|  |
| --- |
|  |
| Electric Supply Customer Management System |
| Synopsis |
|  |
| **Dipanwita Dey ( 105136520 ) Susmita Podder( 105140695 )** |
|  |

|  |
| --- |
|  |

Table of Contents

[1. Introduction 4](#_Toc330410211)

[1.1 Background 4](#_Toc330410212)

[1.2 Objective 4](#_Toc330410213)

[1.3 Purpose and Scope 5](#_Toc330410214)

[1.3.1 Purpose 5](#_Toc330410215)

[Scope 6](#_Toc330410216)

[1.3.2 Applicability 6](#_Toc330410217)

[1.4 achievement 6](#_Toc330410218)

[1.5 Organisation of report 6](#_Toc330410219)

[2. SURVEY OF TECHNOLOGY 6](#_Toc330410220)

[Programming FRAMEWORK (.NET 4) 7](#_Toc330410221)

[Programming Language (C#) 7](#_Toc330410222)

[Database - MySQL 7](#_Toc330410223)

[3. REQUIREMENTS AND ANALYSIS 8](#_Toc330410224)

[3.1 Problem Definition 8](#_Toc330410225)

[3.1.1 Existing System 8](#_Toc330410226)

[3.1.2 Documents maintained 9](#_Toc330410227)

[3.1.3 Work To Be Done 10](#_Toc330410228)

[3.2 Requirements Specification 10](#_Toc330410229)

[3.2.1 Functional Requirements 10](#_Toc330410230)

[3.2.2 Non-Functional Requirements 13](#_Toc330410231)

[3.3 Planning and Scheduling 13](#_Toc330410232)

[Gantt chart 13](#_Toc330410233)

[Tracking Gantt 14](#_Toc330410234)

[Pert chart (Network Diagram) 14](#_Toc330410235)

[3.4 Hardware and Software Requirements 15](#_Toc330410236)

[3.4.1 Hardware Requirements 15](#_Toc330410237)

[3.4.2 Software Requirements 15](#_Toc330410238)

[3.5 PRELIMINARY PRODUCT DESCRIPTION 16](#_Toc330410239)

[3.6 CONCEPTUAL MODELS 17](#_Toc330410240)

[3.6.1 E-R Diagram 17](#_Toc330410241)

[3.6.2 Context Diagram 20](#_Toc330410242)

[3.6.3 Data Flow Diagram 20](#_Toc330410243)

[4. system design 21](#_Toc330410244)

[4.1 Basic Modules 21](#_Toc330410245)

[4.2 data design 21](#_Toc330410246)

[4.2.1 Schema Design 21](#_Toc330410247)

[4.2.2 data integrity and constraints 21](#_Toc330410248)

[4.3 procedural design 21](#_Toc330410249)

[4.3.1 logic diagrams 21](#_Toc330410250)

[4.3.2 data structures 21](#_Toc330410251)

[4.3.3 Algoithitms and design 24](#_Toc330410252)

[4.4 user interface design 24](#_Toc330410253)

[4.4.1 main window 24](#_Toc330410254)

[4.4.2 login window 25](#_Toc330410255)

[4.4.3 customer browser window 26](#_Toc330410256)

[4.4.4 payment window 27](#_Toc330410257)

[4.4.5 connection window 27](#_Toc330410258)

[4.4.6 estimate quotation window 28](#_Toc330410259)

[4.4.7 work order window 28](#_Toc330410260)

[4.4.8 settings window 29](#_Toc330410261)

[4.4.9 contractor billl window 29](#_Toc330410262)

[4.4.10 contractor browser window 30](#_Toc330410263)

[4.4.11 employeee browser window 30](#_Toc330410264)

[4.4.12 report window 31](#_Toc330410265)

[4.4.13 search window 32](#_Toc330410266)

[4.5 security issues 33](#_Toc330410267)

[4.6 test cases design 33](#_Toc330410268)

[5. Implimentation And Testing 36](#_Toc330410269)

[5.1 Implimentation Approaches 36](#_Toc330410270)

[5.2 coding details and code efficiency 36](#_Toc330410271)

[5.2.1 code efficiency 36](#_Toc330410272)

[5.3 testing approach 36](#_Toc330410273)

[5.3.1 unit testing 36](#_Toc330410274)

[5.3.2 integrated testing 36](#_Toc330410275)

[5.4 modification and improvements 36](#_Toc330410276)

[6. result and discussion 36](#_Toc330410277)

[6.1 test reports 36](#_Toc330410278)

[user documentation 37](#_Toc330410279)

[7. conclusions 37](#_Toc330410280)

[7.1 conclusion 37](#_Toc330410281)

[7.2 limitation of the system 37](#_Toc330410282)

[7.3 future scope of the project 37](#_Toc330410283)

[8. REFERENCES 38](#_Toc330410284)

# Introduction

## Background

Electricity is the ultimate need for almost every citizen of our country. With the advent of new technology we are more dependent on electricity now. Currently the electric supply offices use paper books and ledgers to track & manage customer applications, complaints. As a result, it takes longer time and extra effort to serve customers with existing inefficient system. Electric Supply Customer Management System (will be referred as **ESCMS** in this document) is acomputerized solution for managing customer needs in Electric Supply offices. Electric Supply Customer Management System will enable electric supply offices to maintain computerized records and manage customer needs more efficiently with help of sophisticated customer management techniques and technologies.

## Objective

The main objective of this project is to automate the process in an Electric Supply office. Electric Supply Customer Management System will be used by Electric Supply Employees to entervarious data about the Customers and their needs. The Employees will update the status of customerrequests; track the progress of the work & transactions made in Electric Supply CustomerManagementSystem. It can generate reports and receipts required to serve customer request andqueries. In a nutshell **ESCMS** will be the backbone of an Electric Supply office and it will be a next generation solution for better customer service and customer satisfaction.

## Purpose and Scope

### Purpose

The purpose of the project is to computerize the workflow of an Electric supply office. The below mentioned diagram shows the existing Customer Management System of Electric Supply offices which will be completely computerized with the help of this software.



The main advantages of Electric Supply Customer Management System over existing traditional paper book and ledger system are given below:

1. It is faster as the computer is doing the searching and fetching of data.

2. It is more efficient as it reduces the need of more employees.

3. It is more accurate.

4. It can generate paper report and main ledgers to help transition from older system.

5. It can store the data in a centralized server so that data will available to use anywhere. So the auditing of the data can be done without physically being present to the site.

## Scope

Currently this software is aimed for a single electric supply office customer management. It can be extended to support networked multiple electric supply office and have a centralized database and to serve wider range of customers of Electric Supply around the country.

We have developed this for Desktop Computers running on Windows Operating System. It can be enhanced to support UNIX / Linux, MAC OSX Operating systems.

Our software will not be integrated with Electric Billing System right now. But in future we can easily extend to support that.

### Applicability

ESCMS will not only solve the complexity of the electric supply management but also speed up the process of a new connection. It is helpful to both users and customers. With its simple to understand GUI, ESCMS helps in managing the entire process more easily, efficiently and accurately for the electric office employees. On the other hand, it helps customers serve better.

## achievement

After completition of the project, we gained overall working knowledge on c#, XAML, MYSQL, WPF and Microsoft Visusal C# 2010 Express Editions. We used to know the therories of programming languages like C# but this project gave us the ability to understand the application of the language in real life and not only that, we now know how an entire business logic can be controlled using a software.

## Organisation of report

This report will carry,

Survey of Technology.

Requirement Analysis.

Planning and Scheduling.

System Design.

Data Design.

Logical Diagrams.

Testing Reports.

Snapshots.

# SURVEY OF TECHNOLOGY

This software will follow Object Oriented Programming Paradigm and use below mentionedareas.

**Front End/ GUI Tools**: Visual Studio 2010, .NET 4.0, C#

**Backend:** MySQL

**Networking Technologies:** TCP/IP

**Operating Systems:** Windows XP, Windows 7

**Application Type:** ERP application, Database Management System.

## Programming FRAMEWORK (.NET 4)

The .NET 4 Framework is Microsoft's platform for building applications that have visually stunning user experiences, seamless and secure communication, and the ability to model a range of business processes. The .Net Framework consists of:

**Common Language Runtime** – provides an abstraction layer over the operating system

**Base Class Libraries** – pre-built code for common low-level programming tasks

**Development frameworks & technologies** – reusable, customizable solutions for larger programming tasks.

The framework's Base Class provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The class library is used by programmers, who combine it with their own code to produce applications.

## Programming Language (C#)

C# is a type-safe, object-oriented language that is simple yet powerful, allowing programmers to build a breadth of applications.

C# is a multi-paradigm programming language encompassing imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines.

It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure.

C# is intended to be a simple, modern, general-purpose, object-oriented programming language.

## Database - MySQL

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout its history.

The MySQL Community Edition includes:

* Pluggable Storage Engine Architecture
* **Multiple Storage Engines**: InnoDB , MyISAM, NDB (MySQL Cluster),Memory ,Merge , Archive, CSV
* MySQL Replication to improve application performance and scalability
* MySQL Partitioning to improve performance and management of large database applications
* Stored Procedures to improve developer productivity

Electric Supply Customer Management System

# REQUIREMENTS AND ANALYSIS

## Problem Definition

### Existing System

The existing system is traditional paper books and ledger system where several registers are maintained to store user request and to track other details about the request. The flow diagram of how a customer request executed now is shown below:



### Documents maintained

**Application Register:** Application Number, Name, Address with Contact Number, Load, Initial

Deposit Amount, Application Received Date, Quotation Amount, Quotation Sent Date, Amount

Received On, Service Connection Number.



* **Service Connection Register:** Service Connection Number, Application Number, Name,Address with Contact Number, Quotation Amount, Amount Received Date, Work Assigned To (Contractor), Work Completed on Date.
* **Meter Movement Register:** Service Connection Number, Name Address with Contact Number, Meter Number, Seal Number, Meter Issue Date, Work Assigned To (Staff), Date of Connection
* **Estimation Sheet per Application:** contains input Application Number, name, Address with Contact Number, Wire Length Required, Angle Type and calculates Weight of Angle, and Quotation amount

### Work To Be Done

We will incorporate the above mentioned workflow of an Electric Supply Customer Management System in an automatic computerized way.

## Requirements Specification

### Functional Requirements

#### Apply for new connection

**Introduction**

Customer can apply for a new connection.

**Inputs**

Relevant customer data like name, address, contact number, type, payment.

**Processing**

Employee will enter the data in the ESCMS and create a new connection entry.

**Outputs**

ESCMS will generate an application number for future reference and will provide customer a acknowledgement receipt.

#### check connection request status

**Introduction**

Customer can check the new connection status.

**Inputs**

Application number & customer name.

**Processing**

Employee will enter application number & customer name in the ESCMS and it will search the status & display in the screen.

**Outputs**

Customer will get the status information from employee and he may request for a printed status also.

#### Create a vendor task

**Introduction**

Employee will create a task for vendor.

**Inputs**

Application number, customer details, task details.

**Processing**

Employee will enter details in the ESCMS and it will pick a vendor & assign the task.

**Outputs**

Vendor will get a notification about the task and a printed work order.

#### vendor task status update

**Introduction**

Vendor will update the task status to employee and receive partial payment.

**Inputs**

Application number, task details, proof of task status.

**Processing**

Employee will enter details in the ESCMS and update the system. System will approve the payment order.

**Outputs**

Vendor will get a notification about the task update and a printed payment order.

#### Generate report

**Introduction**

Employee will choose the kind report to be printed and system will create the details of the report and print it.

**Inputs**

Report Type, Area, Time Frame.

**Processing**

Employee will enter details in the ESCMS and the system will collate data. System will print the report.

**Outputs**

A printed report will generated.

### Non-Functional Requirements

* **Efficiency**:

It will be efficient as it reduces manual labor and searching.

* **Backup**:

The employees will take regular print out of daily reports and take back up. Digital back up can be taken in a regular interval.

* **Documentation**:

ESCMS will have user manual and help documents.

* **Maintainability**:

It is designed such a way that it can be maintained with minimal effort.

* **Performance**:

The response time of ESCMS will be very fast. So it will be efficient enough to cater the customer.

* **Privacy**:

The data will be encrypted and the user data will not be shared with third party.

* **Security**:

ESCMS will use secure connection and enhanced security measures to protect data.

