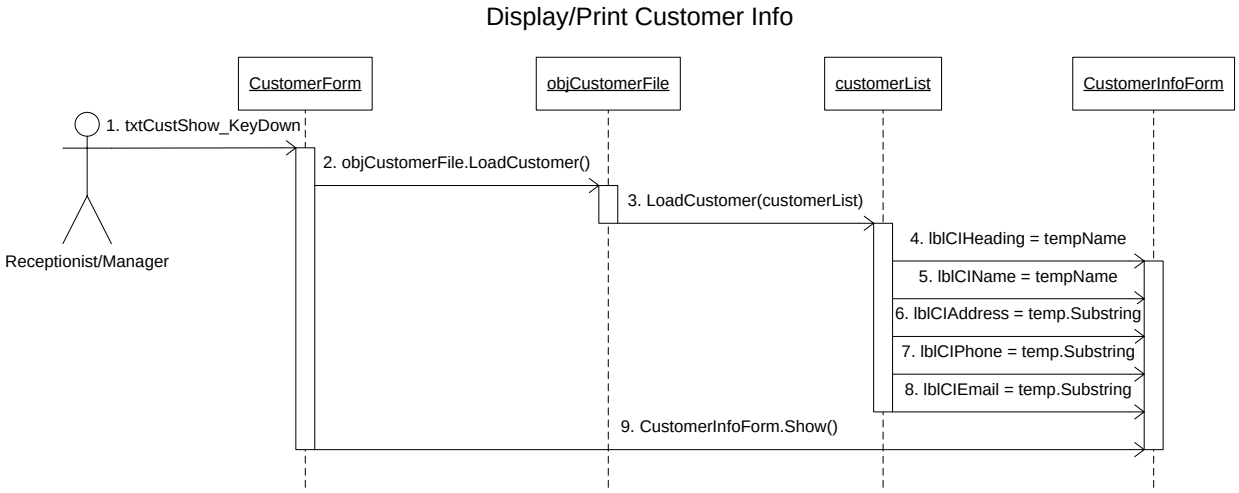


Figure 2.6

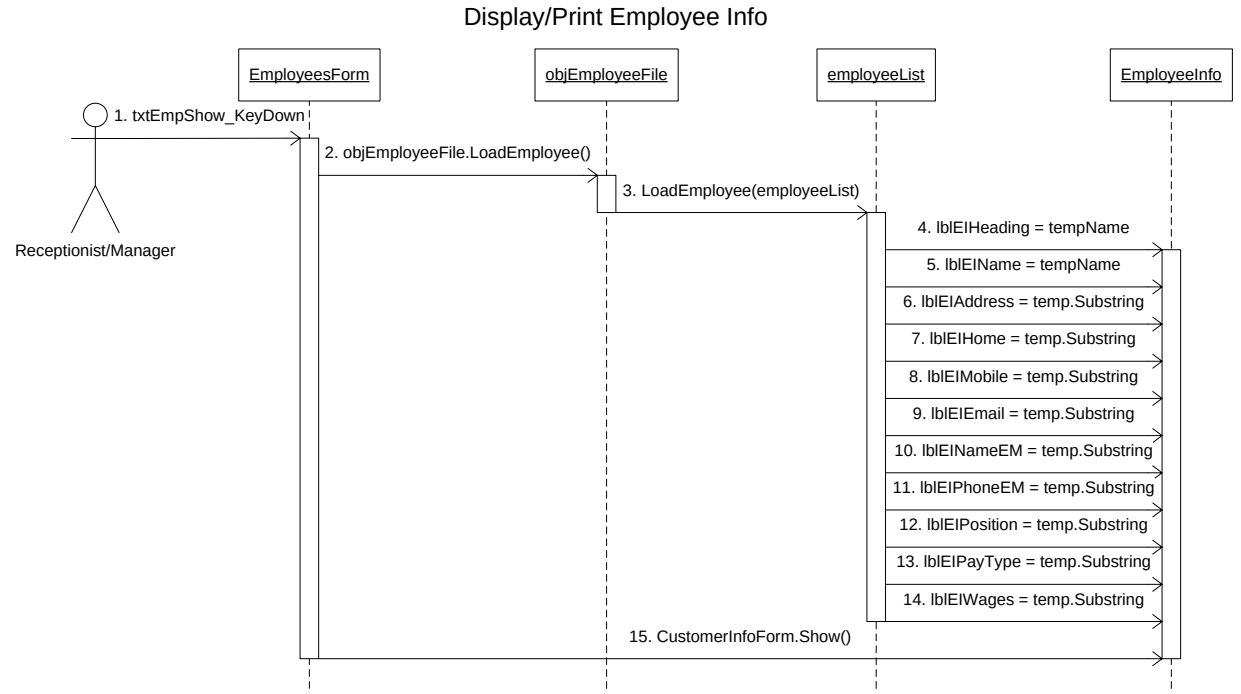


Figure 2.7

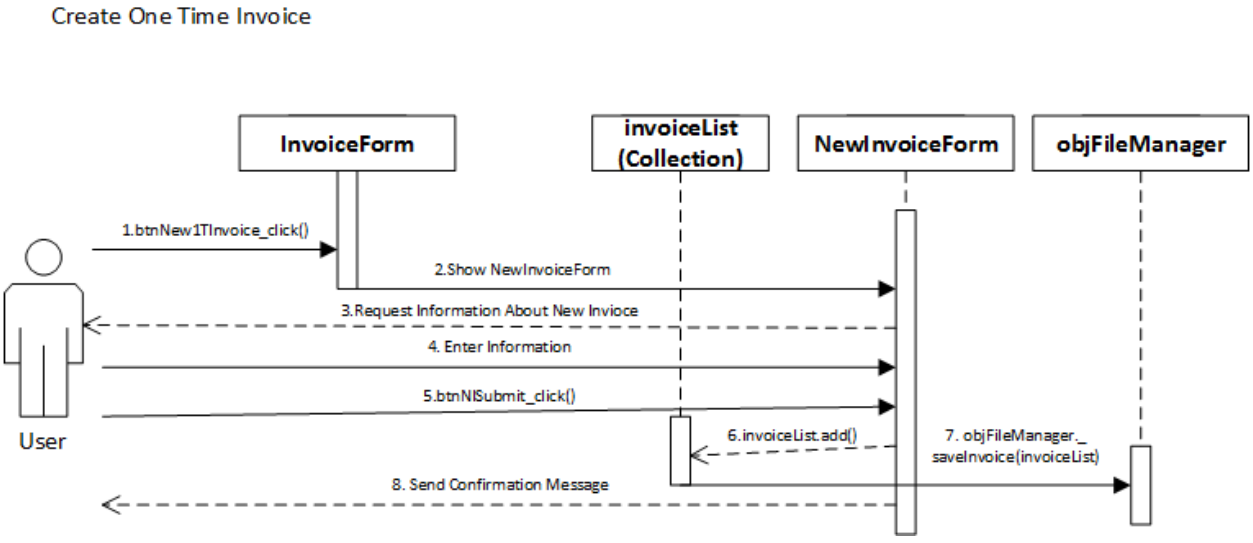


Figure 2.8

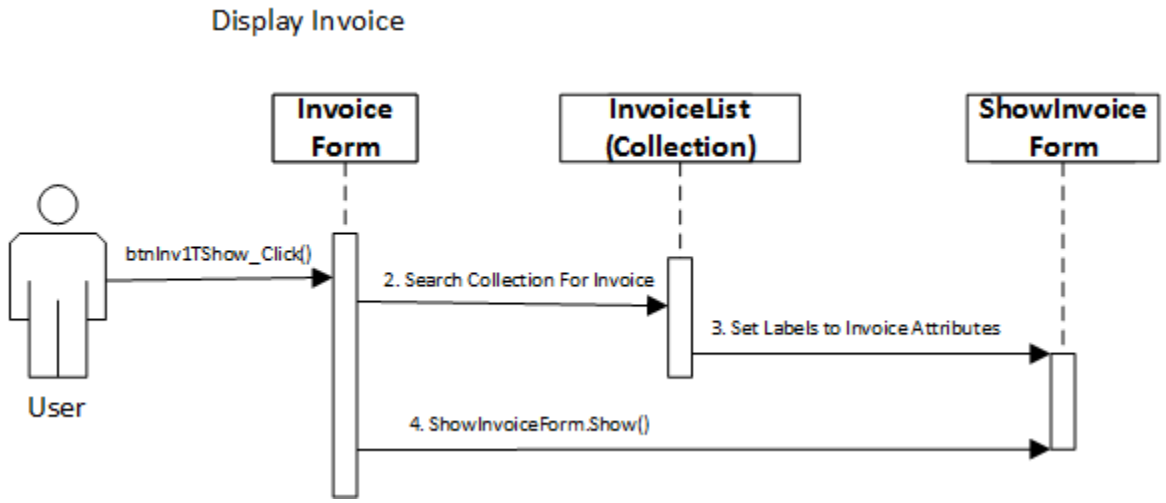


Figure 2.9

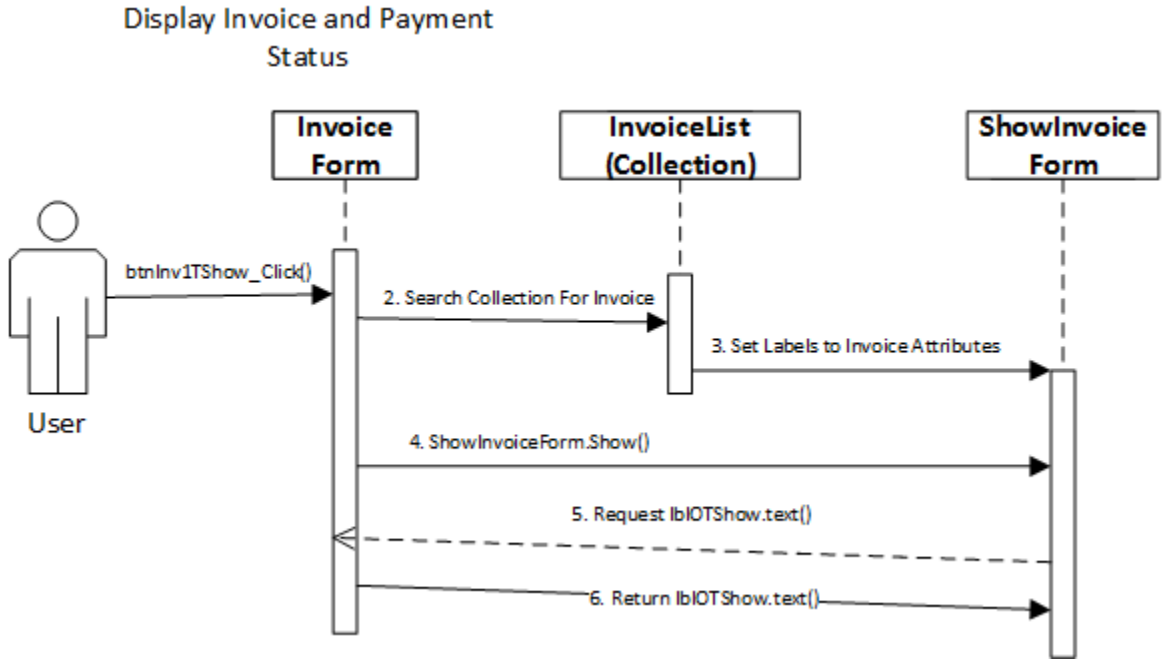


Figure 2.10

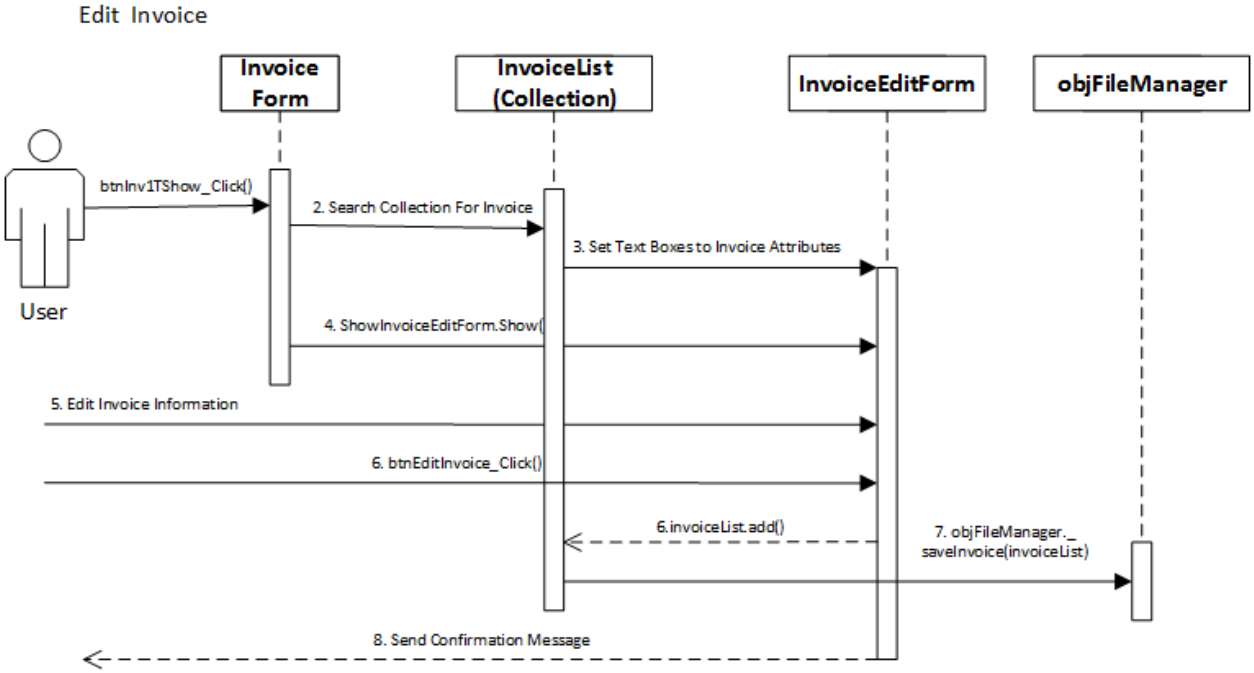


Figure 2.11

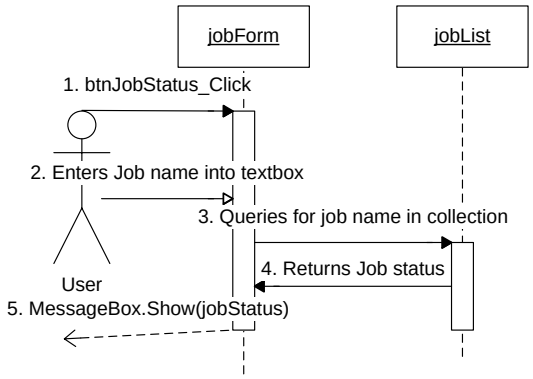


Figure 2.12

Delete Invoice

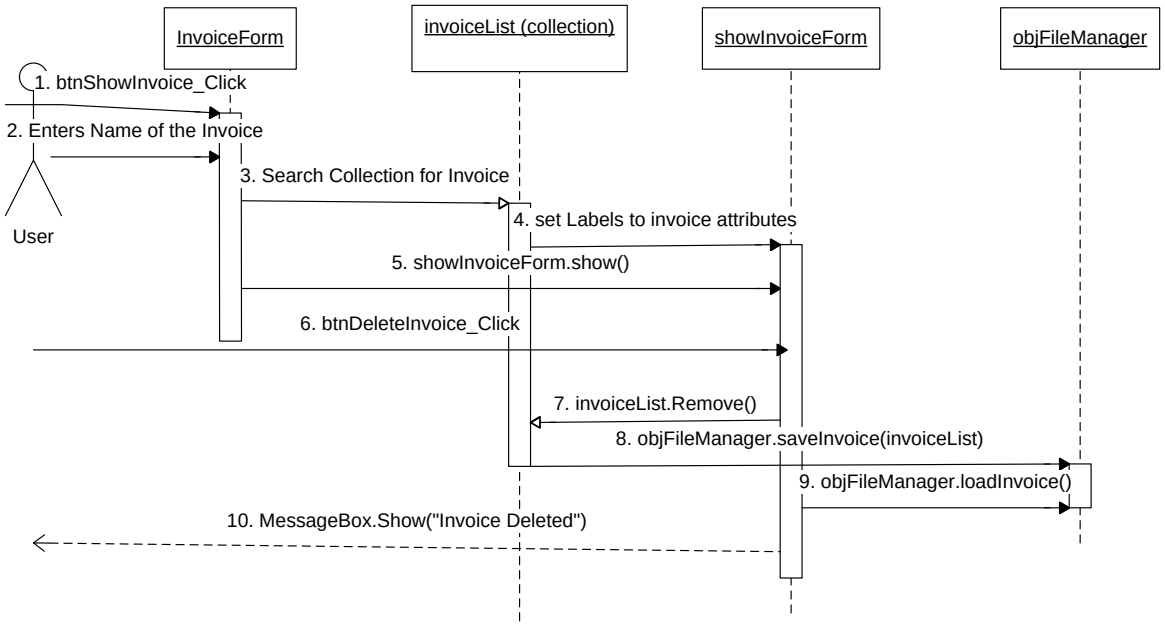


Figure 2.13

Display Pending Jobs

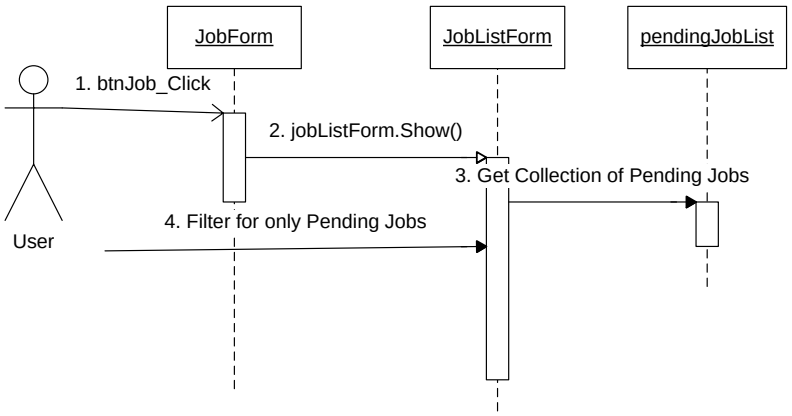


Figure 2.14

Display Completed Jobs

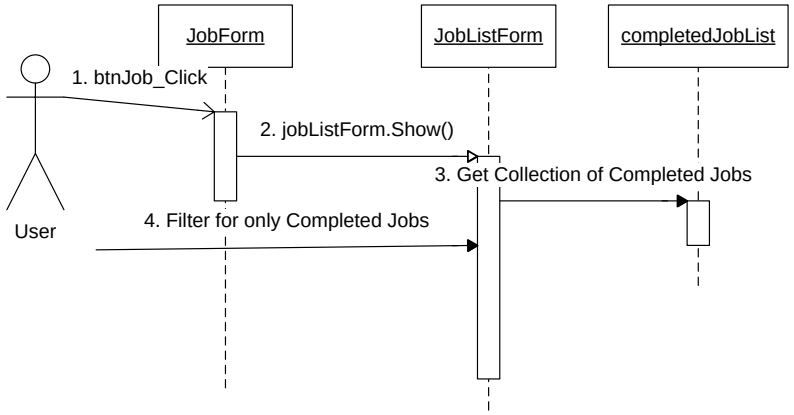


Figure 2.15

Delete Job

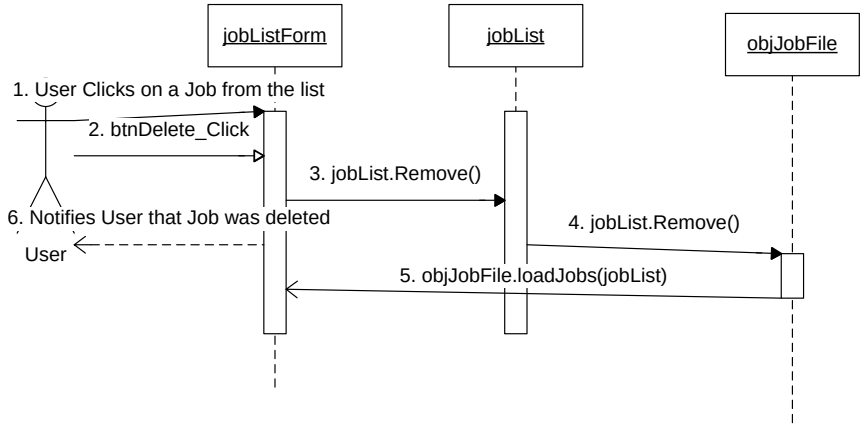


Figure 2.16

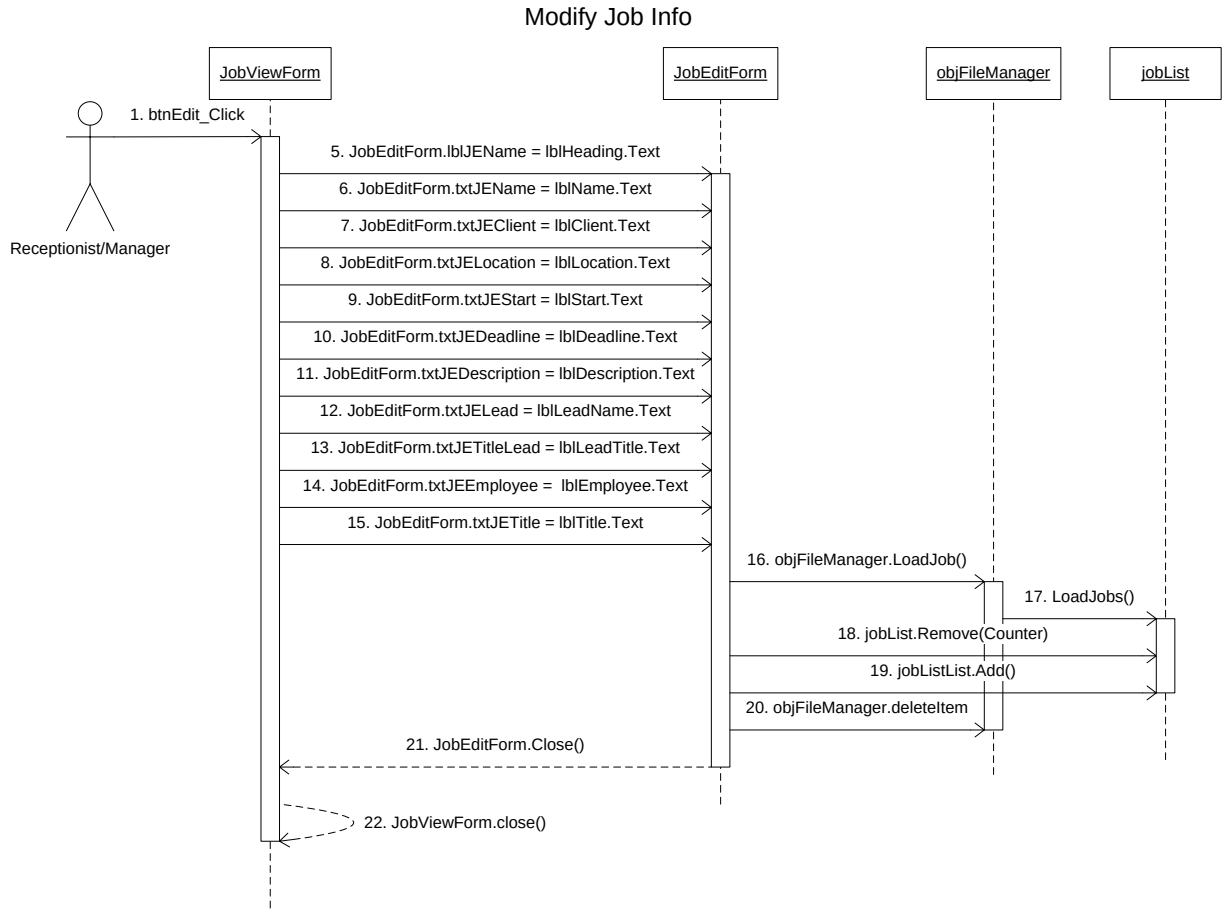
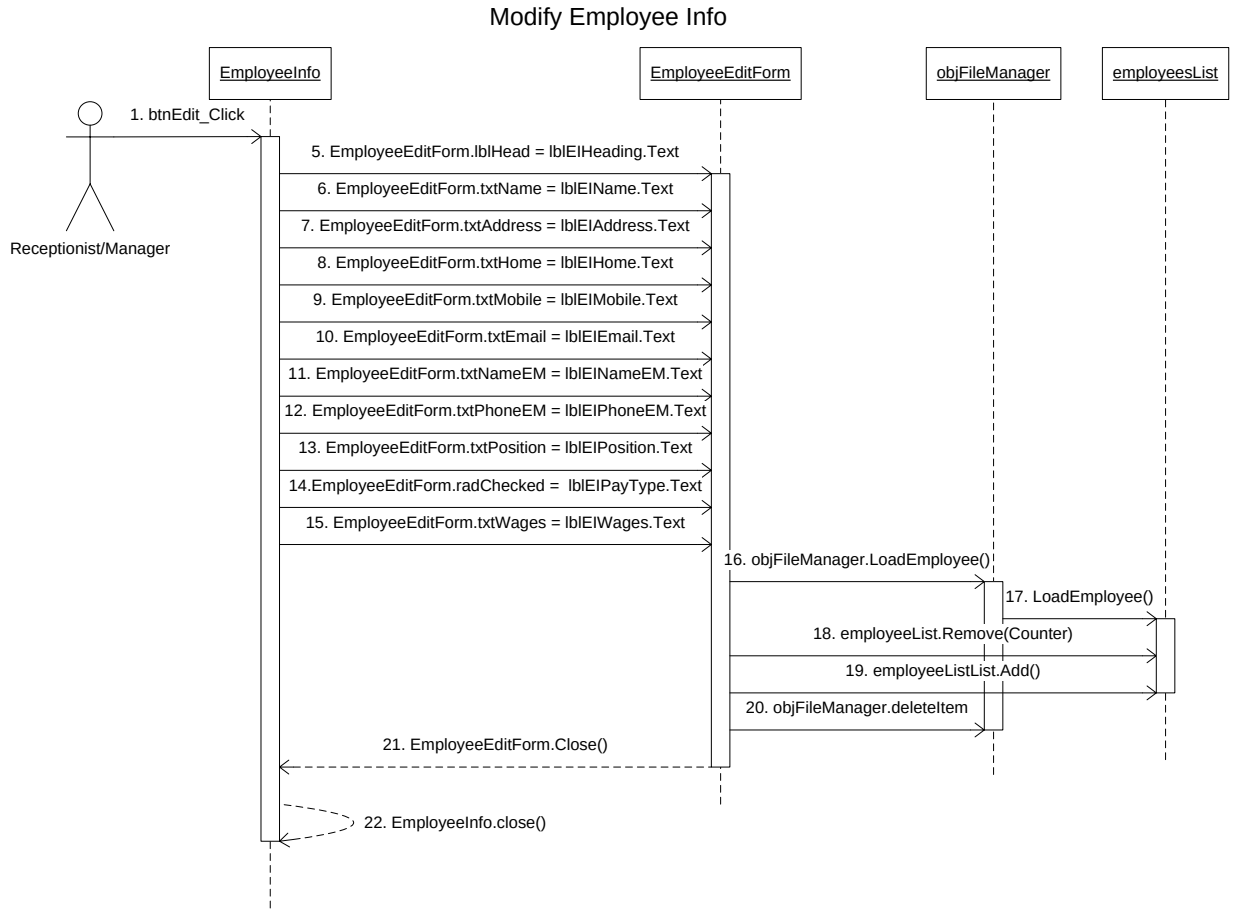

Modify Job Info
JobViewForm
JobEditForm
objFileManager
jobList
Receptionist/Manager
1. btnEdit_Click
5. JobEditForm.lblJEName = lblHeading.Text
6. JobEditForm.txtJEName = lblName.Text
7. JobEditForm.txtJEClient = lblClient.Text
8. JobEditForm.txtJELocation = lblLocation.Text
9. JobEditForm.txtJEStart = lblStart.Text
10. JobEditForm.txtJEDeadline = lblDeadline.Text
11. JobEditForm.txtJEDescription = lblDescription.Text
12. JobEditForm.txtJELead = lblLeadName.Text
13. JobEditForm.txtJETitleLead = lblLeadTitle.Text
14. JobEditForm.txtJEEmployee = lblEmployee.Text
15. JobEditForm.txtJETitle = lblTitle.Text
16. objFileManager.LoadJob()
17. LoadJobs()
18. jobList.Remove(Counter)
19. jobListList.Add()
20. objFileManager.deleteItem
21. JobEditForm.Close()
22. JobViewForm.close()

Figure 2.17



CRC Cards

The following three tables show the CRC cards for *clsInvoice, clsInvoiceListManager,* and *clsFileManager* classes. CRC cards depict a class's responsibilities, as well as any collaborators the class works with in completing their responsibilities.

Table 2.1

clsInvoice	
Responsibility	**Collaborator**
Create a new instance of an invoice	

Table 2.2

clsInvoiceListManager	
Responsibility	**Collaborator**
Display information of an invoice	InvoiceListForm
Add new invoices to a collection	clsInvoice

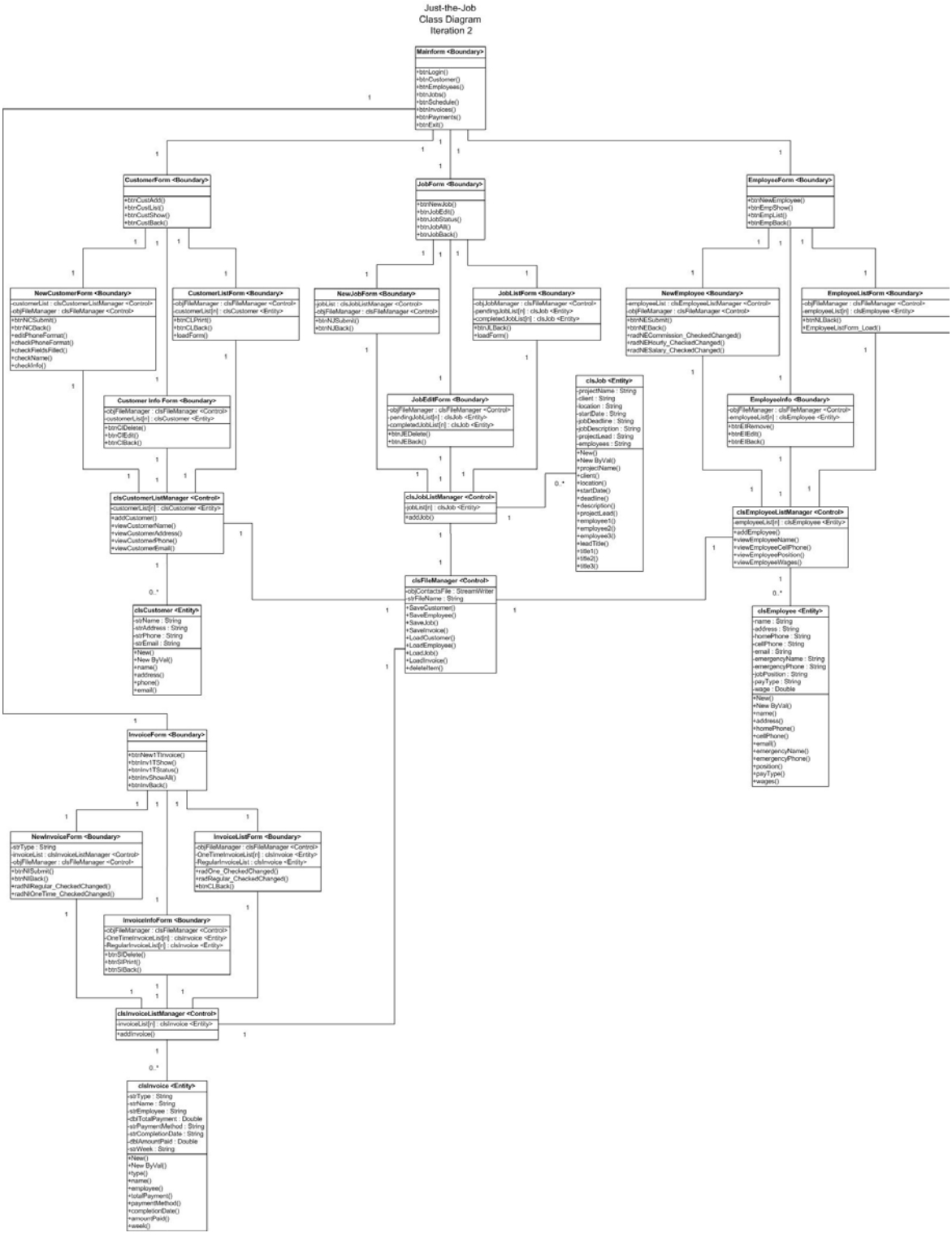
Table 2.3

clsFileManager	
Responsibility	**Collaborator**
Save Customers, Employees, Invoices, Jobs, and Appointments to database.	streamWriter
Load Customers, Employees, Invoices, Jobs, and Appointments from database.	streamReader

Class Diagram

The figure below shows the class diagram through the second iteration. This includes the invoice additions, as well as all of the boundary forms for editing, removing, and viewing an individual object.

