|  |
| --- |
|  |
| **ELECTRIC SUPPLY CUSTOMER MANAGEMENT SYSTEM** |
| MCA 6th SEMESTER  MCSP-060(PROJECT REPORT) |
|  |
| **SANGITA MONDAL**  **ENROLLMENT NO: 105140503** |
|  |

|  |
| --- |
|  |

Table of Contents

[Table of Contents 1](#_Toc351966565)

[1. Introduction 6](#_Toc351966566)

[1.1 Background 6](#_Toc351966567)

[1.2 Objective 6](#_Toc351966568)

[System Analysis 7](#_Toc351966569)

[Identification of Need: 7](#_Toc351966570)

[Preliminary Investigation: 7](#_Toc351966571)

[Feasibility Study: 7](#_Toc351966572)

[Project Planning & Scheduling: 7](#_Toc351966573)

[Gantt chart 7](#_Toc351966574)

[Tracking GantT 8](#_Toc351966575)

[Pert chart (Network Diagram) 8](#_Toc351966576)

[Electric Supply Customer Management System 9](#_Toc351966577)

[Software requirement specifications (SRS): 9](#_Toc351966578)

[Functional Requirements 9](#_Toc351966579)

[Apply for new connection 9](#_Toc351966580)

[**Introduction** 9](#_Toc351966581)

[**Inputs** 10](#_Toc351966582)

[**Processing** 10](#_Toc351966583)

[**Outputs** 10](#_Toc351966584)

[check connection request status 10](#_Toc351966585)

[**Introduction** 10](#_Toc351966586)

[**Inputs** 10](#_Toc351966587)

[**Processing** 10](#_Toc351966588)

[**Outputs** 10](#_Toc351966589)

[Create a vendor task 10](#_Toc351966590)

[**Introduction** 10](#_Toc351966591)

[**Inputs** 10](#_Toc351966592)

[**Processing** 11](#_Toc351966593)

[**Outputs** 11](#_Toc351966594)

[Vendor will get a notification about the task and a printed work order. 11](#_Toc351966595)

[vendor task status update 11](#_Toc351966596)

[**Introduction** 11](#_Toc351966597)

[**Inputs** 11](#_Toc351966598)

[**Processing** 11](#_Toc351966599)

[**Outputs** 11](#_Toc351966600)

[Vendor will get a notification about the task update and a printed payment order. 11](#_Toc351966601)

[Generate report 11](#_Toc351966602)

[**Introduction** 11](#_Toc351966603)

[**Inputs** 11](#_Toc351966604)

[**Processing** 12](#_Toc351966605)

[**Outputs** 12](#_Toc351966606)

[Non-Functional Requirements 13](#_Toc351966607)

[Data models 13](#_Toc351966608)

[Data Flow Diagram 13](#_Toc351966609)

[E-R Diagram 16](#_Toc351966610)

[**Relationship between Entities:** 17](#_Toc351966611)

[Electric Supply office has Customers1 : N 17](#_Toc351966612)

[Class Diagram 19](#_Toc351966613)

[Sequence diagrams 21](#_Toc351966614)

[1. system design 21](#_Toc351966615)

[1.1 Basic Modules 21](#_Toc351966616)

[*ECMS contains following main modules:* 22](#_Toc351966617)

[1.2 data design 23](#_Toc351966618)

[1.2.1 Schema Design 23](#_Toc351966619)

[1.2.2 data integrity and constraints 25](#_Toc351966620)

[1.3 procedural design 25](#_Toc351966621)

[1.3.1 logic diagrams 25](#_Toc351966622)

[1.3.2 data structures 25](#_Toc351966623)

[1.3.3 ALGOrITHITMS AND DESIGN 29](#_Toc351966624)

[1.4 user interface design 29](#_Toc351966625)

[1.4.1 main window 29](#_Toc351966626)

[1.4.2 login window 30](#_Toc351966627)

[1.4.3 customer browser window 31](#_Toc351966628)

[1.4.4 payment window 32](#_Toc351966629)

[1.4.5 connection window 32](#_Toc351966630)

[1.4.6 estimate quotation window 33](#_Toc351966631)

[1.4.7 work order window 33](#_Toc351966632)

[1.4.8 settings window 34](#_Toc351966633)

[1.4.9 contractor billl window 34](#_Toc351966634)

[1.4.10 contractor browser window 35](#_Toc351966635)

[1.4.11 employeee browser window 35](#_Toc351966636)

[1.4.12 report window 36](#_Toc351966637)

[1.4.13 search window 37](#_Toc351966638)

[1.5 security issues 38](#_Toc351966639)

[1.6 test cases design 38](#_Toc351966640)

[Coding 41](#_Toc351966641)

[Complete Project Coding 41](#_Toc351966642)

[Comments and Description of Coding segments 41](#_Toc351966643)

[Comments and API Documentation 41](#_Toc351966644)

[Description of coding 41](#_Toc351966645)

[Standardization of the coding 41](#_Toc351966646)