* **Usability**:

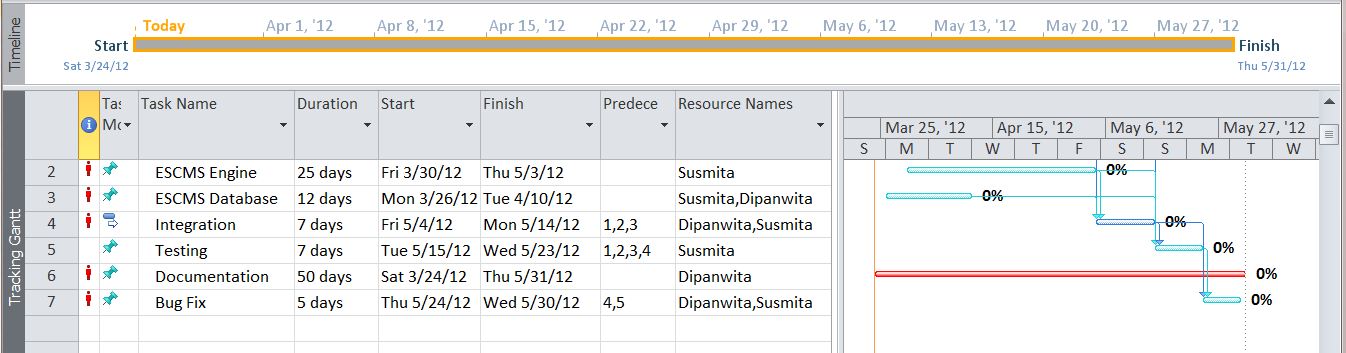
It will be very user friendly and usable by any person with minimal computer knowledge.

## Planning and Scheduling

### Gantt chart



### Tracking Gantt



### Pert chart (Network Diagram)



## Hardware and Software Requirements

### Hardware Requirements

* Computer that has a 1.6GHz or faster processor
* 1 GB (32 Bit) or 2 GB (64 Bit) RAM
* 10 MB of available hard disk space
* DVD-ROM Drive / USB Port

### Software Requirements

* Windows XP (x86) with Service Pack 3 / Windows Vista (x86 & x64) with
* Service Pack 2 / Windows 7 (x86 & x64)
* Microsoft .NET 4.0

## PRELIMINARY PRODUCT DESCRIPTION

Electric Supply Customer Management System will upgrade the existing system, so it needs to havegood support for existing system as well. It will collect the data from customers & employees andpopulate records which will match existing paper book registers. So that employees can take printout and maintain similar records. Electric Supply Customer Management System consists of threemain modules:

* ESCMS GUI
* ESCMS Engine
* ESCMS Database



## CONCEPTUAL MODELS

### E-R Diagram

We will design a RDBMS for File Management System. The entities and their attributes are listed below. Attributes in Bold letter is the unique key.



**Relationship between Entities:**

Electric Supply office has Customers1: N

Electric Supply office has Contractors1: N

Electric Supply office has Employees1: N

Customer does Requests 1: N

Electric Supply serves Requests 1: N

User uses Service Connection 1: N

Employees provides Estimates M: N





### Context Diagram



### Data Flow Diagram



# system design

## Basic Modules

*ECMS contains following main modules:*

* ECMS GUI: all the codes containing WPF GUI designing are written in this module. It contains many sub modules such as:

1. Connection: all the codes of Connection window and its sub windows are written in this module.
2. Contractor: all the codes of contractor window and its sub windows are written in this module.
3. Contractor Bill: This module contains the designing codes of the contractor bill.
4. Customer: Contains codes related to customer window such as Customer details window, adding new customer, updating and deleting them.
5. Employee: All the codes of the employee browser are separated in this module.

* ECMS Controller: all the logics of the entire application are written in this module. The ECMS Controller module controls the logical data flow direction of the entire application, such as what is going to take place when we click on the ‘view’ button in inside the employee browser or how the application assigns a new contractor for a new connection etc.
* ECMS Storage: All the logics related to data storage are written in this module. This module is controlled by ECMS Controller for various database related actions.
* ECMS Style: This module contains all the style definitions of various GUI tool such as buttons, textboxes etc. The style definitions written in this module are used by various sub modules of ECMS GUI whenever the style for a tool is to be defined.

## data design

### Schema Design

|  |  |  |
| --- | --- | --- |
| **APPLICATION\_REGISTER** | |  |
| **Column Name** | **Datatype** | **Default** |
| apps\_no | string | NOT NULL |
| payment\_id | string | NULL |
| customerId | string | NULL |
| received\_date | DATETIME | NULL |
| estimateId | DOUBLE | NULL |
| service\_connection\_no | string | NULL |
|  |  |  |
|  |  |  |
| **CONTRUCTOR** |  |  |
| **Column Name** | **Datatype** | **Default** |
| Id | string | NOT NULL |
| Name | string | NULL |
| Address | string | NULL |
| Contact | string | NULL |
| contract\_details | string | NULL |
|  |  |  |
|  |  |  |
| **CUSTOMER** |  |  |
| **Column Name** | **Datatype** | **Default** |
| Id | string | NOT NULL |
| Name | string | NULL |
| Address | string | NULL |
| Contact | string | NULL |
|  |  |  |
|  |  |  |
| **EMPLOYEE** |  |  |
| **Column Name** | **Datatype** | **default** |
| Id | string | NOT NULL |
| Name | string | NULL |
| Address | string | NULL |
| Contact | string | NULL |
| post\_type | string | NULL |
| Doj | DATETIME | NULL |
| Department | string | NULL |
|  |  |  |
|  |  |  |
| **ESTIMATE** |  |  |
| **Column Name** | **Datatype** | **Default** |
| appNo | string | NOT NULL |
| Estimator | string | TBA |
| wireLength | DOUBLE | 0 |
| angleType | string | SHORT |
| angleWeight | DOUBLE | 0 |
| amountQuotation | DOUBLE | 0 |
| Contractor | string | TBA |
|  |  |  |
|  |  |  |
| **METER\_GER** |  |  |
| **Column Name** | **Datatype** | **Default** |
| Id | string | NOT NULL |
| connection\_no | string | NULL |
| connection\_name | string | NULL |
| Address | string | NULL |
| Contact | string | NULL |
| meter\_no | string | NULL |
| seal\_no | string | NULL |
| issue\_date | DATETIME | NULL |
| work\_assign\_to | string | NULL |
|  |  |  |
|  |  |  |
| **METER\_REGISTER** |  |  |
| **Column Name** | **Datatype** | **Default** |
| Id | string | NOT NULL |
| connection\_no | string | NULL |
| connection\_name | string | NULL |
| Address | string | NULL |
| Contact | string | NULL |
| meter\_no | string | NULL |
| seal\_no | string | NULL |
| issue\_date | DATETIME | NULL |
| work\_assign\_to | string | NULL |
|  |  |  |
|  |  |  |
| **PAYMENT** |  |  |
| **Column Name** | **Datatype** | **Default** |
| Id | string | NOT NULL |
| customerId | string | NULL |
| Amount | DOUBLE | NULL |
| Dop | DATETIME | NULL |
|  |  |  |
|  |  |  |
| **SERVICE\_CONNECTION\_REGISTER** | |  |
| **Column Name** | **Datatype** | **Default** |
| Id | string | NOT NULL |
| apps\_no | string | NULL |
| Name | string | NULL |
| Address | string | NULL |
| Contact | string | NULL |
| wire\_length | string | NULL |
| initial\_deposite\_amount | DOUBLE | NULL |
| angle\_type | string | NULL |
| angle\_weight | string | NULL |
| quotation\_amount | DOUBLE | NULL |

### data integrity and constraints

## procedural design

### logic diagrams

### data structures

/\* ContractorInfo.cs \*/

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespaceESCMSData

{

publicclassContractorInfo

{

publicstring id { get; set; }

publicstring name { get; set; }

publicstring address { get; set; }

publicstring contact { get; set; }

publicstring details { get; set; }

}

}

/\* CustomerInfo.cs \*/

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespaceESCMSData

{

publicclassCustomerInfo

{

publicstring id { get; set; }

publicstring name { get; set; }

publicstring address { get; set; }

publicstring contact { get; set; }

}

}

/\* EmployeeInfo.cs \*/

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespaceESCMSData

{

publicenumPostType

{

DataEnterer,

Officer,

Estimator,

Admin

}

publicclassEmployeeInfo

{

publicstring id { get; set; }

publicstring name { get; set; }

publicstring address { get; set; }

publicstring contact { get; set; }

publicPostTypepostType { get; set; }

publicDateTimedoj { get; set; }

publicstring department { get; set; }

}

}

/\*estimateInfo.cs\*/

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespaceESCMSData

{

publicenumAngleType

{

Short,

Long

}

publicclassestimateInfo

{

publicstringappsNo { get; set; }

publicstring name { get; set; }

publicstring address { get; set; }

publicstring contact { get; set; }

publicstring estimator { get; set; }

publicdoubleinitialAmount { get; set; }

publicdoublewireLength { get; set; }

publicdoubleangleWeight { get; set; }

publicAngleTypeangleType { get; set; }

publicdoublequotationAmount { get; set; }

publicstring contractor { get; set; }

}

}

/\*NewConnectionInfo.cs\*/

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespaceESCMSData

{

publicclassNewConnectionInfo

{

publicstringappsNo { get; set; }

publicDateTimereceivedDate { get; set; }

publicstringpaymentId { get; set; }

publicDateTimeamountReceivedOn { get; set; }

publicdoubleinitialAmount { get; set; }

publicstringcustomerId { get; set; }

publicstring name { get; set; }

publicstring address { get; set; }

publicstring phone { get; set; }

publicdoublequotationAmount { get; set; }

publicDateTimequotationSendDate { get; set; }

publicstringserviceConnectionNo { get; set; }

}

}

/\*PaymentInfo.cs\*/

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespaceESCMSData

{

publicclassPaymentInfo

{

publicstring id { get; set; }

publicstring name { get; set; }

publicstringcustomerId { get; set; }

publicdouble amount { get; set; }

publicDateTimedop { get; set; }

}

}

### ALGOrITHITMS AND DESIGN

Algorithm for searching:

Following algorithm is used to search and view a customer details by its id-

Input taken by GUI: A user provides the id of a specific customer inside a textbox. The text of that textbox is then taken inside the data module. The data module carries the data to the Controller module.

Searching is done inside the database: The controller module sends the data to the Storage module through the data module and tells it to search any available answer or answers inside the specific table of the database For example, in this case, the query would be ,’select \* from customer where id = @id’.

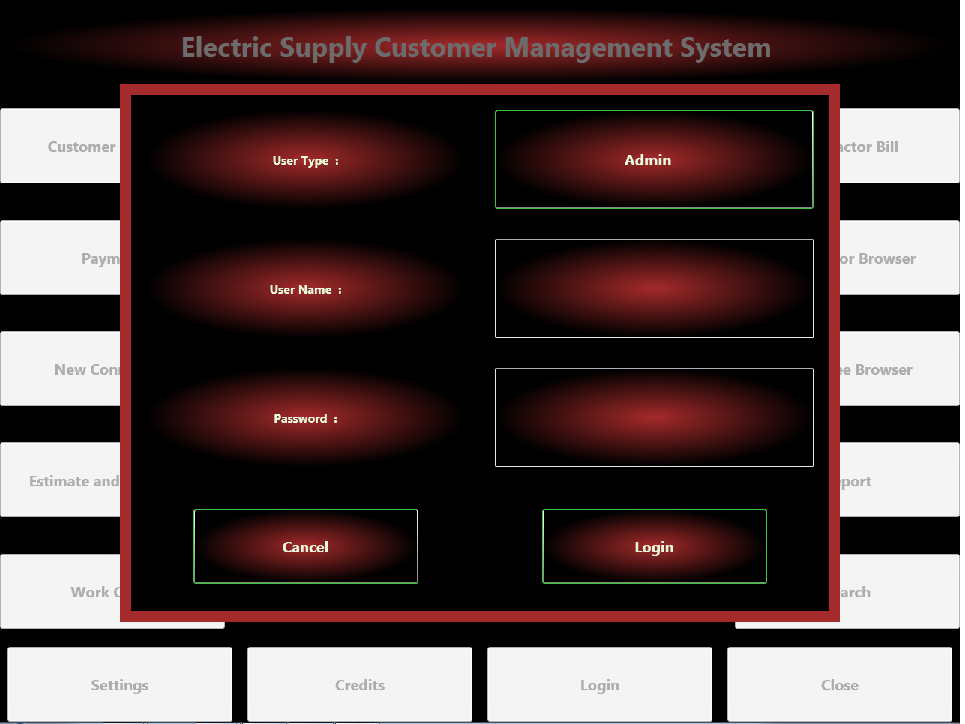
Output is shown at the GUI: The storage module does the desired searching and sends the output to the controller module through the data module and the output is displayed to the user through a list view.