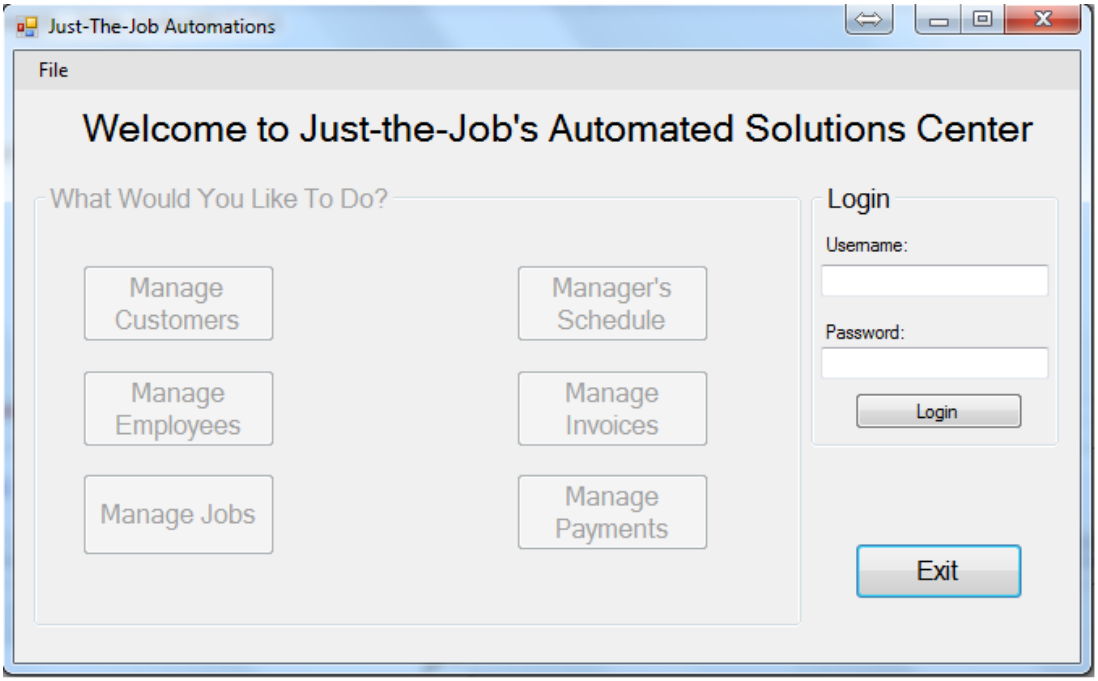
Figure 2.18



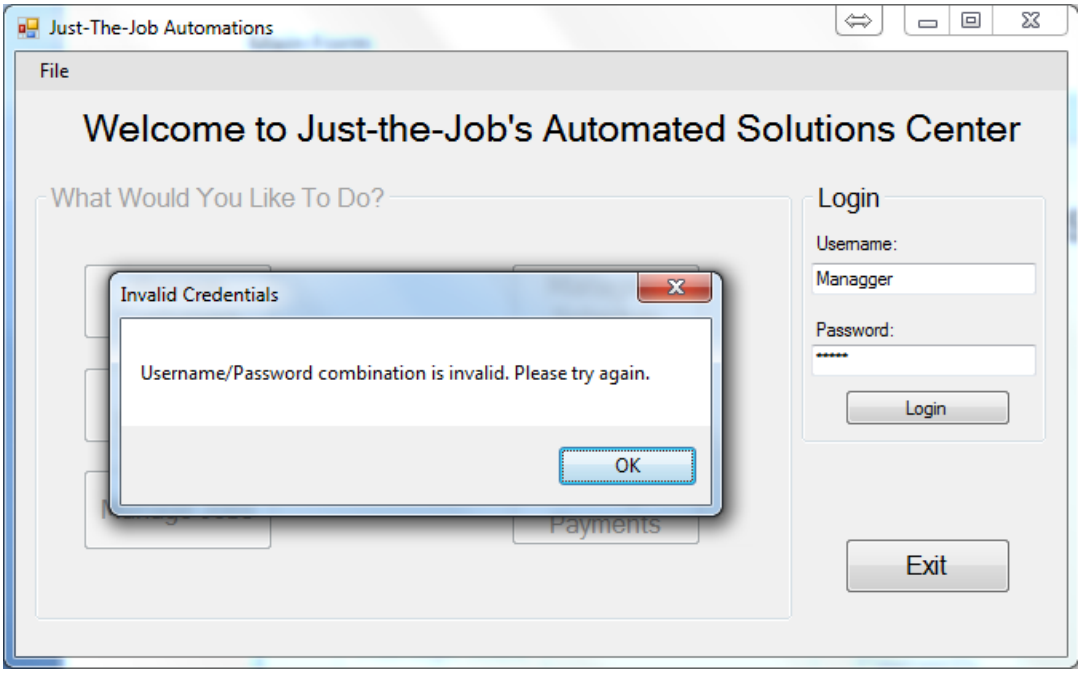
GUI Screenshots

The following images are screen shots of the program interface based on varied user input. Shown are the cases for both correct and incorrect cases for most of the events.

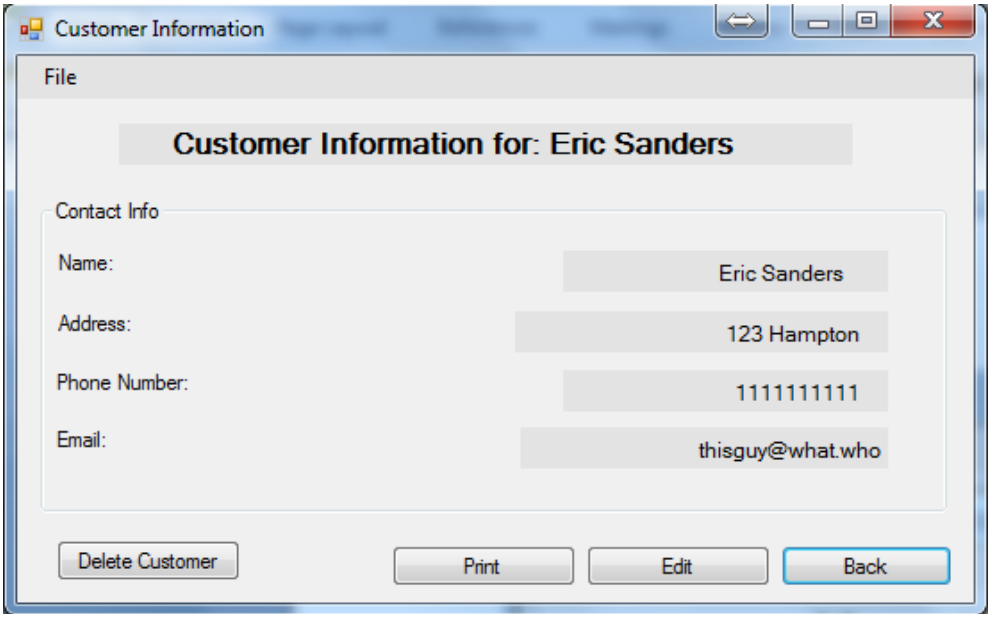
Main Form



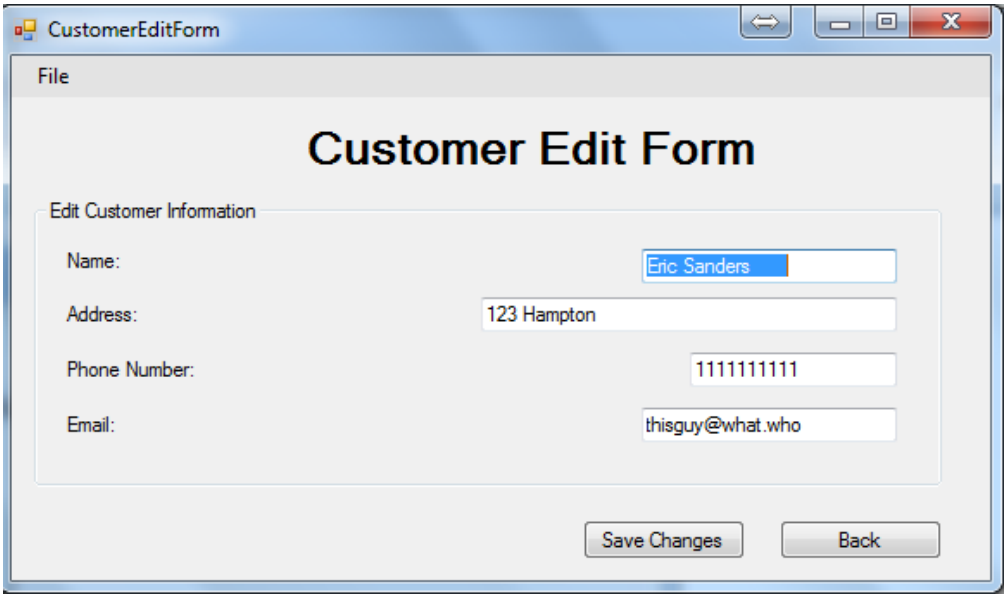
Main form (Incorrect User Name Entered)



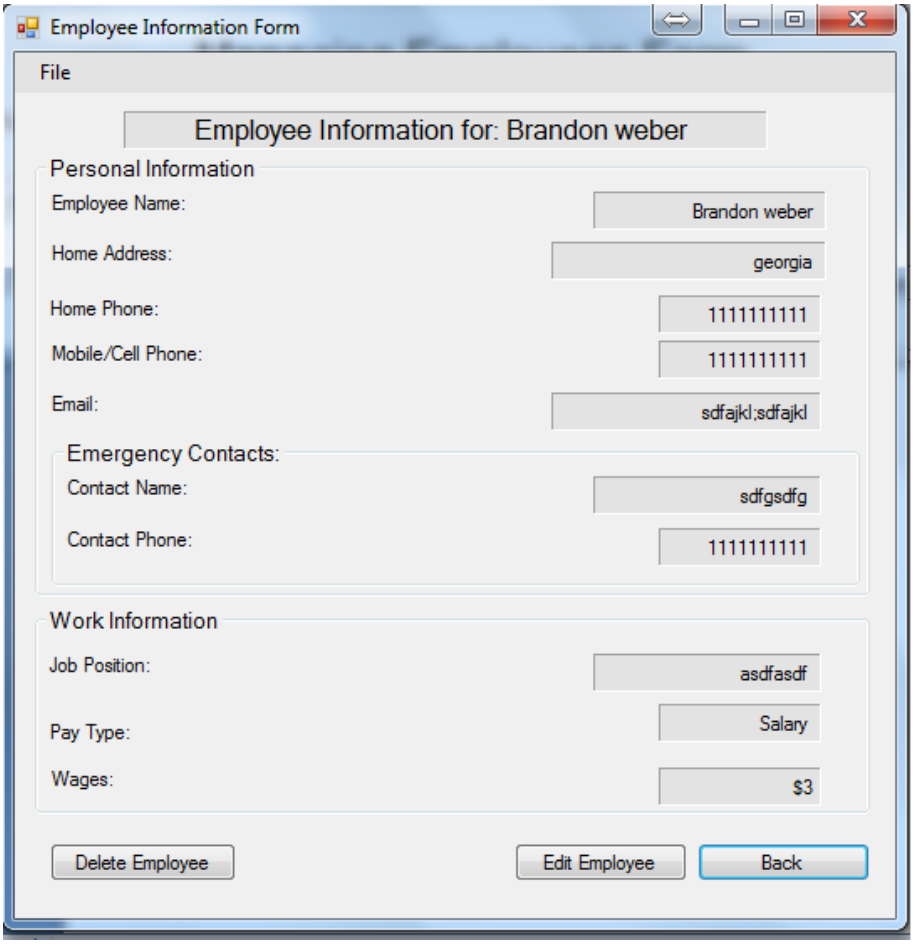
Customer Info Form



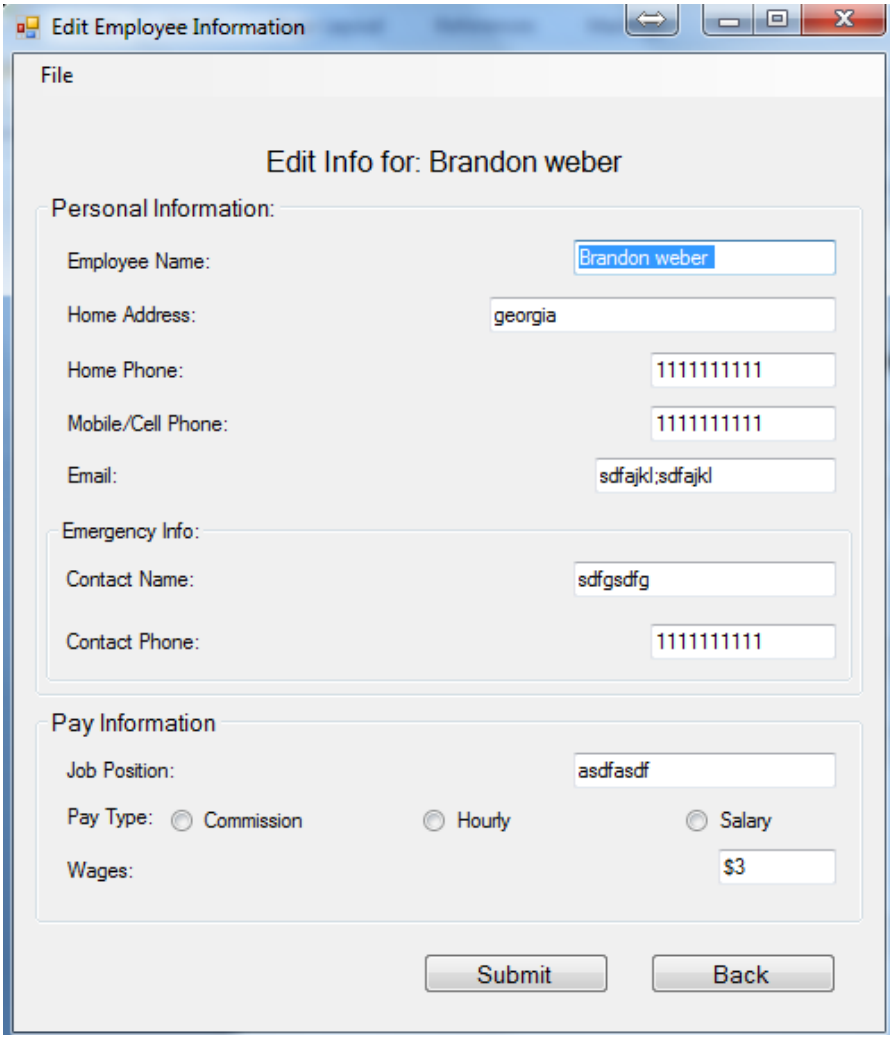
Customer Edit Form



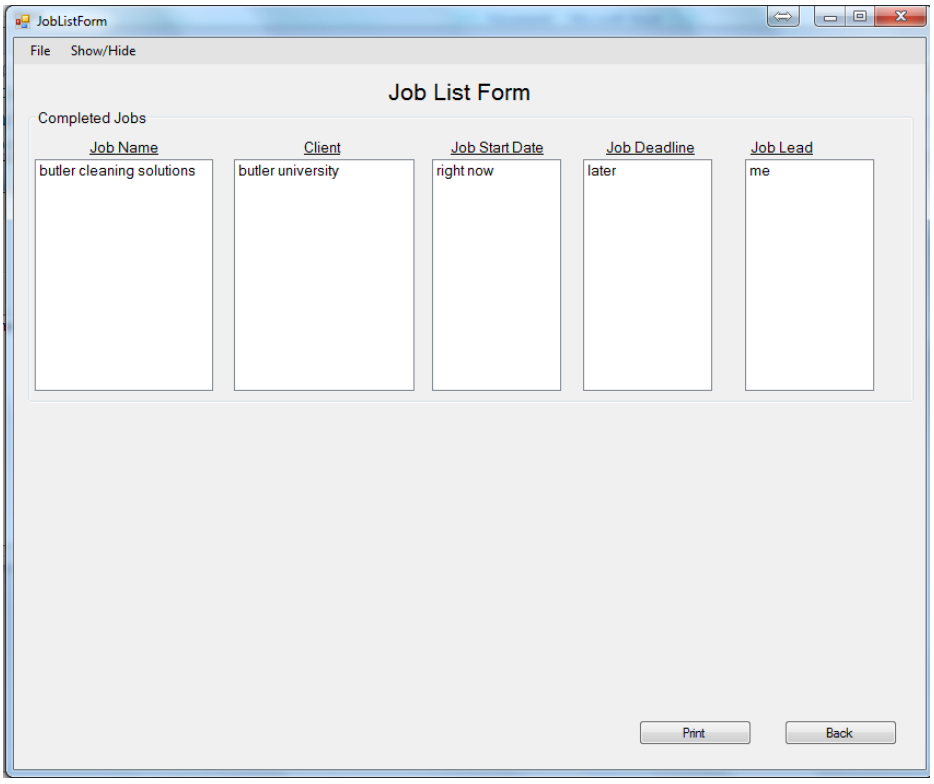
Employee Info Form



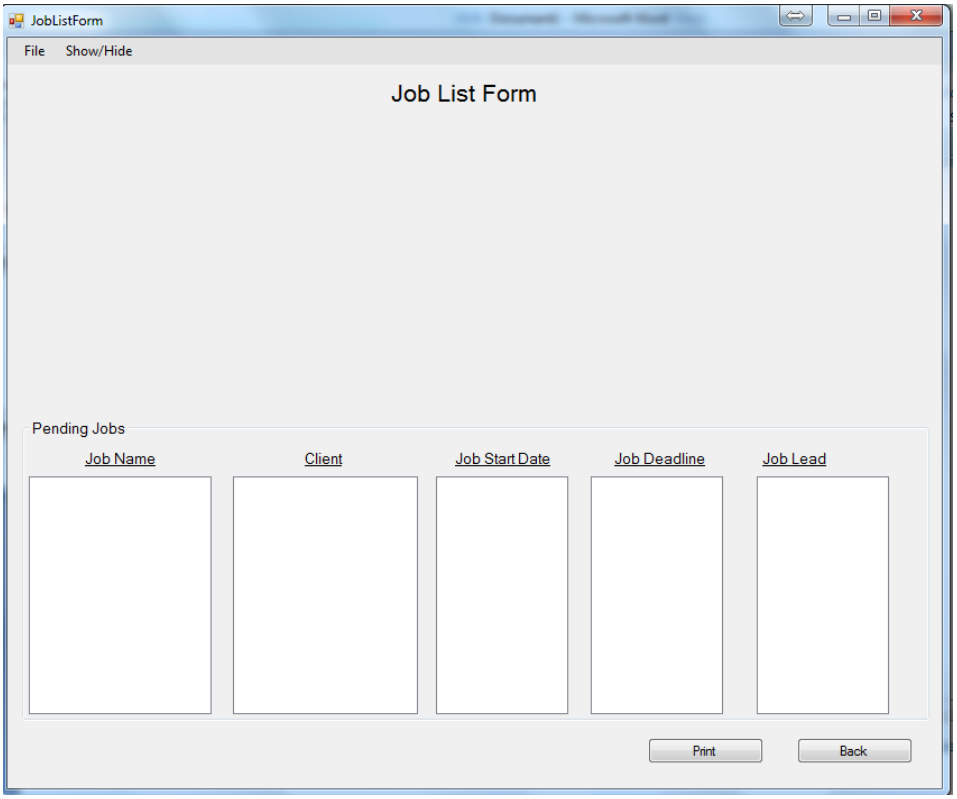
Edit Employee Info



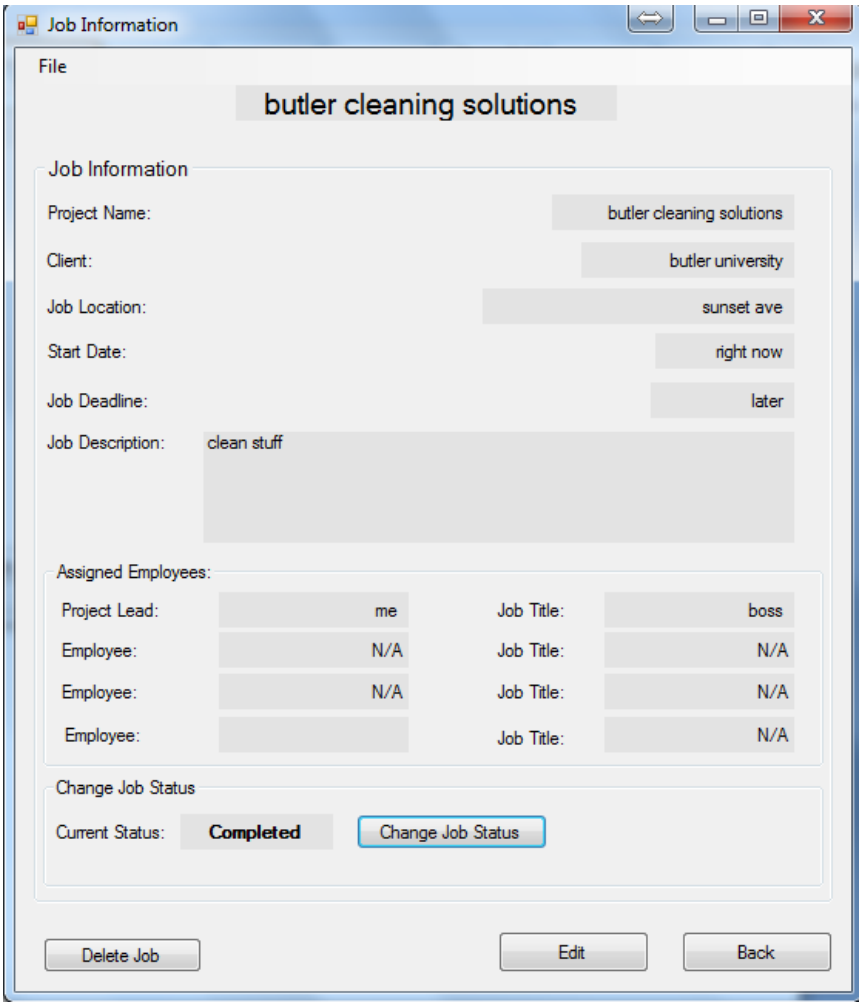
Display Completed Jobs



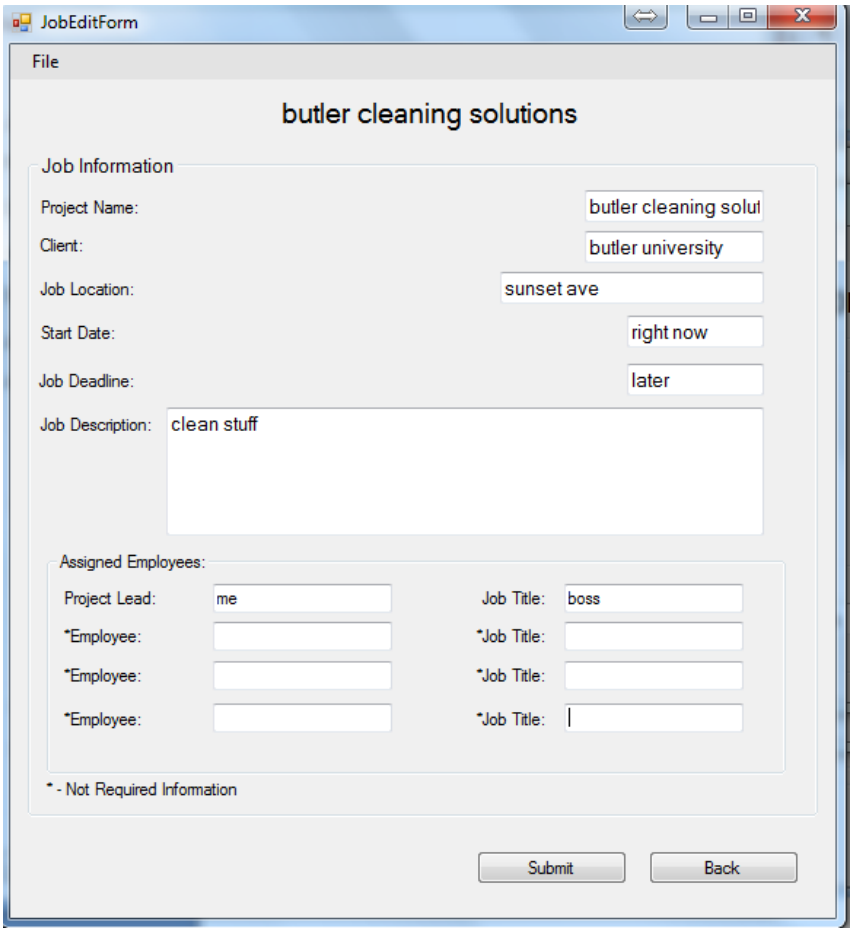
Display Pending Jobs



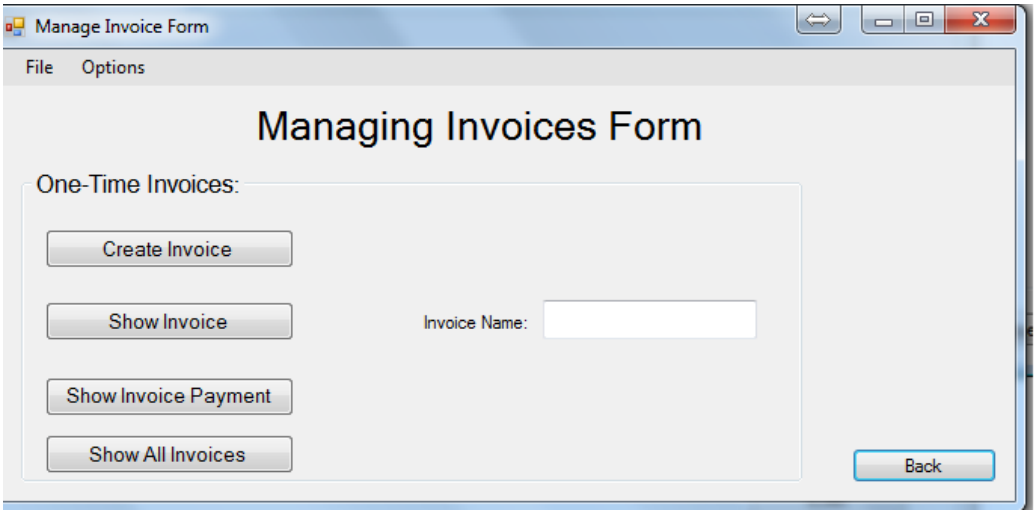
View Job Info



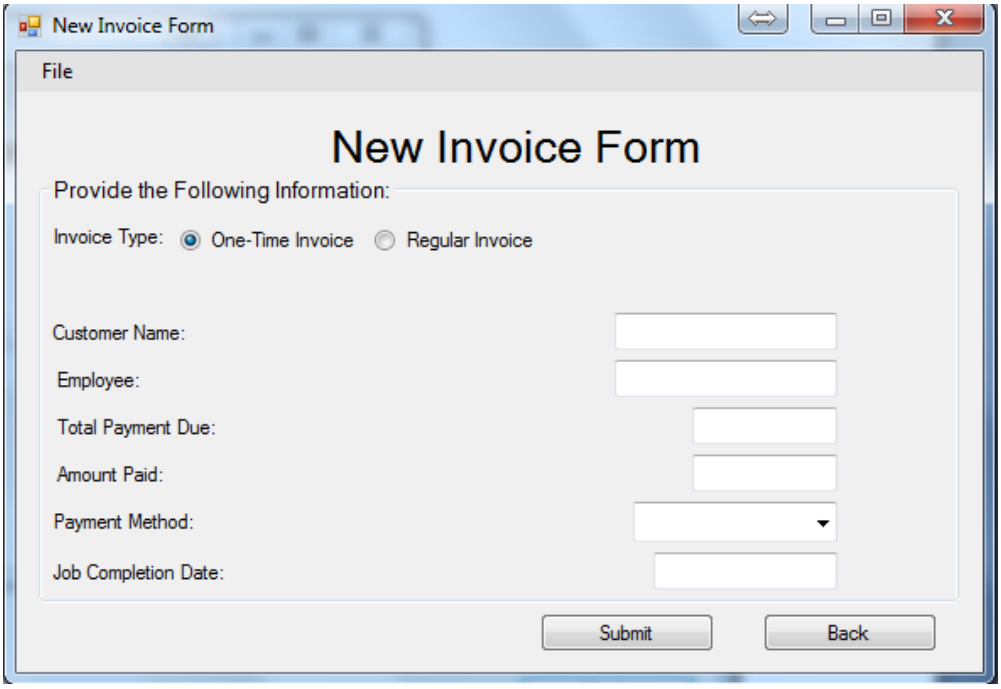
Edit Job Info



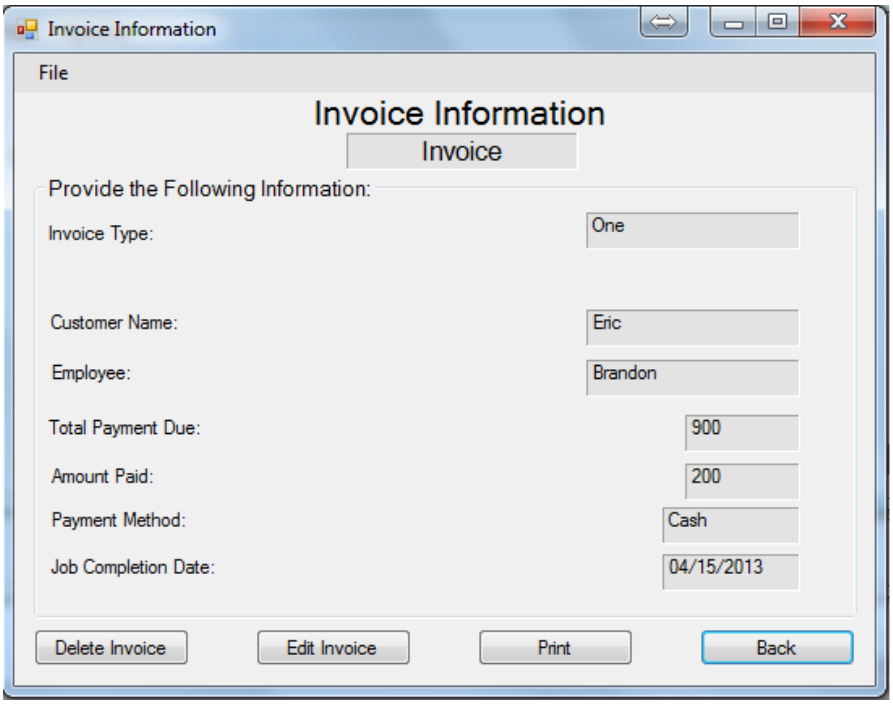
Manage Invoices



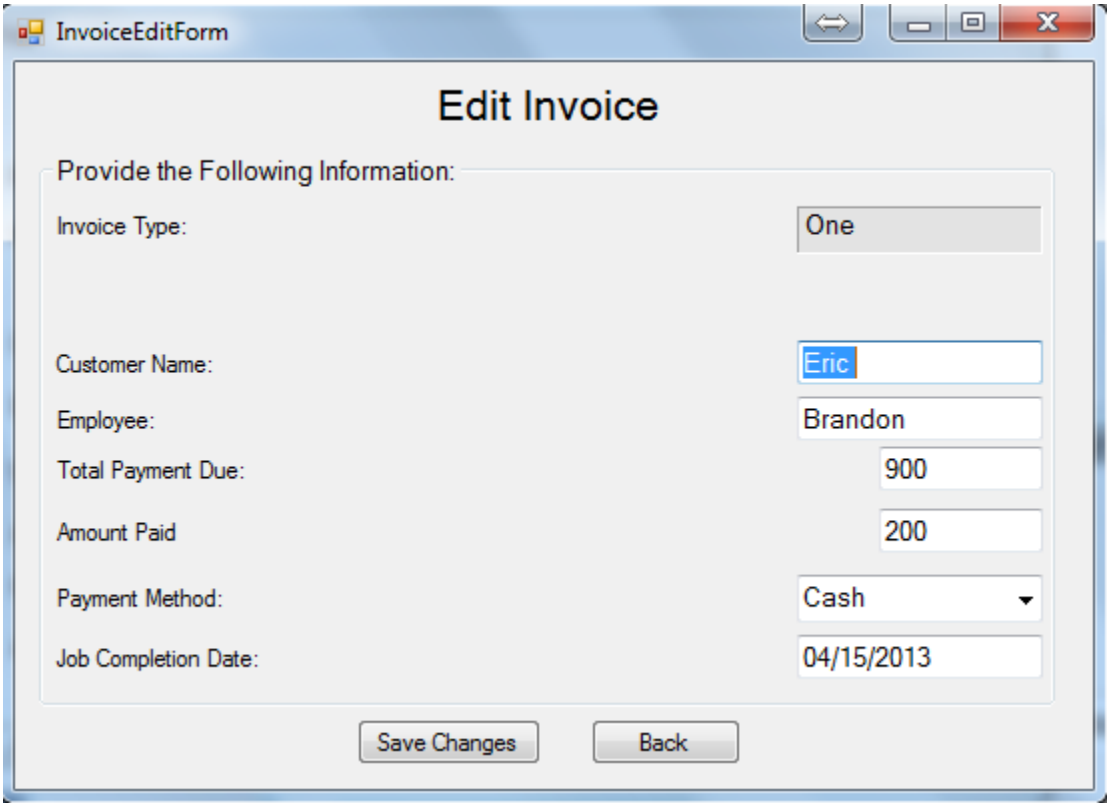
New Invoice



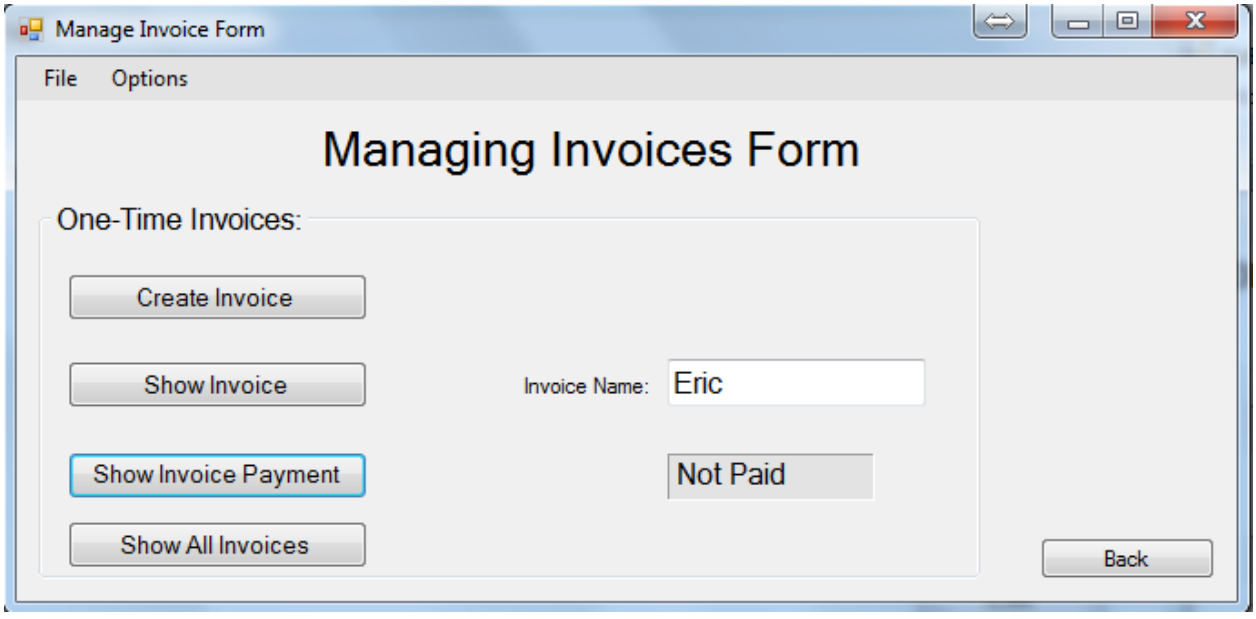
Show Invoice



Edit Invoice



Show Invoice Payment



Show All Invoices

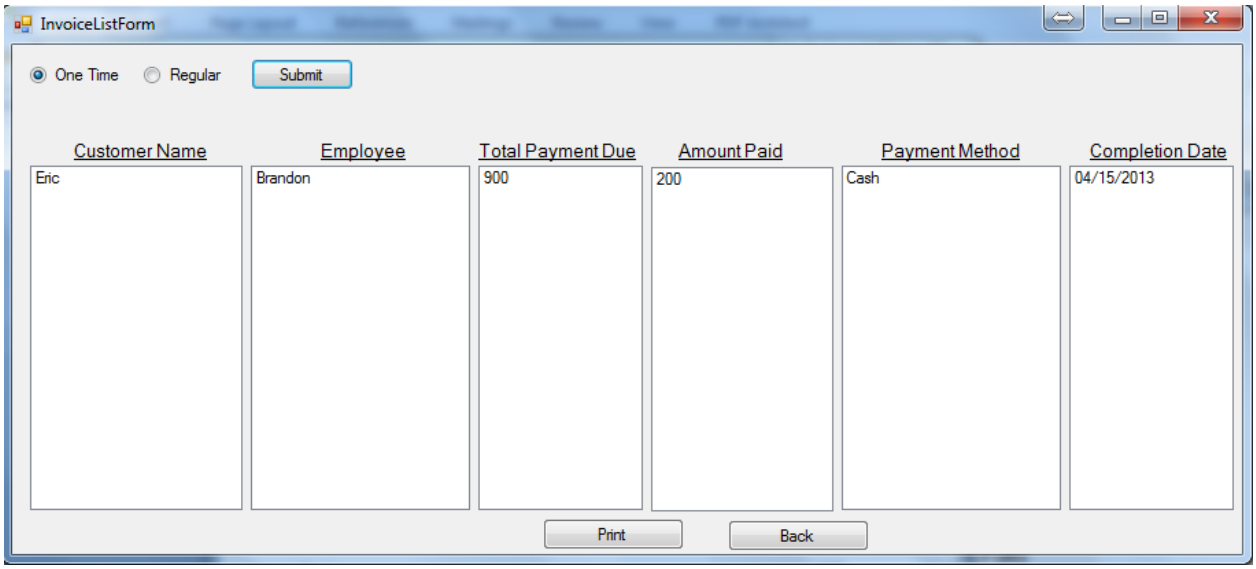


Figure 2.19 GUI

Iteration 2 code

The following contains the all of the code that was implemented for iteration 2. This includes the code from the three classes implemented as well as the forms that were created or updated in the iteration.