[Code Efficiency 41](#_Toc351966647)

[Error handling 41](#_Toc351966648)

[Parameters calling/passing 42](#_Toc351966649)

[Validation checks 42](#_Toc351966650)

[Testing 42](#_Toc351966651)

[Testing techniques and testing strategies used 42](#_Toc351966652)

[Database & Data Integrity Testing 42](#_Toc351966653)

[Functional Testing: 42](#_Toc351966654)

[Regression Testing: 43](#_Toc351966655)

[User Interface Testing: 43](#_Toc351966656)

[Performance Profiling: 44](#_Toc351966657)

[Load Testing: 44](#_Toc351966658)

[Stress Testing: 45](#_Toc351966659)

[Volume Testing: 45](#_Toc351966660)

[Security & Access Control Testing: 46](#_Toc351966661)

[Failover & Recovery Testing: 46](#_Toc351966662)

[Configuration Testing: 46](#_Toc351966663)

[Installation/Deploy & Back out Testing: 46](#_Toc351966664)

[Post Production Testing: 47](#_Toc351966665)

[Unit Testing: 47](#_Toc351966666)

[Smoke Testing: 47](#_Toc351966667)

[Data Migration Testing: 47](#_Toc351966668)

[Test reports for Unit Test Cases and System Test Cases 48](#_Toc351966669)

[Test reports for Unit Test Cases 48](#_Toc351966670)

[Test reports for System Test Cases 50](#_Toc351966671)

[Debugging and Code improvement: 51](#_Toc351966672)

[System Security measures: 51](#_Toc351966673)

[Database/data security: 51](#_Toc351966674)

[Creation of User profiles and access rights 51](#_Toc351966675)

[Cost Estimation of the Project along with Cost Estimation Model 52](#_Toc351966676)

[Estimation of development effort 52](#_Toc351966677)

[Estimation of development time 52](#_Toc351966678)

[Reports 53](#_Toc351966679)

[future scope of the project 53](#_Toc351966680)

[limitation of the system 54](#_Toc351966681)

[Bibliography 54](#_Toc351966682)

[Website 54](#_Toc351966683)

[Books 54](#_Toc351966684)

[Appendices 55](#_Toc351966685)

[IDE Used: 55](#_Toc351966686)

[Visual Studio 2010 55](#_Toc351966687)

[Front End - WPF (Windows Presentation Framework) 57](#_Toc351966688)

[Extensible application Markup Language (XAML) 60](#_Toc351966689)

[Programming Framework 61](#_Toc351966690)

[.NET 4.5 61](#_Toc351966691)

[Database/backend: 68](#_Toc351966692)

[MySQL 68](#_Toc351966693)

[ide for Database 71](#_Toc351966694)

[MySQL workbench 72](#_Toc351966695)

[Programming Language 73](#_Toc351966696)

[C# - C sharp 73](#_Toc351966697)

[Dia for Diagram Drawing &Modeling 75](#_Toc351966698)

[Google Spreadsheet Interface: 76](#_Toc351966699)

[Cacoo:: online drawing tool 77](#_Toc351966700)

[Creating Diagrams 77](#_Toc351966701)

[Collaboration 77](#_Toc351966702)

[Sharing Diagrams 77](#_Toc351966703)

[Managing Diagrams 78](#_Toc351966704)

[Languages and Time Zones 78](#_Toc351966705)

[Security 78](#_Toc351966706)

[API 78](#_Toc351966707)

[Access Token:https://cacoo.com/oauth/access\_token 79](#_Toc351966708)

[Version Control System : GitHub 79](#_Toc351966709)

[Description 79](#_Toc351966710)

[Limitations and constraints 79](#_Toc351966711)

# 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.

# System Analysis

## Identification of Need:

The aim 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.