## user interface design

### main window



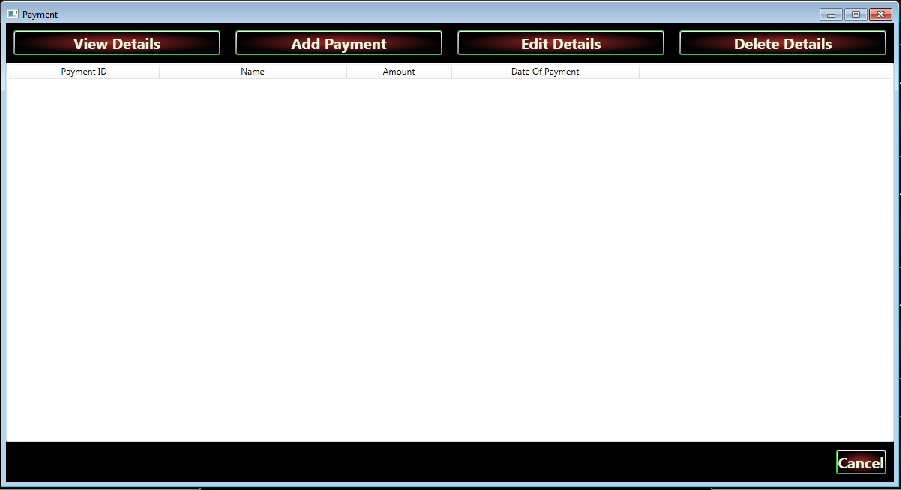
### login window



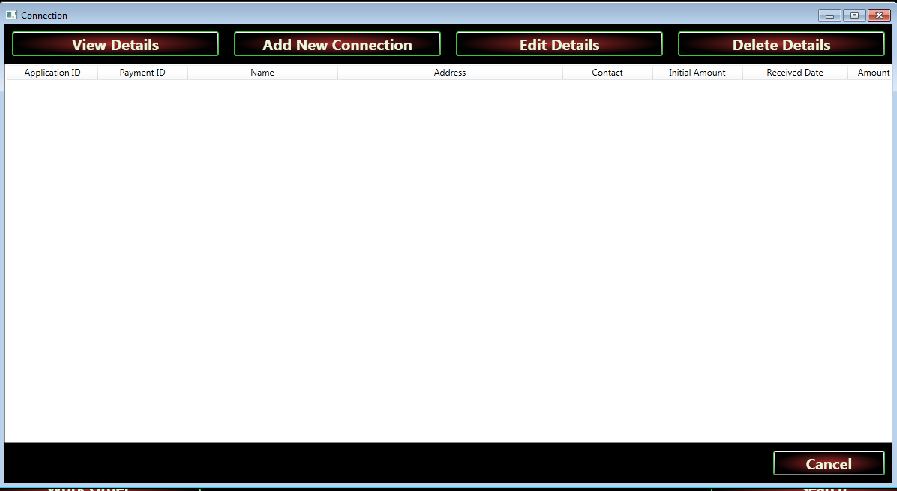
### customer browser window



### payment window



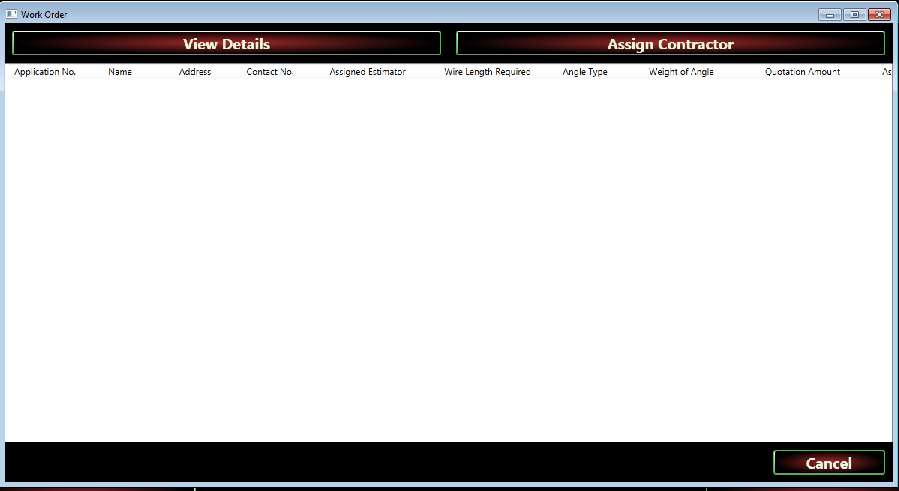
### connection window



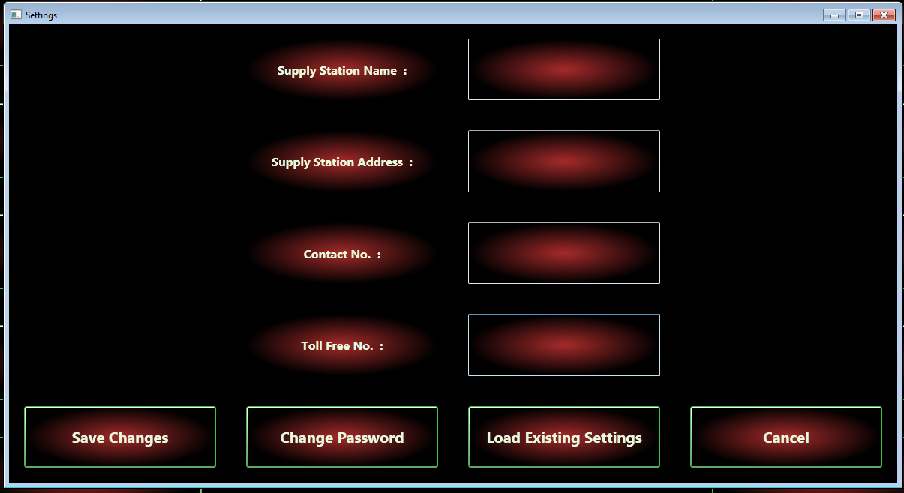
### estimate quotation window



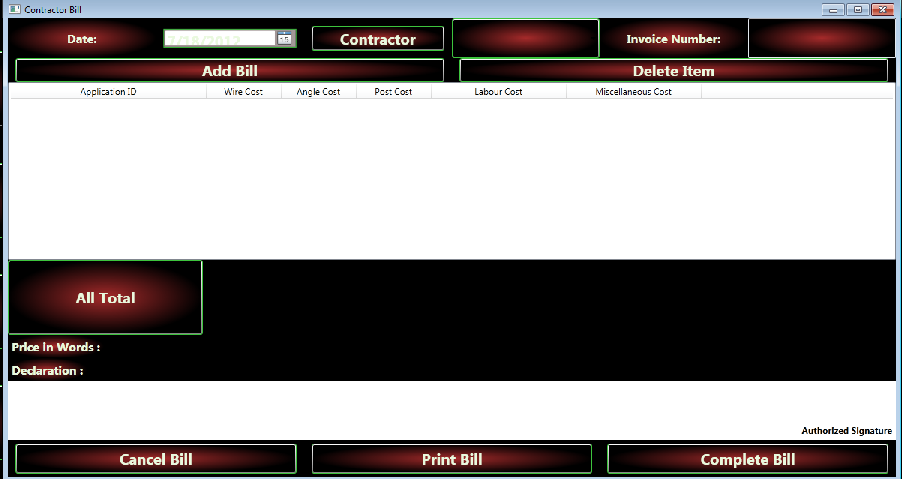
### work order window



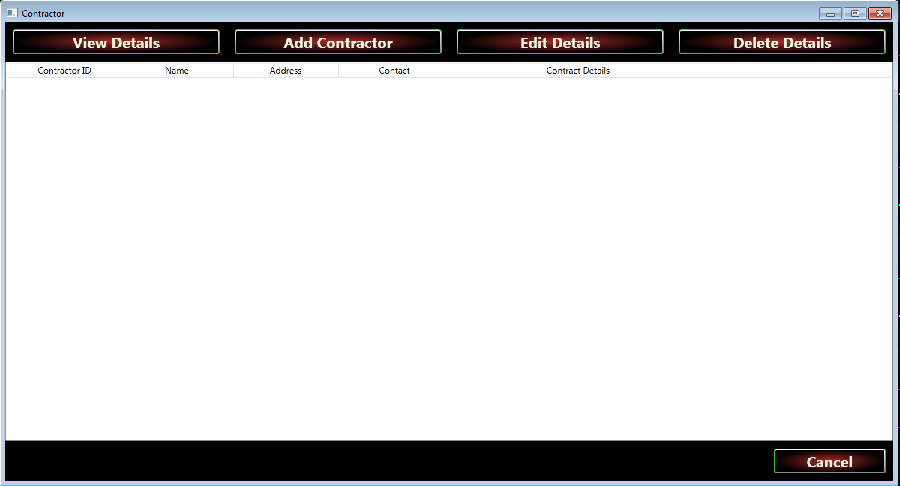
### settings window



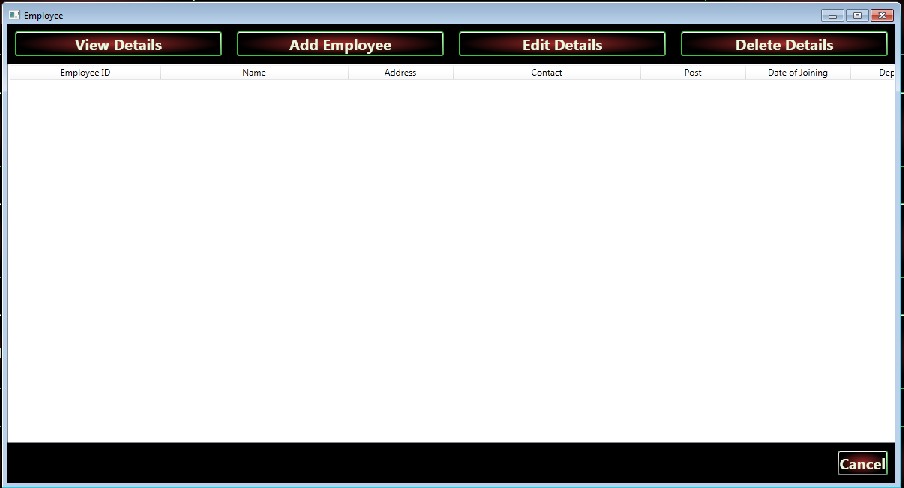
### contractor billl window



### contractor browser window



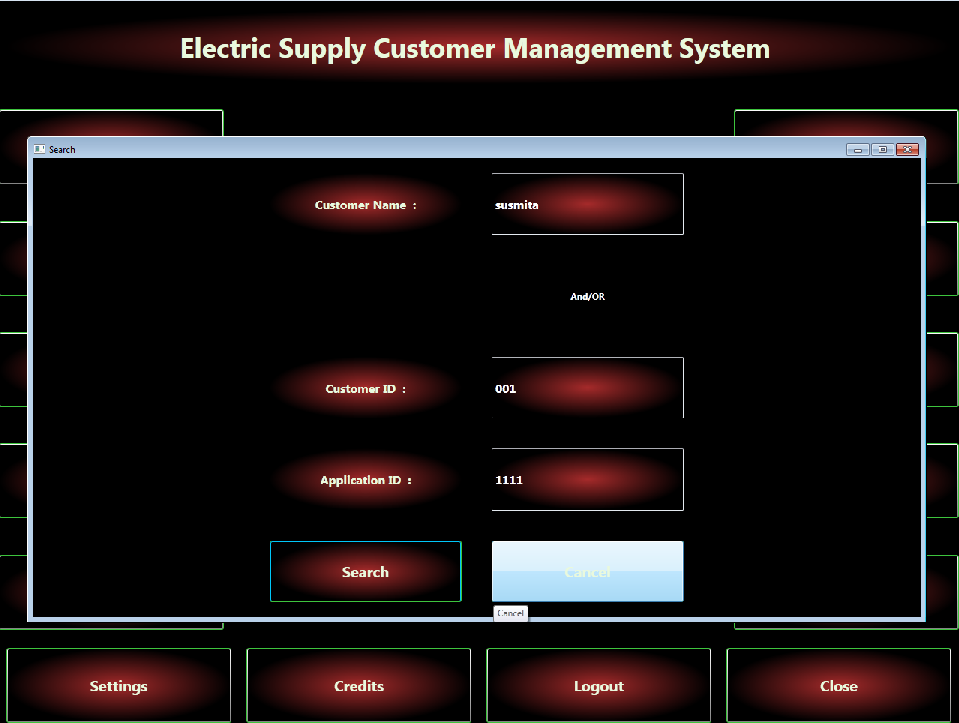
### employeee browser window



### report window



### search window



## security issues

ESCMS provides password based security for its user. In terms of security unauthorized access will be denied and register user will be able to change as necessary. User must impute a valid **user name** and **password** to login to the application. Without a valid **user name** and **password** no one can access to the application. This way it prevents misuse of application and database.