clsInvoice

```
Public Class clsInvoice

    Private strType As String
    Private strName As String
    Private strEmployee As String
    Private dblTotalPayment As Double
    Private strPaymentMethod As String
    Private strCompletionDate As String
    Private dblAmountPaid As Double
    Private strWeek As String
    Private strIdNumber As String


    Public Sub New(ByVal idNumber, ByVal type, ByVal name, ByVal employee, ByVal
totalPayment,
                   ByVal paymentMethod, ByVal completionDate, ByVal amountPaid)

        strIdNumber = idNumber
        strType = type
        strName = name
        strEmployee = employee
        dblTotalPayment = totalPayment
        strPaymentMethod = paymentMethod
        strCompletionDate = completionDate
        dblAmountPaid = amountPaid

    End Sub

    Sub New()

    End Sub

    Public Property type As String
        Get
            Return strType
        End Get
        Set(ByVal value As String)
            strType = value
        End Set
    End Property

    Public Property name As String
        Get
            Return strName
        End Get
        Set(ByVal value As String)
            strName = value
        End Set
    End Property
```

```
Public Property employee As String
    Get
        Return strEmployee
    End Get
    Set(ByVal value As String)
        strEmployee = value
    End Set
End Property

Public Property totalPayment As Double
    Get
        Return dblTotalPayment
    End Get
    Set(ByVal value As Double)
        dblTotalPayment = value
    End Set
End Property

Public Property paymentMethod As String
    Get
        Return strPaymentMethod
    End Get
    Set(ByVal value As String)
        strPaymentMethod = value
    End Set
End Property

Public Property completionDate As String
    Get
        Return strCompletionDate
    End Get
    Set(ByVal value As String)
        strCompletionDate = value
    End Set
End Property

Public Property amountPaid As Double
    Get
        Return dblAmountPaid
    End Get
    Set(ByVal value As Double)
        dblAmountPaid = value
    End Set
End Property
Public Property week As String
    Get
        Return strWeek
    End Get
    Set(ByVal value As String)
        strWeek = value
    End Set
End Property

Public Property idNumber As String
    Get
        Return strIdNumber
    End Get
```

```
        Set(ByVal value As String)
            strIdNumber = value
        End Set
    End Property
End Class
```

clsInvoiceListManager

```
Public Class clsInvoiceListManager
    Private colInvoices As New Collection

    Public Sub addInvoice(ByVal idNumber As String, ByVal type As String, ByVal name As
String, ByVal employee As String, ByVal totalPayment As Double,
                ByVal paymentMethod As String, ByVal completionDate As String, ByVal
amountPaid As Double, ByVal week As String)
        Dim objInvoice As New clsInvoice

        objInvoice.idNumber = idNumber
        objInvoice.type = type
        objInvoice.name = name
        objInvoice.employee = employee
        objInvoice.totalPayment = totalPayment
        objInvoice.paymentMethod = paymentMethod
        objInvoice.completionDate = completionDate
        objInvoice.amountPaid = amountPaid
        objInvoice.week = week

        colInvoices.Add(objInvoice)
    End Sub

    Public Sub viewType(ByRef guiDisplay As ListBox)
        Dim invoice As clsInvoice

        For Each invoice In colInvoices
            guiDisplay.Items.Add(invoice.type)
        Next
    End Sub

    Public Sub viewName(ByRef guiDisplay As ListBox)
        Dim invoice As clsInvoice

        For Each invoice In colInvoices
            guiDisplay.Items.Add(invoice.name)
        Next
    End Sub
    Public Sub viewEmployee(ByRef guiDisplay As ListBox)
        Dim invoice As clsInvoice

        For Each invoice In colInvoices
            guiDisplay.Items.Add(invoice.employee)
        Next
    End Sub
    Public Sub viewTotalPayment(ByRef guiDisplay As ListBox)
        Dim invoice As clsInvoice

        For Each invoice In colInvoices
            guiDisplay.Items.Add(invoice.totalPayment)
        Next
```

```
    End Sub
    Public Sub viewPaymentMethod(ByRef guiDisplay As ListBox)
        Dim invoice As clsInvoice

        For Each invoice In colInvoices
            guiDisplay.Items.Add(invoice.paymentMethod)
        Next
    End Sub
    Public Sub viewCompletionDate(ByRef guiDisplay As ListBox)
        Dim invoice As clsInvoice

        For Each invoice In colInvoices
            guiDisplay.Items.Add(invoice.completionDate)
        Next
    End Sub
    Public Sub viewAmountPaid(ByRef guiDisplay As ListBox)
        Dim invoice As clsInvoice

        For Each invoice In colInvoices
            guiDisplay.Items.Add(invoice.amountPaid)
        Next
    End Sub
    Public Sub viewWeek(ByRef guiDisplay As ListBox)
        Dim invoice As clsInvoice

        For Each invoice In colInvoices
            guiDisplay.Items.Add(invoice.week)
        Next
    End Sub
End Class
```

clsFileManager

```
Imports System.IO

Public Class clsFileManager
    Private strFileName As String

    'Used to save a Customer in the database
    Public Sub SaveCustomer(ByVal name As String, ByVal address As String, ByVal phone As
String, ByVal email As String)
        Dim objContactsFile As StreamWriter
        strFileName = "Customers.txt"

        objContactsFile = File.AppendText(strFileName)

        objContactsFile.WriteLine("<name>" + name + " <address>" + address + " <phone>" +
phone + " <email>" + email)


        objContactsFile.Close()
    End Sub

    'Used to save a Customer in the database
    Public Sub SaveEmployee(ByVal name As String, ByVal address As String, ByVal home As
String, ByVal mobile As String,
```

```
                            ByVal email As String, ByVal nameEM As String, ByVal phone As
String, ByVal position As String,
                            ByVal payType As String, ByVal pay As String)
        Dim objContactsFile As StreamWriter
        strFileName = "Employees.txt"

        objContactsFile = File.AppendText(strFileName)

        objContactsFile.WriteLine("<name>" + name + " <address>" + address + " <home>" +
home + " <mobile>" + mobile +
                                  " <email>" + email + " <nameEM>" + nameEM + " <phone>"
+ phone + " <position>" + position +
                                  " <payType>" + payType + "<pay>" + pay)
        objContactsFile.Close()
    End Sub

    'Used to save a Customer in the database
    Public Sub SaveJob(ByVal name As String, ByVal client As String, ByVal location As
String, ByVal start As String, ByVal deadLine As String, ByVal description As String,
                       ByVal lead As String, ByVal employee1 As String, ByVal employee2
As String, ByVal employee3 As String, ByVal leadTitle As String, ByVal title1 As String,
                       ByVal title2 As String, ByVal title3 As String)
        Dim objContactsFile As StreamWriter

        'put in pending jobs list
        strFileName = "PendingJobs.txt"

        objContactsFile = File.AppendText(strFileName)

        objContactsFile.WriteLine("<name>" + name + " <client>" + client + " <location>"
+ location + " <start>" + start +
                                  " <deadLine>" + deadLine + " <description>" +
description + " <lead>" + lead + " <employee1>" + employee1 +
                                  " <employee2>" + employee2 + " <employee3>" + employee3
+ " <leadTitle>" + leadTitle + " <title1>" + title1 +
                                  " <title2>" + title2 + " <title3>" + title3)
        objContactsFile.Close()

        'put in allJobs list
        strFileName = "Jobs.txt"
        objContactsFile = File.AppendText(strFileName)

        objContactsFile.WriteLine("<name>" + name + " <client>" + client + " <location>"
+ location + " <start>" + start +
                                  " <deadLine>" + deadLine + " <description>" +
description + " <lead>" + lead + " <employee1>" + employee1 +
                                  " <employee2>" + employee2 + " <employee3>" + employee3
+ " <leadTitle>" + leadTitle + " <title1>" + title1 +
                                  " <title2>" + title2 + " <title3>" + title3)
        objContactsFile.Close()
    End Sub

    Public Sub SaveAppointment(ByVal name As String, ByVal location As String, ByVal
dateX As String, ByVal time As String, ByVal length As String, ByVal brief As String)
        Dim objContactsFile As StreamWriter
        Try
            strFileName = "Appointments.txt"
```

```
            objContactsFile = File.AppendText(strFileName)

            objContactsFile.WriteLine("<name>" + name + " <location>" + location + "
<date>" + dateX + " <time>" + time +
                                      " <length>" + length + " <brief>" + brief)

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
        objContactsFile.Close()
    End Sub

    Public Sub SaveInvoice(ByVal idNumber As String, ByVal type As String, ByVal name As
String, ByVal employee As String, ByVal payment As String, ByVal paymentMethod As String,
                       ByVal complete As String, ByVal amountPaid As String, ByVal week
As String)
        Dim objContactsFile As StreamWriter
        If type = "One" Then
            strFileName = "OneTimeInvoices.txt"

            objContactsFile = File.AppendText(strFileName)

            objContactsFile.WriteLine("<id>" + idNumber + " <name>" + name + " <type>" +
type + " <employee>" + employee + " <payment>" + payment +
                                      " <paymentMethod>" + paymentMethod + " <complete>" +
complete + " <amountPaid>" + amountPaid + " <date>" + week)
            objContactsFile.Close()
        ElseIf type = "Regular" Then
            strFileName = "RegularInvoices.txt"

            objContactsFile = File.AppendText(strFileName)

            objContactsFile.WriteLine("<id>" + idNumber + " <name>" + name + " <type>" +
type + " <employee>" + employee + " <payment>" + payment +
                                      " <paymentMethod>" + paymentMethod + " <complete>" +
complete + " <amountPaid>" + amountPaid + " <date>" + week)
            objContactsFile.Close()
        End If
    End Sub

    Public Sub SavePayment(ByVal invoiceID As String, ByVal amountPaid As String)
        Dim objContactsFile As StreamWriter
        Try
            strFileName = "Payments.txt"

            objContactsFile = File.AppendText(strFileName)

            objContactsFile.WriteLine("<invoiceID>" + invoiceID + " <amountPaid>" +
amountPaid)

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
        objContactsFile.Close()
    End Sub

    Public Sub LoadPayments(ByRef payments As Collection)
        Dim objLoadPaymentsFile As StreamReader
```

```
        Try
            objLoadPaymentsFile = File.OpenText("Payments.txt")
            Dim i As Integer
            i = 1
            While objLoadPaymentsFile.Peek <> -1
                payments.Add(objLoadPaymentsFile.ReadLine(), i)
                i = i + 1
            End While

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
        objLoadPaymentsFile.Close()
    End Sub

    Public Sub LoadInvoice(ByRef oneTimeInvoiceList As Collection, ByRef
regularInvoiceList As Collection)
        Dim objOneTimeInvoiceFile As StreamReader
        Dim objRegularInvoiceFile As StreamReader
        Try
            objOneTimeInvoiceFile = File.OpenText("OneTimeInvoices.txt")

            Dim i As Integer
            i = 1
            While objOneTimeInvoiceFile.Peek <> -1
                oneTimeInvoiceList.Add(objOneTimeInvoiceFile.ReadLine(), i)
                i = i + 1
            End While
        Catch ex As Exception
            'MessageBox.Show(ex.Message, "Error")
        End Try
        objOneTimeInvoiceFile.Close()
        Try
            objRegularInvoiceFile = File.OpenText("RegularInvoices.txt")

            Dim i As Integer
            i = 1
            While objRegularInvoiceFile.Peek <> -1
                regularInvoiceList.Add(objRegularInvoiceFile.ReadLine(), i)
                i = i + 1
            End While

        Catch ex As Exception
            'MessageBox.Show(ex.Message, "Error")
        End Try
        objRegularInvoiceFile.Close()
    End Sub
    Public Sub LoadCustomer(ByRef customerList As Collection)
        Dim objCustomerFile As StreamReader
        Try
            objCustomerFile = File.OpenText("Customers.txt")

            Dim i As Integer
            i = 1
            While objCustomerFile.Peek <> -1
                customerList.Add(objCustomerFile.ReadLine(), i)
                i = i + 1
            End While
```

```
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
        objCustomerFile.Close()
    End Sub

    Public Sub LoadEmployee(ByRef employeeList As Collection)
        Dim objEmployeeFile As StreamReader
        Try
            objEmployeeFile = File.OpenText("Employees.txt")
            Dim i As Integer
            i = 1
            While objEmployeeFile.Peek <> -1
                employeeList.Add(objEmployeeFile.ReadLine(), i)
                i = i + 1
            End While

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
        objEmployeeFile.Close()
    End Sub

    Public Sub LoadJob(ByRef pendingJobList As Collection, ByRef completedJobList As
Collection)
        Dim objJobFile As StreamReader
        Try
            objJobFile = File.OpenText("PendingJobs.txt")
            Dim i As Integer
            i = 1
            While objJobFile.Peek <> -1
                pendingJobList.Add(objJobFile.ReadLine(), i)
                i = i + 1
            End While

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
        objJobFile.Close()
        Try
            objJobFile = File.OpenText("CompletedJobs.txt")
            Dim i As Integer
            i = 1
            While objJobFile.Peek <> -1
                completedJobList.Add(objJobFile.ReadLine(), i)
                i = i + 1
            End While

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
        objJobFile.Close()
    End Sub

    Public Sub LoadAppointment(ByRef appointmentList As Collection)
        Dim objAppointmentFile As StreamReader
        Try
```

```
            objAppointmentFile = File.OpenText("Appointments.txt")

            Dim i As Integer
            i = 1
            While objAppointmentFile.Peek <> -1
                appointmentList.Add(objAppointmentFile.ReadLine(), i)
                i = i + 1
            End While

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error")
        End Try
        objAppointmentFile.Close()
    End Sub

    Public Sub deleteItem(ByRef list As Collection, ByRef fileName As String)
        Dim objReplaceFile As StreamWriter
        File.Replace(fileName, "Clear.txt", "Dumpster.txt")
        Try
            objReplaceFile = File.AppendText(fileName)

            For Each item In list
                objReplaceFile.WriteLine(item)
            Next

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error Message")
        End Try
        objReplaceFile.Close()
    End Sub
End Class
```

CustomerInfoForm

```
Public Class CustomerInfoForm
    Dim objFileManager As New clsFileManager
    Dim customerList As New Collection

    'Closes form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub btnCIBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCIBack.Click
        Me.Close()
    End Sub

    Private Sub btnEdit_Click(sender As Object, e As EventArgs) Handles btnEdit.Click
        CustomerEditForm.txtName.Text = lblCIName.Text
        CustomerEditForm.txtAddress.Text = lblCIAddress.Text
        CustomerEditForm.txtPhone.Text = lblCIPhone.Text
        CustomerEditForm.txtEmail.Text = lblCIEmail.Text

        CustomerEditForm.Show()
```

```
    End Sub
    'Deletes Customer from database. Manager action only. (prompt verification)
    Private Sub btnCIDelete_Click_1(sender As Object, e As EventArgs) Handles
btnCIDelete.Click
        Dim temp As String
        Dim tempName As String
        Dim counter As Integer
        counter = 1

        objFileManager.LoadCustomer(customerList)

        For Each cust In customerList
            temp = cust
            temp = temp.Substring(6)                    'Removes beginning <name> tags
            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempName = lblCIName.Text) Then
                customerList.Remove(counter)
            End If
            counter = counter + 1
        Next
        objFileManager.deleteItem(customerList, "Customers.txt")
        MessageBox.Show("Customer Deleted", "Remove Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        Me.Close()
    End Sub
End Class
```

CustomerEditForm

```
Public Class CustomerEditForm
    Dim objFileManager As New clsFileManager
    Dim customerList As New Collection

    Private Sub BackToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub btnBack_Click(sender As Object, e As EventArgs) Handles btnBack.Click
        Me.Close()
    End Sub

    Private Sub btnSave_Click(sender As Object, e As EventArgs) Handles btnSave.Click
        Dim temp As String
        Dim tempName As String
        Dim counter As Integer = 1

        Dim phoneCorrect As Boolean = False
        Dim nameCorrect As Boolean = False
        Dim fieldsFilled As Boolean = False
        Dim phoneNum As String = txtPhone.Text

        Dim objCheck As New clsCheck

        phoneNum = objCheck.editPhone(phoneNum)
        phoneCorrect = objCheck.checkPhone(phoneNum)
        nameCorrect = objCheck.checkName(txtName.Text)
```

```
        fieldsFilled = checkFieldsFilled()

        If phoneCorrect = True And nameCorrect = True And fieldsFilled = True Then
            objFileManager.LoadCustomer(customerList)
            For Each cust In customerList
                temp = cust
                temp = temp.Substring(6)                    'Removes beginning <name>
tags
                tempName = temp.Substring(0, temp.IndexOf("<"))

                If (tempName = CustomerInfoForm.lblCIName.Text) Then
                    customerList.Remove(counter)
                End If
                counter = counter + 1
            Next
            customerList.Add("<name>" + txtName.Text.Trim + " <address>" +
txtAddress.Text.Trim + " <phone>" + phoneNum.Trim + " <email>" + txtEmail.Text.Trim)

            objFileManager.deleteItem(customerList, "Customers.txt")

            MessageBox.Show("Customer Info Changed", "Change Successful",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            CustomerInfoForm.Close()
            Me.Close()
        End If
    End Sub
    'Checks to see if all the fields are filled out
    Private Function checkFieldsFilled() As Boolean

        If txtName.Text.Length = 0 Or txtAddress.Text.Length = 0 Or txtPhone.Text.Length
= 0 Or txtEmail.Text.Length = 0 Then
            MessageBox.Show("Please fill in all fields", "Missing information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function
End Class
```

EmployeeInfo

```
Public Class EmployeeInfo
    Dim objFileManager As New clsFileManager
    Dim employeeList As New Collection

    'Opens The Modify Employee Form
    Private Sub btnEIEdit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEIEdit.Click
        'need a form here or something.
        EmployeeEditForm.txtName.Text = lblEIName.Text
        EmployeeEditForm.txtAddress.Text = lblEIAddress.Text
        EmployeeEditForm.txtHome.Text = lblEIHome.Text
        EmployeeEditForm.txtMobile.Text = lblEIMobile.Text
        EmployeeEditForm.txtEmail.Text = lblEIEmail.Text
        EmployeeEditForm.txtNameEM.Text = lblEINameEM.Text
        EmployeeEditForm.txtPhoneEM.Text = lblEIPhoneEM.Text
        EmployeeEditForm.txtPosition.Text = lblEIPosition.Text
        EmployeeEditForm.lblPay.Text = lblEIWages.Text
```

```
        EmployeeEditForm.lblHead.Text = "Edit Info for: " + lblEIName.Text

        If lblEIPayType.Text = "Salary" Then
            EmployeeEditForm.radSalary.Checked = True
            EmployeeEditForm.txtSalary.Text = lblEIWages.Text
        ElseIf lblEIPayType.Text = "Commission" Then
            EmployeeEditForm.radCommission.Checked = True
            EmployeeEditForm.txtCommWages.Text = lblEIWages.Text
        ElseIf lblEIPayType.Text = "Hourly" Then
            EmployeeEditForm.radHourly.Checked = True
            EmployeeEditForm.txtHourly.Text = lblEIWages.Text
        End If

        EmployeeEditForm.Show()

    End Sub
    'Close Form
    Private Sub btnEIBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEIBack.Click
        Me.Close()
    End Sub
    'Close Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    'Deletes the Employee from database. Manager action only
    Private Sub btnRemove_Click_1(sender As Object, e As EventArgs) Handles
btnRemove.Click
        Dim temp As String
        Dim tempName As String
        Dim counter As Integer
        counter = 1

        objFileManager.LoadEmployee(employeeList)

        For Each empl In employeeList
            temp = empl
            temp = temp.Substring(6)                    'Removes beginning <name> tags
            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempName = lblEIName.Text) Then
                employeeList.Remove(counter)
            End If
            counter = counter + 1
        Next
        objFileManager.deleteItem(employeeList, "Employees.txt")
        MessageBox.Show("Employee deleted", "Delete Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        Me.Close()
    End Sub
End Class
```

EmployeeEditForm

```
Public Class EmployeeEditForm
    Dim objFileManager As New clsFileManager
```

```
    Dim employeeList As New Collection
    'closes form
    Private Sub btnBack_Click(sender As Object, e As EventArgs) Handles btnBack.Click
        Me.Close()
    End Sub
    'set lblPay to "Hourly rate:" & set txtCommWage, lblCommission, and txtCommission to
visible
    Private Sub radNECommission_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radCommission.CheckedChanged
        txtCommWages.Visible = True
        lblPay.Text = "Commission:"
        lblCommWages.Visible = True
        lblSalaryWages.Visible = False
        lblHourlyWages.Visible = False
        txtHourly.Visible = False
        txtSalary.Visible = False
    End Sub
    'set lblPay to "Hourly rate:" & set txtHourly to visible
    Private Sub radNEHourly_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radHourly.CheckedChanged
        txtCommWages.Visible = False
        txtHourly.Visible = True
        lblHourlyWages.Visible = True
        lblCommWages.Visible = False
        lblSalaryWages.Visible = False
        txtHourly.Focus()
        txtSalary.Visible = False
        lblPay.Text = "Hourly rate:"

    End Sub

    Private Sub radSalary_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles radSalary.CheckedChanged
        txtCommWages.Visible = False
        lblCommWages.Visible = False
        lblHourlyWages.Visible = False
        lblSalaryWages.Visible = True
        txtHourly.Visible = False
        txtSalary.Visible = True
        txtSalary.Focus()
        lblPay.Text = "Salary:"

    End Sub
    'closes form
    Private Sub BackToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
BackToolStripMenuItem.Click
        Me.Close()
    End Sub
    'Removes old item from collection and adds modified info to collection. updates
textfile
    Private Sub btnSubmit_Click(sender As Object, e As EventArgs) Handles btnSubmit.Click
        Dim temp As String
        Dim tempName As String
        Dim counter As Integer
        Dim employeeNameCorrect As Boolean = False 'Employee Name
        Dim homePhoneCorrect As Boolean = False 'Home Phone
        Dim cellPhoneCorrect As Boolean = False 'Cell Phone
        Dim emPhoneCorrect As Boolean = False 'Emergency Contact Phone Number
```

```
        Dim emNameCorrect As Boolean = False 'Emergency Contact Name
        Dim fieldsFilled As Boolean = False
        'Phone Numbers
        Dim emPhone As String
        Dim homePhone As String
        Dim cellPhone As String

        Dim objCheck As New clsCheck

        employeeNameCorrect = objCheck.checkName(txtName.Text)
        emNameCorrect = objCheck.checkName(txtNameEM.Text)

        emPhone = objCheck.editPhone(txtPhoneEM.Text)
        emPhoneCorrect = objCheck.checkPhone(emPhone)
        homePhone = objCheck.editPhone(txtHome.Text)
        homePhoneCorrect = objCheck.checkPhone(homePhone)
        cellPhone = objCheck.editPhone(txtMobile.Text)
        cellPhoneCorrect = objCheck.checkPhone(cellPhone)
        fieldsFilled = checkFieldsFilled()


        counter = 1
        If employeeNameCorrect = True And homePhoneCorrect = True And cellPhoneCorrect =
True And emPhoneCorrect = True And
    emNameCorrect = True And fieldsFilled = True Then
            objFileManager.LoadEmployee(employeeList)
            For Each empl In employeeList
                temp = empl
                temp = temp.Substring(6)                    'Removes beginning <name>
tags
                tempName = temp.Substring(0, temp.IndexOf("<"))

                If (tempName = EmployeeInfo.lblEIName.Text) Then
                    employeeList.Remove(counter)
                End If
                counter = counter + 1
            Next

            If radSalary.Checked = True Then
                employeeList.Add("<name>" + txtName.Text.Trim + " <address>" +
txtAddress.Text.Trim + " <home>" + txtHome.Text.Trim + " <mobile>" + txtMobile.Text.Trim
+
                                 " <email>" + txtEmail.Text.Trim + " <nameEM>" +
txtNameEM.Text.Trim + " <phone>" + txtPhoneEM.Text.Trim + " <position>" +
txtPosition.Text.Trim +
                                 " <payType>" + "Salary" + " <pay>" +
txtSalary.Text.Trim)
            ElseIf radCommission.Checked = True Then
                employeeList.Add("<name>" + txtName.Text.Trim + " <address>" +
txtAddress.Text.Trim + " <home>" + txtHome.Text.Trim + " <mobile>" + txtMobile.Text.Trim
+
                                    " <email>" + txtEmail.Text.Trim + " <nameEM>" +
txtNameEM.Text.Trim + " <phone>" + txtPhoneEM.Text.Trim + " <position>" +
txtPosition.Text.Trim +
                                    " <payType>" + "Commission" + " <pay>" +
txtCommWages.Text.Trim)
            ElseIf radHourly.Checked = True Then
```

```
                employeeList.Add("<name>" + txtName.Text.Trim + " <address>" +
txtAddress.Text.Trim + " <home>" + txtHome.Text.Trim + " <mobile>" + txtMobile.Text.Trim
+
                                       " <email>" + txtEmail.Text.Trim + " <nameEM>" +
txtNameEM.Text.Trim + " <phone>" + txtPhoneEM.Text.Trim + " <position>" +
txtPosition.Text.Trim +
                                       " <payType>" + "Hourly" + " <pay>" +
txtHourly.Text.Trim)
            End If

            objFileManager.deleteItem(employeeList, "Employees.txt")

            MessageBox.Show("Employee Information Changed", "Change Successful",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            EmployeeInfo.Close()
            Me.Close()

        End If
    End Sub

    Private Function checkFieldsFilled() As Boolean
        If txtAddress.Text = "" Or txtName.Text = "" Or txtEmail.Text = "" Or
txtHome.Text = "" Or txtMobile.Text = "" Or txtNameEM.Text = "" Or txtPhoneEM.Text = ""
Or txtPosition.Text = "" Then
            MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False

        ElseIf radCommission.Checked = True Then
            If txtCommWages.Text = "" Then
                MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                Return False
            End If
        ElseIf radHourly.Checked = True Then
            If txtHourly.Text = "" Then
                MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                Return False
            End If
        ElseIf radSalary.Checked = True Then
            If txtSalary.Text = "" Then
                MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                Return False
            End If
        ElseIf radCommission.Checked = False And radHourly.Checked = False And
radSalary.Checked = False Then
            MessageBox.Show("Fill in all fields please", "Missing Information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function

End Class
```

JobViewForm

```
Public Class JobViewForm
    Dim objFileManager As New clsFileManager
    Dim jobPendingList As New Collection
    Dim jobCompleteList As New Collection
    Dim jobList As New clsJobListManager

    Private Sub BackToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub JobViewForm_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        If lblEmployee1.Text = " " Then
            lblEmployee1.Text = "N/A"
        End If
        If lblEmployee2.Text = " " Then
            lblEmployee2.Text = "N/A"
        End If
        If lblEmployee3.Text = " " Then
            lblEmployee3.Text = "N/A"
        End If
        If lblTitle1.Text = " " Then
            lblTitle1.Text = "N/A"
        End If
        If lblTitle2.Text = " " Then
            lblTitle2.Text = "N/A"
        End If
        If lblTitle3.Text = "" Then
            lblTitle3.Text = "N/A"
        End If
    End Sub

    Private Sub btnBack_Click(sender As Object, e As EventArgs) Handles btnBack.Click
        Me.Close()

    End Sub

    Private Sub btnEdit_Click(sender As Object, e As EventArgs) Handles btnEdit.Click
        JobEditForm.lblJEName.Text = lblName.Text
        JobEditForm.txtJEName.Text = lblName.Text
        JobEditForm.txtJEClient.Text = lblClient.Text
        JobEditForm.txtJELocation.Text = lblLocation.Text
        JobEditForm.txtJEStart.Text = lblStart.Text
        JobEditForm.txtDeadline.Text = lblDeadline.Text
        JobEditForm.txtJEDescription.Text = lblDescription.Text
        JobEditForm.txtJELead.Text = lblLeadName.Text
        JobEditForm.txtJETitleLead.Text = lblLeadTitle.Text

        If lblEmployee1.Text = "N/A" Then
            JobEditForm.txtJEEmployee1.Text = ""
        Else
            JobEditForm.txtJEEmployee1.Text = lblEmployee1.Text
        End If

        If lblEmployee2.Text = "N/A" Then
            JobEditForm.txtJEEmployee2.Text = ""
```

```
        Else
            JobEditForm.txtJEEmployee2.Text = lblEmployee2.Text
        End If

        If lblEmployee3.Text = "N/A" Then
            JobEditForm.txtJEEmployee3.Text = ""
        Else
            JobEditForm.txtJEEmployee3.Text = lblEmployee3.Text
        End If

        If lblTitle1.Text = "N/A" Then
            JobEditForm.txtJETitle1.Text = ""
        Else
            JobEditForm.txtJETitle1.Text = lblTitle1.Text
        End If

        If lblTitle2.Text = "N/A" Then
            JobEditForm.txtJETitle2.Text = ""
        Else
            JobEditForm.txtJETitle2.Text = lblTitle2.Text
        End If

        If lblTitle3.Text = "N/A" Then
            JobEditForm.txtJETitle3.Text = ""
        Else
            JobEditForm.txtJETitle3.Text = lblTitle3.Text
        End If

        'open form
        JobEditForm.Show()

    End Sub

    Private Sub btnChangeStatus_Click(sender As Object, e As EventArgs) Handles
btnChangeStatus.Click
        Dim temp As String
        Dim tempName As String
        Dim counter As Integer

        Dim emp1 As String
        Dim emp2 As String
        Dim emp3 As String
        Dim title1 As String
        Dim title2 As String
        Dim title3 As String

        'set these variables to those labels
        emp1 = lblEmployee1.Text
        emp2 = lblEmployee2.Text
        emp3 = lblEmployee3.Text
        title1 = lblTitle1.Text
        title2 = lblTitle2.Text
        title3 = lblTitle3.Text

        'set variables to empty if nothing is entered
        If lblEmployee1.Text = "N/A" Then
            emp1 = ""
        End If
```

```
        If lblEmployee2.Text = "N/A" Then
            emp2 = ""
        End If
        If lblEmployee3.Text = "N/A" Then
            emp3 = ""
        End If
        If lblTitle1.Text = "N/A" Then
            title1 = ""
        End If
        If lblTitle2.Text = "N/A" Then
            title2 = ""
        End If
        If lblTitle3.Text = "N/A" Then
            title3 = ""
        End If

        counter = 1
        objFileManager.LoadJob(jobPendingList, jobCompleteList)
        If lblStatus.Text = "Pending" Then
            For Each job In jobPendingList
                temp = job
                temp = temp.Substring(6)                    'Removes beginning <name>
tags
                tempName = temp.Substring(0, temp.IndexOf("<"))
                'MessageBox.Show(tempName)
                If (tempName = lblName.Text) Then
                    jobCompleteList.Add(jobPendingList.Item(counter))

                    'remove job from pending list
                    jobPendingList.Remove(counter)
                    'MessageBox.Show(jobPendingList.Item(counter))

                    lblStatus.Text = "Completed"
                    Exit For
                End If
                counter = counter + 1
            Next


            objFileManager.deleteItem(jobPendingList, "PendingJobs.txt")
            objFileManager.deleteItem(jobCompleteList, "CompletedJobs.txt")
            MessageBox.Show("Status changed to Completed", "Status Changed",
MessageBoxButtons.OK, MessageBoxIcon.Information)

        ElseIf lblStatus.Text = "Completed" Then
            For Each job2 In jobCompleteList
                temp = job2
                temp = temp.Substring(6)                    'Removes beginning <name>
tags
                tempName = temp.Substring(0, temp.IndexOf("<"))

                If (tempName = lblName.Text) Then
                    jobPendingList.Add(jobCompleteList.Item(counter))

                    'remove job from pending list
                    jobCompleteList.Remove(counter)

                    lblStatus.Text = "Pending"
```

```
                    Exit For
                End If
                counter = counter + 1
            Next
            'MessageBox.Show(jobPendingList.Item(counter))
            objFileManager.deleteItem(jobCompleteList, "CompletedJobs.txt")
            objFileManager.deleteItem(jobPendingList, "PendingJobs.txt")
            MessageBox.Show("Status changed to Pending", "Status Changed",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        End If
        Me.Close()
    End Sub

    Private Sub btnJEDelete_Click(sender As Object, e As EventArgs) Handles
btnJEDelete.Click
        Dim temp As String
        Dim tempName As String
        Dim counter As Integer
        counter = 1

        objFileManager.LoadJob(jobPendingList, jobCompleteList)

        For Each pend In jobPendingList
            temp = pend
            temp = temp.Substring(6)                    'Removes beginning <name> tags
            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempName = lblName.Text) Then
                jobPendingList.Remove(counter)
            End If
            counter = counter + 1
        Next
        objFileManager.deleteItem(jobPendingList, "PendingJobs.txt")
        counter = 1
        For Each comp In jobCompleteList
            temp = comp
            temp = temp.Substring(6)                    'Removes beginning <name> tags
            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempName = lblName.Text) Then
                jobCompleteList.Remove(counter)
            End If
            counter = counter + 1
        Next
        objFileManager.deleteItem(jobCompleteList, "CompletedJobs.txt")

        MessageBox.Show("Job Deleted", "Remove Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        Me.Close()
    End Sub
End Class
```