For data security there will be a digital backup and restore system. It is possible to take backup after some period of time.

## test cases design

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case Id | Item | Description | Actual Result |
| ESCMS-001 | Login | Enter Type, user Id and Password for log in. | Successfully Logged in. |
| ESCMS-002 | Cancel | Select Cancel to close login window | Successfully Canceled. |
| ESCMS-003 | Add Customer | To add a new customer enter the CustomerID,Name,Addresss and  Contact No. of the customer. | New Customer is added to the Electric Supply Customer Management System. |
| ESCMS-004 | ViewCustomer | Show the Customer details. | Successfully displayed. |
| ESCMS-005 | EditCustomer | Select the Customer and click the Edit option. Now edit the Customer Details and submit the details. | Connection details successfully updated. |
| ESCMS-006 | Delete Customer | Select the Customer and click the Delete option. | Customer account successfully deleted. |
| ESCMS-007 | OK | Select OK to close Customers window | Successfully Closed. |
| ESCMS-008 |  | To pay sufficient money and must give the application for a new Connection. |  |
| ESCMS-009 |  | Enter CustomerID , Payment Amount , and Pay Date etc. | Successfully updated and display the result. |
| ESCMS-010 | Edit Details | Select the Application ID and click the Edit option. Now edit the Connection Details and submit the details. | Connection Details Successfully updated. |
| ESCMS-011 | Delete Details | Select the Application ID and click the Delete option. | Successfully deleted the Application. |
| ESCMS-012 | OK | Select OK to close Connection window | Successfully Closed. |
| ESCMS-013 | View Details | Show the available Connection details. | Successfully displayed. |
| ESCMS-014 | Assign Contractor | To distribute the work into the Contractors and specify the name of the Contactor for assigning the jobs. | Successfully Assigned. |
| ESCMS-015 | View Details | Select the Contractor name and press the View Option. | Successfully displayed the details. |
| ESCMS-016 | Assign Estimation | Enter Application No. ,Name , Addresss , Contact No, Assigned Estimator, Wire length required,AngleType,Weight of Angle and Quotation Amount for Assigning The Estimation. | Successfully Assigned the Estimation.. |
| ESCMS-017 | Update Quotation | Select Application ID and enter the new Quotation correspondind to the Application ID and press Update Quotation option. | Quotation successfully updated. |
| ESCMS-018 | Add Bill | Enter Date ,Contractor Name , Invoice No. of the Contractor. | Bill is successfully generated. |
| ESCMS-019 | Delete Item | Select the Contractor ID and click the Delete option. | Contractor account successfully deleted. |
| ESCMS-020 | OK | Select OK to close Contractor window | Successfully Closed. |
| ESCMS-021 | Print Bill | Print the Bill of the Contractor for paying his remulation. | Successfully print the Bill. |
| ESCMS-022 | Report | Show the Details about Application received in certain time. | Successfully displayed. |
| ESCMS-023 | Add Employee | Enter Employee ID , Name, Contact No., Post ,Date of joining, DEPARTMENT etc. to add Employee Details. | Successfully new Employee details created. |
| ESCMS-024 | Edit Employee | Select Employee ID and Name and press the Edit option to edit the Employee Details. | Successfully updated the Employee details . |
| ESCMS-025 | Delete Employee | Select Employee ID and Name and press the Delete option to delete the Employee Details. | Successfully deleted. |

# Implimentation And Testing

## Implimentation Approaches

We started the project aiming to develop it such a way that after completing of the project we can easily study the codes for understanding how it works. We used one of the best features of object oriented programming, the modular approach perfectly and made all the codes well separated from each other according to its purpose. For example, we have kept all the database interaction related codes in a single CS page and we have modularized it even further inside the page. We have separated all the GUI tools style definitions in a single resource files. We have separated the data classes in a different cs page. So we can not only study the codes or find errors in the codes easily, we can also use them in other projects in no times which saves lots of time and effort.

## coding details and code efficiency

Codes of ESCMS has been pasted into the report in three different parts, first part contains GUI related codings, second part is created using Controller modules and the third part shows all the database intyeraction related codings.

GUI Related Coding Details:

Mainwindow:

|  |
| --- |
| <Window x:Class="ECMS.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Electric Supply Customer Management System" Left="10" Top="10" Background="Black" WindowState="Maximized" Loaded="Window\_Loaded">  <Window.Resources>sssss  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel Name="mainDocPanel" IsEnabled="False">  <Label DockPanel.Dock="Top" Content="Electric Supply Customer Management System " Style="{StaticResource bigLabelStyle}"></Label>  <UniformGrid DockPanel.Dock="Bottom" Rows="1" Height="100">    <Button Content="Settings" ToolTip="Change Settings" Name="settingsBtn" Style="{StaticResource ControlBtnStyle}" Click="settingsBtn\_Click" ></Button>  <Button Content="Credits" ToolTip="Credits" Name="creditBtn" Style="{StaticResource ControlBtnStyle}" Click="creditBtn\_Click"></Button>  <Button Name="loginBtn" Content="Login" Click="loginBtn\_Click" Style="{StaticResource ControlBtnStyle}"></Button>  <Button Name="closeBtn" ToolTip="Close This Application" Click="closeBtn\_Click" Style="{StaticResource ControlBtnStyle}">Close</Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="Left" Columns="1">    <Button Content="Customer Browser" ToolTip="Customer Details" Name="customerBrowserBtn" Style="{StaticResource ControlBtnStyle}" Click="customerBrowserBtn\_Click" ></Button>  <Button Content="Payment" ToolTip="Payment Details" Name="paymentBtn" Style="{StaticResource ControlBtnStyle}" Click="paymentBtn\_Click" ></Button>  <Button Content="New Connection" ToolTip="Apply For New Connection Details" Name="newConnectionBtn" Click="newConnectionBtn\_Click" Style="{StaticResource ControlBtnStyle}" ></Button>  <Button Content="Estimate and Quotation" ToolTip="Estimate and Quotation Details" Name="estimateBtn" Style="{StaticResource ControlBtnStyle}" Click="estimateBtn\_Click" ></Button>  <Button Content="Work Order" ToolTip="Work Order Details" Style="{StaticResource ControlBtnStyle}" Name="workOrderBtn" Click="workOrderBtn\_Click" ></Button>    </UniformGrid>  <UniformGrid DockPanel.Dock="Right" Columns="1">    <Button Content="Contractor Bill" ToolTip="Contractor Bill Details" Name="contractorBillBtn" Style="{StaticResource ControlBtnStyle}" Click="contractorBillBtn\_Click"></Button>  <Button Content="Contractor Browser" ToolTip="Contractor Details" Name="contractorBrowserBtn" Style="{StaticResource ControlBtnStyle}" Click="contractorBrowserBtn\_Click" ></Button>  <Button Content="Employee Browser" ToolTip="Employee Details" Name="employeeBrowserBtn" Style="{StaticResource ControlBtnStyle}" Click="employeeBrowserBtn\_Click" ></Button>  <Button Content="Report" ToolTip="View Report Details" Name="reportBtn" Style="{StaticResource ControlBtnStyle}" Click="reportBtn\_Click"></Button>  <Button Name="searchBtn" ToolTip="Search" Style="{StaticResource ControlBtnStyle}" Click="searchBtn\_Click\_1">Search</Button>  </UniformGrid>  <Image Source="/ECMS;component/Images/Light-Bulb.jpg" Width="Auto" Height="Auto" ></Image>  </DockPanel>  </Window> |

Coding of Style Resource Page: This resource page contains styling definitions of all the GUI tools used in this project and this page isreffered to whereevere we have used those tools inside the GUI.

|  |
| --- |
| <ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">  <Style x:Key="ControlBtnStyle" TargetType="Button">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="20" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="BorderBrush" Value="#FF3DC43D" />  <Setter Property="Width" Value="300" />  <Setter Property="Height" Value="100" />  <Setter Property="OpacityMask" Value="White" />  <Setter Property="Cursor" Value="Hand" />  </Style>  <Style x:Key="smallControlBtnStyle" TargetType="Button">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="20" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="BorderBrush" Value="#FF3DC43D" />  <Setter Property="Margin" Value="10"/>  <Setter Property="OpacityMask" Value="White" />  <Setter Property="Cursor" Value="Hand" />  </Style>  <Style x:Key="bigLabelStyle" TargetType="Label">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="36" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="Height" Value="100" />  <Setter Property="HorizontalContentAlignment" Value="Center" />  <Setter Property="VerticalContentAlignment" Value="Center" />  <Setter Property="Margin" Value="10"/>  </Style>  <Style x:Key="labelStyle" TargetType="Label">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="16" />  <Setter Property="FontWeight" Value="Bold" />  <!--<Setter Property="Height" Value="100" />-->  <Setter Property="HorizontalContentAlignment" Value="Center" />  <Setter Property="VerticalContentAlignment" Value="Center" />  <!--<Setter Property="Margin" Value="10"/>-->  </Style>  <Style x:Key="textboxStyle" TargetType="TextBox">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="White" />  <Setter Property="FontSize" Value="16" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="VerticalContentAlignment" Value="Center" />  </Style>  <Style x:Key="PasswordBoxStyle" TargetType="PasswordBox">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="16" />  <Setter Property="FontWeight" Value="Bold" />  <!--<Setter Property="Height" Value="100" />-->  <!--<Setter Property="HorizontalContentAlignment" Value="Center" />-->  <Setter Property="VerticalContentAlignment" Value="Center" />  <!--<Setter Property="Margin" Value="10"/>-->  </Style>  <Style x:Key="CBStyle" TargetType="ComboBox">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="20" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="BorderBrush" Value="#FF3DC43D" />  <Setter Property="Margin" Value="20"/>  <Setter Property="OpacityMask" Value="White" />  <Setter Property="HorizontalContentAlignment" Value="Center" />  <Setter Property="VerticalContentAlignment" Value="Center" />  </Style>  <Style x:Key="CBStyleForName" TargetType="ComboBox">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="20" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="BorderBrush" Value="#FF3DC43D" />  <!--<Setter Property="Margin" Value="20"/>-->  <Setter Property="OpacityMask" Value="White" />  <Setter Property="HorizontalContentAlignment" Value="Center" />  <Setter Property="VerticalContentAlignment" Value="Center" />  </Style>  <Style x:Key="CBItemStyle" TargetType="ComboBoxItem">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="20" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="BorderBrush" Value="#FF3DC43D" />  <Setter Property="OpacityMask" Value="White" />  </Style>  <Style x:Key="DPStyle" TargetType="DatePicker" >  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Brown" Offset="0" />  <GradientStop Color="Black" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="20" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="BorderBrush" Value="#FF3DC43D" />  <Setter Property="Margin" Value="10" />  <Setter Property="VerticalContentAlignment" Value="Center" />  <Setter Property="OpacityMask" Value="White" />  </Style>  </ResourceDictionary> |

Coding of Customer Browser GUI:

|  |
| --- |
| <Window x:Class="ECMS.Customer"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Customer"  Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black"  >  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid DockPanel.Dock="top" Rows="1">  <Button Content="View Customers" ToolTip="View or Refresh" Name="viewCustomerBtn" Style="{StaticResource smallControlBtnStyle}" Click="viewCustomerBtn\_Click"></Button>  <Button Content="Add Customer" ToolTip="Click To Add New Customer" Name="addCustomerBtn" Style="{StaticResource smallControlBtnStyle}" Click="addCustomerBtn\_Click"></Button>  <Button Content="Edit Customer" ToolTip="Edit Details" Name="editCustomerBtn" Style="{StaticResource smallControlBtnStyle}" Click="editCustomerBtn\_Click"></Button>  <Button Content="Delete Customer" ToolTip="Delete Selected Details" Name="deleteCustomerBtn" Style="{StaticResource smallControlBtnStyle}" Click="deleteCustomerBtn\_Click"></Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="Bottom" Rows="1" Columns="7">  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="top">  <ListView Name="customerView" ItemsSource="{Binding customerCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="200" Header="Customer ID." DisplayMemberBinding="{Binding id}" />  <GridViewColumn Width="300" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="400" Header="Address" DisplayMemberBinding="{Binding address}" />  <GridViewColumn Width="200" Header="Contact" DisplayMemberBinding="{Binding contact}" />  </GridView>  </ListView.View>  </ListView>  </UniformGrid>  </DockPanel>  </Window> |