JobEditForm

```
Public Class JobEditForm

    Dim objFileManager As New clsFileManager
    Dim jobPendingList As New Collection
```

```
    Dim jobCompletedList As New Collection
    'Prints Job information form
    Private Sub PrintToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)

    End Sub
    'Closes form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub
    'Closes Form
    Private Sub btnJEBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnJEBack.Click
        Me.Close()
    End Sub
    'Deletes the job from the database. Manager action only
    Private Sub btnJEDelete_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)

    End Sub

    Private Sub btnSubmit_Click(sender As Object, e As EventArgs) Handles btnSubmit.Click
        Dim temp As String
        Dim tempName As String
        Dim counter As Integer

        Dim objCheck As New clsCheck
        Dim dateStart As String
        Dim dateEnd As String

        Dim dateStartCorrect As Boolean = False
        Dim dateEndCorrect As Boolean = False
        Dim fieldsFilled As Boolean = False

        dateStart = objCheck.editDate(txtJEStart.Text)
        dateStartCorrect = objCheck.checkDate(dateStart)
        dateEnd = objCheck.editDate(txtDeadline.Text)
        dateEndCorrect = objCheck.checkDate(dateEnd)
        fieldsFilled = checkFieldsFilled()
        counter = 1

        If dateStartCorrect = True And dateEndCorrect = True And fieldsFilled = True Then
            objFileManager.LoadJob(jobPendingList, jobCompletedList)
            If JobViewForm.lblStatus.Text = "Pending" Then
                For Each job In jobPendingList
                    temp = job
                    temp = temp.Substring(6)                    'Removes beginning <name>
tags
                    tempName = temp.Substring(0, temp.IndexOf("<"))

                    If (tempName = JobViewForm.lblName.Text) Then
                        jobPendingList.Remove(counter)
                    End If
                    counter = counter + 1
                Next
                jobPendingList.Add("<name>" + txtJEName.Text.Trim + " <client>" +
txtJEClient.Text.Trim + " <location>" + txtJELocation.Text.Trim + " <start>" +
```

```
dateStart.Trim + " <deadLine>" + dateEnd.Trim + " <description>" +
txtJEDescription.Text.Trim + " <lead>" + txtJELead.Text.Trim +
                                    " <employee1>" + txtJEEmployee1.Text.Trim + "
<employee2>" + txtJEEmployee2.Text.Trim + " <employee3>" + txtJEEmployee3.Text.Trim +
"<leadTitle>" + txtJETitleLead.Text.Trim + " <title1>" + txtJETitle1.Text.Trim + "
<title2>" + txtJETitle2.Text.Trim + " <title3>" + txtJETitle3.Text.Trim)

                objFileManager.deleteItem(jobPendingList, "PendingJobs.txt")

                MessageBox.Show("The Job has been edited successfully", "Change
Successful", MessageBoxButtons.OK, MessageBoxIcon.Information)
                JobViewForm.Close()
                Me.Close()
            ElseIf JobViewForm.lblStatus.Text = "Completed" Then
                For Each job2 In jobCompletedList
                    temp = job2
                    temp = temp.Substring(6)                    'Removes beginning <name>
tags
                    tempName = temp.Substring(0, temp.IndexOf("<"))

                    If (tempName = JobViewForm.lblName.Text) Then
                        jobCompletedList.Remove(counter)
                    End If
                    counter = counter + 1
                Next
                jobCompletedList.Add("<name>" + txtJEName.Text.Trim + " <client>" +
txtJEClient.Text.Trim + " <location>" + txtJELocation.Text.Trim + " <start>" +
dateStart.Trim + " <deadLine>" + dateEnd.Trim + " <description>" +
txtJEDescription.Text.Trim + " <lead>" + txtJELead.Text.Trim +
                                    " <employee1>" + txtJEEmployee1.Text.Trim + "
<employee2>" + txtJEEmployee2.Text.Trim + " <employee3>" + txtJEEmployee3.Text.Trim +
"<leadTitle>" + txtJETitleLead.Text.Trim + " <title1>" + txtJETitle1.Text.Trim + "
<title2>" + txtJETitle2.Text.Trim + " <title3>" + txtJETitle3.Text.Trim)

                objFileManager.deleteItem(jobCompletedList, "CompletedJobs.txt")

                MessageBox.Show("The Job has been edited successfully", "Change
Successful", MessageBoxButtons.OK, MessageBoxIcon.Information)
                JobViewForm.Close()
                Me.Close()
            End If
        End If
    End Sub
    Private Function checkFieldsFilled() As Boolean
        If txtJEClient.Text = "" Or txtDeadline.Text = "" Or txtJEDescription.Text = ""
Or txtJELead.Text = "" Or
            txtJETitleLead.Text = "" Or txtJELocation.Text = "" Or txtJEName.Text = "" Or
txtJEStart.Text = "" Then
            MessageBox.Show("Please fill in all fields", "Missing information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function
End Class
```

InvoiceForm

```
Public Class InvoiceForm
    'Opens AddNew 1-Time Invoice Form
    Private Sub btnNew1TInvoice_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNew1TInvoice.Click
        NewInvoiceForm.Show()
        NewInvoiceForm.radNIOneTime.Checked = True
        lblStatus.Visible = False
        lblOTShow.Visible = False
    End Sub
    'create input box to search for invoice name
    Private Sub btnInv1TShow_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnInv1TShow.Click
        Dim objInvoiceFile As New clsFileManager
        Dim OneTimeInvoiceList As New Collection
        Dim RegularInvoiceList As New Collection
        Dim payments As New Collection
        Dim tempPay As String
        Dim tempInvID As String
        Dim tempPaid As String
        Dim totalPaid As Double
        Dim temp As String          'temporary string
        Dim tempID As String
        Dim tempName As String 'Hold the testing customer name for search function
        Dim strInvoiceID As String
        objInvoiceFile.LoadInvoice(OneTimeInvoiceList, RegularInvoiceList)
        objInvoiceFile.loadPayments(payments)
        totalPaid = 0

        strInvoiceID = InputBox("Enter the Invoice ID you want to search for.", "Invoice
ID")

        For Each inv In OneTimeInvoiceList
            temp = inv

            temp = temp.Substring(4)                    'Removes beginning <name> tags
            tempID = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)
            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempID.Trim = strInvoiceID.Trim) Then
                'Set contents
                InvoiceInfoForm.lblIIHeading.Text = tempName
                InvoiceInfoForm.lblIIName.Text = tempName
                InvoiceInfoForm.lblID.Text = tempID

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)

                InvoiceInfoForm.lblIIType.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)

                InvoiceInfoForm.lblIIEmployee.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
```

```
                temp = temp.Substring(9)

                InvoiceInfoForm.lblIIPayment.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(15)

                InvoiceInfoForm.lblIIMethod.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)

                InvoiceInfoForm.lblIICompleteDate.Text = temp.Substring(0,
temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(12)

                For Each pay In payments
                    tempPay = pay
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempPaid = tempPay.Substring(0).Trim
                    If tempInvID = tempID Then
                        totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
                    End If
                Next

                InvoiceInfoForm.lblIIAmountPaid.Text = totalPaid

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)

                InvoiceInfoForm.lblIIWeek.Text = temp
                InvoiceInfoForm.Show()
                Exit Sub
            End If
        Next
        For Each inv In RegularInvoiceList
            temp = inv
            temp = temp.Substring(4)

            tempID = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)                    'Removes beginning <name> tags

            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempID.Trim = strInvoiceID.Trim) Then
                'Set contents
                InvoiceInfoForm.lblIIHeading.Text = tempName
                InvoiceInfoForm.lblIIName.Text = tempName
                InvoiceInfoForm.lblID.Text = tempID

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)
```

```
                InvoiceInfoForm.lblIIType.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)

                InvoiceInfoForm.lblIIEmployee.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(9)

                InvoiceInfoForm.lblIIPayment.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(15)

                InvoiceInfoForm.lblIIMethod.Text = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)

                InvoiceInfoForm.lblIICompleteDate.Text = temp.Substring(0,
temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(12)

                For Each pay In payments
                    tempPay = pay
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempPaid = tempPay.Substring(0).Trim
                    If tempInvID = tempID Then
                        totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
                    End If
                Next

                InvoiceInfoForm.lblIIAmountPaid.Text = totalPaid

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)

                InvoiceInfoForm.lblIIWeek.Text = temp
                InvoiceInfoForm.Show()
                Exit Sub
            End If
        Next
        MessageBox.Show("There is no invoice by that name in the database.", "Invoice Not
Found", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Sub

    'search database based on info entered in input box
    Private Sub btnInv1TStatus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnInv1TStatus.Click
        Dim objInvoiceFile As New clsFileManager
        Dim OneTimeInvoiceList As New Collection
        Dim RegularInvoiceList As New Collection
```

```
        Dim payments As New Collection
        Dim temp As String          'temporary string
        Dim tempName As String      'Hold the testing customer name for search function
        Dim dblAmountPaid As Double
        Dim dblPayment As Double
        Dim strInvoiceID As String
        Dim tempID As String
        Dim totalPaid As Double
        Dim tempPay As String
        Dim tempInvID As String
        Dim tempPaid As String

        objInvoiceFile.LoadInvoice(OneTimeInvoiceList, RegularInvoiceList)
        objInvoiceFile.loadPayments(payments)

        strInvoiceID = InputBox("Enter the Invoice ID number you want to search for.",
"Invoice ID")

        lblStatus.Visible = True
        lblOTShow.Visible = True

        For Each inv In OneTimeInvoiceList
            temp = inv
            temp = temp.Substring(4)
            tempID = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)                    'Removes beginning <name> tags
            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempID.ToLower().Trim = strInvoiceID.Trim) Then

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(9)

                dblPayment = temp.Substring(0, temp.IndexOf("<"))

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(15)

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(10)

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(12)

                For Each pay In payments
                    tempPay = pay
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempPaid = tempPay.Substring(0).Trim
```

```
            If tempInvID = tempID Then
                totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
            End If
        Next
        dblAmountPaid = totalPaid

        If dblPayment - dblAmountPaid <= 0 Then
            lblOTShow.Text = "Paid"
        Else
            lblOTShow.Text = "Not Paid"
        End If

        Exit Sub
    End If
Next
For Each inv In RegularInvoiceList
    temp = inv
    temp = temp.Substring(4)
    tempID = temp.Substring(0, temp.IndexOf("<"))

    temp = temp.Substring(temp.IndexOf("<"))
    temp = temp.Substring(6)                    'Removes beginning <name> tags
    tempName = temp.Substring(0, temp.IndexOf("<"))

    If (tempID.Trim = strInvoiceID.Trim) Then

        temp = temp.Substring(temp.IndexOf("<"))
        temp = temp.Substring(6)

        temp = temp.Substring(temp.IndexOf("<"))
        temp = temp.Substring(10)

        temp = temp.Substring(temp.IndexOf("<"))
        temp = temp.Substring(9)

        dblPayment = temp.Substring(0, temp.IndexOf("<"))

        temp = temp.Substring(temp.IndexOf("<"))
        temp = temp.Substring(15)

        temp = temp.Substring(temp.IndexOf("<"))
        temp = temp.Substring(10)

        temp = temp.Substring(temp.IndexOf("<"))
        temp = temp.Substring(12)

        For Each pay In payments
            tempPay = pay
            tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
            tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
            tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
            tempPaid = tempPay.Substring(0).Trim
            If tempInvID = tempID Then
                totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
            End If
        Next
        dblAmountPaid = totalPaid
```

```
                If dblPayment - dblAmountPaid = 0 Then
                    lblOTShow.Text = "Paid"
                Else
                    lblOTShow.Text = "Not Paid"
                End If

                Exit Sub
            End If
        Next
        MessageBox.Show("There is no invoice by that ID in the database.", "Invoice Not
Found", MessageBoxButtons.OK, MessageBoxIcon.Error)

    End Sub

    'Closes InvoiceForm
    Private Sub btnInvBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnInvBack.Click
        Me.Close()
    End Sub

    'Closes Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub btnInvShowAll_Click(sender As Object, e As EventArgs) Handles
btnInvShowAll.Click
        InvoiceListForm.Show()
    End Sub
End Class
```

InvoiceListForm

```
Public Class InvoiceListForm
    Dim objFileManager As New clsFileManager
    Dim OneTimeInvoiceList As New Collection
    Dim RegularInvoiceList As New Collection

    Private Sub radOne_CheckedChanged(sender As Object, e As EventArgs) Handles
radOne.CheckedChanged
        objFileManager.LoadInvoice(OneTimeInvoiceList, RegularInvoiceList)

        lstComplete.Items.Clear()
        lstCustomerName.Items.Clear()
        lstID.Items.Clear()
        lstMethod.Items.Clear()
        lstPaid.Items.Clear()
        lstPayment.Items.Clear()

        Dim i As Integer
        Dim counter As Integer
        Dim temp As String
        Dim tempType As String
        Dim tempName As String
        Dim totalPaid As Double
        Dim tempID As String
        Dim tempPay As String
```

```
        Dim tempInvID As String
        Dim tempPaid As String
        Dim payments As New Collection

        objFileManager.loadPayments(payments)

        i = 1
        counter = OneTimeInvoiceList.Count
        While counter > 0
            tempType = OneTimeInvoiceList.Item(i)
            tempType = tempType.Substring(6)                    'Removes beginning <name>
tags
            tempName = tempType.Substring(0, tempType.IndexOf("<"))

            tempType = tempType.Substring(tempType.IndexOf("<"))
            tempType = tempType.Substring(6) 'removes <type> tags
            tempType = tempType.Substring(0, tempType.IndexOf("<"))

            'Set contents
            temp = OneTimeInvoiceList.Item(i)

            temp = temp.Substring(4)
            tempID = temp.Substring(0, temp.IndexOf("<"))
            lstID.Items.Add(tempID)

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)
            lstCustomerName.Items.Add(temp.Substring(0, temp.IndexOf("<")))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)

            'Do nothing for type

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(10)
            'not pulling employees into list anymore
            'lstEmployee.Items.Add(temp.Substring(0, temp.IndexOf("<")))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(9)

            lstPayment.Items.Add(temp.Substring(0, temp.IndexOf("<")))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(15)

            lstMethod.Items.Add(temp.Substring(0, temp.IndexOf("<")))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(10)

            lstComplete.Items.Add(temp.Substring(0, temp.IndexOf("<")))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(12)

            totalPaid = 0
```

```
            For Each pay In payments
                tempPay = pay
                tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                tempPaid = tempPay.Substring(0).Trim
                If tempInvID = tempID Then
                    totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
                End If
            Next

            lstPaid.Items.Add(totalPaid)

            counter = counter - 1
            i = i + 1
        End While
    End Sub

    Private Sub radRegular_CheckedChanged(sender As Object, e As EventArgs) Handles
radRegular.CheckedChanged
        objFileManager.LoadInvoice(OneTimeInvoiceList, RegularInvoiceList)

        lstComplete.Items.Clear()
        lstCustomerName.Items.Clear()
        lstID.Items.Clear()
        lstMethod.Items.Clear()
        lstPaid.Items.Clear()
        lstPayment.Items.Clear()

        Dim i As Integer
        Dim counter As Integer
        Dim temp As String
        Dim tempType As String
        Dim tempID As String
        Dim tempName As String
        Dim tempWeek As String
        Dim totalPaid As Double
        Dim tempPay As String
        Dim tempInvID As String
        Dim tempPaid As String
        Dim payments As New Collection

        objFileManager.loadPayments(payments)

            i = 1
        counter = RegularInvoiceList.Count
            While counter > 0
            tempWeek = RegularInvoiceList.Item(i)
            tempWeek = tempWeek.Substring(4)
            tempWeek = tempWeek.Substring(tempWeek.IndexOf("<"))

            tempWeek = tempWeek.Substring(6)                    'Removes beginning <name>
tags
                tempName = tempWeek.Substring(0, tempWeek.IndexOf("<"))

                tempWeek = tempWeek.Substring(tempWeek.IndexOf("<"))
                tempWeek = tempWeek.Substring(6) 'removes <type> tags
                tempType = tempWeek.Substring(0, tempWeek.IndexOf("<"))
```

```
        tempWeek = tempWeek.Substring(tempWeek.IndexOf("<"))
        tempWeek = tempWeek.Substring(10) 'removes <Employee> tags

        tempWeek = tempWeek.Substring(tempWeek.IndexOf("<"))
        tempWeek = tempWeek.Substring(9) 'removes <Payment> tags

        tempWeek = tempWeek.Substring(tempWeek.IndexOf("<"))
        tempWeek = tempWeek.Substring(15) 'removes <paymentMethod> tags

        tempWeek = tempWeek.Substring(tempWeek.IndexOf("<"))
        tempWeek = tempWeek.Substring(10) 'removes <complete> tags

        tempWeek = tempWeek.Substring(tempWeek.IndexOf("<"))
        tempWeek = tempWeek.Substring(12) 'removes <AmountPaid> tags

        tempWeek = tempWeek.Substring(tempWeek.IndexOf("<"))
        tempWeek = tempWeek.Substring(6) 'removes <week> tags
        tempWeek = tempWeek.Substring(0)

    'Set contents
    temp = RegularInvoiceList.Item(i)

    temp = temp.Substring(4)
    tempID = temp.Substring(0, temp.IndexOf("<"))
    lstID.Items.Add(tempID)

    temp = temp.Substring(temp.IndexOf("<"))
    temp = temp.Substring(6)
    lstCustomerName.Items.Add(temp.Substring(0, temp.IndexOf("<")))

    temp = temp.Substring(temp.IndexOf("<"))
    temp = temp.Substring(6)

    'Do nothing for type

    temp = temp.Substring(temp.IndexOf("<"))
    temp = temp.Substring(10)

    'not pulling employee anymore
    'lstEmployee.Items.Add(temp.Substring(0, temp.IndexOf("<")))

    temp = temp.Substring(temp.IndexOf("<"))
    temp = temp.Substring(9)

    lstPayment.Items.Add(temp.Substring(0, temp.IndexOf("<")))

    temp = temp.Substring(temp.IndexOf("<"))
    temp = temp.Substring(15)

    lstMethod.Items.Add(temp.Substring(0, temp.IndexOf("<")))

    temp = temp.Substring(temp.IndexOf("<"))
    temp = temp.Substring(10)

    lstComplete.Items.Add(temp.Substring(0, temp.IndexOf("<")))

    temp = temp.Substring(temp.IndexOf("<"))
```

```
            temp = temp.Substring(12)


            totalPaid = 0
            For Each pay In payments
                tempPay = pay
                tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                tempPaid = tempPay.Substring(0).Trim
                If tempInvID = tempID Then
                    totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
                End If
            Next

            lstPaid.Items.Add(totalPaid)

            counter = counter - 1
            i = i + 1
        End While
    End Sub

    Private Sub btnCLBack_Click(sender As Object, e As EventArgs) Handles btnCLBack.Click
        Me.Close()
    End Sub

    Private Sub BackToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
BackToolStripMenuItem.Click
        Me.Close()
    End Sub
End Class
```

InvoiceEditForm

```
Public Class InvoiceEditForm

    Private Sub InvoiceEditForm_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        If lblEIType.Equals("Regular") Then
            lblEIWeek.Visible = True
            txtEIWeek.Visible = True
        End If
    End Sub

    Private Sub btnSave_Click(sender As Object, e As EventArgs) Handles btnSave.Click
        Dim temp As String
        Dim tempName As String
        Dim counter As Integer = 1
        Dim objFileManager As New clsFileManager
        Dim OneTimeInvoiceList As New Collection
        Dim RegularInvoiceList As New Collection
        Dim tempID As String

        Dim fieldsFilled As Boolean = False
        Dim cNameCorrect As Boolean = False 'Customer Name
        Dim eNameCorrect As Boolean = False 'Employee Name
        Dim dateCorrect As Boolean = False
        Dim completeDateCorrect As Boolean = False
```

```
        Dim invCompleteDate As String
        Dim invDate As String
        Dim objCheck As New clsCheck

        invDate = objCheck.editDate(txtEIWeek.Text)
        dateCorrect = objCheck.checkDate(invDate)
        invCompleteDate = objCheck.editDate(txtEIComplete.Text)
        completeDateCorrect = objCheck.checkDate(invCompleteDate)
        cNameCorrect = objCheck.checkName(txtEIName.Text)
        eNameCorrect = objCheck.checkName(txtEIEmployee.Text)
        fieldsFilled = checkFieldsFilled()
        objFileManager.LoadInvoice(OneTimeInvoiceList, RegularInvoiceList)
        If cNameCorrect = True And eNameCorrect = True And dateCorrect = True And
completeDateCorrect = True And fieldsFilled = True Then
            If InvoiceInfoForm.lblIIType.Text.Trim = "One" Then
                For Each inv In OneTimeInvoiceList
                    temp = inv
                    temp = temp.Substring(4)
                    tempID = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(6)                          'Removes beginning
<name> tags
                    tempName = temp.Substring(0, temp.IndexOf("<"))

                    If (tempID = InvoiceInfoForm.lblID.Text) Then
                        OneTimeInvoiceList.Remove(counter)
                    End If
                    counter = counter + 1
                Next
                OneTimeInvoiceList.Add("<id>" + lblID.Text.Trim + " <name>" +
txtEIName.Text.Trim + " <type>" + lblEIType.Text.Trim + " <employee>" +
txtEIEmployee.Text.Trim + " <payment>" + txtEIPayment.Text.Trim +
                                        " <paymentMethod>" + cmbEIMethod.Text.Trim + "
<complete>" + invCompleteDate.Trim + " <amountPaid>" + txtEIPaid.Text.Trim + " <week>" +
invDate)

                objFileManager.deleteItem(OneTimeInvoiceList, "OneTimeInvoices.txt")
            ElseIf InvoiceInfoForm.lblIIType.Text.Trim = "Regular" Then
                For Each inv In RegularInvoiceList
                    temp = inv
                    temp = temp.Substring(4)
                    tempID = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(6)                          'Removes beginning
<name> tags
                    tempName = temp.Substring(0, temp.IndexOf("<"))

                    If (tempID = InvoiceInfoForm.lblID.Text) Then
                        RegularInvoiceList.Remove(counter)
                    End If
                    counter = counter + 1
                Next
                RegularInvoiceList.Add("<id>" + lblID.Text.Trim + " <name>" +
txtEIName.Text.Trim + " <type>" + lblEIType.Text.Trim + " <employee>" +
txtEIEmployee.Text.Trim + " <payment>" + txtEIPayment.Text.Trim +
```

```
                                        " <paymentMethod>" + cmbEIMethod.Text.Trim + "
<complete>" + invCompleteDate.Trim + " <amountPaid>" + txtEIPaid.Text.Trim + " <week>" +
invDate.Trim)

                objFileManager.deleteItem(RegularInvoiceList, "RegularInvoices.txt")
            End If

            MessageBox.Show("Invoice Edited Successfully", "Change Successful",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            InvoiceInfoForm.Close()
            Me.Close()
        End If
    End Sub
    Private Function checkFieldsFilled() As Boolean
        If txtEIWeek.Text.Length = 0 Or txtEIComplete.Text.Length = 0 Or
txtEIEmployee.Text.Length = 0 Or
            txtEIName.Text.Length = 0 Or txtEIPaid.Text.Length = 0 Or
txtEIPayment.Text.Length = 0 Or
            cmbEIMethod.Text.Length = 0 Then
            MessageBox.Show("Please fill in all fields", "Missing information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function

    Private Sub btnBack_Click(sender As Object, e As EventArgs) Handles btnBack.Click
        Me.Close()
    End Sub
End Class
```

InvoiceInfoForm

```
Public Class InvoiceInfoForm
    Dim objFileManager As New clsFileManager
    Dim OneTimeInvoiceList As New Collection
    Dim RegularInvoiceList As New Collection
    'Closes the form
    Private Sub btnSIBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnIIBack.Click
        Me.Close()
    End Sub
    'Closes the form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

    Private Sub btnSIDelete_Click(sender As Object, e As EventArgs) Handles
btnIIDelete.Click
        Dim temp As String
        Dim tempName As String
        Dim tempID As String
        Dim counter As Integer
        counter = 1

        objFileManager.LoadInvoice(OneTimeInvoiceList, RegularInvoiceList)
```

```
        For Each inv In OneTimeInvoiceList
            temp = inv
            temp = temp.Substring(4)

            tempID = temp.Substring(0, temp.IndexOf("<"))
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)                    'Removes beginning <name> tags
            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempID = lblID.Text) Then
                OneTimeInvoiceList.Remove(counter)
            End If
            counter = counter + 1
        Next
        objFileManager.deleteItem(OneTimeInvoiceList, "OneTimeInvoices.txt")
        counter = 1

        For Each inv In RegularInvoiceList
            temp = inv
            temp = temp.Substring(4)

            tempID = temp.Substring(0, temp.IndexOf("<"))
            temp = temp.Substring(6)                    'Removes beginning <name> tags
            tempName = temp.Substring(0, temp.IndexOf("<"))

            If (tempID = lblID.Text) Then
                RegularInvoiceList.Remove(counter)
            End If
            counter = counter + 1
        Next
        objFileManager.deleteItem(RegularInvoiceList, "RegularInvoices.txt")
        MessageBox.Show("Invoice Deleted Successfully", "Invoice Deleted",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        Me.Close()
    End Sub

    Private Sub btnSIEdit_Click(sender As Object, e As EventArgs) Handles btnIIEdit.Click
        InvoiceEditForm.lblID.Text = lblID.Text
        InvoiceEditForm.lblEIType.Text = lblIIType.Text
        InvoiceEditForm.txtEIWeek.Text = lblIIWeek.Text
        InvoiceEditForm.txtEIName.Text = lblIIName.Text
        InvoiceEditForm.txtEIEmployee.Text = lblIIEmployee.Text
        InvoiceEditForm.txtEIPayment.Text = lblIIPayment.Text
        InvoiceEditForm.txtEIPaid.Text = lblIIAmountPaid.Text
        InvoiceEditForm.cmbEIMethod.Text = lblIIMethod.Text
        InvoiceEditForm.txtEIComplete.Text = lblIICompleteDate.Text

        InvoiceEditForm.Show()
    End Sub
End Class
```

NewInvoiceForm

```
Public Class NewInvoiceForm
    Private strType As String
    Dim invoiceList As New clsInvoiceListManager
    Dim objFileManager As New clsFileManager
    'Closes the form
```

```
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub
    'Closes the form
    Private Sub btnNIBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNIBack.Click
        Me.Close()
    End Sub
    'Add the invoice to the appropriate database depending on the radio button selection
    Private Sub btnNISubmit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNISubmit.Click
        Dim fieldsFilled As Boolean = False
        Dim cNameCorrect As Boolean = False 'Customer Name
        Dim eNameCorrect As Boolean = False 'Employee Name
        Dim dateCorrect As Boolean = False
        Dim completeDateCorrect As Boolean = False
        Dim invCompleteDate As String
        Dim invDate As String
        Dim objCheck As New clsCheck

        If radNIRegular.Checked = True Then
            strType = "Regular"
        Else
            strType = "One" 'one time
        End If
        invDate = objCheck.editDate(txtNIWeek.Text)
        dateCorrect = objCheck.checkDate(invDate)
        invCompleteDate = objCheck.editDate(txtNIComplete.Text)
        completeDateCorrect = objCheck.checkDate(invCompleteDate)
        cNameCorrect = objCheck.checkName(txtNIName.Text)
        eNameCorrect = objCheck.checkName(txtNIEmployee.Text)
        fieldsFilled = checkFieldsFilled()

        If cNameCorrect = True And eNameCorrect = True And dateCorrect = True And
completeDateCorrect = True And fieldsFilled = True Then
            invoiceList.addInvoice(txtID.Text, strType, txtNIName.Text,
txtNIEmployee.Text, CDbl(txtNIPay.Text),
                                   cmbPaymentMethod.Text, invCompleteDate,
CDbl(txtNIPaid.Text), invDate)
            objFileManager.SaveInvoice(txtID.Text, strType, txtNIName.Text,
txtNIEmployee.Text, txtNIPay.Text,
                                   cmbPaymentMethod.Text, invCompleteDate, txtNIPaid.Text,
invDate)

            MessageBox.Show("Invoice Added", "Invoice Added", MessageBoxButtons.OK,
MessageBoxIcon.Information)
            Me.Close()
        End If
    End Sub

    Private Function checkFieldsFilled() As Boolean
        If txtNIWeek.Text.Length = 0 Or txtNIComplete.Text.Length = 0 Or
txtNIEmployee.Text.Length = 0 Or
            txtNIName.Text.Length = 0 Or txtNIPaid.Text.Length = 0 Or
txtNIPay.Text.Length = 0 Or
            cmbPaymentMethod.Text.Length = 0 Then
```

```
            MessageBox.Show("Please fill in all fields", "Missing information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function
End Class
```

Chapter 3- Iteration 3

The figure below shows our final class diagram, including all changes made for iteration three. Note the addition of the payments and schedule trees that connect on the bottom right (blue and yellow on the colored figure.) Also included, is the class clsCheck that is accessed by almost every form, and is used to check phone numbers, dates, etc.