Coding for Add New Customer GUI:

|  |
| --- |
| <Window x:Class="ECMS.AddCustomer"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Add Customer" Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black"  >  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid Columns="4">  <Label></Label>  <Label Content="Name :" Style="{StaticResource labelStyle}"/>  <TextBox Name="nameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Address :" Style="{StaticResource labelStyle}" />  <TextBox Name="addressTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Contact No. :" Style="{StaticResource labelStyle}"/>  <TextBox Name="contactNoTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  <Button Content="Submit" ToolTip="Submit" Name="submitBtn" Style="{StaticResource smallControlBtnStyle}" Click="submitBtn\_Click"></Button>  <Label></Label>  </UniformGrid>  </DockPanel>  </Window> |

Coding for Payment Window GUI:

|  |
| --- |
| <Window x:Class="ECMS.Payment"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Payment"  Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black"  >  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid DockPanel.Dock="top" Rows="1">  <Button Content="View Details" ToolTip="View or Refresh" Name="viewConnectionBtn" Style="{StaticResource smallControlBtnStyle}" Click="viewConnectionBtn\_Click"></Button>  <Button Content="Add Payment" ToolTip="Click To Add New Payment" Name="addPaymentBtn" Style="{StaticResource smallControlBtnStyle}" Click="addPaymentBtn\_Click"></Button>  <Button Content="Edit Details" ToolTip="Edit Details" Name="editConnectionBtn" Style="{StaticResource smallControlBtnStyle}" Click="editConnectionBtn\_Click"></Button>  <Button Content="Delete Details" ToolTip="Delete Selected Details" Name="deleteConnectionBtn" Style="{StaticResource smallControlBtnStyle}" Click="deleteConnectionBtn\_Click"></Button>    </UniformGrid>    <Button DockPanel.Dock="Bottom" HorizontalAlignment="Right" Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>    <UniformGrid DockPanel.Dock="top">  <ListView Name="paymentView" ItemsSource="{Binding paymentCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="200" Header="Payment ID" DisplayMemberBinding="{Binding id}" />  <GridViewColumn Width="250" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="140" Header="Amount" DisplayMemberBinding="{Binding amount}" />  <GridViewColumn Width="250" Header="Date Of Payment" DisplayMemberBinding="{Binding dop}" />  </GridView>  </ListView.View>  </ListView>  </UniformGrid>  </DockPanel>  </Window> |

Coding for Add Payment GUI:

|  |
| --- |
| <Window x:Class="ECMS.AddPayment"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Add Payment" Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black" Loaded="Window\_Loaded">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid Columns="4">  <Label></Label>  <Label Content="Payment ID :" Style="{StaticResource labelStyle}"/>  <Label Content="" Name="paymentIdLbl" Style="{StaticResource labelStyle}"></Label>  <Label></Label>  <Label></Label>  <Label Content="Name :" Style="{StaticResource labelStyle}"/>  <ComboBox Name="nameTxtbox" SelectedIndex="0" Style="{StaticResource CBStyle}" ItemsSource="{Binding customerCollection}" DisplayMemberPath="name" SelectedValuePath="id" ></ComboBox>  <Button Name="addCustomerBtn" Style="{StaticResource smallControlBtnStyle}" Click="addCustomerBtn\_Click">Add New Customer</Button>  <Label></Label>  <Label Content="Amount :" Style="{StaticResource labelStyle}" />  <TextBox Name="amountTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Date of Payment :" Style="{StaticResource labelStyle}"/>  <DatePicker Name="dopDatePicker" Style="{StaticResource DPStyle}"></DatePicker>  <Label></Label>  <Label></Label>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  <Button Content="Submit" ToolTip="Submit" Name="submitBtn" Style="{StaticResource smallControlBtnStyle}" Click="submitBtn\_Click"></Button>  <Label></Label>  </UniformGrid>  </DockPanel>  </Window> |

Coding for Estimate and Quotation Window:

|  |
| --- |
| <Window x:Class="ECMS.EstimationQuotationDisplay"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Estimate And Quotation"  Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black"  >  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid DockPanel.Dock="top" Rows="1">  <Button Content="View Details" ToolTip="View or Refresh" Name="viewEstimateBtn" Click="viewEstimateBtn\_Click" Style="{StaticResource smallControlBtnStyle}"></Button>  <Button Content="Assign Estimator" ToolTip="Click To Update or Assign Estimator" Name="editBtn" Style="{StaticResource smallControlBtnStyle}" Click="editBtn\_Click"></Button>  <Button Content="Update Quotation" ToolTip="Click To Update or Add Quotation" Name="addPaymentBtn" Style="{StaticResource smallControlBtnStyle}" Click="addPaymentBtn\_Click"></Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="Bottom" Rows="1" Columns="7">  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="top">  <ListView Name="estimateView" ItemsSource="{Binding estimateCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="100" Header="Application No." DisplayMemberBinding="{Binding appsNo}" />  <GridViewColumn Width="100" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="100" Header="Address" DisplayMemberBinding="{Binding address}" />  <GridViewColumn Width="100" Header="Contact No." DisplayMemberBinding="{Binding contact}" />  <GridViewColumn Width="160" Header="Assigned Estimator" DisplayMemberBinding="{Binding estimator}" />  <GridViewColumn Width="160" Header="Wire Length Required" DisplayMemberBinding="{Binding wireLength}" />  <GridViewColumn Width="100" Header="Angle Type" DisplayMemberBinding="{Binding angleType}" />  <GridViewColumn Width="160" Header="Weight of Angle " DisplayMemberBinding="{Binding angleWeight}" />  <GridViewColumn Width="160" Header="Quotation Amount" DisplayMemberBinding="{Binding quotationAmount}" />  </GridView>  </ListView.View>  </ListView>  </UniformGrid>  </DockPanel>  </Window> |

Coding for Assign Estimator Window:

|  |
| --- |
| <Window x:Class="ECMS.AddEstimator"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Add Estimator"  Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black" Loaded="Window\_Loaded">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid Columns="4">  <Label></Label>  <Label Content="Application No. :" Style="{StaticResource labelStyle}"/>  <TextBox Name="applicationNoTB" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Name :" Style="{StaticResource labelStyle}"/>  <TextBox Name="nameTxtbox" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Address :" Style="{StaticResource labelStyle}" />  <TextBox Name="addressTxtbox" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Contact No. :" Style="{StaticResource labelStyle}"/>  <TextBox Name="contactNoTxtbox" IsEnabled="false" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Assigned Estimator :" Style="{StaticResource labelStyle}"/>  <ComboBox Name="assignedEstimatorCB" Style="{StaticResource CBStyle}" SelectedIndex="0" ItemsSource="{Binding estimatorCollection}" DisplayMemberPath="name" SelectedValuePath="id" ></ComboBox>  <Label></Label>  <Label></Label>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  <Button Content="Submit" ToolTip="Submit" Name="submitBtn" Style="{StaticResource smallControlBtnStyle}" Click="submitBtn\_Click"></Button>  <Label></Label>  </UniformGrid>  </DockPanel>  </Window> |

Coding for Update Quotation Window:

|  |
| --- |
| <Window x:Class="ECMS.AddQuotation"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Add Quotation"  Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black" Loaded="Window\_Loaded">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid Columns="4">  <Label></Label>  <Label Content="Application No. :" Style="{StaticResource labelStyle}"/>  <TextBox Name="applicationNoTB" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Name :" Style="{StaticResource labelStyle}"/>  <TextBox Name="nameTxtbox" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Address :" Style="{StaticResource labelStyle}" />  <TextBox Name="addressTxtbox" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Contact No. :" Style="{StaticResource labelStyle}"/>  <TextBox Name="contactNoTxtbox" IsEnabled="false" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Assigned Estimator :" Style="{StaticResource labelStyle}"/>  <ComboBox Name="assignedEstimatorCB" Style="{StaticResource CBStyleForName}" SelectedIndex="0" ItemsSource="{Binding estimatorCollection}" DisplayMemberPath="name" SelectedValuePath="id"></ComboBox>  <Label></Label>  <Label></Label>  <Label Content="Wire Length Required :" Style="{StaticResource labelStyle}"/>  <TextBox Name="wireLengthRequiredTB" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Angle Type :" Style="{StaticResource labelStyle}"/>  <ComboBox Name="angleTypeCB" Style="{StaticResource CBStyleForName}" SelectedIndex="0">  <ComboBoxItem Style="{StaticResource CBItemStyle}">Short</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Long</ComboBoxItem>  </ComboBox>  <Label></Label>  <Label></Label>  <Label Content="Weight of Angle Calculation :" Style="{StaticResource labelStyle}"/>  <TextBox Name="weightofAngleCalculationTB" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Quotation Amount :" Style="{StaticResource labelStyle}"/>  <TextBox Name="quotationAmountTB" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  <Button Content="Submit" ToolTip="Submit" Name="submitBtn" Style="{StaticResource smallControlBtnStyle}" Click="submitBtn\_Click"></Button>  <Label></Label>  </UniformGrid>  </DockPanel>  </Window> |

Coding for Work Order Window:

|  |
| --- |
| <Window x:Class="ECMS.WorkOrder"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Work Order"  Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black"  >  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid DockPanel.Dock="top" Rows="1">  <Button Content="View Details" ToolTip="View Or Refresh" Name="viewWorkOrderBtn" Style="{StaticResource smallControlBtnStyle}" Click="viewWorkOrderBtn\_Click"></Button>  <Button Content="Assign Contractor" ToolTip="Click To Assign Contractor" Name="addWorkOrderBtn" Style="{StaticResource smallControlBtnStyle}" Click="addWorkOrderBtn\_Click"></Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="Bottom" Rows="1" Columns="7">  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Label></Label>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="top">  <ListView Name="workOrderView" ItemsSource="{Binding estimateCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="100" Header="Application No." DisplayMemberBinding="{Binding appsNo}" />  <GridViewColumn Width="100" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="100" Header="Address" DisplayMemberBinding="{Binding address}" />  <GridViewColumn Width="100" Header="Contact No." DisplayMemberBinding="{Binding contact}" />  <GridViewColumn Width="160" Header="Assigned Estimator" DisplayMemberBinding="{Binding estimator}" />  <GridViewColumn Width="160" Header="Wire Length Required" DisplayMemberBinding="{Binding wireLength}" />  <GridViewColumn Width="100" Header="Angle Type" DisplayMemberBinding="{Binding angleType}" />  <GridViewColumn Width="160" Header="Weight of Angle " DisplayMemberBinding="{Binding angleWeight}" />  <GridViewColumn Width="160" Header="Quotation Amount" DisplayMemberBinding="{Binding quotationAmount}" />  <GridViewColumn Width="160" Header="Assigned Contractor" DisplayMemberBinding="{Binding contractor }" />  </GridView>  </ListView.View>  </ListView>  </UniformGrid>  </DockPanel>  </Window> |

Coding for Assign Contractor Window:

|  |
| --- |
| <Window x:Class="ECMS.AssignContractor"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:sys="clr-namespace:System;assembly=mscorlib"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Add Quotation"  Height="650" Width="1200"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  Background="Black" Loaded="Window\_Loaded">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel>  <UniformGrid Columns="4">  <Label></Label>  <Label Content="Application No. :" Style="{StaticResource labelStyle}"/>  <TextBox Name="applicationNoTB" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Name :" Style="{StaticResource labelStyle}"/>  <TextBox Name="nameTxtbox" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Address :" Style="{StaticResource labelStyle}" />  <TextBox Name="addressTxtbox" Style="{StaticResource textboxStyle}" IsEnabled="false"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Contact No. :" Style="{StaticResource labelStyle}"/>  <TextBox Name="contactNoTxtbox" IsEnabled="false" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Assigned Estimator :" Style="{StaticResource labelStyle}"/>  <TextBox Name="assignedEstimatorTxtbox" IsEnabled="false" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Wire Length Required :" Style="{StaticResource labelStyle}"/>  <TextBox Name="wireLengthRequiredTB" IsEnabled="false" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Angle Type :" Style="{StaticResource labelStyle}"/>  <TextBox Name="angleTypeTB" IsEnabled="false" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Weight of Angle Calculation :" Style="{StaticResource labelStyle}"/>  <TextBox Name="weightofAngleCalculationTB" IsEnabled="false" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Quotation Amount :" Style="{StaticResource labelStyle}"/>  <TextBox Name="quotationAmountTB" IsEnabled="false" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Assigned Contractor :" Style="{StaticResource labelStyle}"/>  <ComboBox Name="assignedContractorCB" Style="{StaticResource CBStyleForName}" SelectedIndex="0" ItemsSource="{Binding contractorCollection}" DisplayMemberPath="name" SelectedValuePath="id">  </ComboBox>  <Label></Label>  <Label></Label>  <Button Content="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  <Button Content="Submit" Name="submitBtn" Style="{StaticResource smallControlBtnStyle}" Click="submitBtn\_Click"></Button>  <Label></Label>  </UniformGrid>  </DockPanel>  </Window> |

Coding Related to ESCMS Engine:

Coding of Main window to Open Other Windows:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Navigation;  using System.Windows.Shapes;  namespace ECMS  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  login loginWindow;  public MainWindow()  {  InitializeComponent();  loginWindow = new login();  loginWindow.OnSucccesfulLogin += new ECMS.login.delegateOnSucccesfulLogin(LoginWindowObj\_OnSucccesfulLogin);  }  private void loginBtn\_Click(object sender, RoutedEventArgs e)  {  if (LoggedIn == false)  {    }  else  {  mainDocPanel.IsEnabled = false;  loginBtn.Content = "Login";  loginBtn.ToolTip = "Click to Login";  LoggedIn = false;  loginWindow.ShowDialog();  }  }  bool LoggedIn = false;  void LoginWindowObj\_OnSucccesfulLogin(bool IsSuccess)  {  if (IsSuccess)  {  mainDocPanel.IsEnabled = true;  loginBtn.Content = "Logout";  loginBtn.ToolTip = "Click to Logout";  LoggedIn = true;  }  else  {  this.Close();  }  }  private void paymentBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Payment PaymentObj = new ECMS.Payment();  PaymentObj.Show();  }  private void closeBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void newConnectionBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Connection ConnectionObj = new ECMS.Connection();  ConnectionObj.ShowDialog();  }  private void employeeBrowserBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Employee EmployeeObj = new ECMS.Employee();  EmployeeObj.Show();  }  private void contractorBrowserBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Contractor ContractorObj = new ECMS.Contractor();  ContractorObj.Show();  }  private void estimateBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.EstimationQuotationDisplay EstimationQuatationDisplayObj = new ECMS.EstimationQuotationDisplay();  EstimationQuatationDisplayObj.Show();  }  private void customerBrowserBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Customer CustomerObj = new ECMS.Customer();  CustomerObj.Show();  }  private void workOrderBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.WorkOrder WorkOrderObj = new ECMS.WorkOrder();  WorkOrderObj.Show();  }  private void settingsBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Settings SettingsObj = new ECMS.Settings();  SettingsObj.Show();  }  private void searchBtn\_Click\_1(object sender, RoutedEventArgs e)  {  ECMS.Search SearchObj = new ECMS.Search();  SearchObj.Show();  }  private void contractorBillBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.ContractorBill ContractorBillObj = new ECMS.ContractorBill();  ContractorBillObj.Show();  }  private void reportBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Report ReportObj = new ECMS.Report();  ReportObj.Show();  }  private void Window\_Loaded(object sender, RoutedEventArgs e)  {  if (!LoggedIn)  {  loginWindow.ShowDialog();  }  }  private void creditBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Credits HelpObj = new ECMS.Credits();  HelpObj.Show();  }  }  } |

Code for Add Customer:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Shapes;  namespace ECMS  {  /// <summary>  /// Interaction logic for AddCustomer.xaml  /// </summary>  public partial class AddCustomer : Window  {  string customerId;  bool isEdit = false;  public AddCustomer(ESCMSData.CustomerInfo info)  {  InitializeComponent();  if (info != null)  {  isEdit = true;  nameTxtbox.Text = info.name;  addressTxtbox.Text = info.address;  contactNoTxtbox.Text = info.contact;  customerId = info.id;  }  }  private void OK\_Btn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void submitBtn\_Click(object sender, RoutedEventArgs e)  {  ESCMSData.CustomerInfo newCustomer = new ESCMSData.CustomerInfo();  newCustomer.id = GenerateId();  newCustomer.name = nameTxtbox.Text;  newCustomer.address = addressTxtbox.Text;  newCustomer.contact = contactNoTxtbox.Text;  if (isEdit == false)  {  newCustomer.id = GenerateId();  ESCMSStorage.DbInteraction.DoRegisterNewCustomer(newCustomer);  }  else  {  newCustomer.id = customerId;  ESCMSStorage.DbInteraction.EditCustomer(newCustomer);  }  this.Close();  }  private string GenerateId()  {  return DateTime.Now.ToOADate().ToString();  }  private void cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  }  } |

Code for Customer Details Window:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Shapes;  using ESCMSData;  using System.Collections.ObjectModel;  namespace ECMS  {  /// <summary>  /// Interaction logic for Customer.xaml  /// </summary>  public partial class Customer : Window  {  ObservableCollection<CustomerInfo> \_customerCollection = new ObservableCollection<CustomerInfo>();  public ObservableCollection<CustomerInfo> customerCollection  {  get  {  return \_customerCollection;  }  }  public Customer()  {  InitializeComponent();  }  private void cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void addCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.AddCustomer AddCustomerObj = new ECMS.AddCustomer(null);  AddCustomerObj.Show();  }  private void viewCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  List<CustomerInfo> customers = ESCMSStorage.DbInteraction.GetAllCustomerList();  \_customerCollection.Clear();  foreach (CustomerInfo customer in customers)  {  \_customerCollection.Add(customer);  }  }  private void deleteCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  CustomerInfo customerToDelete = GetSelectedItem();  if (customerToDelete != null)  {  \_customerCollection.Remove(customerToDelete);  ESCMSStorage.DbInteraction.DeleteCustomer(customerToDelete.id);  }  }  private CustomerInfo GetSelectedItem()  {  CustomerInfo customerToDelete = null;  if (customerView.SelectedIndex == -1)  MessageBox.Show("Please Select an Item");  else  {  CustomerInfo i = (CustomerInfo)customerView.SelectedItem;  customerToDelete = \_customerCollection.Where(item => item.id.Equals(i.id)).First();  }  return customerToDelete;  }  private void editCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  CustomerInfo customerToEdit = GetSelectedItem();  if (customerToEdit != null)  {  ECMS.AddCustomer AddCustomerObj = new ECMS.AddCustomer(customerToEdit);  AddCustomerObj.Show();  }  }  }  } |

Codings for Estimate and Quotation Window:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Shapes;  using System.Collections.ObjectModel;  using ESCMSData;  namespace ECMS  {  /// <summary>  /// Interaction logic for EstimationQuatationDisplay.xaml  /// </summary>  public partial class EstimationQuotationDisplay : Window  {  ObservableCollection<estimateInfo> \_estimateCollection = new ObservableCollection<estimateInfo>();  public ObservableCollection<estimateInfo> estimateCollection  {  get  {  return \_estimateCollection;  }  }  public EstimationQuotationDisplay()  {  InitializeComponent();  }  private void addPaymentBtn\_Click(object sender, RoutedEventArgs e)  {  estimateInfo estimateToEdit = GetSelectedItem();  if (estimateToEdit != null)  {  ECMS.AddQuotation EstimateQuotationObj = new ECMS.AddQuotation(estimateToEdit);  EstimateQuotationObj.Show();  }  }  public estimateInfo GetSelectedItem()  {  estimateInfo estimateToDelete = null;  if (estimateView.SelectedIndex == -1)  MessageBox.Show("Please Select an Item");  else  {  estimateInfo i = (estimateInfo)estimateView.SelectedItem;  estimateToDelete = \_estimateCollection.Where(item => item.appsNo.Equals(i.appsNo)).First();  }  return estimateToDelete;  }  private void cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void viewEstimateBtn\_Click(object sender, RoutedEventArgs e)  {  List<estimateInfo> estimates = ESCMSStorage.DbInteraction.GetAllEstimateList();  \_estimateCollection.Clear();  foreach (estimateInfo esm in estimates)  {  string cusId = ESCMSStorage.DbInteraction.GetNewConnectionCustomerId(esm.appsNo);  CustomerInfo cusInfo = ESCMSStorage.DbInteraction.GetCustomerInfo(cusId);  esm.name = cusInfo.name;  esm.address = cusInfo.address;  esm.contact = cusInfo.contact;  \_estimateCollection.Add(esm);  }  }  private void editBtn\_Click(object sender, RoutedEventArgs e)  {  estimateInfo estimateToEdit = GetSelectedItem();  if (estimateToEdit != null)  {  ECMS.AddEstimator AddEstimatorObj = new ECMS.AddEstimator(estimateToEdit);  AddEstimatorObj.Show();  }  }  }  } |

Codings for Assign Estimator Window:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Shapes;  using ESCMSData;  using System.Collections.ObjectModel;  namespace ECMS  {  /// <summary>  /// Interaction logic for AddEstimator.xaml  /// </summary>  public partial class AddEstimator : Window  {  ObservableCollection<EmployeeInfo> \_estimatorCollection = new ObservableCollection<EmployeeInfo>();  public ObservableCollection<EmployeeInfo> estimatorCollection  {  get  {  return \_estimatorCollection;  }  }  public AddEstimator(estimateInfo info)  {  InitializeComponent();  applicationNoTB.Text = info.appsNo;  nameTxtbox.Text = info.name;  addressTxtbox.Text = info.address;  contactNoTxtbox.Text = info.contact;  assignedEstimatorCB.Text = info.estimator;  List<EmployeeInfo> estimators = ESCMSStorage.DbInteraction.GetAllEstimatorList();  \_estimatorCollection.Clear();  foreach (EmployeeInfo estimator in estimators)  {  \_estimatorCollection.Add(estimator);  }  }  private void cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void Window\_Loaded(object sender, RoutedEventArgs e)  {  List<EmployeeInfo> estimators = ESCMSStorage.DbInteraction.GetAllEmployeeList();  \_estimatorCollection.Clear();  foreach (EmployeeInfo estimator in estimators)  {  \_estimatorCollection.Add(estimator);  }  }  private void submitBtn\_Click(object sender, RoutedEventArgs e)  {  estimateInfo estimateData = new ESCMSData.estimateInfo();  estimateData.appsNo = applicationNoTB.Text;  estimateData.estimator = assignedEstimatorCB.Text;  ESCMSStorage.DbInteraction.assignEstimator(estimateData);  this.Close();  }  }  } |

Codings for Update Quotation Window:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Shapes;  using ESCMSData;  using System.Collections.ObjectModel;  namespace ECMS  {  /// <summary>  /// Interaction logic for AddQuotation.xaml  /// </summary>  public partial class AddQuotation : Window  {  string estimateQuotationId;  bool isEdit = false;  ObservableCollection<EmployeeInfo> \_estimatorCollection = new ObservableCollection<EmployeeInfo>();  public ObservableCollection<EmployeeInfo> estimatorCollection  {  get  {  return \_estimatorCollection;  }  }  ObservableCollection<NewConnectionInfo> \_newConnectionCollection = new ObservableCollection<NewConnectionInfo>();  public ObservableCollection<NewConnectionInfo> newConnectionCollection  {  get  {  return \_newConnectionCollection;  }  }  public AddQuotation(estimateInfo info)  {  InitializeComponent();  applicationNoTB.Text = info.appsNo;  nameTxtbox.Text = info.name;  addressTxtbox.Text = info.address;  contactNoTxtbox.Text = info.contact;  assignedEstimatorCB.Text = info.estimator;  wireLengthRequiredTB.Text = Convert.ToString(info.wireLength);  weightofAngleCalculationTB.Text = Convert.ToString(info.angleWeight);  quotationAmountTB.Text = Convert.ToString(info.quotationAmount);  List<EmployeeInfo> estimators = ESCMSStorage.DbInteraction.GetAllEstimatorList();  \_estimatorCollection.Clear();  foreach (EmployeeInfo estimator in estimators)  {  \_estimatorCollection.Add(estimator);  }  }  private void cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void submitBtn\_Click(object sender, RoutedEventArgs e)  {  estimateInfo estimateData = new ESCMSData.estimateInfo();  estimateData.appsNo = applicationNoTB.Text;  estimateData.estimator = assignedEstimatorCB.Text;  estimateData.wireLength = Convert.ToDouble(wireLengthRequiredTB.Text);  estimateData.angleType = (AngleType)Enum.Parse(typeof(AngleType), angleTypeCB.Text, true);  estimateData.angleWeight = Convert.ToDouble(weightofAngleCalculationTB.Text);  estimateData.quotationAmount = Convert.ToDouble(quotationAmountTB.Text);  ESCMSStorage.DbInteraction.EditEstimate(estimateData);  this.Close();  }  private void Window\_Loaded(object sender, RoutedEventArgs e)  {  List<EmployeeInfo> estimators = ESCMSStorage.DbInteraction.GetAllEmployeeList();  \_estimatorCollection.Clear();  foreach (EmployeeInfo estimator in estimators)  {  \_estimatorCollection.Add(estimator);  }  }  }  } |