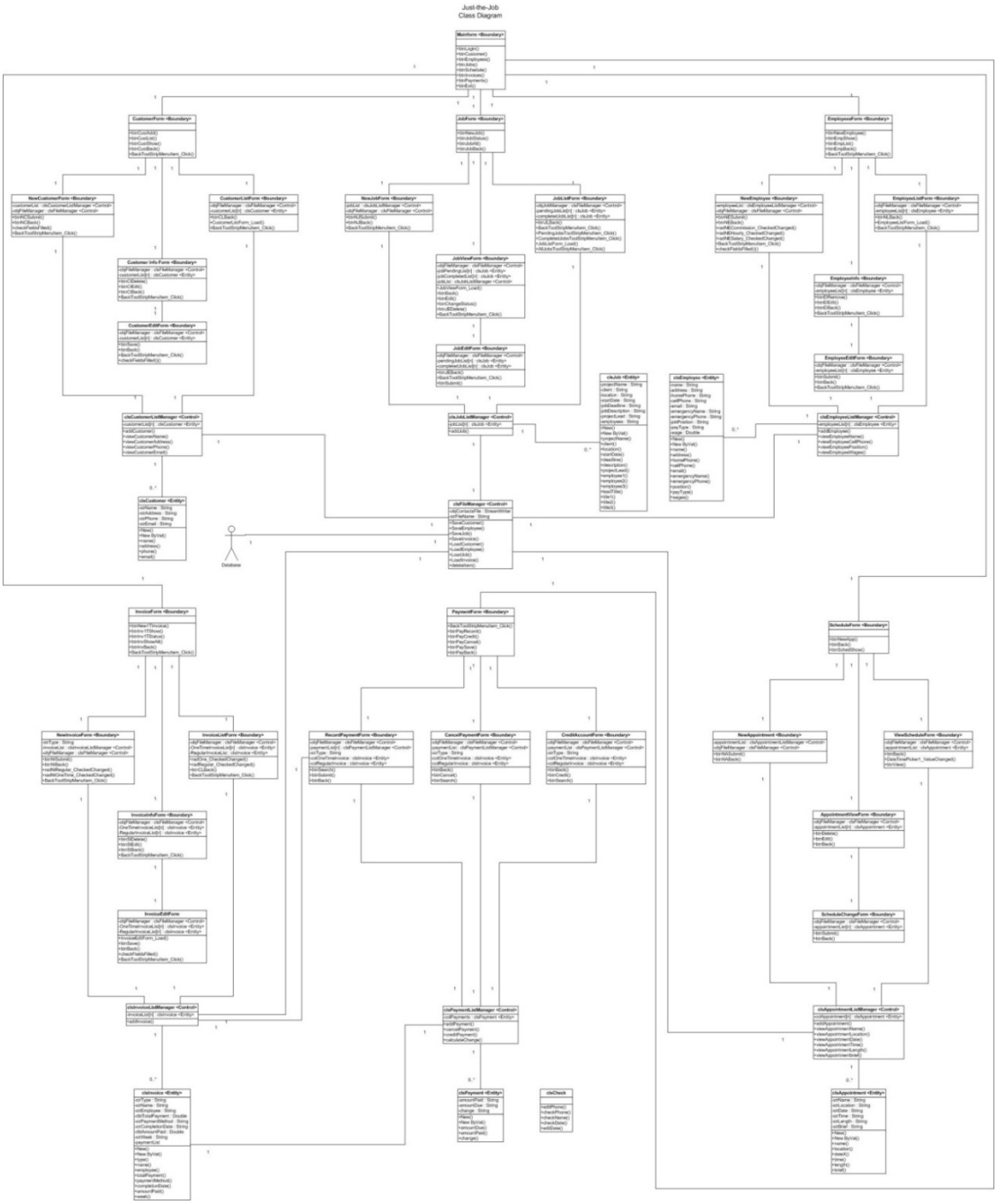


Figure 3.1 – Class diagram

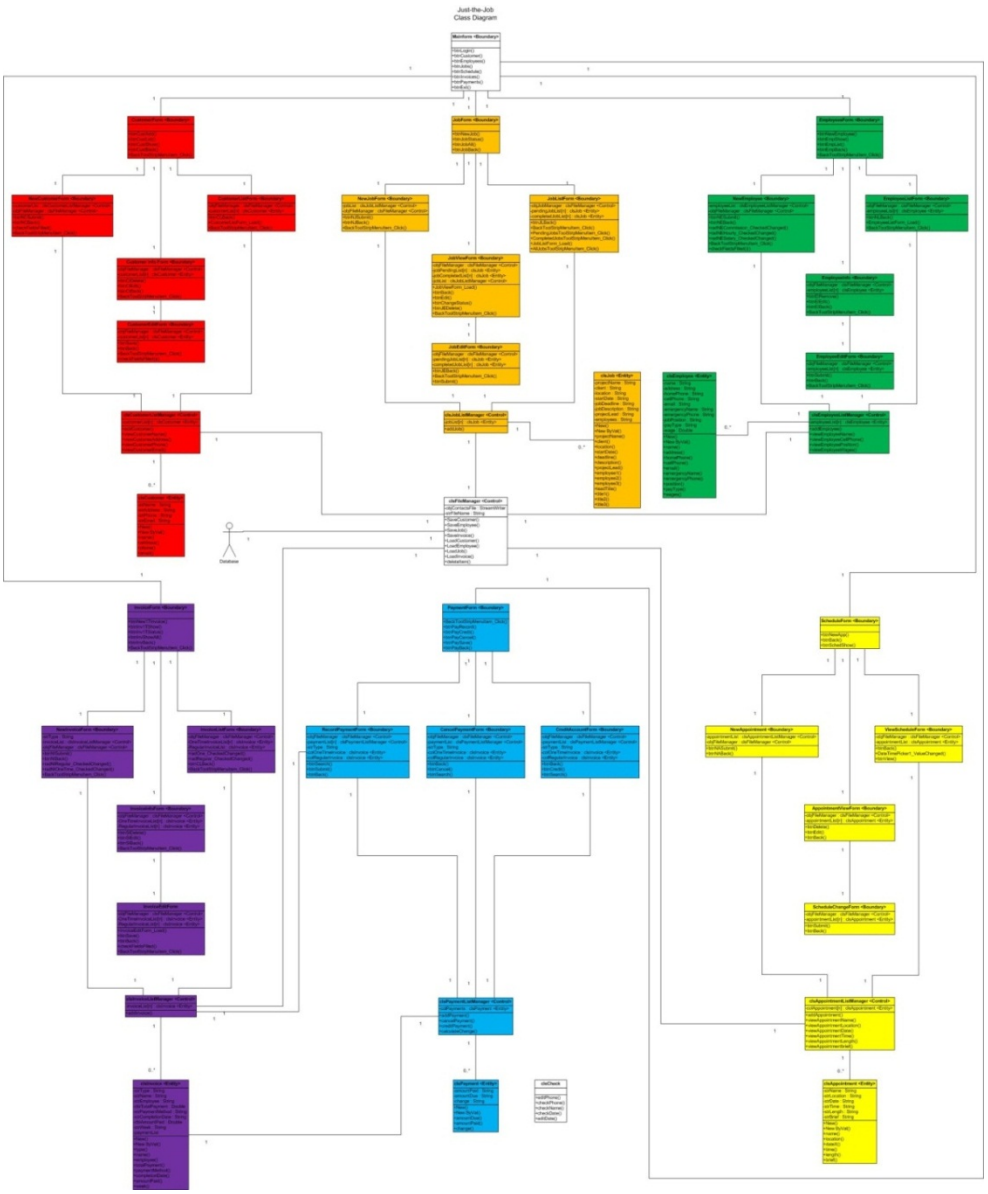


Figure 3.2 – Class diagram with colors

Legend

- Customer
- Job
- Employee
- Invoice
- Payment
- Schedule

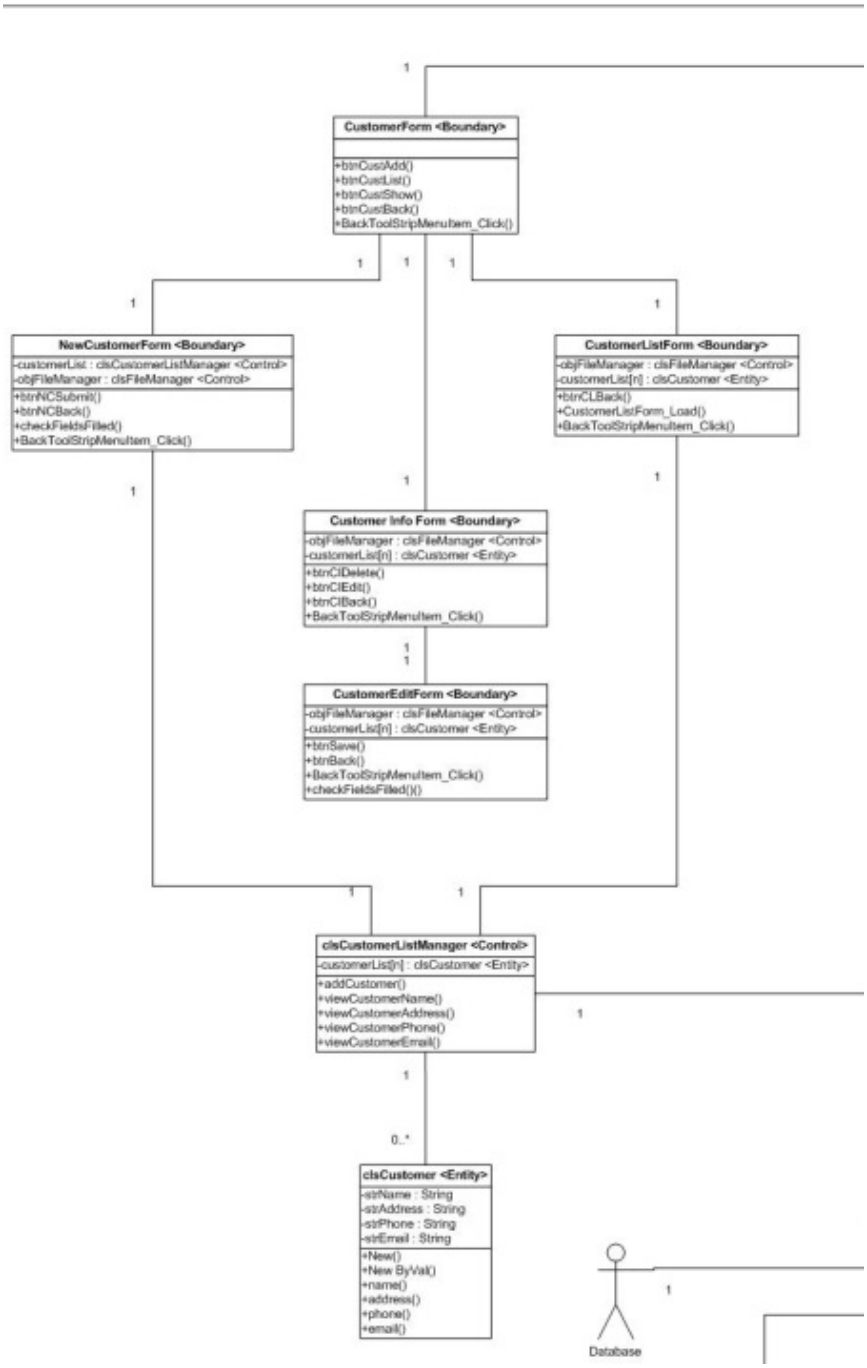


Figure 3.3 – Customer tree of class diagram

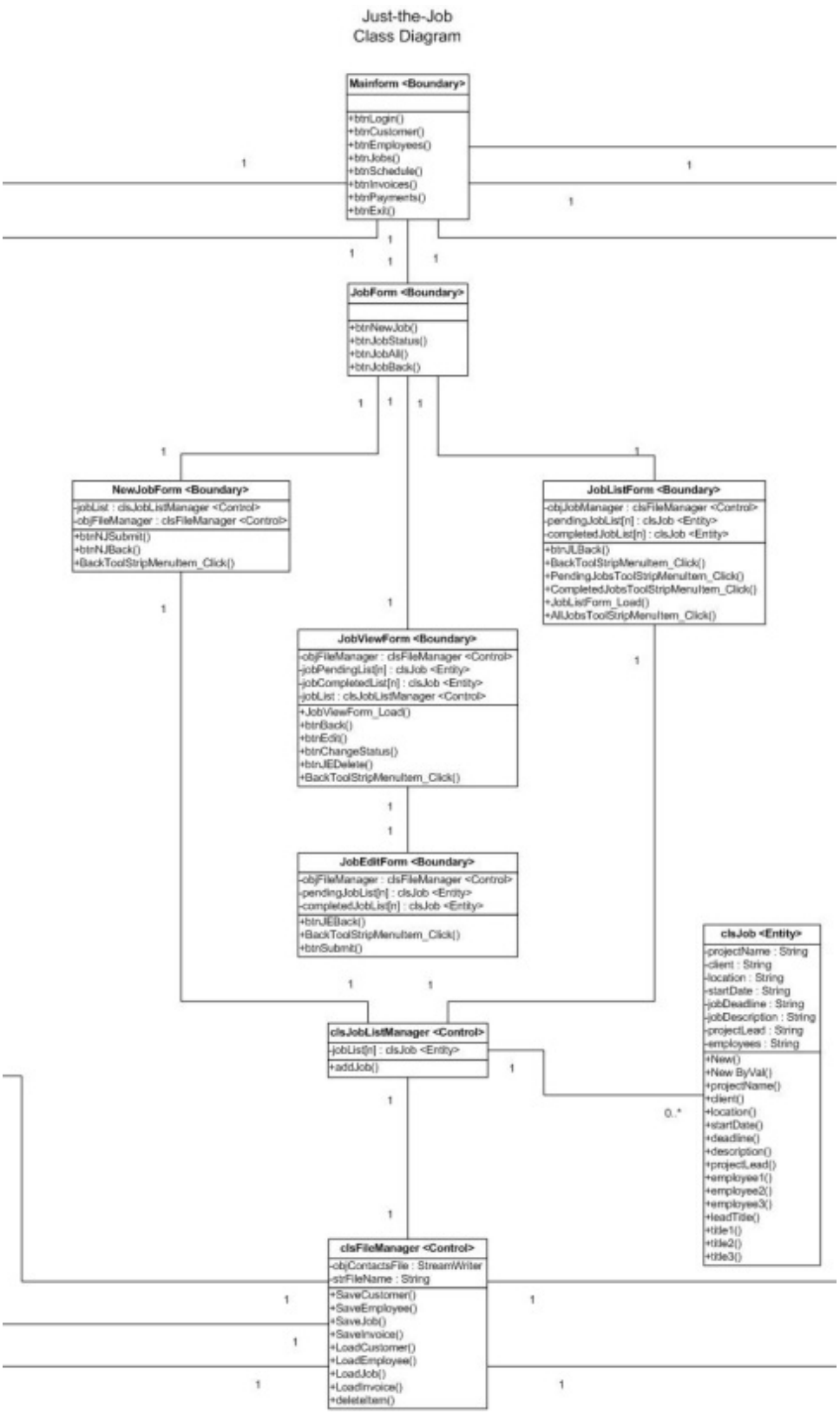


Figure 3.4 –Job tree of class diagram

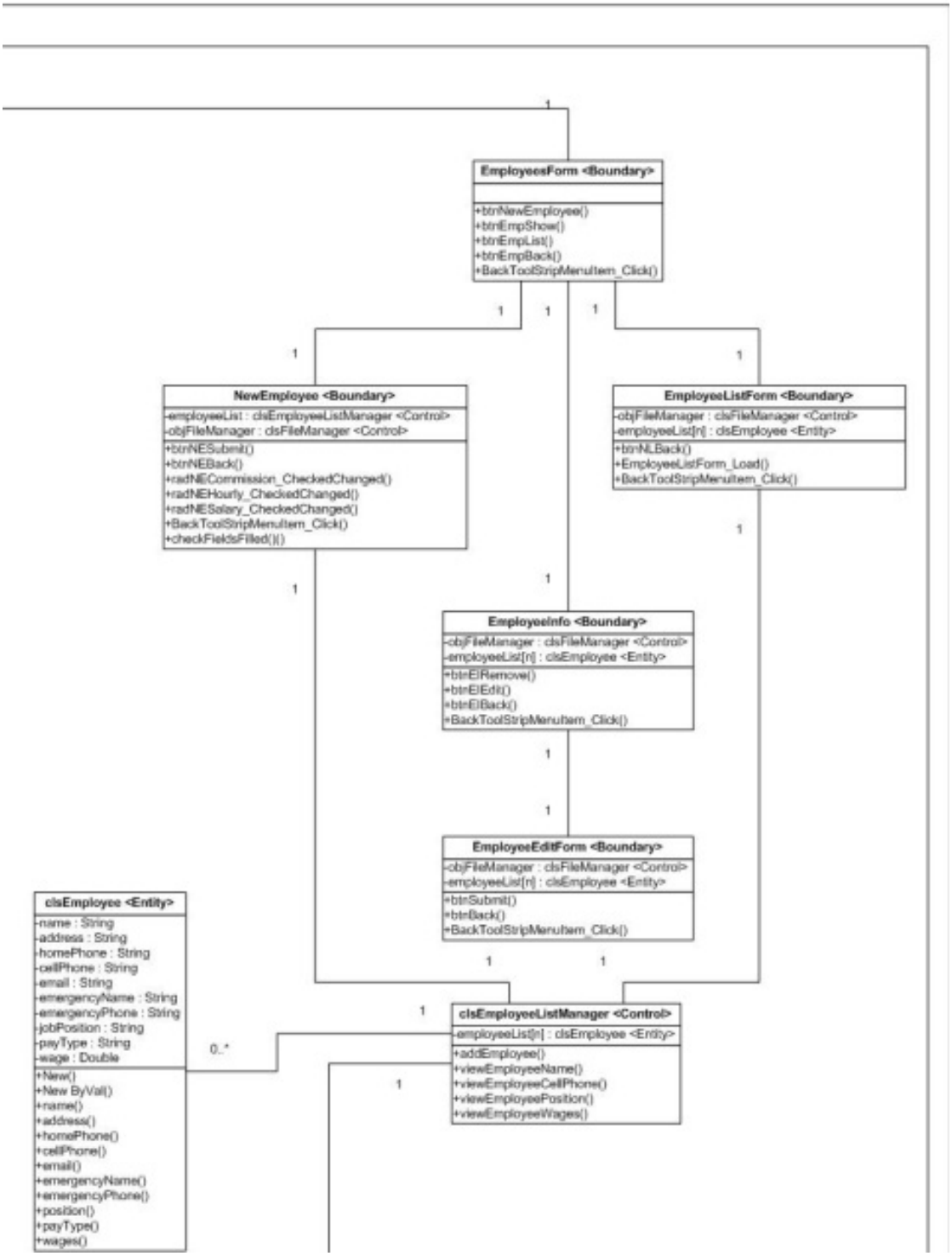


Figure 3.5 – Employee tree of class diagram

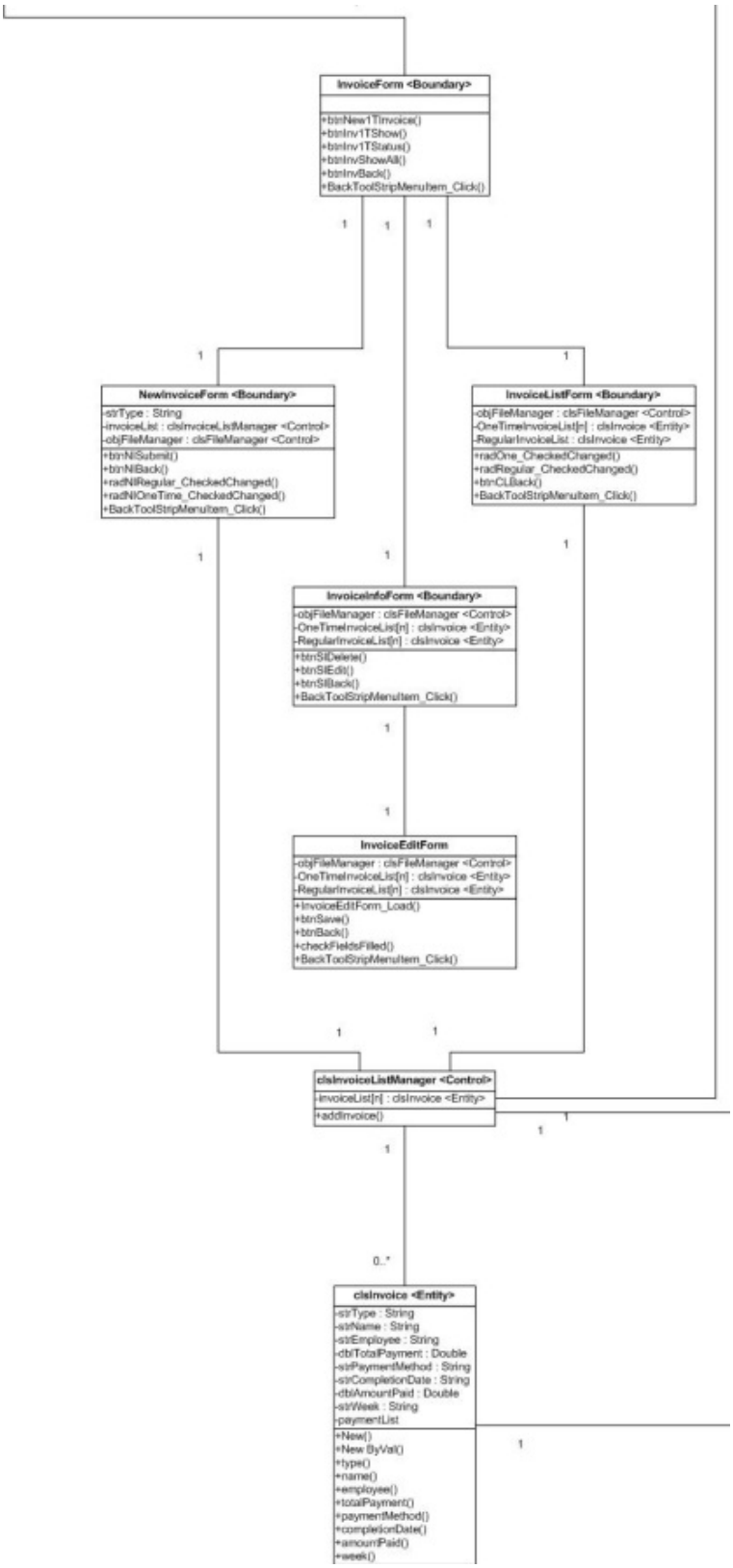


Figure 3.6 – Invoice tree of class diagram

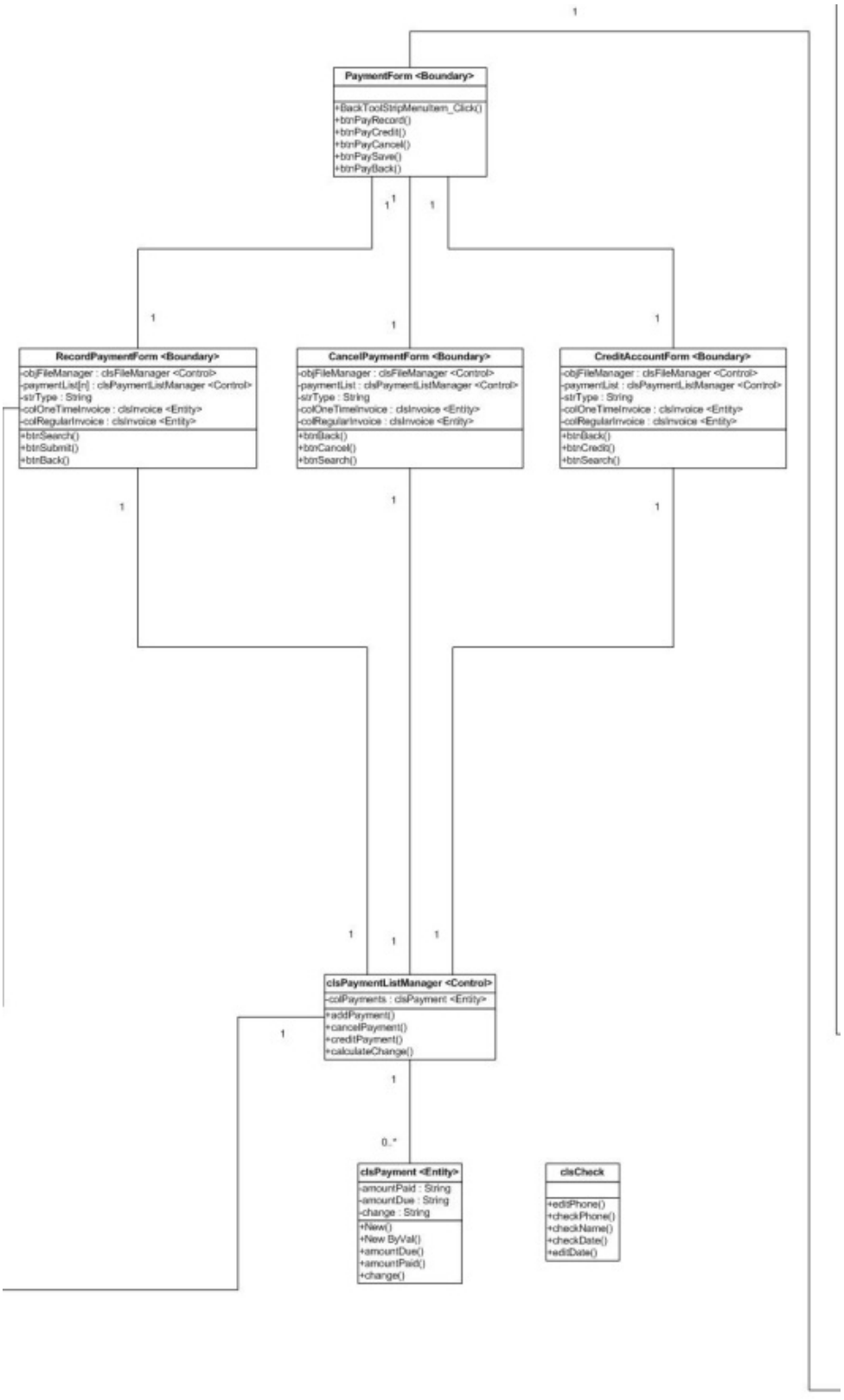


Figure 3.7 – Payment tree of class diagram

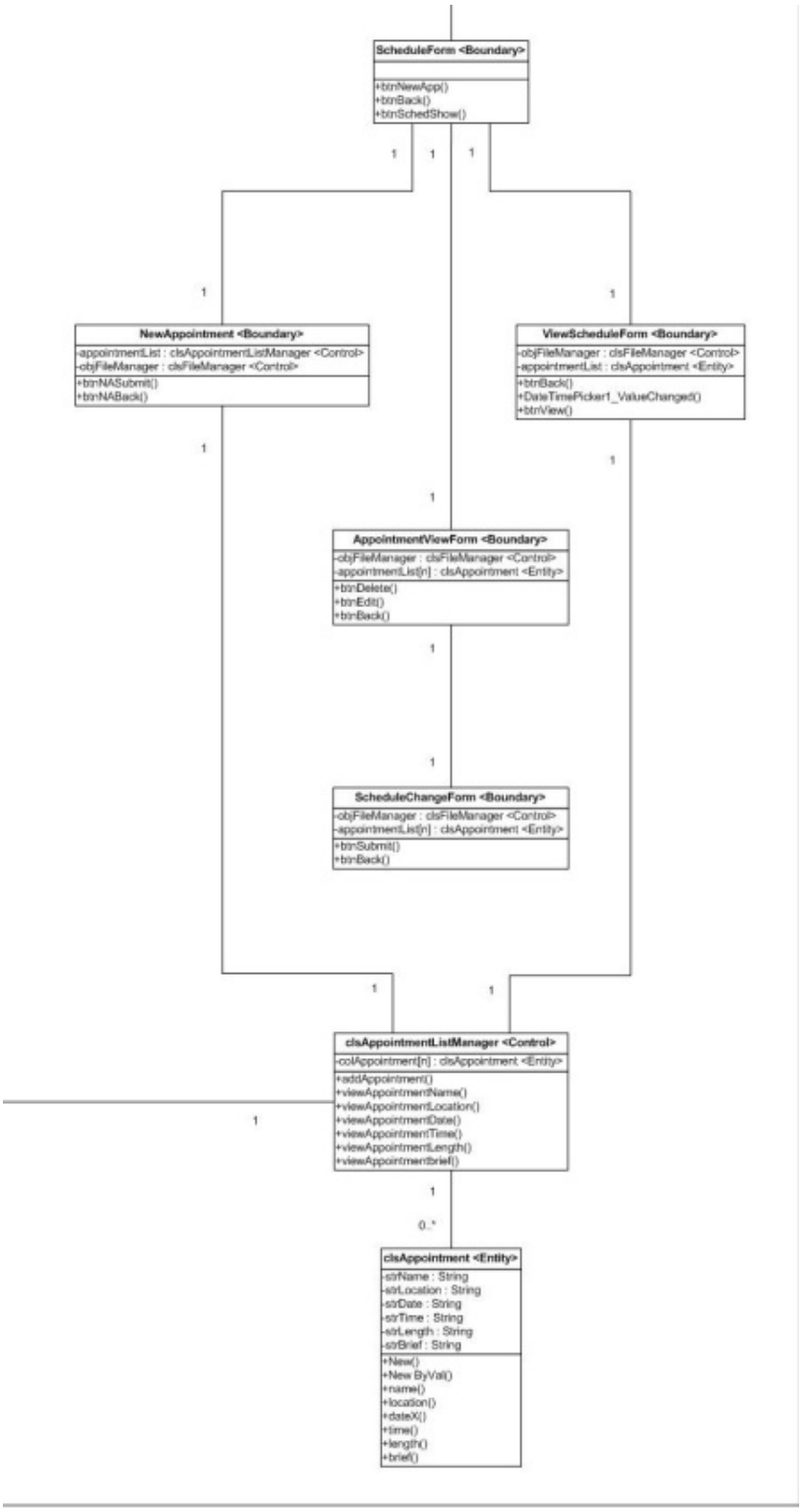


Figure 3.8 – Appointment tree of class diagram

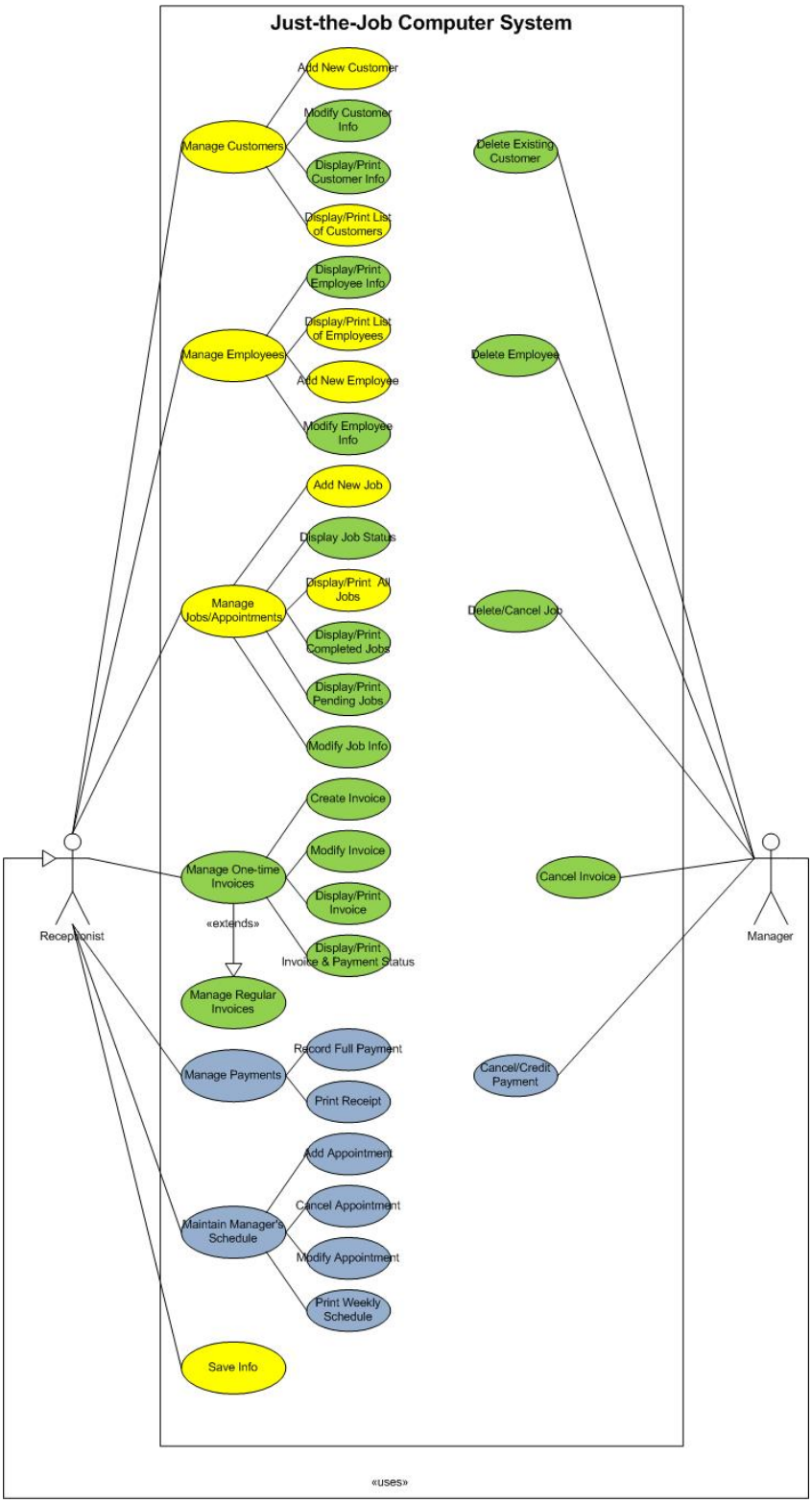


Figure 3.9 – Final use case model, new additions in blue

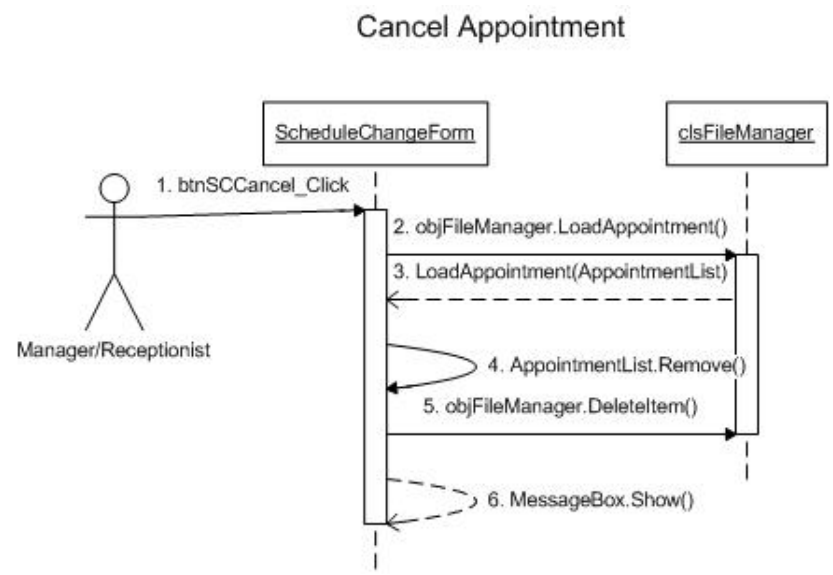


Figure 3.10

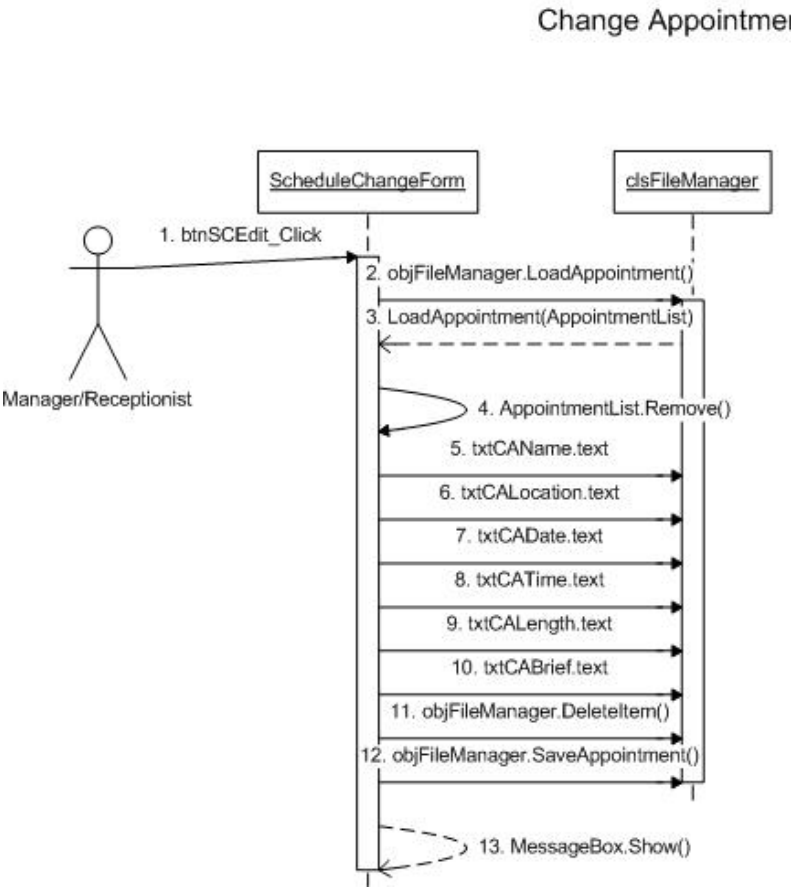


Figure 3.11

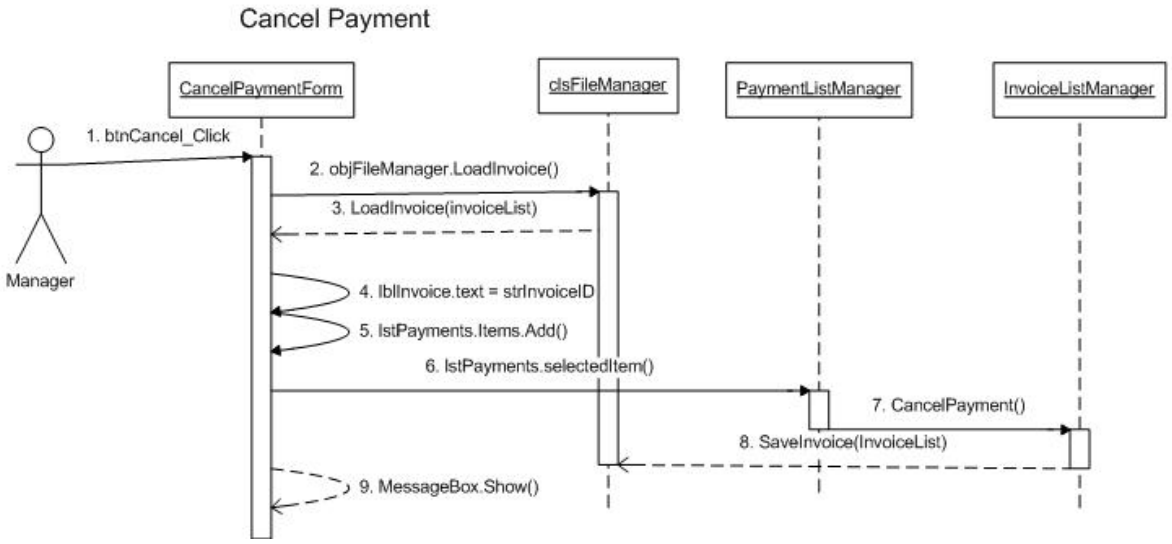


Figure 3.12

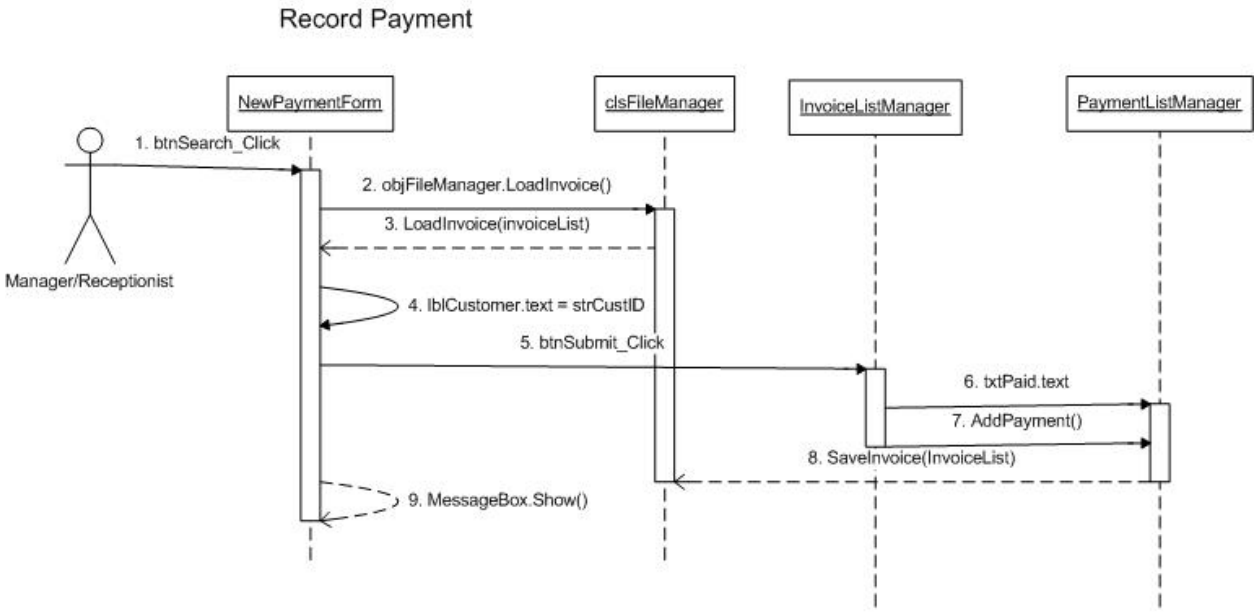


Figure 3.13

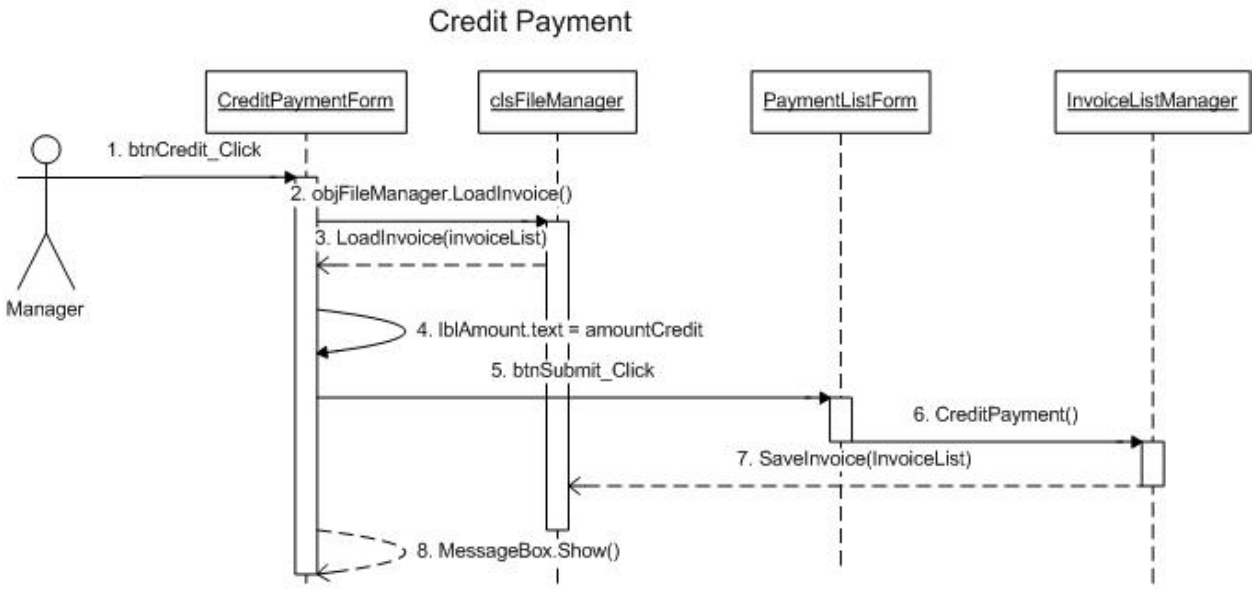


Figure 3.14

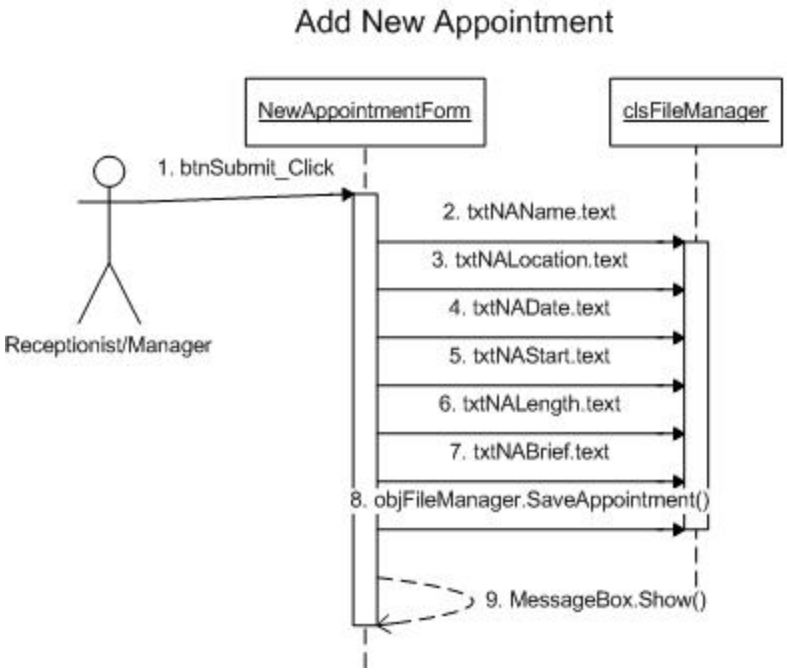


Figure 3.15

Table 3.1

clsAppointment	
Responsibilty	**Collaborator**
Create a new instance of an appointment	