Codings Related to Database Interaction of the Entire Project(DBInteraction.cs):

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ESCMSData;  namespace ESCMSStorage  {  public static class DbInteraction  {  static string passwordCurrent = "technicise";  static string dbmsCurrent = "escmsdatabase";  private static MySql.Data.MySqlClient.MySqlConnection OpenDbConnection()  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=" + passwordCurrent + ";database=" + dbmsCurrent + ";persist security info=False");  //open the connection  if (msqlConnection.State != System.Data.ConnectionState.Open)  msqlConnection.Open();  return msqlConnection;  }  #region NewConnection  public static int DoRegisterNewNewConnection(NewConnectionInfo newConnectionDetails)  {  return DoRegisterNewNewConnectionInDb(newConnectionDetails);  }  private static int DoRegisterNewNewConnectionInDb(NewConnectionInfo newConnectionDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO application\_register(apps\_no,payment\_id,customerId,received\_date) "  + "VALUES(@apps\_no,@payment\_id,@customerId,@received\_date)";  msqlCommand.Parameters.AddWithValue("@apps\_no", newConnectionDetails.appsNo);  msqlCommand.Parameters.AddWithValue("@payment\_id", newConnectionDetails.paymentId);  msqlCommand.Parameters.AddWithValue("@customerId", newConnectionDetails.customerId);  msqlCommand.Parameters.AddWithValue("@received\_date", newConnectionDetails.receivedDate);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<NewConnectionInfo> GetAllNewConnectionList()  {  return QueryAllNewConnectionList();  }  private static List<NewConnectionInfo> QueryAllNewConnectionList()  {  List<NewConnectionInfo> NewConnectionList = new List<NewConnectionInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From application\_register;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  NewConnectionInfo NewConnection = new NewConnectionInfo();  NewConnection.appsNo = msqlReader.GetString("apps\_no");  NewConnection.customerId = msqlReader.GetString("customerId");  NewConnection.paymentId = msqlReader.GetString("payment\_id");  NewConnection.receivedDate = msqlReader.GetDateTime("received\_date");  NewConnectionList.Add(NewConnection);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return NewConnectionList;  }  public static void DeleteNewConnection(string newConnectionToDelete)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "DELETE FROM application\_register WHERE apps\_no=@newConnectionIdToDelete";  msqlCommand.Parameters.AddWithValue("@newConnectionIdToDelete", newConnectionToDelete);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static void EditNewConnection(NewConnectionInfo newConnectionToEdit)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "UPDATE application\_register SET payment\_id=@payment\_id,received\_date=@received\_date WHERE apps\_no=@apps\_no";  msqlCommand.Parameters.AddWithValue("@payment\_id", newConnectionToEdit.paymentId);  msqlCommand.Parameters.AddWithValue("@received\_date", newConnectionToEdit.receivedDate);  msqlCommand.Parameters.AddWithValue("@apps\_no", newConnectionToEdit.appsNo);  msqlCommand.ExecuteNonQuery();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static string GetNewConnectionCustomerId(string appNo)  {  string cusId = null;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select customerId From application\_register WHERE apps\_no=@apps\_no;";  msqlCommand.Parameters.AddWithValue("@apps\_no", appNo);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  msqlReader.Read();  cusId = msqlReader.GetString("customerId");  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return cusId;  }  #endregion  #region Employee  public static int DoRegisterNewEmployee(EmployeeInfo employeeDetails)  {  return DoRegisterNewEmployeeInDb(employeeDetails);  }  private static int DoRegisterNewEmployeeInDb(EmployeeInfo employeeDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO employee(id,name,address,contact,post\_type,doj,department) "  + "VALUES(@id,@name,@address,@contact,@post\_type,@doj,@department)";  msqlCommand.Parameters.AddWithValue("@id", employeeDetails.id);  msqlCommand.Parameters.AddWithValue("@name", employeeDetails.name);  msqlCommand.Parameters.AddWithValue("@address", employeeDetails.address);  msqlCommand.Parameters.AddWithValue("@contact", employeeDetails.contact);  msqlCommand.Parameters.AddWithValue("@post\_type", employeeDetails.postType);  msqlCommand.Parameters.AddWithValue("@doj", employeeDetails.doj);  msqlCommand.Parameters.AddWithValue("@department", employeeDetails.department);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<EmployeeInfo> GetAllEmployeeList()  {  return QueryAllEmployeeList();  }  private static List<EmployeeInfo> QueryAllEmployeeList()  {  List<EmployeeInfo> EmployeeList = new List<EmployeeInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From employee;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  EmployeeInfo Employee = new EmployeeInfo();  Employee.id = msqlReader.GetString("id");  Employee.name = msqlReader.GetString("name");  Employee.address = msqlReader.GetString("address");  Employee.contact = msqlReader.GetString("contact");  Employee.postType = (PostType)Enum.Parse(typeof(PostType), msqlReader.GetString("post\_type"), true);  Employee.doj = msqlReader.GetDateTime("doj");  Employee.department = msqlReader.GetString("department");  EmployeeList.Add(Employee);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return EmployeeList;  }  public static void DeleteEmployee(string employeeToDelete)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "DELETE FROM employee WHERE id= @employeeIdToDelete";  msqlCommand.Parameters.AddWithValue("@employeeIdToDelete", employeeToDelete);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static void EditEmployee(EmployeeInfo employeeToEdit)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "UPDATE employee SET name=@name,address=@address,contact=@contact,post\_type=@postType,doj=@doj WHERE id=@id";  msqlCommand.Parameters.AddWithValue("@name", employeeToEdit.name);  msqlCommand.Parameters.AddWithValue("@address", employeeToEdit.address);  msqlCommand.Parameters.AddWithValue("@contact", employeeToEdit.contact);  msqlCommand.Parameters.AddWithValue("@postType", employeeToEdit.postType);  msqlCommand.Parameters.AddWithValue("@doj", employeeToEdit.doj);  msqlCommand.Parameters.AddWithValue("@department", employeeToEdit.department);  msqlCommand.Parameters.AddWithValue("@id", employeeToEdit.id);  msqlCommand.ExecuteNonQuery();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  #endregion  #region Contractor  public static int DoRegisterNewContractor(ContractorInfo contractorDetails)  {  return DoRegisterNewContractorInDb(contractorDetails);  }  private static int DoRegisterNewContractorInDb(ContractorInfo contractorDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();    try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO contractor(id,name,address,contact,contract\_details) "  + "VALUES(@id,@name,@address,@contact,@contract\_details)";  msqlCommand.Parameters.AddWithValue("@id", contractorDetails.id);  msqlCommand.Parameters.AddWithValue("@name", contractorDetails.name);  msqlCommand.Parameters.AddWithValue("@address", contractorDetails.address);  msqlCommand.Parameters.AddWithValue("@contact", contractorDetails.contact);  msqlCommand.Parameters.AddWithValue("@contract\_details", contractorDetails.details);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<ContractorInfo> GetAllContractorList()  {  return QueryAllContractorList();  }  private static List<ContractorInfo> QueryAllContractorList()  {  List<ContractorInfo> ContractorList = new List<ContractorInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From contractor;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  ContractorInfo Contractor = new ContractorInfo();  Contractor.id = msqlReader.GetString("id");  Contractor.name = msqlReader.GetString("name");  Contractor.address = msqlReader.GetString("address");  Contractor.contact = msqlReader.GetString("contact");  Contractor.details = msqlReader.GetString("contract\_details");  ContractorList.Add(Contractor);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return ContractorList;  }  public static void DeleteContractor(string contractorToDelete)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "DELETE FROM contractor WHERE id= @contractorIdToDelete";  msqlCommand.Parameters.AddWithValue("@contractorIdToDelete", contractorToDelete);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static void EditContractor(ContractorInfo contractorToEdit)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "UPDATE contractor SET name=@name,address=@address,contact=@contact,contract\_details=@details WHERE id=@id";  msqlCommand.Parameters.AddWithValue("@name", contractorToEdit.name);  msqlCommand.Parameters.AddWithValue("@address", contractorToEdit.address);  msqlCommand.Parameters.AddWithValue("@contact", contractorToEdit.contact);  msqlCommand.Parameters.AddWithValue("@details", contractorToEdit.details);  msqlCommand.Parameters.AddWithValue("@id", contractorToEdit.id);  msqlCommand.ExecuteNonQuery();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  #endregion  #region Payment  public static List<PaymentInfo> GetUnassignedPaymentList()  {  List<PaymentInfo> PaymentList = new List<PaymentInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From payment;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  PaymentInfo Payment = new PaymentInfo();  Payment.id = msqlReader.GetString("id");  Payment.customerId = msqlReader.GetString("customerId");  Payment.amount = msqlReader.GetDouble("amount");  Payment.dop = msqlReader.GetDateTime("dop");  PaymentList.Add(Payment);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return PaymentList;  }  public static int DoRegisterNewPayment(PaymentInfo peaymentDetails)  {  return DoRegisterNewPaymentInDb(peaymentDetails);  }  private static int DoRegisterNewPaymentInDb(PaymentInfo peaymentDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();    try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO payment(id,customerId,amount,dop) "  + "VALUES(@id,@customerId,@amount,@dop)";  msqlCommand.Parameters.AddWithValue("@id", peaymentDetails.id);  msqlCommand.Parameters.AddWithValue("@customerId", peaymentDetails.customerId);  msqlCommand.Parameters.AddWithValue("@amount", peaymentDetails.amount);  msqlCommand.Parameters.AddWithValue("@dop", peaymentDetails.dop);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<PaymentInfo> GetAllPaymentList()  {  return QueryAllPaymentList();  }  private static List<PaymentInfo> QueryAllPaymentList()  {  List<PaymentInfo> PaymentList = new List<PaymentInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From payment;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  PaymentInfo Payment = new PaymentInfo();  Payment.id = msqlReader.GetString("id");  Payment.customerId = msqlReader.GetString("customerId");  Payment.amount = msqlReader.GetDouble("amount");  Payment.dop = msqlReader.GetDateTime("dop");  PaymentList.Add(Payment);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return PaymentList;  }  public static void DeletePayment(string paymentToDelete)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "DELETE FROM payment WHERE id=@paymentIdToDelete";  msqlCommand.Parameters.AddWithValue("@paymentIdToDelete", paymentToDelete);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static void EditPayment(PaymentInfo paymentToEdit)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "UPDATE payment SET customerId=@customerId,amount=@amount,dop=@dop WHERE id=@id";  msqlCommand.Parameters.AddWithValue("@customerId", paymentToEdit.customerId);  msqlCommand.Parameters.AddWithValue("@amount", paymentToEdit.amount);  msqlCommand.Parameters.AddWithValue("@dop", paymentToEdit.dop);  msqlCommand.Parameters.AddWithValue("@id", paymentToEdit.id);  msqlCommand.ExecuteNonQuery();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static PaymentInfo GetPaymentInfo(string paymentId)  {  PaymentInfo payment = new PaymentInfo();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From payment WHERE id=@id;";  msqlCommand.Parameters.AddWithValue("@id", paymentId);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  msqlReader.Read();  payment.id = msqlReader.GetString("id");  payment.customerId = msqlReader.GetString("customerId");  payment.amount = msqlReader.GetDouble("amount");  payment.dop = msqlReader.GetDateTime("dop");  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return payment;  }  #endregion  #region Estimator  public static List<EmployeeInfo> GetAllEstimatorList()  {  List<EmployeeInfo> EmployeeList = new List<EmployeeInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From employee WHERE post\_type=@post;";  msqlCommand.Parameters.AddWithValue("@post", "Estimator");  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  EmployeeInfo Employee = new EmployeeInfo();  Employee.id = msqlReader.GetString("id");  Employee.name = msqlReader.GetString("name");  Employee.address = msqlReader.GetString("address");  Employee.contact = msqlReader.GetString("contact");  Employee.postType = (PostType)Enum.Parse(typeof(PostType), msqlReader.GetString("post\_type"), true);  Employee.doj = msqlReader.GetDateTime("doj");  Employee.department = msqlReader.GetString("department");  EmployeeList.Add(Employee);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return EmployeeList;  }  public static int DoRegisterNewEstimate(estimateInfo estimateToEdit)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();    try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "insert into estimate(appNo)" + "values(@appNo)";  msqlCommand.Parameters.AddWithValue("@appNo", estimateToEdit.appsNo);  msqlCommand.ExecuteNonQuery();  // msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static void EditEstimate(estimateInfo estimateToEdit)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "UPDATE estimate SET estimator=@estimator,wireLength=@wireLength,angleType=@angleType,angleWeight=@angleWeight,amountQuotation=@amountQuotation WHERE appNo=@appNo";  msqlCommand.Parameters.AddWithValue("@estimator", estimateToEdit.estimator);  msqlCommand.Parameters.AddWithValue("@wireLength", estimateToEdit.wireLength);  msqlCommand.Parameters.AddWithValue("@angleType", estimateToEdit.angleType);  msqlCommand.Parameters.AddWithValue("@angleWeight", estimateToEdit.angleWeight);  msqlCommand.Parameters.AddWithValue("@amountQuotation", estimateToEdit.quotationAmount);  msqlCommand.Parameters.AddWithValue("@appNo", estimateToEdit.appsNo);  msqlCommand.ExecuteNonQuery();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static List<estimateInfo> GetAllEstimateList()  {  List<estimateInfo> estimateList = new List<estimateInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From estimate;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  estimateInfo estimate = new estimateInfo();  estimate.appsNo = msqlReader.GetString("appNo");  estimate.wireLength = msqlReader.GetDouble("wireLength");  estimate.angleType = (AngleType)Enum.Parse(typeof(AngleType), msqlReader.GetString("angleType"), true);  estimate.angleWeight = msqlReader.GetDouble("angleWeight");  estimate.quotationAmount = msqlReader.GetDouble("amountQuotation");  estimate.estimator = msqlReader.GetString("estimator");  estimateList.Add(estimate);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return estimateList;  }  #endregion  #region Customer  public static CustomerInfo GetCustomerInfo(string cusId)  {  CustomerInfo customer = new CustomerInfo();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From customer WHERE id=@id;";  msqlCommand.Parameters.AddWithValue("@id", cusId);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  msqlReader.Read();  customer.name = msqlReader.GetString("name");  customer.address = msqlReader.GetString("address");  customer.contact = msqlReader.GetString("contact");  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return customer;  }  public static int DoRegisterNewCustomer(CustomerInfo customerDetails)  {  return DoRegisterNewCustomerInDb(customerDetails);  }  private static int DoRegisterNewCustomerInDb(CustomerInfo customerDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  //define the connection used by the command object  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "INSERT INTO customer(id,name,address,contact) "  + "VALUES(@id,@name,@address,@contact)";  msqlCommand.Parameters.AddWithValue("@id", customerDetails.id);  msqlCommand.Parameters.AddWithValue("@name", customerDetails.name);  msqlCommand.Parameters.AddWithValue("@address", customerDetails.address);  msqlCommand.Parameters.AddWithValue("@contact", customerDetails.contact);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<CustomerInfo> GetAllCustomerList()  {  return QueryAllCustomerList();  }  private static List<CustomerInfo> QueryAllCustomerList()  {  List<CustomerInfo> CustomerList = new List<CustomerInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From customer;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  CustomerInfo Customer = new CustomerInfo();  Customer.id = msqlReader.GetString("id");  Customer.name = msqlReader.GetString("name");  Customer.address = msqlReader.GetString("address");  Customer.contact = msqlReader.GetString("contact");  CustomerList.Add(Customer);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return CustomerList;  }  public static void DeleteCustomer(string customerToDelete)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "DELETE FROM customer WHERE id=@customerIdToDelete";  msqlCommand.Parameters.AddWithValue("@customerIdToDelete", customerToDelete);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static void EditCustomer(CustomerInfo employeeToEdit)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;    msqlCommand.CommandText = "UPDATE customer SET name=@name,address=@address,contact=@contact WHERE id=@id";  msqlCommand.Parameters.AddWithValue("@name", employeeToEdit.name);  msqlCommand.Parameters.AddWithValue("@address", employeeToEdit.address);  msqlCommand.Parameters.AddWithValue("@contact", employeeToEdit.contact);  msqlCommand.Parameters.AddWithValue("@id", employeeToEdit.id);  msqlCommand.ExecuteNonQuery();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  #endregion  public static void assignEstimator(estimateInfo estimateToEdit)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "UPDATE estimate SET estimator=@estimator WHERE appNo=@appNo";  msqlCommand.Parameters.AddWithValue("@estimator", estimateToEdit.estimator);  msqlCommand.Parameters.AddWithValue("@appNo", estimateToEdit.appsNo);  msqlCommand.ExecuteNonQuery();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  public static List<estimateInfo> GetAllEstimateListWithContractor()  {  List<estimateInfo> estimateList = new List<estimateInfo>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "Select \* From estimate;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  estimateInfo estimate = new estimateInfo();  estimate.appsNo = msqlReader.GetString("appNo");  estimate.wireLength = msqlReader.GetDouble("wireLength");  estimate.angleType = (AngleType)Enum.Parse(typeof(AngleType), msqlReader.GetString("angleType"), true);  estimate.angleWeight = msqlReader.GetDouble("angleWeight");  estimate.quotationAmount = msqlReader.GetDouble("amountQuotation");  estimate.estimator = msqlReader.GetString("estimator");  estimate.contractor = msqlReader.GetString("contractor");  estimateList.Add(estimate);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return estimateList;  }  public static void assignContractor(estimateInfo contractorToEdit)  {  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlCommand.CommandText = "UPDATE estimate SET contractor=@contractor WHERE appNo=@appNo";  msqlCommand.Parameters.AddWithValue("@contractor", contractorToEdit.contractor);  msqlCommand.Parameters.AddWithValue("@appNo", contractorToEdit.appsNo);  msqlCommand.ExecuteNonQuery();  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  }  }  } |

### code efficiency

We started working on the project keeping in mind that we must develop it in a way that it not only provides a very easy to use GUI but also provide a fast and flexible service to the users. We know that a particular work can be done in more than one ways. We have tried all the options and then chose the one which provides the fastest and most secure performance. First of all, we have used the latest technologies of Microsoft like visual studio 2010 as IDE and WPF as GUI to keep our application’s performance few steps ahead. We have studiesd all the rules of software development life cycle and applied them to keep our application felexible. We have given special attention to the storage related codes. We have avoided all the unnecessary database codes and kept them as short as possible without harming our purpose so that insertion, updation, deletion and fetching of data take place flexibly. You can see the result as a user; our application does all the works very smoothly.

## testing approach

We have vigorously tested the application to make it error free and smooth. To achieve our goal we tested the modules differently inside the codes and then tested the entire application as a whole to mark its drawbacks.

### unit testing

The modules we tested and corrected are,

* ESCMS GUI: all the GUI related issues were tracked and solved while testing this module.
* ESCMS ENGINE: all the bugs and errors encountered in the controller section were solved.
* ESCMS DATA: issues related to data passing from one module to others were tracked and corrected in this module.
* ESCMS STORAGE: tracked and solved the storage related problems in this module.

### integrated testing

## modification and improvements

Following modification and improvements of the application can be done in the future:

* A new feature can be added for the customers to take complains from them when they are having problems with their connection. When a customer lodges a complaint, it could be saved and sent to the specific persons for solving them. Through the complaint id, we can update the status of the complaint by querying to the customer.
* The application could be expanded to make it manage multiple electric supplies or even an entire region.
* We can integrate the application with a website through which a user can see his bill status and other details can lodge a complaint and check its status as well.
* The application could be made more secure by adding a backup and restore feature as it holds many information data losing which could really be a huge mess.

# result and discussion

## test reports

|  |  |  |
| --- | --- | --- |
| Test Case Id | Comment | Status |
| ESCMS-001 | NA | PASS |
| ESCMS-002 | NA | PASS |

|  |  |  |
| --- | --- | --- |
| ESCMS-003 | NA | PASS |
| ESCMS-004 | NA | PASS |
| ESCMS-005 | NA | PASS |
| ESCMS-006 | NA | PASS |
| ESCMS-007 | NA | PASS |
| ESCMS-008 | NA | PASS |
| ESCMS-009 | NA | PASS |
| ESCMS-010 | NA | PASS |
| ESCMS-011 | NA | PASS |
| ESCMS-012 | NA | PASS |
| ESCMS-013 | NA | PASS |
| ESCMS-014 | NA | PASS |
| ESCMS-015 | NA | PASS |
| ESCMS-016 | NA | PASS |
| ESCMS-017 | NA | PASS |
| ESCMS-018 | NA | PASS |
| ESCMS-019 | NA | PASS |
| ESCMS-020 | NA | PASS |
| ESCMS-021 | NA | PASS |
| ESCMS-022 | NA | PASS |
| ESCMS-023 | NA | PASS |
| ESCMS-024 | NA | PASS |
| ESCMS-025 | NA | PASS |

## user documentation

Though it provides many features to a user, ECMS is developed to complete a single important task, managing the electric supply. In this documentation I am going describe how it works.

* A customer who needs a new electric connection comes to the electric office. To get a new connection he needs to register his name first which is stored inside the database and can be viewed through the customer browser.
* After that he pays the amount of money required for the new connection. Payment can be viewed and added through the payment window.
* When the payment is complete his name is visible inside the Add new connection window and the electric office employees add his name for new connection.
* His name after that comes inside the estimate and quotation window with other necessary details. A new estimator is assigned against his name and the estimator after visiting the customer’s house fills up the necessary quotation field.
* When the quotation fields are filled up, the name of the customer with all other details come inside the assign contractor window. The contractor is then informed to visit the customer and provide a new connection at his home.
* .The contractor, after providing a new connection gives a bill to the office which is saved inside the billing window and the money given to the contractor is deducted from the initial payment done by the customer.

# conclusions

## conclusion

We wanted to select a project that not only includes challenging work but also provides us a clear concept of real life project and applicability of languages and technologies we are using for completing the project. We chose C sharp as our programming language as it is one of the best and most used programming language in the IT industry. We used WPF, the latest GUI technology of Microsoft to provide an easy to use and splendid user interface. MYSQL is used here which is the most popular (and free) DBMS software in the world. We started the work keeping in mind that we must develop a flexible and fast application so that it becomes popular among the users. Our modular approach has made it easier to understand, easy to debug application. We can reuse the codes in different applications as well. Our vigorous testing has made it an error free application. It is password protected so all the data kept in the application is secured. With a few improvements and modifications it could be a very useful application in real time.

## limitation of the system

Following modification and improvements of the application can be done in the future:

* A new feature can be added for the customers to take complains from them when they are having problems with their connection. When a customer lodges a complaint, it could be saved and sent to the specific persons for solving them. Through the complaint id, we can update the status of the complaint by querying to the customer.
* The application could be expanded to make it manage multiple electric supplies or even an entire region.
* We can integrate the application with a website through which a user can see his bill status and other details can lodge a complaint and check its status as well.
* The application could be made more secure by adding a backup and restore feature as it holds many information data losing which could really be a huge mess.

## future scope of the project

* Currently this software is aimed for a single electric supply office customer management. It can be extended to support networked multiple electric supply office and have a centralized database and to serve wider range of customers of Electric Supply around the country.
* To support UNIX / Linux, MAC OSX Operating systems.
* Integration with Electric Billing System.

# REFERENCES

* <http://en.wikipedia.org>
* <http://msdn.microsoft.com/en-us/>
* <http://www.microsoft.com/en-us/default.aspx>
* <http://www.codeplex.com/>
* <http://stackoverflow.com/>
* <http://www.codeguru.com/>
* [http://www.w3schools.com](http://www.w3schools.com/)
* [www.mysql.org](http://www.mysql.org)
* Electric Supply Professionals
* **Programming C#** - E. R. Balaguruswamy