Table 3.2

clsAppointmentListManager	
Responsibilty	**Collaborator**
Display information of an appointment	
Add new appointment to a collection	clsAppointment

Table 3.3

clsPayment	
Responsibility	**Collaborator**
Create a new instance of a payment	

Table 3.4

clsPaymentListManager	
Responsibilty	**Collaborator**
Display information of a payment	
Add new payment to a collection	clsPayment

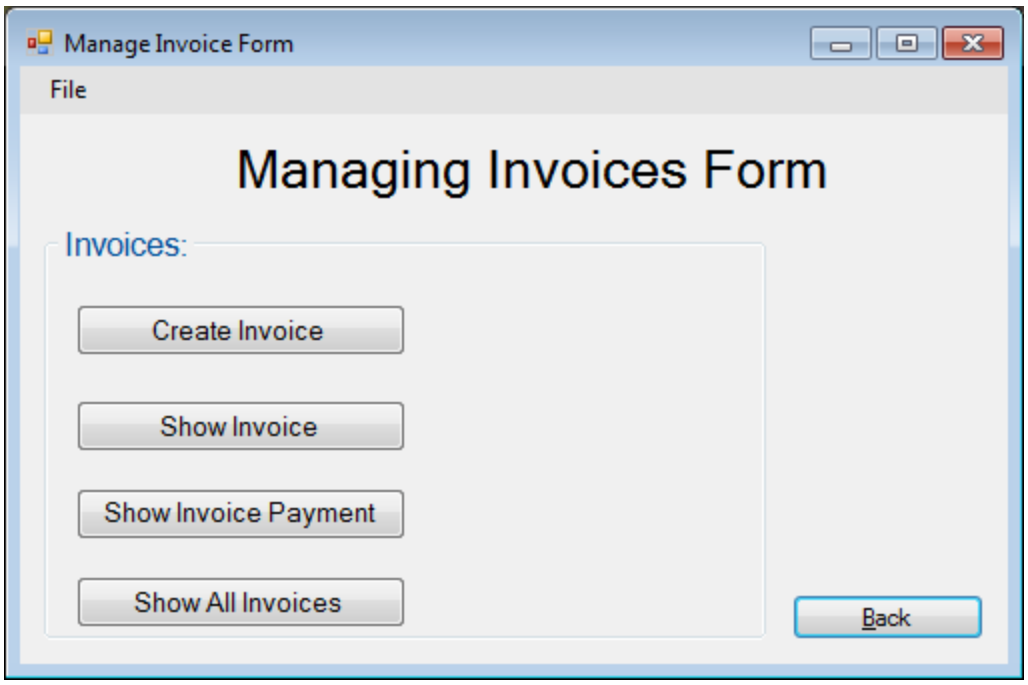


Figure 3.16

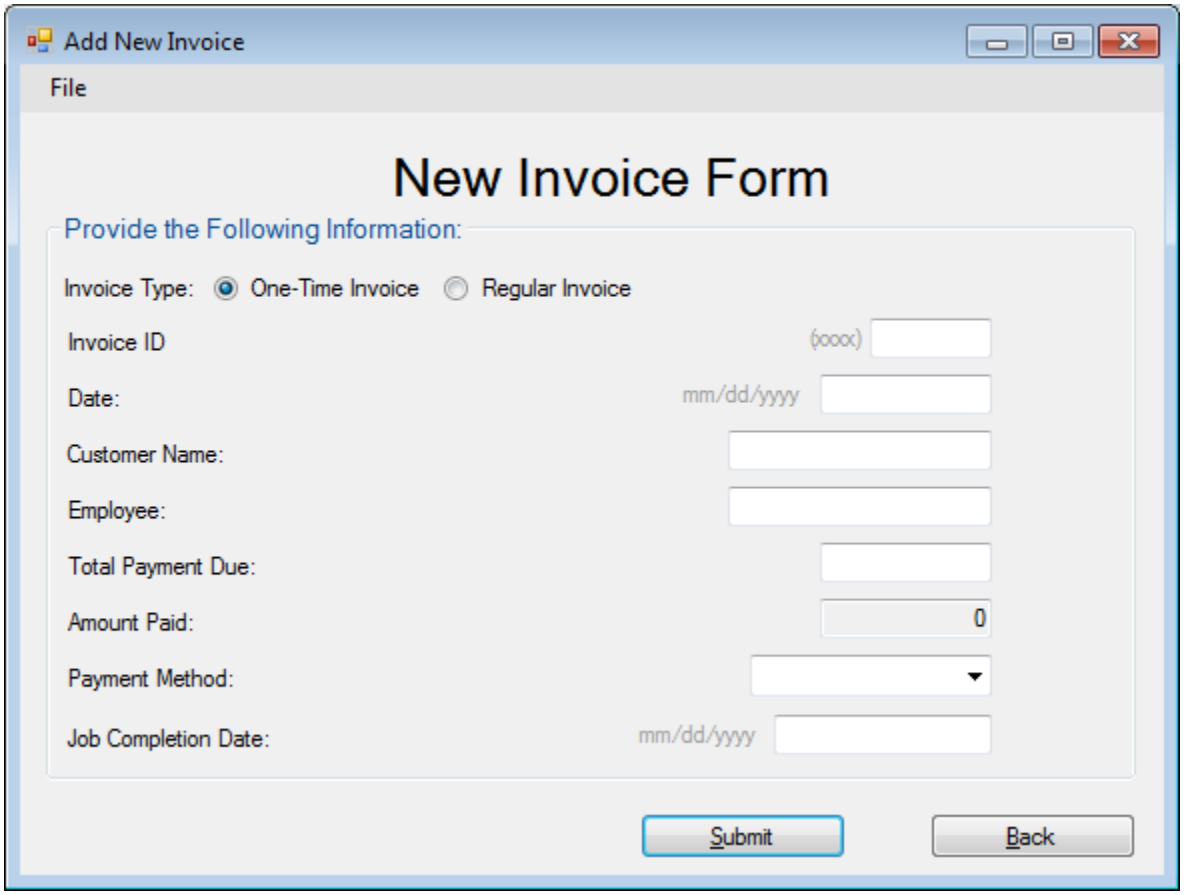


Figure 3.17

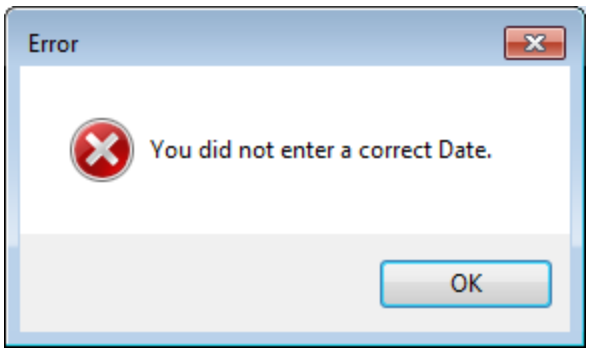


Figure 3.18

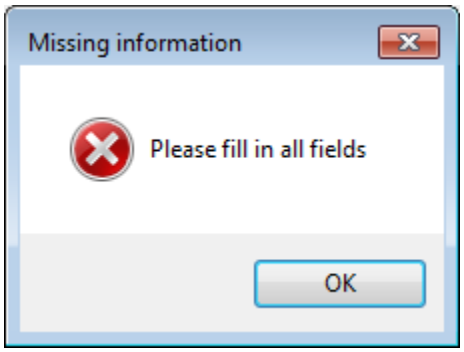


Figure 3.19

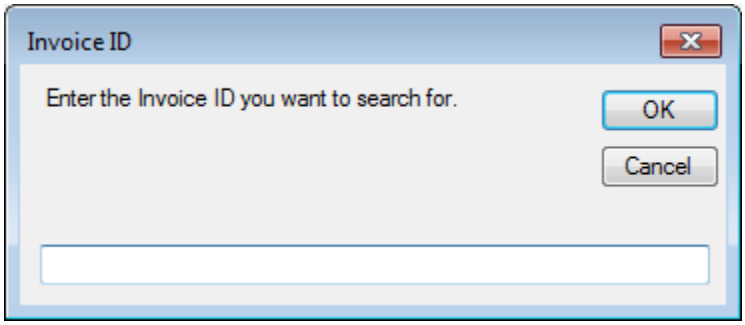


Figure 3.20

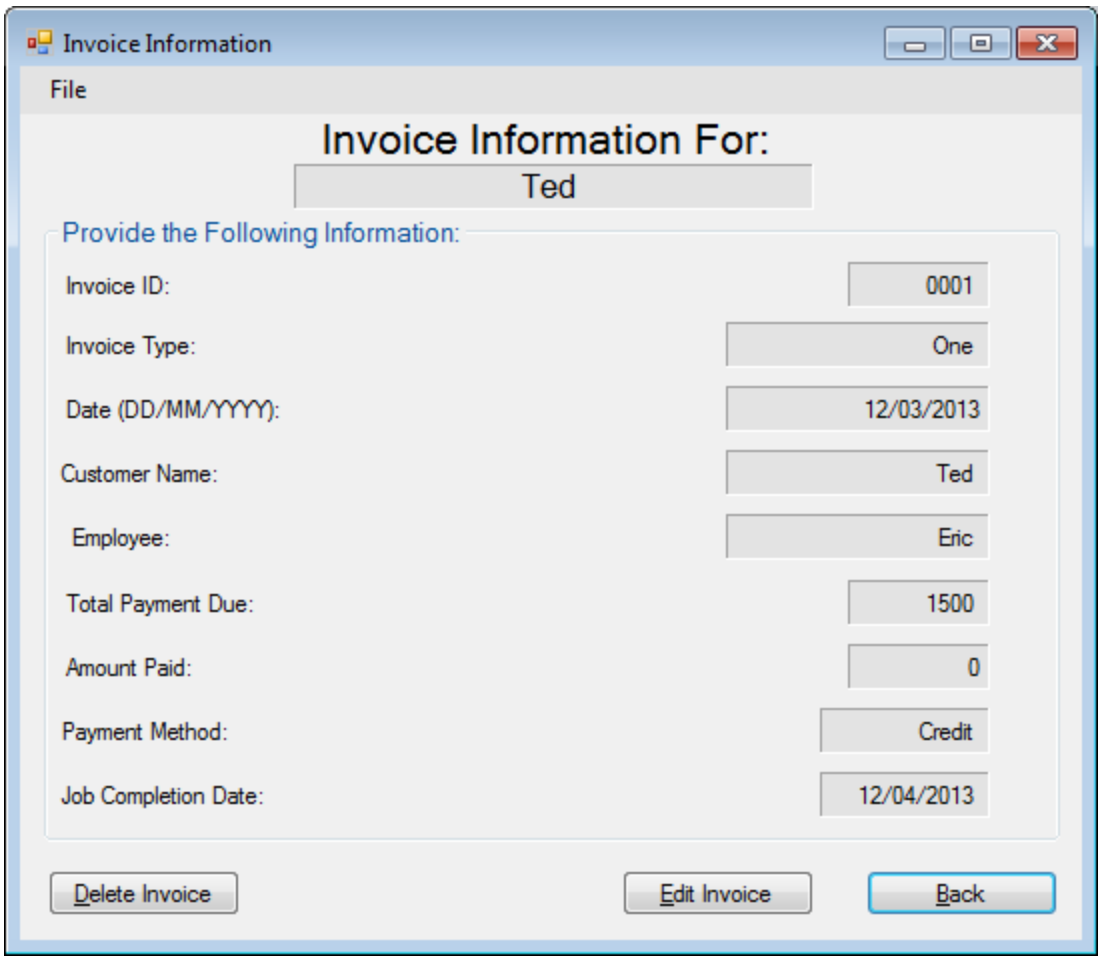

Invoice Information
File
Invoice Information For:
Ted
Provide the Following Information:
Invoice ID:
0001
Invoice Type:
One
Date (DD/MM/YYYY):
12/03/2013
Customer Name:
Ted
Employee:
Eric
Total Payment Due:
1500
Amount Paid:
0
Payment Method:
Credit
Job Completion Date:
12/04/2013
Delete Invoice
Edit Invoice
Back

Figure 3.21

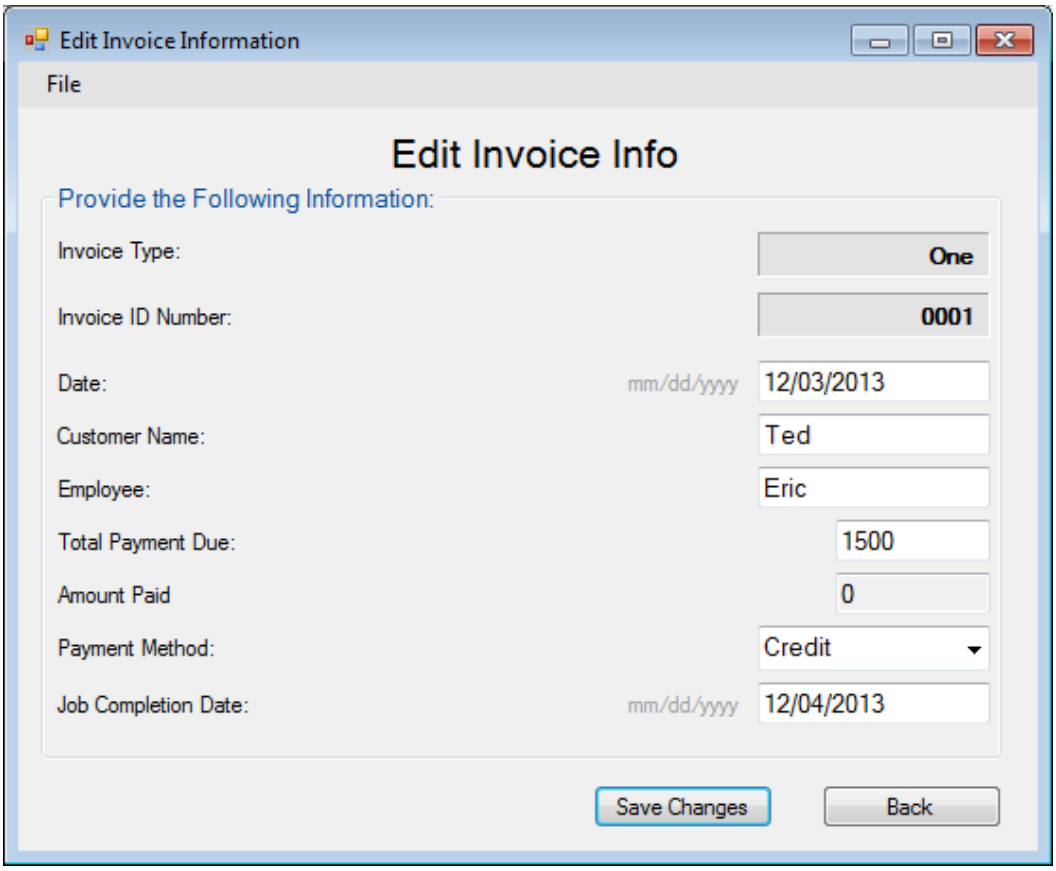


Figure 3.22

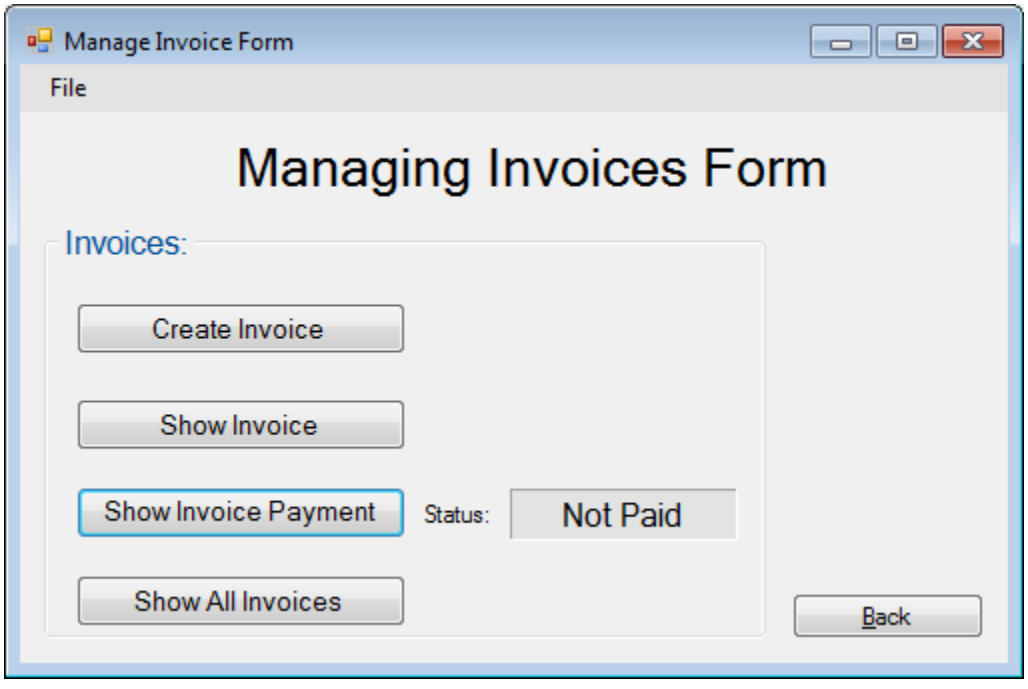


Figure 3.23

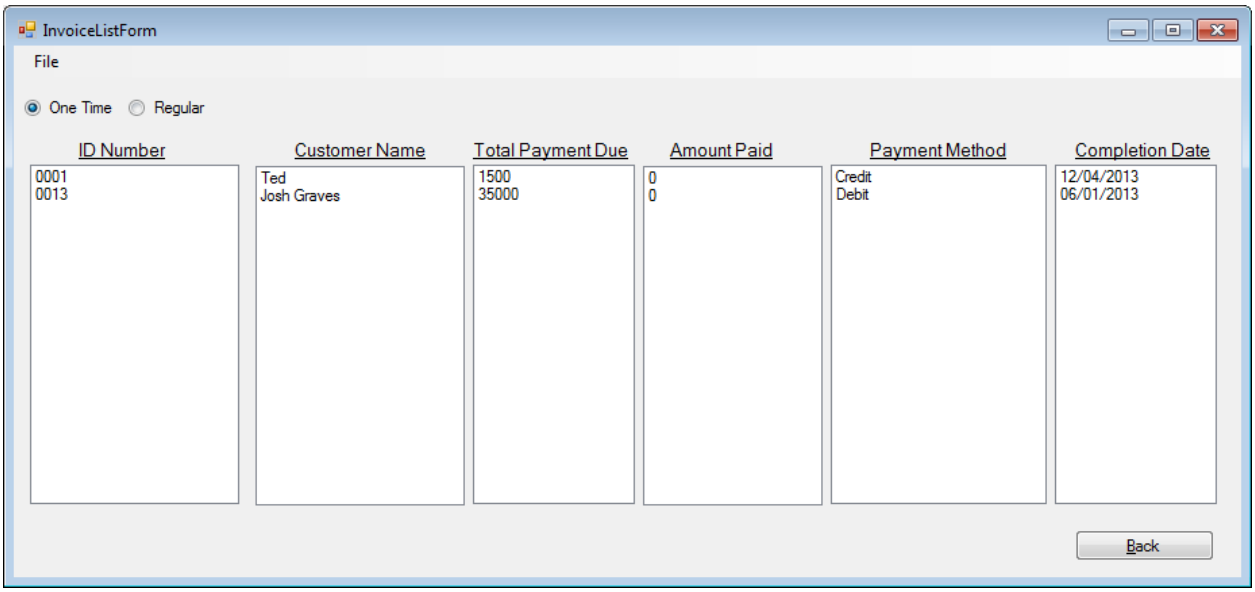


Figure 3.24

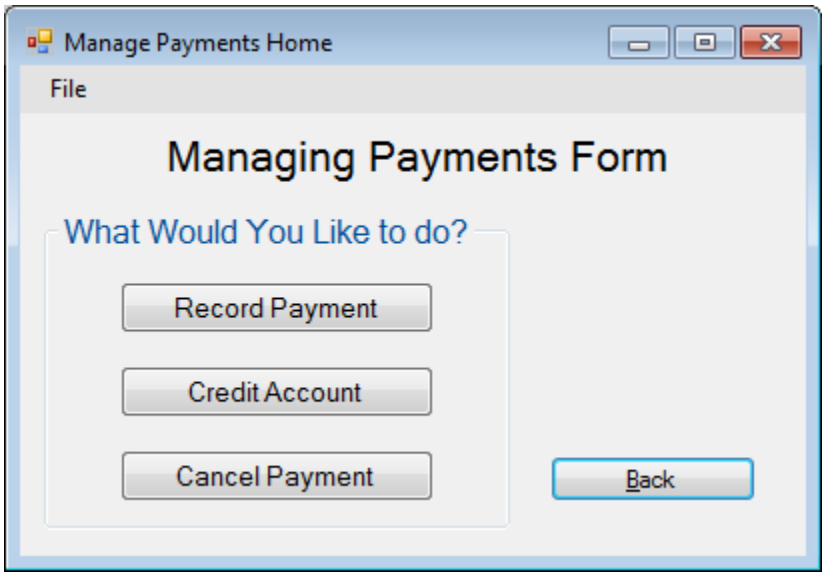


Figure 3.25

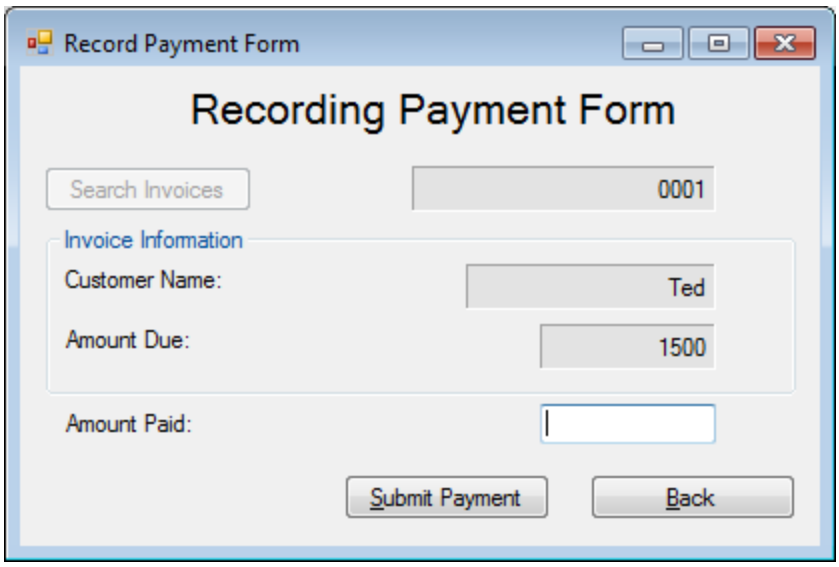


Figure 3.26

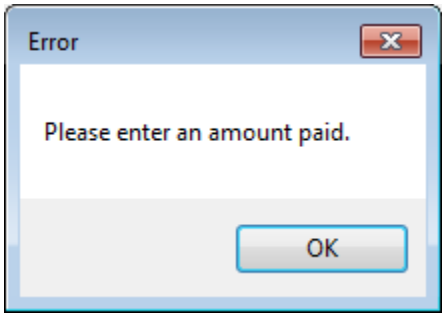


Figure 3.27

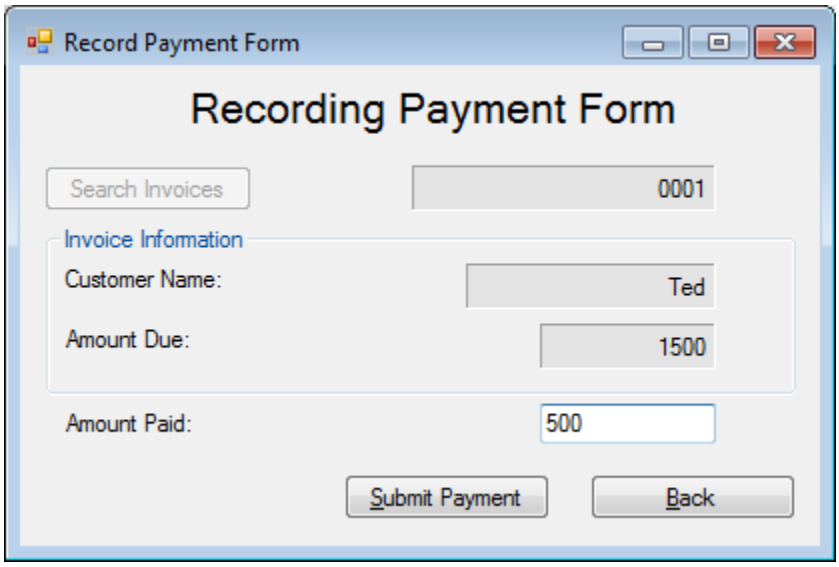


Figure 3.28

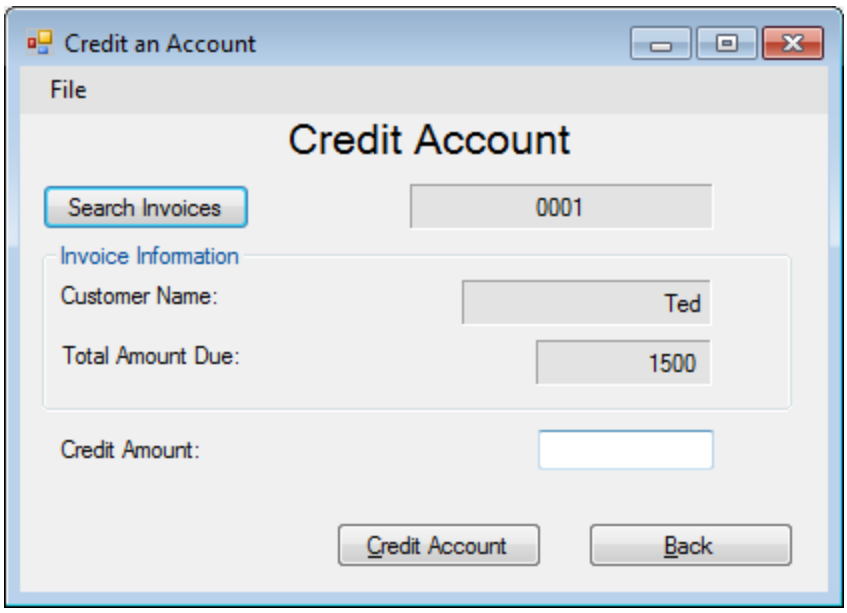


Figure 3.29

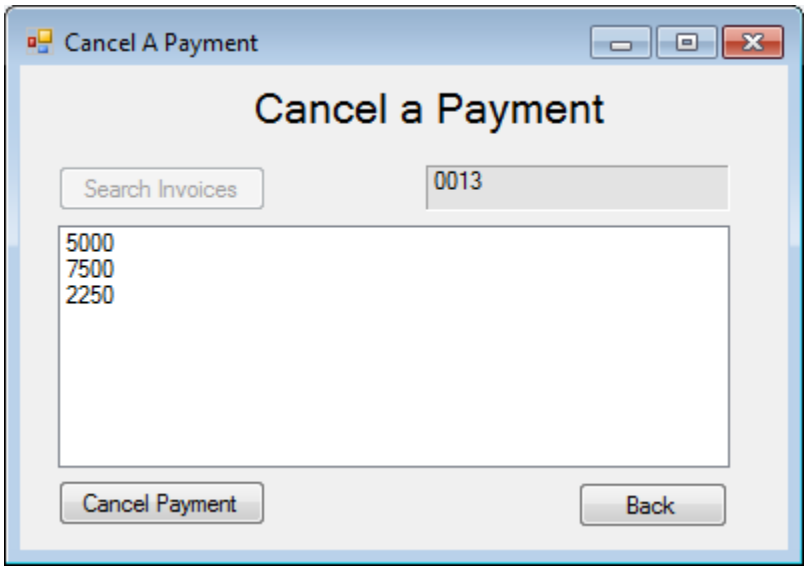


Figure 3.30

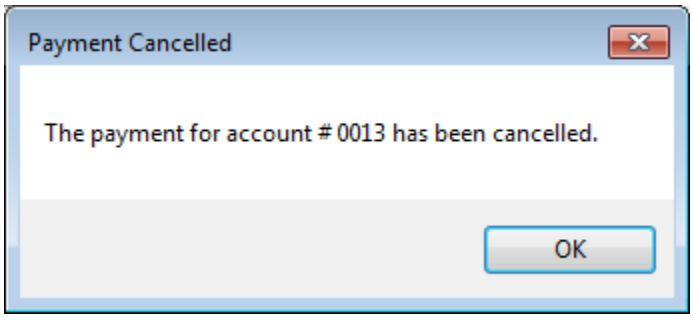


Figure 3.31

Chapter 3 Conclusion

We we're given two extra opportunities to add functionality, but we were unable to complete them before the final presentation. Our design would not have easily accepted the new changes, so we would not have had time to implement either change. One of the changes was to add a premium customer, but this would have required changes to the customer, jobs, employees, and invoices. The other option was to add exceptional employees, client reviews, and evaluation forms. Both of the options were far too time consuming to add in the final week.

clsAppointment

```
Public Class clsAppointment
    Private strName As String
    Private strLocation As String
    Private strDate As String
    Private strTime As String
    Private strLength As String
    Private strBrief As String

    Public Sub New()
        strName = ""
        strLocation = ""
        strDate = ""
        strTime = ""
        strLength = ""
        strBrief = ""
    End Sub

    Public Sub New(ByVal name As String, ByVal location As String, ByVal dateX As String,
ByVal time As String, ByVal length As String, ByVal brief As String)
        strName = name
        strLocation = location
        strDate = dateX
        strTime = time
        strLength = length
        strBrief = brief
    End Sub

    Public Property name As String
        Get
            Return strName
        End Get
        Set(ByVal value As String)

            strName = value

        End Set
    End Property

    Public Property location As String
        Get
            Return strLocation
        End Get
        Set(ByVal value As String)

            strLocation = value
        End Set
    End Property

    Public Property dateX As String
        Get
            Return strDate
        End Get
        Set(ByVal value As String)
            strDate = value
        End Set
    End Property
```

```
    Public Property time As String
        Get
            Return strTime
        End Get
        Set(ByVal value As String)

            strTime = value
        End Set
    End Property
    Public Property length As String
        Get
            Return strLength
        End Get
        Set(ByVal value As String)

            strLength = value
        End Set
    End Property
    Public Property brief As String
        Get
            Return strBrief
        End Get
        Set(ByVal value As String)

            strBrief = value
        End Set
    End Property
End Class
```

clsAppointmentListManager

```
Public Class clsAppointmentListManager
    Private colAppointment As New Collection

    Public Sub addAppointment(ByVal name As String, ByVal location As String, ByVal dateX
As String, ByVal time As String, ByVal length As String, ByVal brief As String)
        Dim objAppointment As New clsAppointment

        objAppointment.name = name
        objAppointment.location = location
        objAppointment.dateX = dateX
        objAppointment.time = time
        objAppointment.length = length
        objAppointment.brief = brief

        colAppointment.Add(objAppointment)
    End Sub

    Public Sub viewAppointmentName(ByRef guiDisplay As ListBox)
        Dim apt As clsAppointment

        For Each apt In colAppointment
            guiDisplay.Items.Add(apt.name)
        Next
    End Sub

    Public Sub viewAppointmentLocation(ByRef guiDisplay As ListBox)
```

```
        Dim apt As clsAppointment

        For Each apt In colAppointment
            guiDisplay.Items.Add(apt.location)
        Next
    End Sub

    Public Sub viewAppointmentDate(ByRef guiDisplay As ListBox)
        Dim apt As clsAppointment

        For Each apt In colAppointment
            guiDisplay.Items.Add(apt.dateX)
        Next
    End Sub

    Public Sub viewAppointmentTime(ByRef guiDisplay As ListBox)
        Dim apt As clsAppointment

        For Each apt In colAppointment
            guiDisplay.Items.Add(apt.time)
        Next
    End Sub
    Public Sub viewAppointmentLength(ByRef guiDisplay As ListBox)
        Dim apt As clsAppointment

        For Each apt In colAppointment
            guiDisplay.Items.Add(apt.length)
        Next
    End Sub
    Public Sub viewAppointmentbrief(ByRef guiDisplay As ListBox)
        Dim apt As clsAppointment

        For Each apt In colAppointment
            guiDisplay.Items.Add(apt.brief)
        Next
    End Sub
    End Class
```

clsPayment

```
Public Class clsPayment
    Private strAmountDue As String
    Private strAmountPaid As String
    Private strChange As String

    Public Sub New()
        strAmountDue = ""
        strAmountPaid = ""
        strChange = ""
    End Sub

    Public Sub New(ByVal due, ByVal paid, ByVal change)
        strAmountDue = due
        strAmountPaid = paid
        strChange = change
    End Sub

    Public Property due As String
```

```
        Get
            Return strAmountDue
        End Get
        Set(value As String)
            strAmountDue = value
        End Set
    End Property

    Public Property paid As String
        Get
            Return strAmountPaid
        End Get
        Set(value As String)
            strAmountPaid = value
        End Set
    End Property

    Public Property change As String
        Get
            Return strChange
        End Get
        Set(value As String)
            strChange = value
        End Set
    End Property
End Class
```

clsPaymentListManager

```
Public Class clsPaymentListManager
    Private colPayments As New Collection

    Public Sub addPayment(ByVal due As String, ByVal paid As String, ByVal change As
String)
        Dim objPayment As New clsPayment

        objPayment.due = due
        objPayment.paid = paid
        objPayment.change = change

        colPayments.Add(objPayment)
    End Sub
End Class
```

clsCheck

```
Public Class clsCheck
    Public Function editPhone(ByVal phoneNum As String)
        Dim areaCode As String = ""
        Dim firstThree As String = ""
        Dim lastFour As String = ""
        Dim rest As String = ""
        phoneNum = phoneNum.Replace(" ", "") ' Remove any spaces. EX: If the user enters
123 456 7890, then just make it 1234567890

        'If the phone nummber is entered is only digits then add in the correct formating
        If IsNumeric(phoneNum) And phoneNum.Length = 10 Then
            areaCode = Mid(phoneNum, 1, 3)
```

```
            areaCode = "(" + areaCode + ")-"
            firstThree = Mid(phoneNum, 4, 3)
            firstThree = firstThree + "-"
            lastFour = Mid(phoneNum, 7, 4)
            phoneNum = areaCode + firstThree + lastFour
            'MessageBox.Show("1)" + phoneNum) 'Can be used to test

            'If the user enters some formating then it needs to be handled differently
        ElseIf phoneNum.Length > 10 Then
            If Mid(phoneNum, 1, 1) <> "(" Then
                phoneNum = "(" + phoneNum
                'MessageBox.Show("2) " + phoneNum) ' can be used to test
            End If
            If Mid(phoneNum, 5, 1) <> ")" Then
                areaCode = Mid(phoneNum, 1, 4)
                areaCode = areaCode + ")"
                rest = Mid(phoneNum, 5, phoneNum.Length)
                phoneNum = areaCode + rest
                'MessageBox.Show("3) " + phoneNum + " a) " + areaCode + " b) " + rest)
'can be used to test
            Else
                areaCode = Mid(phoneNum, 1, 5)
            End If
            If Mid(phoneNum, 6, 1) <> "-" Then
                areaCode = areaCode + "-"
                rest = Mid(phoneNum, 6, phoneNum.Length)
                phoneNum = areaCode + rest
                ' MessageBox.Show("4) " + phoneNum + " a) " + areaCode + " b) " + rest) '
can be used to test
            Else
                areaCode = Mid(phoneNum, 1, 6)
            End If
            If Mid(phoneNum, 10, 1) <> "-" Then
                firstThree = Mid(phoneNum, 7, 3)
                firstThree = firstThree + "-"
                lastFour = Mid(phoneNum, 10, phoneNum.Length)
                phoneNum = areaCode + firstThree + lastFour
                'MessageBox.Show("5) " + phoneNum + " a) " + areaCode + " b) " +
firstThree + " c) " + lastFour) ' can be used to test
            End If

            'MessageBox.Show("6)" + phoneNum) ' can be used to test
        End If
        Return phoneNum
    End Function
    Public Function checkPhone(ByVal phoneNum As String) As Boolean
        Dim iCnt As Integer
        Dim numCount As Integer
        Dim x As String
        phoneNum = phoneNum.Replace(" ", "") ' Remove any spaces. EX: If the user enters
123 456 7890, then just make it 1234567890

        'If the user did not enter 10 numbers
        For iCnt = 1 To phoneNum.Length
            If IsNumeric(Mid(phoneNum, iCnt, 1)) Then
                numCount = numCount + 1
            End If
        Next
```

```
        If numCount <> 10 Then
            MessageBox.Show("You did not enter ten numbers for the phone number. Please
enter ten numbes for the phone numer",
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If

        'Checks to make sure that the phone number only consists of only numbers and
( or ) or -
        For iCnt = 1 To phoneNum.Length
            x = Mid(phoneNum, iCnt, 1)
            If Not IsNumeric(x) Then
                If x <> "(" And x <> ")" And x <> "-" Then
                    MessageBox.Show("Please only use numbers and the following symbols
for formating phone numbers: '( ) -'",
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                    Return False
                End If
                If x = "(" And iCnt <> 1 Then
                    MessageBox.Show("Please only use numbers and the following symbols
for formating phone numbers: '( ) -'",
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                    Return False
                End If
                If x = ")" And iCnt <> 5 Then
                    MessageBox.Show("Please only use numbers and the following symbols
for formating phone numbers: '( ) -'",
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                    Return False
                End If
                If x = "-" And iCnt <> 6 And iCnt <> 10 Then
                    MessageBox.Show("Please only use numbers and the following symbols
for formating phone numbers: '( ) -'",
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                    Return False
                End If

            End If

        Next

        Return True
    End Function
    Public Function editDate(ByVal dateX As String)
        Dim iCnt As Integer
        Dim slashCount As Integer
        Dim day As String = ""
        Dim month As String = ""
        Dim year As String = ""
        Dim rest As String = ""
        dateX = dateX.Replace(" ", "") ' Remove any spaces.

        For iCnt = 1 To dateX.Length
            If (Mid(dateX, iCnt, 1)) = "/" Then
                slashCount = slashCount + 1
            End If
        Next
```

```
        If slashCount = 2 Then
            If Mid(dateX, 3, 1) = "/" Then 'If they enter 2 digits for the day
                day = Mid(dateX, 1, 3)
                rest = Mid(dateX, 4, dateX.Length)
                dateX = day + rest
                'MessageBox.Show("1) " + dateX + " a) " + day + " b) " + rest) 'can be
used to test
            Else 'They did not enter 2 digits
                day = Mid(dateX, 1, 2)
                day = "0" + day
                rest = Mid(dateX, 3, dateX.Length)
                dateX = day + rest
                'MessageBox.Show("2) " + dateX + " a) " + day + " b) " + rest) 'can be
used to test
            End If
            If Mid(dateX, 6, 1) = "/" Then 'If they enter 2 digits for the month
                month = Mid(dateX, 4, 3)
                year = Mid(dateX, 7, dateX.Length)
                dateX = day + month + year
                ' MessageBox.Show("3) " + dateX + " a) " + day + " b) " + month + " c) "
+ year) ' can be used to test
            Else ' If they did not enter 2 digits
                month = Mid(dateX, 4, 2)
                month = "0" + month
                year = Mid(dateX, 6, dateX.Length)
                dateX = day + month + year
                'MessageBox.Show("4) " + dateX + " a) " + day + " b) " + month + " c) " +
year) ' can be used to test
            End If

            'MessageBox.Show("5)" + dateX) ' can be used to test
        End If
        Return dateX
    End Function
    Public Function checkDate(ByVal dateX As String)
        Dim iCnt As Integer
        Dim numCount As Integer
        Dim slashCount As Integer
        Dim x As String
        dateX = dateX.Replace(" ", "") ' Remove any spaces.

        'If the user did not enter 8 Numbers
        For iCnt = 1 To dateX.Length
            If IsNumeric(Mid(dateX, iCnt, 1)) Then
                numCount = numCount + 1
            End If
        Next
        If numCount <> 8 Then
            MessageBox.Show("You did not enter a correct Date.",
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If

        For iCnt = 1 To dateX.Length
            If (Mid(dateX, iCnt, 1)) = "/" Then
                slashCount = slashCount + 1
            End If
        Next
```

```
        If slashCount <> 2 Then
            MessageBox.Show("You did not enter a correct Date.",
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        'Checks to make sure that the date consists of only numbers and '/'
        For iCnt = 1 To dateX.Length
            x = Mid(dateX, iCnt, 1)
            If Not IsNumeric(x) Then
                If x <> "/" Then
                    MessageBox.Show("Please only use numbers and the following symbols
for formating phone numbers: '( ) -'",
                            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                    Return False
                End If

            End If

        Next

        Return True
    End Function
    Public Function checkName(ByVal name As String)
        Dim x As String
        name = name.Replace(" ", "") 'remove spaces when checking
        name = LCase(name)
        For iCnt = 1 To Len(name)
            x = Mid(name, iCnt, 1)
            If x < "a" Or x > "z" Then
                MessageBox.Show("Please use letters only for your name", "Letters only",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                Return False
            End If
        Next iCnt
        Return True
    End Function
End Class
```

NewAppointment

```
Public Class NewAppointment
    Dim appointmentList As New clsAppointmentListManager
    Dim objFileManager As New clsFileManager
    'Close Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub
    'Close Form
    Private Sub btnNABack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNABack.Click
        Me.Close()
    End Sub
    'Adds the Appointment to the Manager's Schedule
    Private Sub btnNASubmit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNASubmit.Click
        Dim objCheck As New clsCheck
        Dim fieldsFilled As Boolean = False
```

```
        Dim nameCorrect As Boolean = False
        Dim dateCorrect As Boolean = False

        Dim appDate As String

        nameCorrect = objCheck.checkName(txtNAName.Text)
        appDate = objCheck.editDate(txtNADate.Text)
        dateCorrect = objCheck.checkDate(appDate)

        fieldsFilled = checkFieldsFilled()
        If fieldsFilled = True And nameCorrect = True And dateCorrect = True Then
            appointmentList.addAppointment(txtNAName.Text, txtNALocation.Text, appDate,
txtNAStart.Text, txtNALength.Text, txtNABrief.Text)
            objFileManager.SaveAppointment(txtNAName.Text, txtNALocation.Text, appDate,
txtNAStart.Text, txtNALength.Text, txtNABrief.Text)
            Me.Close()
        End If

    End Sub

    'Checks to see if all the fields are filled out
    Private Function checkFieldsFilled() As Boolean

        If txtNAName.Text.Length = 0 Or txtNALocation.Text.Length = 0 Or
txtNADate.Text.Length = 0 Or txtNAStart.Text.Length = 0 Or txtNALength.Text.Length = 0
Then
            MessageBox.Show("Please fill in all fields", "Missing information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function

End Class
```

ScheduleForm

```
Public Class ScheduleForm

    'Display dialog box to input name of appointment, then show a form that contains that
appointment's information along
    'with an option to cancel it
    Private Sub btnSchedChange_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        ScheduleChangeForm.Show()
    End Sub

    'Display the new appointment form
    Private Sub btnNewApp_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNewApp.Click
        NewAppointment.Show()
    End Sub

    'save all changed info
    Private Sub btnSchedSave_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Me.Close()
    End Sub
```

```
    'close schedule form
    Private Sub btnSchedBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSchedBack.Click
        Me.Close()
    End Sub
    'Open form containing the weekly schedule
    Private Sub btnSchedShow_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSchedShow.Click
        ViewScheduleForm.Show()
    End Sub
    'Saves Info
    Private Sub SaveToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles SaveToolStripMenuItem.Click

    End Sub
    'Closes Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub
End Class
```

ViewScheduleForm

```
Public Class ViewScheduleForm
    Dim objAppointmentManager As New clsFileManager

    'close the form
    Private Sub btnVSBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnVSBack.Click
        Me.Close()
    End Sub
    'Prints the week's schedule
    Private Sub btnNAPrint_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)

    End Sub

    'Close the form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub
    'Opens AppointmentView form
    Private Sub btnView_Click(sender As Object, e As EventArgs) Handles btnView.Click
        Try
            AppointmentViewForm.lblDate.Text =
lsvSchedule.SelectedItems(0).SubItems(0).Text
            AppointmentViewForm.lblName.Text =
lsvSchedule.SelectedItems(0).SubItems(1).Text
            AppointmentViewForm.lblLocation.Text =
lsvSchedule.SelectedItems(0).SubItems(2).Text
            AppointmentViewForm.lblTime.Text =
lsvSchedule.SelectedItems(0).SubItems(3).Text
            AppointmentViewForm.lblLength.Text =
lsvSchedule.SelectedItems(0).SubItems(4).Text
            AppointmentViewForm.lblBrief.Text =
lsvSchedule.SelectedItems(0).SubItems(5).Text
```

```
            AppointmentViewForm.Show()
        Catch ex As Exception
            MessageBox.Show("Please select an appointment from the list to view", "No
Appointment Selected", MessageBoxButtons.OK, MessageBoxIcon.Information)
        End Try

    End Sub

    Private Sub DateTimePicker1_ValueChanged(sender As Object, e As EventArgs) Handles
DateTimePicker1.ValueChanged
        Dim strStart As String
        Dim strEnd As String
        strStart = DateTimePicker1.Value
        strStart = strStart.Substring(0, strStart.LastIndexOf("/") + 5)
        lblVSStart.Text = strStart

        strEnd = DateTimePicker1.Value.AddDays(6)
        strEnd = strEnd.Substring(0, strEnd.LastIndexOf("/") + 5)
        lblVSEnd.Text = strEnd
        searchDates()
    End Sub
    Public Function searchDates()
        Dim AppointmentList As New Collection
        Dim objCheck As New clsCheck
        objAppointmentManager.LoadAppointment(AppointmentList)
        Dim i As Integer
        Dim x As Integer = 0
        Dim counter As Integer
        Dim temp As String
        Dim name As String
        Dim loc As String
        Dim appDate As String
        Dim time As String
        Dim length As String
        Dim brief As String
        Dim tempDate As String

        lsvSchedule.Items.Clear()

        i = 1

        counter = AppointmentList.Count
        While counter > 0
            temp = AppointmentList.Item(i)
            temp = temp.Substring(6)
            name = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(10)
            loc = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)
            appDate = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6)
            time = temp.Substring(0, temp.IndexOf("<"))
```

```
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(8)
            length = temp.Substring(0, temp.IndexOf("<"))

            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(7)

            brief = temp.Substring(0)
            x = 0
            While x <= 7
                tempDate = DateTimePicker1.Value.AddDays(x)
                tempDate = tempDate.Substring(0, tempDate.LastIndexOf("/") + 5)
                tempDate = objCheck.editDate(tempDate)
                If appDate.Contains(tempDate) Then

                    Dim item = lsvSchedule.Items.Add(appDate)
                    item.SubItems.Add(name)
                    item.SubItems.Add(loc)
                    item.SubItems.Add(time)
                    item.SubItems.Add(length)
                    item.SubItems.Add(brief)

                    Exit While
                End If
                x = x + 1
            End While
            counter = counter - 1
            i = i + 1
        End While
        Return 0
    End Function
End Class
```

ScheduleChangeForm

```
Public Class ScheduleChangeForm
    Dim objFileManager As New clsFileManager
    Dim appointmentList As New Collection
    'Closes Form
    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub
    'Closes Form
    Private Sub btnSCBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSCBack.Click
        Me.Close()
    End Sub


    'Checks to see if all the fields are filled out
    Private Function checkFieldsFilled() As Boolean

        If txtCAName.Text.Length = 0 Or txtCALocation.Text.Length = 0 Or
txtCADate.Text.Length = 0 Or txtCATime.Text.Length = 0 Or txtCALength.Text.Length = 0
Then
```

```
            MessageBox.Show("Please fill in all fields", "Missing information",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False
        End If
        Return True
    End Function
    'Submits the changes made to the appointment to the database

    Private Sub btnSCEdit_Click(sender As Object, e As EventArgs) Handles btnSCEdit.Click
        Dim objCheck As New clsCheck
        Dim fieldsFilled As Boolean = False
        Dim nameCorrect As Boolean = False
        Dim dateCorrect As Boolean = False

        Dim appDate As String
        Dim temp As String
        Dim tempName As String
        Dim tempDate As String
        Dim counter As Integer = 1


        nameCorrect = objCheck.checkName(txtCAName.Text)
        appDate = objCheck.editDate(txtCADate.Text)
        dateCorrect = objCheck.checkDate(appDate)

        fieldsFilled = checkFieldsFilled()
        If fieldsFilled = True And nameCorrect = True And dateCorrect = True Then
            objFileManager.LoadAppointment(appointmentList)

            For Each apt In appointmentList
                temp = apt
                temp = temp.Substring(6)        'Removes beginning <name> tags and gets
the date
                tempName = temp.Substring(0, temp.IndexOf("<"))
                temp = temp.Substring(temp.IndexOf("<"))
                'remove <location> tags
                temp = temp.Substring(10)
                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6) 'removes <date> tags and gets the name
                tempDate = temp.Substring(0, temp.IndexOf("<"))
                If (tempName = AppointmentViewForm.lblName.Text) And (tempDate =
AppointmentViewForm.lblDate.Text) Then
                    appointmentList.Remove(counter)
                End If
                counter = counter + 1
            Next
            appointmentList.Add("<name>" + txtCAName.Text.Trim + " <location>" +
txtCALocation.Text.Trim + " <date>" + appDate.Trim + " <time>" + txtCATime.Text.Trim +
                                    " <length>" + txtCALength.Text.Trim + " <brief>" +
txtCABrief.Text.Trim)

            objFileManager.deleteItem(appointmentList, "Appointments.txt")
            ViewScheduleForm.lsvSchedule.Items.Clear()
            ViewScheduleForm.searchDates()

            AppointmentViewForm.Close()
            Me.Close()
```

```
        End If
    End Sub
End Class
```

AppointmentViewForm

```
Public Class AppointmentViewForm
    Dim objFileManager As New clsFileManager
    Dim appointmentList As New Collection


    Private Sub btnDelete_Click(sender As Object, e As EventArgs) Handles btnDelete.Click
        Dim temp As String
        Dim tempName As String
        Dim tempDate As String
        Dim counter As Integer
        counter = 1

        objFileManager.LoadAppointment(appointmentList)


        For Each apt In appointmentList
            temp = apt
            temp = temp.Substring(6)        'Removes beginning <name> tags and gets the
date
            tempName = temp.Substring(0, temp.IndexOf("<"))
            temp = temp.Substring(temp.IndexOf("<"))
            'remove <location> tags
            temp = temp.Substring(10)
            temp = temp.Substring(temp.IndexOf("<"))
            temp = temp.Substring(6) 'removes <date> tags and gets the name
            tempDate = temp.Substring(0, temp.IndexOf("<"))

            If (tempName = lblName.Text) And (tempDate = lblDate.Text) Then

                appointmentList.Remove(counter)
            End If
            counter = counter + 1
        Next

        objFileManager.deleteItem(appointmentList, "Appointments.txt")
        MessageBox.Show("Appointment Deleted", "Remove Successful", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        ViewScheduleForm.lsvSchedule.Items.Clear()
        ViewScheduleForm.searchDates()
        Me.Close()
    End Sub

    Private Sub btnEdit_Click(sender As Object, e As EventArgs) Handles btnEdit.Click
        ScheduleChangeForm.txtCABrief.Text = lblBrief.Text
        ScheduleChangeForm.txtCADate.Text = lblDate.Text
        ScheduleChangeForm.txtCALength.Text = lblLength.Text
        ScheduleChangeForm.txtCALocation.Text = lblLocation.Text
        ScheduleChangeForm.txtCAName.Text = lblName.Text
        ScheduleChangeForm.txtCATime.Text = lblTime.Text
        ScheduleChangeForm.Show()
    End Sub
```

```
    Private Sub btnNABack_Click(sender As Object, e As EventArgs) Handles btnNABack.Click
        Me.Close()
    End Sub
End Class
```

PaymentForm

```
Public Class PaymentForm
    'Credits a payment. Prompts verification that user is manager before proceeding
    Private Sub btnPayCredit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPayCredit.Click
        CreditAccountForm.Show()
    End Sub
    'Opens a dialog box to search for invoice. Returns an error message if the invoice
doesn't exist Otherwise, it removes the payment from the invoice
    '. Manager action only
    Private Sub btnPayCancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPayCancel.Click
        CancelPaymentForm.Show()
    End Sub
    'opens form to record payment. Form will contain information on the invoice as well
as the payment amount
    Private Sub btnPayRecord_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPayRecord.Click
        RecordPaymentForm.Show()
    End Sub
    'Prints payment receipt

    Private Sub btnPayBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPayBack.Click
        Me.Close()
    End Sub

    Private Sub BackToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles BackToolStripMenuItem.Click
        Me.Close()
    End Sub

End Class
```

RecordPaymentForm

```
Public Class RecordPaymentForm
    Dim strType As String
    Dim colOneTimeInvoice As New Collection
    Dim colRegularInvoice As New Collection
    Dim payments As New Collection
    Dim objFileManager As New clsFileManager

    'Opens dialog box to allow searching database for invoice. If entered invoice doesn't
exist, it returns an error message. Otherwise, it
    'sets all attribute Labels to visible and pulls in the information needed
    'Closes the form
    Private Sub btnRPBack_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBack.Click
        Me.Close()
    End Sub
```

```
    'Submit payment into invoice
    Private Sub btnRPSubmit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSubmit.Click
        If txtPaid.Text <> "" Then
            objFileManager.SavePayment(lblInvoice.Text, txtPaid.Text)
            MessageBox.Show("Payment made to invoice #" + lblInvoice.Text, "Payment
Recorded", MessageBoxButtons.OK, MessageBoxIcon.Information)
            Me.Close()
        Else
            MessageBox.Show("Please enter an amount paid.", "Error", MessageBoxButtons.OK)
        End If
    End Sub
    Private Sub btnSearch_Click(sender As Object, e As EventArgs) Handles btnSearch.Click
        Dim strId As String
        Dim strInvoiceId As String
        Dim amount As Double
        Dim paid As Double
        Dim tempPay As String
        Dim tempInvID As String
        Dim tempPaid As String
        Dim tempID As String
        Dim totalPaid As Double

        amount = 0
        paid = 0
        totalPaid = 0

        objFileManager.LoadPayments(payments)
        objFileManager.LoadInvoice(colOneTimeInvoice, colRegularInvoice)

        strInvoiceId = InputBox("Enter the Invoice ID you want to search for.", "Invoice
ID")
        For Each inv In colOneTimeInvoice
            inv = inv.subString(4)
            strId = inv.subString(0, inv.indexOf("<")).Trim
            inv = inv.subString(inv.indexof("<"))
            inv = inv.substring(6)
            If (strInvoiceId.Trim = strId.Trim) Then
                lblCustomer.Text = inv.substring(0, inv.indexof("<")).Trim
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(6)
                strType = inv.substring(0, inv.indexof("<"))
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(10)
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(9)
                'lblAmount.Text = inv.subString(0, inv.indexof("<"))
                amount = inv.subString(0, inv.indexof("<"))
                inv = inv.subString(inv.indexof(">") + 1)
                inv = inv.subString(inv.indexof(">") + 1)
                inv = inv.subString(inv.indexof(">") + 1)
                paid = inv.subString(0, inv.indexof("<"))
                For Each pay In payments
                    tempPay = pay
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempPaid = tempPay.Substring(0).Trim
```

```
                    If tempInvID.Trim = strId.Trim Then
                        totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
                    End If
                Next
                lblAmount.Text = (amount - totalPaid)
                lblInvoice.Text = strId
            End If
        Next
        For Each inv In colRegularInvoice
            inv = inv.subString(4)
            strId = inv.subString(0, inv.indexOf("<")).Trim
            inv = inv.subString(inv.indexof("<"))
            inv = inv.substring(6)
            If (strInvoiceId.Trim = strId.Trim) Then
                lblCustomer.Text = inv.substring(0, inv.indexof("<")).Trim
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(6)
                strType = inv.substring(0, inv.indexof("<"))
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(10)
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(9)
                'lblAmount.Text = inv.subString(0, inv.indexof("<"))
                amount = inv.subString(0, inv.indexof("<"))
                inv = inv.subString(inv.indexof(">") + 1)
                inv = inv.subString(inv.indexof(">") + 1)
                inv = inv.subString(inv.indexof(">") + 1)
                paid = inv.subString(0, inv.indexof("<"))
                For Each pay In payments
                    tempPay = pay
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempPaid = tempPay.Substring(0).Trim
                    If tempInvID.Trim = strId.Trim Then
                        totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
                    End If
                Next
                lblAmount.Text = (amount - totalPaid)
                lblInvoice.Text = strId
            End If
        Next
        btnSearch.Enabled = False
    End Sub

    Private Sub txtPaid_KeyDown(sender As Object, e As KeyEventArgs) Handles
txtPaid.KeyDown
        If (e.KeyCode = Keys.Enter) Then
            If txtPaid.Text <> "" Then
                objFileManager.SavePayment(lblInvoice.Text, txtPaid.Text)
                MessageBox.Show("Payment made to invoice #" + lblInvoice.Text, "Payment
Recorded", MessageBoxButtons.OK, MessageBoxIcon.Information)
                Me.Close()
            Else
                MessageBox.Show("Please enter an amount paid.", "Error",
MessageBoxButtons.OK)
            End If
        End If
```

```
    End Sub
End Class
```

CancelPaymentForm

```
Public Class CancelPaymentForm
    Dim strType As String
    Dim colOneTimeInvoice As New Collection
    Dim colRegularInvoice As New Collection
    Dim payments As New Collection
    Dim objFileManager As New clsFileManager

    Private Sub btnSearch_Click(sender As Object, e As EventArgs) Handles btnSearch.Click
        Dim strId As String
        Dim strInvoiceId As String
        Dim tempPay As String
        Dim tempInvID As String
        Dim tempPaid As String

        objFileManager.LoadInvoice(colOneTimeInvoice, colRegularInvoice)
        objFileManager.loadPayments(payments)

        strInvoiceId = InputBox("Enter the Invoice ID you want to search for.", "Invoice
ID")

        lstPayments.Items.Clear()
        For Each inv In colOneTimeInvoice
            inv = inv.subString(4)
            strId = inv.subString(0, inv.indexOf("<")).Trim
            inv = inv.subString(inv.indexof("<"))
            inv = inv.substring(6)
            If (strInvoiceId.Trim = strId.Trim) Then
                lblInvoice.Text = strInvoiceId
                For Each pay In payments
                    tempPay = pay
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempPaid = tempPay.Substring(0).Trim
                    If tempInvID.Trim = strInvoiceId.Trim Then
                        lstPayments.Items.Add(tempPaid)
                    End If
                Next
            End If
        Next
        For Each inv In colRegularInvoice
            inv = inv.subString(4)
            strId = inv.subString(0, inv.indexOf("<")).Trim
            inv = inv.subString(inv.indexof("<"))
            inv = inv.substring(6)
            If (strInvoiceId.Trim = strId.Trim) Then
                lblInvoice.Text = strInvoiceId
                For Each pay In payments
                    tempPay = pay
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempPaid = tempPay.Substring(0).Trim
```

```
                    If tempInvID.Trim = strInvoiceId.Trim Then
                        lstPayments.Items.Add(tempPaid)
                    End If
                Next
            End If
        Next
        btnSearch.Enabled = False
    End Sub

    Private Sub btnBack_Click(sender As Object, e As EventArgs) Handles btnBack.Click
        Me.Close()
    End Sub

    Private Sub btnCancel_Click(sender As Object, e As EventArgs) Handles btnCancel.Click
        Dim tempPay As String
        Dim tempInvID As String
        Dim tempPaid As String
        Dim counter As Integer
        Dim gotIt As Boolean
        gotIt = False
        counter = 1

        If lblInvoice.Text <> "" Then
            If lstPayments.SelectedIndex <> -1 Then
                For Each pay In payments
                    tempPay = pay
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                    tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                    tempPaid = tempPay.Substring(0).Trim
                    If tempPaid.Trim = lstPayments.SelectedItem.Trim And tempInvID.Trim =
lblInvoice.Text.Trim And gotIt = False Then
                        payments.Remove(counter)
                        gotIt = True
                        objFileManager.deleteItem(payments, "Payments.txt")
                        MessageBox.Show("The payment for account # " +
lblInvoice.Text.Trim + " has been cancelled.", "Payment Cancelled", MessageBoxButtons.OK)
                        Me.Close()
                    End If
                    counter = counter + 1
                Next
            Else
                MessageBox.Show("Please select a Payment", "No Payment Selected",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            End If
        Else
            MessageBox.Show("Please select an invoice", "No Invoice Selected",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End If
    End Sub
End Class
```

CreditAccountForm

```
Public Class CreditAccountForm
    Dim strType As String
    Dim colOneTimeInvoice As New Collection
    Dim colRegularInvoice As New Collection
```

```
    Dim objFileOpener As New clsFileManager
    Dim objFileCloser As New clsFileManager
    Dim strInvoiceId As String

    Private Sub btnSearch_Click(sender As Object, e As EventArgs) Handles btnSearch.Click
        Dim strId As String

        objFileOpener.LoadInvoice(colOneTimeInvoice, colRegularInvoice)

        strInvoiceId = InputBox("Enter the Invoice ID you want to search for.", "Invoice
ID")
        For Each inv In colOneTimeInvoice
            inv = inv.subString(4)
            strId = inv.subString(0, inv.indexOf("<")).Trim
            inv = inv.subString(inv.indexof("<"))
            inv = inv.substring(6)
            If (strInvoiceId.Trim = strId.Trim) Then
                lblCustomer.Text = inv.substring(0, inv.indexof("<")).Trim
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(6)
                strType = inv.substring(0, inv.indexof("<"))
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(10)
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(9)
                lblAmount.Text = inv.subString(0, inv.indexof("<"))
                lblInvoice.Text = strId
            End If
        Next
        For Each inv In colRegularInvoice
            inv = inv.subString(4)
            strId = inv.subString(0, inv.indexOf("<")).Trim
            inv = inv.subString(inv.indexof("<"))
            inv = inv.substring(6)
            If (strInvoiceId.Trim = strId.Trim) Then
                lblCustomer.Text = inv.substring(0, inv.indexof("<")).Trim
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(6)
                strType = inv.substring(0, inv.indexof("<"))
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(10)
                inv = inv.subString(inv.indexof("<"))
                inv = inv.substring(9)
                lblAmount.Text = inv.subString(0, inv.indexof("<"))
                lblInvoice.Text = strId
            End If
        Next
    End Sub

    Private Sub btnSubmit_Click(sender As Object, e As EventArgs) Handles btnSubmit.Click
        Dim temp As String
        Dim tempID As String
        Dim tempName As String
        Dim strInvoiceID As String
        Dim payments As New Collection
        Dim tempPay As String
        Dim tempPaid As String
        Dim totalPaid As Double
```

```
        Dim tempInvId As String
        Dim counter As Integer

        Dim id As String
        Dim name As String
        Dim type As String
        Dim employee As String
        Dim payment As String
        Dim paymentMethod As String
        Dim complete As String
        Dim amountPaid As String
        Dim week As String

        strInvoiceID = lblInvoice.Text
        totalPaid = 0
        counter = 1
        objFileOpener.loadPayments(payments)
        objFileOpener.LoadInvoice(colOneTimeInvoice, colRegularInvoice)

        If txtCredit.Text <> "" Then
            For Each inv In colOneTimeInvoice
                temp = inv

                temp = temp.Substring(4)                    'Removes beginning <name>
tags
                tempID = temp.Substring(0, temp.IndexOf("<"))
                id = tempID

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)
                tempName = temp.Substring(0, temp.IndexOf("<"))
                name = tempName

                If (tempID.Trim = strInvoiceID.Trim) Then
                    'Set contents
                    InvoiceInfoForm.lblIIHeading.Text = tempName
                    InvoiceInfoForm.lblIIName.Text = tempName
                    InvoiceInfoForm.lblID.Text = tempID

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(6)

                    type = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(10)

                    employee = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(9)

                    payment = (CDbl(temp.Substring(0, temp.IndexOf("<"))) -
txtCredit.Text)

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(15)
```

```
                    paymentMethod = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(10)

                    complete = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(12)

                    For Each pay In payments
                        tempPay = pay
                        tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                        tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                        tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                        tempPaid = tempPay.Substring(0).Trim
                        If tempInvId.Trim = tempID.Trim Then
                            totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
                        End If
                    Next

                    amountPaid = totalPaid

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(6)

                    week = temp

                    For Each invoice In colOneTimeInvoice
                        temp = invoice
                        temp = temp.Substring(4)
                        tempID = temp.Substring(0, temp.IndexOf("<"))

                        temp = temp.Substring(temp.IndexOf("<"))
                        temp = temp.Substring(6)                            'Removes
beginning <name> tags
                        tempName = temp.Substring(0, temp.IndexOf("<"))

                        If (tempID = InvoiceInfoForm.lblID.Text) Then
                            colOneTimeInvoice.Remove(counter)
                        End If
                        counter = counter + 1
                    Next
                    colOneTimeInvoice.Add("<id>" + id.Trim + " <name>" + name.Trim + "
<type>" + type.Trim + " <employee>" + employee.Trim + " <payment>" + payment.Trim +
                                          " <paymentMethod>" + paymentMethod.Trim + "
<complete>" + complete.Trim + " <amountPaid>" + amountPaid.Trim + " <week>" + week.Trim)

                    objFileCloser.deleteItem(colOneTimeInvoice, "OneTimeInvoices.txt")
                    MessageBox.Show("The payment for account # " + lblInvoice.Text.Trim +
" has been credited", "Account Credited", MessageBoxButtons.OK,
MessageBoxIcon.Information)
                    Me.Close()
                    Exit Sub
                End If
            Next
            counter = 1
            For Each inv In colRegularInvoice
```

```
                temp = inv
                temp = temp.Substring(4)

                tempID = temp.Substring(0, temp.IndexOf("<"))
                id = tempID

                temp = temp.Substring(temp.IndexOf("<"))
                temp = temp.Substring(6)                    'Removes beginning <name>
tags

                tempName = temp.Substring(0, temp.IndexOf("<"))
                name = tempName

                If (tempID.Trim = strInvoiceID.Trim) Then
                    'Set contents
                    InvoiceInfoForm.lblIIHeading.Text = tempName
                    InvoiceInfoForm.lblIIName.Text = tempName
                    InvoiceInfoForm.lblID.Text = tempID

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(6)

                    type = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(10)

                    employee = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(9)

                    payment = (CDbl(temp.Substring(0, temp.IndexOf("<"))) -
txtCredit.Text)

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(15)

                    paymentMethod = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(10)

                    complete = temp.Substring(0, temp.IndexOf("<"))

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(12)

                    For Each pay In payments
                        tempPay = pay
                        tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                        tempInvID = tempPay.Substring(0, tempPay.IndexOf("<"))
                        tempPay = tempPay.Substring(tempPay.IndexOf(">") + 1)
                        tempPaid = tempPay.Substring(0).Trim
                        If tempInvId.Trim = tempID.Trim Then
                            totalPaid = CDbl(totalPaid) + CDbl(tempPaid)
                        End If
                    Next
```

```
                    amountPaid = totalPaid

                    temp = temp.Substring(temp.IndexOf("<"))
                    temp = temp.Substring(6)

                    week = temp

                    For Each invoice In colRegularInvoice
                        temp = invoice
                        temp = temp.Substring(4)
                        tempID = temp.Substring(0, temp.IndexOf("<"))

                        temp = temp.Substring(temp.IndexOf("<"))
                        temp = temp.Substring(6)                            'Removes
beginning <name> tags
                        tempName = temp.Substring(0, temp.IndexOf("<"))

                        If (tempID = InvoiceInfoForm.lblID.Text) Then
                            colRegularInvoice.Remove(counter)
                        End If
                        counter = counter + 1
                    Next
                    colRegularInvoice.Add("<id>" + id.Trim + " <name>" + name.Trim + "
<type>" + type.Trim + " <employee>" + employee.Trim + " <payment>" + payment.Trim +
                                          " <paymentMethod>" + paymentMethod.Trim + "
<complete>" + complete.Trim + " <amountPaid>" + amountPaid.Trim + " <week>" + week.Trim)
                    objFileCloser.deleteItem(colRegularInvoice, "RegularInvoices.txt")
                    MessageBox.Show("The payment for account # " + lblInvoice.Text.Trim +
" has been credited", "Account Credited", MessageBoxButtons.OK,
MessageBoxIcon.Information)
                    Me.Close()
                    Exit Sub
                End If
            Next
            MessageBox.Show("There is no invoice by that name in the database.", "Invoice
Not Found", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End If
    End Sub

    Private Sub btnBack_Click(sender As Object, e As EventArgs) Handles btnBack.Click
        Me.Close()
    End Sub
End Class
```

Chapter 4- Conclusions

Throughout this project, we were to implement the object-oriented approach in completing this program. This approach allowed us to complete the program in iterations, which in turn allowed us to maintain an overall agile program. For all of us, this project was the first opportunity to try out this new approach to programming, and for the most part, all of us had a positive experience working with in the object-oriented approach to programming.

Looking back at the project and how long it took us to complete it, we definitely came to the conclusion that the object-oriented method of programming takes significantly longer than our standard hack-and-slash coding techniques. This is mainly because we wouldn't have to create all of the UML documentation that goes along with programming. However, had we not done all of the documentation before coding the program, there would be no way that our program would be as complete or as agile as it is now. I think that it's obvious to each of us that the object-oriented method to programming is much more efficient than many other methods such as the waterfall or spiral techniques that we learned about at the beginning of the class.

Our views of an object-oriented approach have changed drastically. We began with an understanding of how it worked, but we were not fully able to grasp the concept until we did this final project. We find it to be very helpful in designing a program. For smaller programs it may take longer to finish the project if you create many diagrams, but for larger programs it is much easier and more time efficient to take an object-oriented approach for the analysis, design and implementation. Taking this approach allows you to model the entire program before ever writing any code. This allows you to figure out what classes you need in your program as well as the functionality of each class.

The largest issue we had with an object-oriented approach for the analysis, design and implementation was that we were not fully ready to design a program of this size with this approach. What we had done early in such diagrams as the class diagram was not reflected by our code that we found to work to our need. We attribute this issue to several reasons such as our level of programming experience, inexperience in visual basic, and our inexperience in UML. However, we still feel that an object-oriented approach for the analysis, design and implementation of software systems is highly beneficial.

Chapter 5- Team Contributions

Team Roles

In this project, each member of our team contributed in some way to virtually every aspect of the project, from the sequence diagrams to the code to the WSR's. However, because of the large scope of the project, we found that each of us were responsible for certain aspects of the project and thus, focused more on that part.

Devin Edmundowicz-

I was primarily responsible for making sure the GUI stayed up to date and the forms were linked correctly. I also spent much of the time coding various segments of the project, such as viewing an object like customer, employee, or job, as well as displaying each of those in a list. Lastly, despite not having a set leader of the group, I was still responsible to send out information as to who is responsible for what each week.

Brandon Weber-

I spent a majority of my time working on the overall design of the program, more specifically, the class diagram. I also worked out clsPayment, clsPaymentListManager, and clsFileManager. I worked with Devin on the use case model, as well much of the code on the forms. I made the GUI TOE table as well, along with the final presentation PowerPoint. In general, we all did sequence diagrams, CRC cards, milestone submissions, and WSR submissions.

Eric Sanders-

For this project, I created a few classes. I created classes such as clsCustomer, clsCustomerListManger, clsInvoice, clsInvoiceListManager, clsAppointment, clsAppointmentManager, and clsCheck. clsCheck was a class I decided to design that is comprised of public functions that check different strings such as phone format, name, and date format. It also edits the phone format and date format so that they are all similar. I did a lot of testing for error handling because we did not have a tester. I also helped work on coding most of the forms and did code cleanup that involved fixing errors in our code.

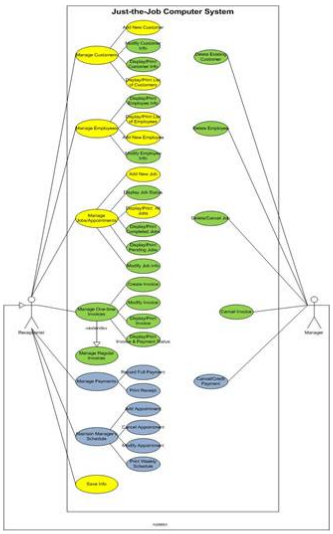
Appendix

Presentation Slides

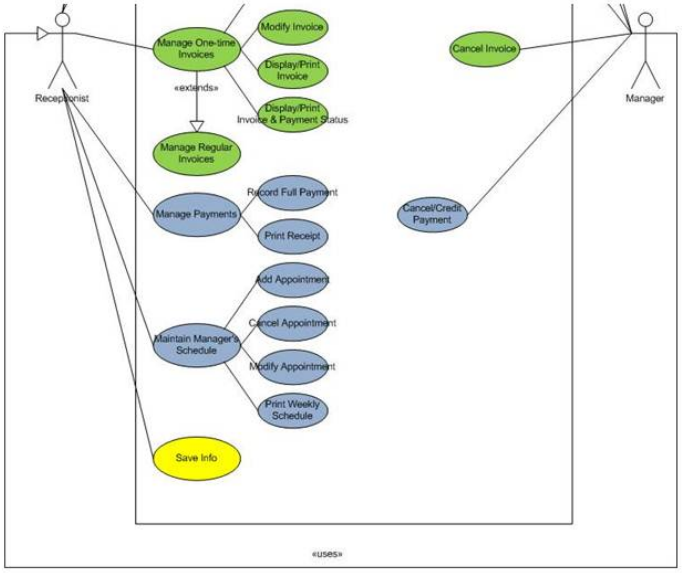
Just-the-Job Cleaning Solutions Business Management Application

Devin Edmundowicz – Developer/GUI
Eric Sanders – Developer/Documentation
Brandon Weber – Developer/Design

Use Case Model



Latest Functionality



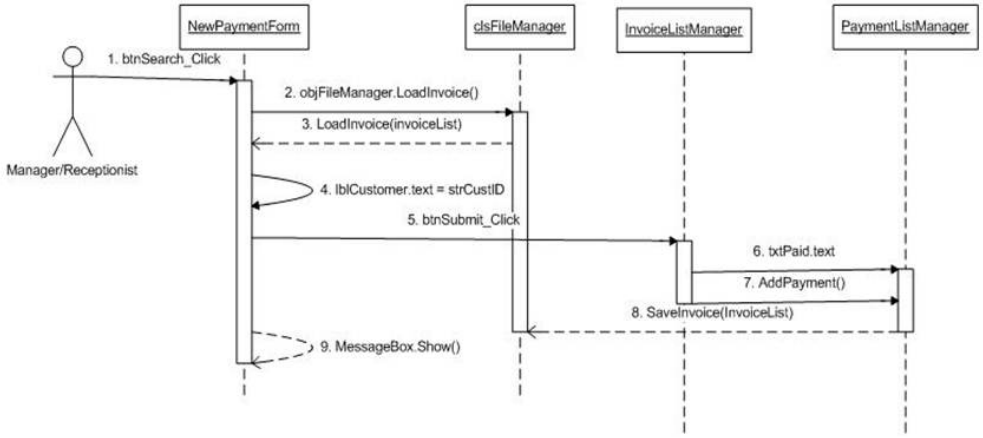
Featured CRC Cards

clsAppointment	
Responsibilty	Collaborator
Create a new instance of an appointment	

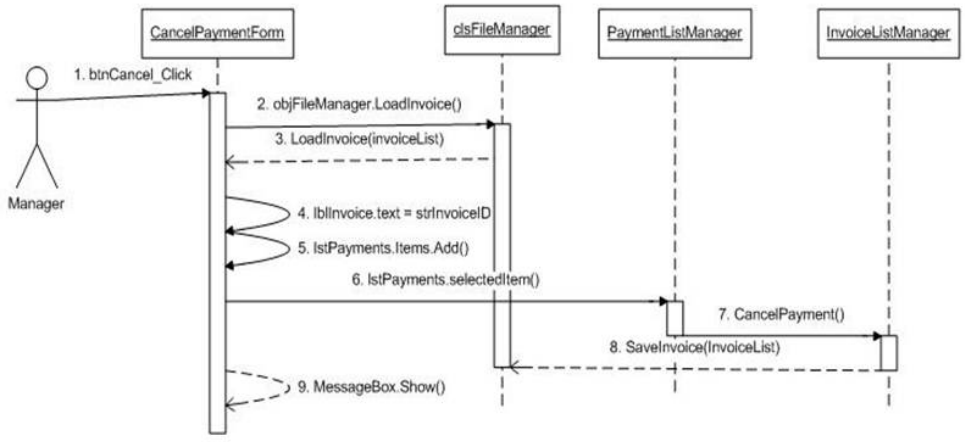
clsAppointmentListManager	
Responsibilty	Collaborator
Display information of an appointment	
Add new appointment to a collection	clsAppointment

clsFileManager	
Responsibility	Collaborator
Save Customers, Employees, Invoices, Jobs, and Appointments to database.	streamWriter
Load Customers, Employees, Invoices, Jobs, and Appointments from database.	streamReader

Record Payment Sequence

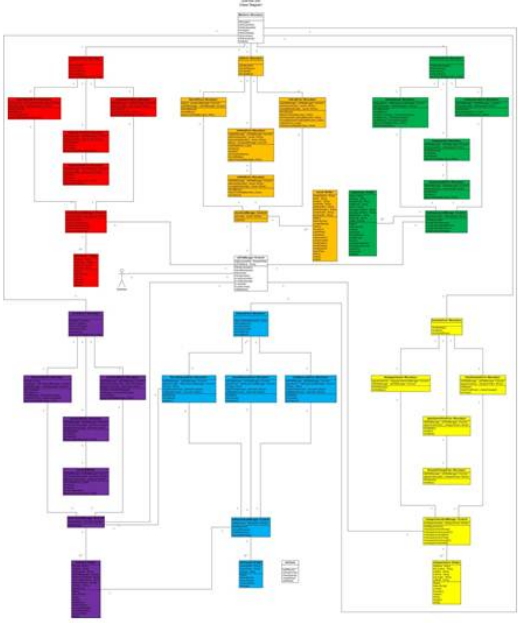


Cancel Payment Sequence

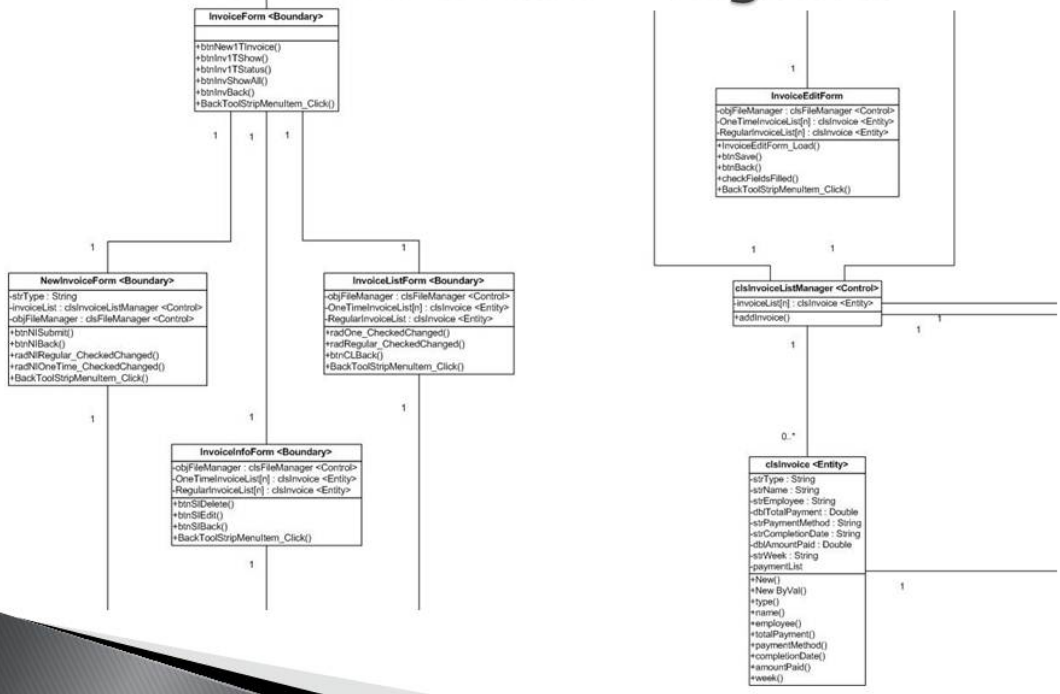


Class Diagram

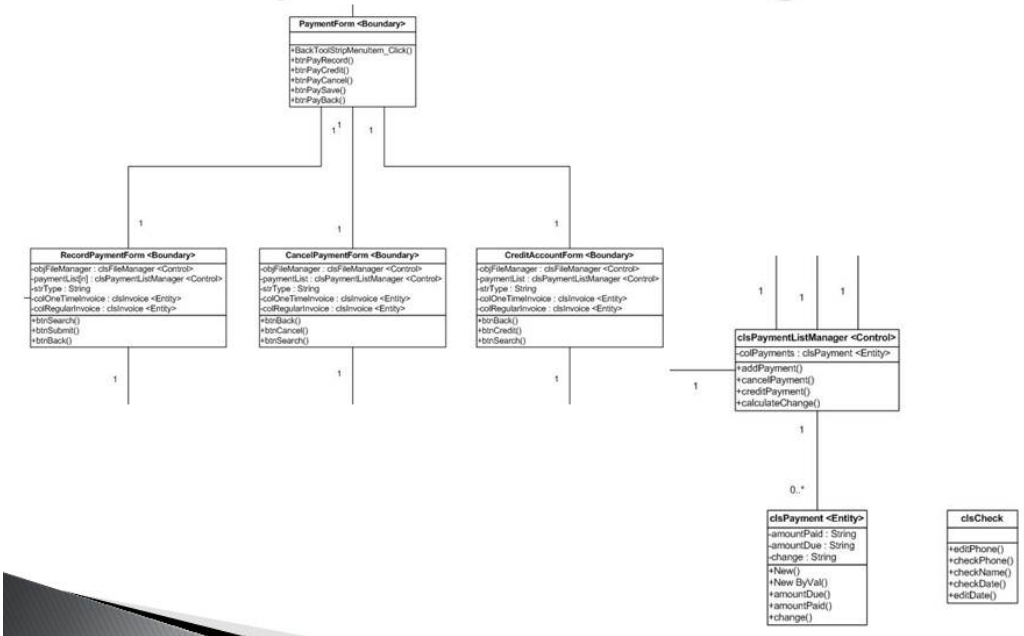
LEGEND
- Customer
- Job
- Employee
- Invoice
- Payment
- Schedule



Invoice Class Diagram



Payment Class Diagram



Tester and Defects

- Error Checking
  - Phone Number
  - Date
- Inconsistent Forms
- Iteration 1 Only

Recent Problems

- Had trouble designing the payments tree
  - Worked on design before code
- Issues implementing a ListView
- Issues overwriting files
  - Needed to close files outside of loops/if statements
- Extensive input checking

Programs Used

- Visual Studio 2010 and 2012
- Visio 2007
- Open Project
- Google Docs
- Microsoft Outlook
- Microsoft Word
- Microsoft PowerPoint

TOE Tables

Mainform.vb

Task	Object	Event
Login to the system: • Username • Password • Login • Activate Groupbox	 UsernameTextBox(txtUsername) PasswordTextBox(txtPassword) LoginButton(btnLogin) MainFormGroupBox(grpMainForm)	 Type Type Click Enable
Open customer management form: • Show form	CustomerButton(btnCustomer)	Click
Open employee management form: • Show form	EmployeeButton(btnEmployees)	Click
Open job management form: • Show form	JobButton(btnJobs)	Click
Open schedule management form: • Show form	ScheduleButton(btnSchedule)	Click
Open invoice management form: • Show form	InvoiceButton(btnInvoices)	Click
Open payment management form: • Show form	PaymentButton(btnPayments)	Click
Exit the program: • Close form	ExitButton(btnExit)	Click

CustomerEditForm.vb

Task	Object	Event
Save changes of a customer: • Name • Address • PhoneNumber • Email • Save	 NameTextBox(txtName) AddressTextBox(txtAddress) PhoneTextBox(txtPhone) EmailTextBox(txtEmail) SaveButton(btnSave)	 Type Type Type Type Click
Close the form without saving changes: • Back	BackButton(btnBack)	Click

CustomerForm.vb

Task	Object	Event
Open new customer form: • Show form	NewCustomerButton(btnCustAdd)	Click
Open customer list form: • Show form	CustomerListButton(btnCustList)	Click
Search individual customer: • Show search box • Customer name • Show form	CustomerInfoButton(btnCustShow) CustomerNameTextBox(txtCustShow) EnterKeyDown(Keys.enter)	Click Type Key Press
Close the form: • Close form	backButton(btnCustBack)	Click

CustomerInfoForm.vb

Task	Object	Event
Delete a customer: • Remove customer	DeleteCustomerButton(btnCIDelete)	Click
Print customer info: • Print customer	PrintButton(btnCIPrint)	Click
Edit customer info: • Show edit form	EditButton(btnCIEdit)	Click
Close the form: • Close form	BackButton(btnCIBack)	Click

CustomerListForm.vb

Task	Object	Event
Load all customers: • Name • Address • Phone • Email	NameListBox(lstName) AddressListBox(lstAddress) PhoneListBox(lstPhone) EmailListBox(lstEmail)	Load Load Load Load
Print list of customers: • Print list	PrintButton(btnCLPrint)	Click
Close the form: • Close form	BackButton(btnCLBack)	Click

EmployeeInfo.vb

Task	Object	Event
Delete an employee: • Remove employee	DeleteButton(btnEIRemove)	Click
Edit an employee: • Show edit form	EditButton(btnEIEdit)	Click
Close the form: • Close form	BackButton(btnEIBack)	Click

EmployeeListForm.vb

Task	Object	Event
Load all employees: • Name • Cell Phone • Job Position • Wages	 NameListBox(lstName) PhoneListBox(lstPhone) PositionListBox(lstPosition) WagesListBox(lstWages)	 Load Load Load Load
Print list of all employees: • Print List	 PrintButton(btnNLPrint)	 Click
Close the form: • Close form	 BackButton(btnNLBack)	 Click

EmployeesForm.vb

Task	Object	Event
Open new employee form: • Show form	 NewEmployeeButton(btnnewEmployee)	 Click
Open employee list form: • Show form	 EmployeeListButton(btnEmpList)	 Click
Search individual employee: • Show search box • Employee name • Show form	 EmployeeInfoButton(btnEmpShow) EmployeeNameTextBox(txtEmpShow) EnterKeyDown(Keys.enter)	 Click Type Key Press
Close the form: • Close form	 backButton(btnEmpBack)	 Click

JobForm.vb

Task	Object	Event
Open new job form: • Open form	 AddJobButton(btnNewJob)	 Click
Open all jobs form: • Open form	 AllJobsButton(btnJobAll)	 Click
Open completed jobs form: • Open form	 CompletedJobsButton(btnJobCompleted)	 Click
Open pending jobs form: • Open form	 PendingJobsButton(btnJobPending)	 Click
View a specific job: • Show search box • Check search box • Show form	 JobInfoButton(btnJobStatus) NameTextBox(txtJobSearch) EnterKeyDown(Keys.enter)	 Click Type Key Press

InvoiceForm.vb

Task	Object	Event
Open new invoice form: • Open form	New1InvoiceButton(btnNew1TInvoice)	Click
Show an invoice: • Check name • Open form	NameTextBox(txtInvShow) Show1TButton(btnInv1TShow)	Type Click
Show an invoice payment: • Check name • Show status box • Display Status	NameTextBox(txtInvShow) Status1TButton(btnInv1TStatus) StatusLabel(lblOTShow)	Type Click Show
Open new invoice form: • Open form	NewRInvoiceButton(btnNewRInvoice)	Click
Show an invoice: • Check name • Open form	NameTextBox(txtInvRShow) ShowRButton(btnInvRShow)	Type Click
Show an invoice payment: • Check name • Show status box • Display Status	NameTextBox(txtInvRShow) StatusButton(btnInvStatus) StatusLabel(lblRShow)	Type Click Show
Close the form: • Close form	BackButton(btnInvBack)	Click

JobEditForm.vb

Task	Object	Event
Delete a job: • Remove Job	DeleteButton(btnJEDelete)	Click
Print a job: • Print job	PrintButton(btnJEPrint)	Click
Close the form: • Close form	BackButton(btnJEBack)	Click

JobListForm.vb

Task	Object	Event
Load all jobs: • Load completed name • Load completed client • Load completed start • Load completed dead • Load completed lead • Load pending name • Load pending client • Load pending start • Load pending dead • Load pending lead	 NameListBox(lstCompletedName) ClientListBox(lstCompletedClient) StartListBox(lstCompletedStart) DeadlineListBox(lstCompletedDeadline) LeadListBox(lstCompletedLead) NameListBox(lstPendingName) ClientListBox(lstPendingClient) StartListBox(lstPendingStart) DeadlineListBox(lstPendingDeadline) LeadListBox(lstPendingLead)	 Load Load Load Load Load Load Load Load Load Load
Print list of jobs: • Print list	 PrintButton(btnJLPrint)	 Click
Close the form: • Close form	 BackButton(btnJLBack)	 Click

JobViewForm.vb

Task	Object	Event
Open edit form: • Open form	 EditButton(btnEdit)	 Click
Change job status: • Change status • Update label	 StatusButton(btnChangeStatus) StatusLabel(lblStatus)	 Click Show
Close the form: • Close form	 BackButton(btnBack)	 Click

NewAppointment.vb

Task	Object	Event
Add a new appointment: • Submit info • Clear name • Clear location • Clear date • Clear time • Clear length • Clear brief	 SubmitButton(btnNASubmit) NameTextBox(txtNAName) LocationTextBox(txtNALocation) DateTextBox(txtNADate) TimeTextBox(txtNAStart) LengthTextBox(txtNALength) BriefTextBox(txtNABrienf)	 Click Clear Clear Clear Clear Clear Clear
Close the form: • Close form	 BackButton(btnNABack)	 Click

NewCustomerForm.vb

Task	Object	Event
Add a new customer: • Submit info • Clear name • Clear address • Clear phone number • Clear email	 SubmitButton(btnNCSubmit) NameTextBox(txtNCName) AddressTextBox(txtNCAddress) PhoneTextBox(txtNCPhone) EmailTextBox(txtNCEmail)	 Click Clear Clear Clear Clear
Close the form: • Close form	 BackButton(btnNCBack)	 Click

NewEmployee.vb

Task	Object	Event
Add a new employee: • Submit info • Clear name • Clear address • Clear phone • Clear cell • Clear email • Clear emergency name • Clear emergency phone • Clear position • Clear radio commission • Clear radio hourly • Clear radio salary • Hide commission text • Hide hourly text • Hide salary text • Hide pay label	 SubmitButton(btnNESubmit) NameTextBox(txtNEName) AddressTextBox(txtNEAddress) PhoneTextBox(txtNEHome) CellTextBox(txtNEMobile) EmailTextBox(txtNEEmail) ENameTextBox(txtNENameEM) EPhoneTextBox(txtNEPhone) PositionTextBox(txtNEPosition) ComRadioButton(radNECommission) HourlyRadioButton(radNEHourly) SalaryRadioButton(radNESalary) ComTextBox(txtNECommWages) HourlyTextBox(txtNEHourly) SalaryTextBox(txtNESalary) PayLabel(lblNEPay)	 Click Clear Clear Clear Clear Clear Clear Clear Clear Clear Clear Clear Hide Hide Hide Hide
Show commission: • Select commission • Show label • Show textbox	 ComRadioButton(radNECommission) PayLabel(lblNEPay) ComTextBox(txtNECommWages)	 Check Show Show
Show hourly: • Select hourly • Show label • Show textbox	 HourlyRadioButton(radNEHourly) PayLabel(lblNEPay) HourlyTextBox(txtNEHourly)	 Check Show Show
Show salary: • Select salary • Show label • Show textbox	 SalaryRadioButton(radNESalary) PayLabel(lblNEPay) SalaryTextBox(txtNESalary)	 Check Show Show
Close the form: • Close form	 BackButton(btnNEBack)	 Click

NewInvoiceForm.vb

Task	Object	Event
Add a new invoice: • Submit info • Clear r-invoice radio • Clear ot-invoice radio • Clear name • Clear employee • Clear payment due • Clear payment method • Clear deadline	SubmitButton(btnNISubmit) RagularRadio(radNIRegular) OneTimeRadio(radNIOneTime) NameTextBox(txtNIName) EmployeeTextBox(txtNIEmployee) PaymentTextBox(txtNIPay) MethodComboBox(comMethod) DeadlineTextBox(txtNIComplete)	Click Clear Clear Clear Clear Clear Clear Clear
Close the form: • Close form	BackButton(btnNIBack)	Click

NewJobForm.vb

Task	Object	Event
Add a new job: • Submit info • Clear name • Clear client • Clear location • Clear start • Clear deadline • Clear description • Clear lead • Clear lead title • Clear employee 1 • Clear emp 1 title • Clear employee 2 • Clear emp 2 title • Clear employee 3 • Clear emp 3 title	SubmitButton(btnNJSubmit) NameTextBox(btnNJName) ClientTextBox(btnNJClient) LocationTextBox(btnNJLocation) StartTextBox(btnNJDate) DeadlineTextBox(btnNJDeadline) DescTextBox(btnNJDescription) LeadTextBox(btnNJLead) LeadTitleTextBox(btnNJLeadTitle) Emp1TextBox(txtNJEmployee1) Emp1TitleTextBox(txtNJTitle1) Emp2TextBox(txtNJEmployee2) Emp2TitleTextBox(txtNJTitle2) Emp3TextBox(txtNJEmployee3) Emp3TitleTextBox(txtNJTitle3)	Click Clear Clear Clear Clear Clear Clear Clear Clear Clear Clear Clear Clear Clear Clear
Close the form: • Close form	BackButton(btnNJBack)	Click

Peer Evaluation Forms

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

2. **All defects detected and reported to the developers were addressed effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

4. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

5. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **The class diagram included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

8. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **The team appeared to be better organized and improved since the beginning of the semester.**
    (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

2. **All defects detected and reported to the developers were addressed effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

3. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

4. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

5. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **The class diagram included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **The team appeared to be better organized and improved since the beginning of the semester.**
    (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3
TEAM NAME: RED TEAM

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

2. **All defects detected and reported to the developers were addressed effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

4. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

5. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **The class diagram included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree) need some work

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **The team appeared to be better organized and improved since the beginning of the semester.**
    (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

2. **All defects detected and reported to the developers were addressed effectively.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

3. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

4. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

5. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

6. **The class diagram included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

8. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

10. **The team appeared to be better organized and improved since the beginning of the semester.**
    (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

2. **All defects detected and reported to the developers were addressed effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

3. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

4. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

5. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

6. **The class diagram included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. **The team appeared to be better organized and improved since the beginning of the semester.**
    (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

2. **All defects detected and reported to the developers were addressed effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

4. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

5. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **The class diagram included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **The team appeared to be better organized and improved since the beginning of the semester.**
    (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

Really good job!

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

2. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

3. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

5. **The updated UML class diagram was clearly improved and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

6. **All the Entity, Boundary and Control classes were presented and explained well.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

8. **The updated UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

10. **The defect-list and recommendations received from the QA(s) was discussed and addressed well.**
    (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

11. **I have the following additional constructive suggestions and/or comments for the team:**

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

2. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

5. **The updated UML class diagram was clearly improved and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **All the Entity, Boundary and Control classes were presented and explained well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The updated UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **The defect-list and recommendations received from the QA(s) was discussed and addressed well.**
    (Strongly Disagree) 1 2 3 4 5 (Strongly Agree) N/A

11. **I have the following additional constructive suggestions and/or comments for the team:**

- Give feedback to user (e.g "want to delete?")
- Email the QA and ask about the defect_list

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

2. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

3. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

4. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

5. **The updated UML class diagram was clearly improved and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **All the Entity, Boundary and Control classes were presented and explained well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The updated UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **The defect-list and recommendations received from the QA(s) was discussed and addressed well.**
    (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

11. **I have the following additional constructive suggestions and/or comments for the team:**

Great added features and functionality

Cool login!

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new functionality for this iteration.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

2. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

3. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

5. **The updated UML class diagram was clearly improved and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

6. **All the Entity, Boundary and Control classes were presented and explained well.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The updated UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. **The defect-list and recommendations received from the QA(s) was discussed and addressed well.**
    (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

11. **I have the following additional constructive suggestions and/or comments for the team:**

You have alot of GUI that could be changed to show more info without having to select through three or four forms.

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: RED

Please rate the team's work and presentation by circling the right number.

1. **The demonstration was complete and included all the expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 (4) (5) (Strongly Agree)

2. **The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

3. **The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

4. **The interaction diagrams seemed to be complete and represented the message passing among classes.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

5. **The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

6. **The GUI design was well modeled and presented.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

8. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

9. **The team appears to be well organized and functioning.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **I have the following additional constructive suggestions and/or comments for the team:**

Shrink screen when you hide completed/pend jobs.

frmNewCustomerForm seems a bit redundant.

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

2. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

3. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

5. **The updated UML class diagram was clearly improved and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

6. **All the Entity, Boundary and Control classes were presented and explained well.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

8. **The updated UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree)

10. **The defect-list and recommendations received from the QA(s) was discussed and addressed well.**
    (Strongly Disagree) 1 2 3 4 5 (Strongly Agree) NA

11. **I have the following additional constructive suggestions and/or comments for the team:**

Nice GUI and functionality
Diagrams could be cleaned up a bit

TEAM EVALUATION AND FEEDBACK FORM

ITERATION 2
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree) Missing print

2. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree) Cluttered, Missing cast confirm info or change, etc.

5. **The updated UML class diagram was clearly improved and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree) good diagram, make sure

6. **All the Entity, Boundary and Control classes were presented and explained well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree) everything connects to what it creates

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The updated UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **The defect-list and recommendations received from the QA(s) was discussed and addressed well.**
    (Strongly Disagree) 1 2 3 4 5 (Strongly Agree) N/A (sorry guys that your tester sucks)

11. **I have the following additional constructive suggestions and/or comments for the team:**

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demonstration was complete and included all the expected functionality for this iteration.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

2. **The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and represented the message passing among classes.**
   (Strongly Disagree) 1 (2) 3 4 5 (Strongly Agree) - didn't match program

5. **The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree) - great diagram

6. **The GUI design was well modeled and presented.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

7. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree) - worried abt interaction diagrams and CRC cards

8. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The team appears to be well organized and functioning.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. **I have the following additional constructive suggestions and/or comments for the team:**

- Need the specific info. the project description asks for each of cust., empty., etc.
- No save, you did some bells and whistles w/o finishing full iteration requirements

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demonstration was complete and included all the expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

2. **The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

3. **The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and represented the message passing among classes.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

5. **The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

6. **The GUI design was well modeled and presented.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

7. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

8. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

9. **The team appears to be well organized and functioning.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. **I have the following additional constructive suggestions and/or comments for the team:**

seems like some classes were not being re used correctly

speak up when presenting

add details to interaction diagrams

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demonstration was complete and included all the expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

2. **The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and represented the message passing among classes.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

5. **The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **The GUI design was well modeled and presented.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

9. **The team appears to be well organized and functioning.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. **I have the following additional constructive suggestions and/or comments for the team:**

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demonstration was complete and included all the expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

2. **The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

3. **The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and represented the message passing among classes.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree) need work

5. **The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

6. **The GUI design was well modeled and presented.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

7. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree) need work

8. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The team appears to be well organized and functioning.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. **I have the following additional constructive suggestions and/or comments for the team:**

focus more on the OO Design.

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demonstration was complete and included all the expected functionality for this iteration.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

2. **The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 (2) 3 4 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and represented the message passing among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

5. **The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **The GUI design was well modeled and presented.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

8. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The team appears to be well organized and functioning.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. **I have the following additional constructive suggestions and/or comments for the team:**

Work on CRC cards and sequence diagrams
add save and load functions, add print functions

TEAM EVALUATION AND FEEDBACK FORM

ITERATION: 1

TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demonstration was complete and included all the expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

2. **The use case diagram and scenarios were complete (i.e. included both basic and alternate flow) and easy to understand.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The CRC cards described effectively all candidate analysis classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 (3) 4 5 (Strongly Agree)

4. **The interaction diagrams seemed to be complete and represented the message passing among classes.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

5. **The class diagram was well organized based on the 3-tier architecture and included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

6. **The GUI design was well modeled and presented.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

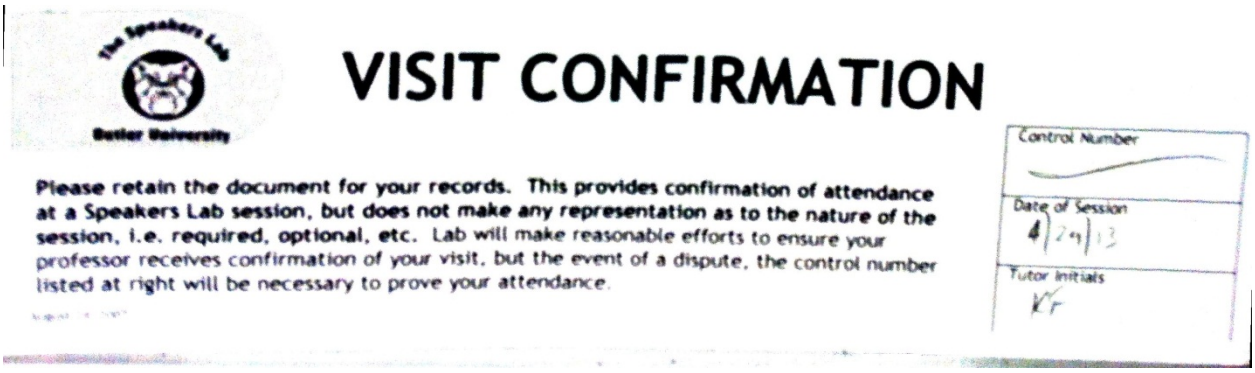
7. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

9. **The team appears to be well organized and functioning.**
   (Strongly Disagree) 1 2 3 (4) 5 (Strongly Agree)

10. **I have the following additional constructive suggestions and/or comments for the team:**

Add file managers
Additional CRC cards could help
and more detail could be added



The Speakers Lab
Butler University

VISIT CONFIRMATION

Please retain the document for your records. This provides confirmation of attendance at a Speakers Lab session, but does not make any representation as to the nature of the session, i.e. required, optional, etc. Lab will make reasonable efforts to ensure your professor receives confirmation of your visit, but the event of a dispute, the control number listed at right will be necessary to prove your attendance.

Control Number	
Date of Session	4/29/13
Tutor Initials	KF

TEAM EVALUATION AND FEEDBACK FORM

ITERATION #: 3
TEAM NAME: Red Team

Please rate the team's work and presentation by circling the right number.

1. **The demo was complete and included all the new expected functionality for this iteration.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

2. **All defects detected and reported to the developers were addressed effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

3. **The use case diagram was clearly improved from the previous iteration and the scenarios were complete (i.e. included both basic and alternate flow).**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

4. **The CRC cards for this iteration described effectively all candidate classes with their responsibilities and collaborators.**
   (Strongly Disagree) 1 2 3 4 5 (Strongly Agree) ✓

5. **The interaction diagrams seemed to be complete and presented the message passing among classes effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

6. **The class diagram included all attributes, operations and relationships among classes.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

7. **The GUI design was improved and presented well.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

8. **The UML design diagrams appear to be accurately mapped to the implementation (code) of the system.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

9. **The overall presentation was well prepared and delivered by the team effectively.**
   (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)

10. **The team appeared to be better organized and improved since the beginning of the semester.**
    (Strongly Disagree) 1 2 3 4 (5) (Strongly Agree)