## Preliminary Investigation:

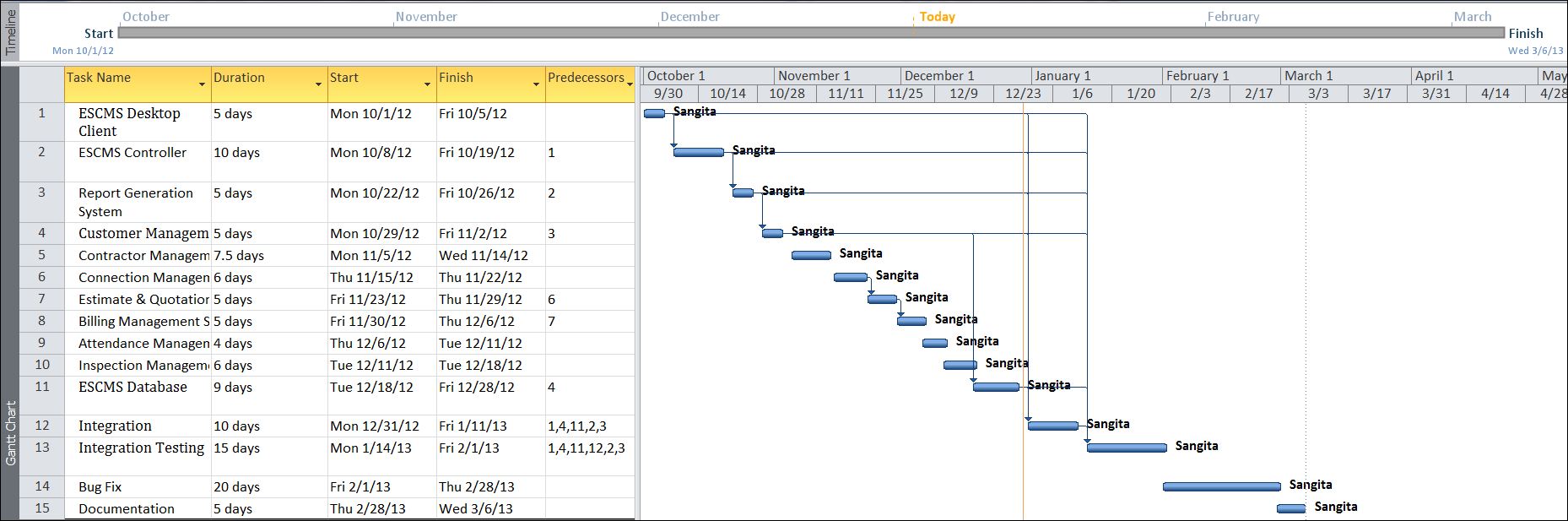
Currently this software is aimed for a single electric supply office customer management. It can be extended to support networked multiple electric supply office and have a centralized database and to serve wider range of customers of Electric Supply around the country.We have developed this for Desktop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

## Feasibility Study:

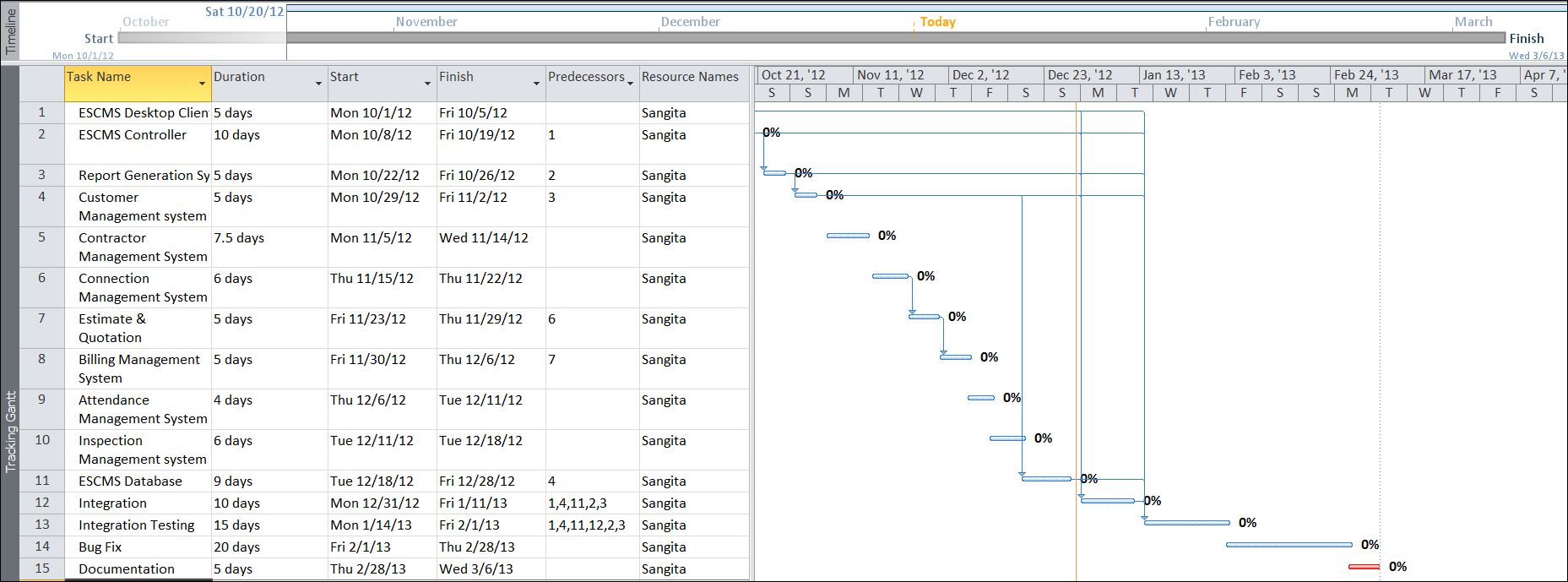
As explained before, people are now completely technology dependant and number of smart phone and computer users is increasing day by day. So, an application like **ESCMS** would be really useful to them. With minimal effort people can manage, keeptrack and generate paper report and main ledgers to help transition from older system. That is why; I know that after completion, this software will be used by thousands of users.

## Project Planning & Scheduling:

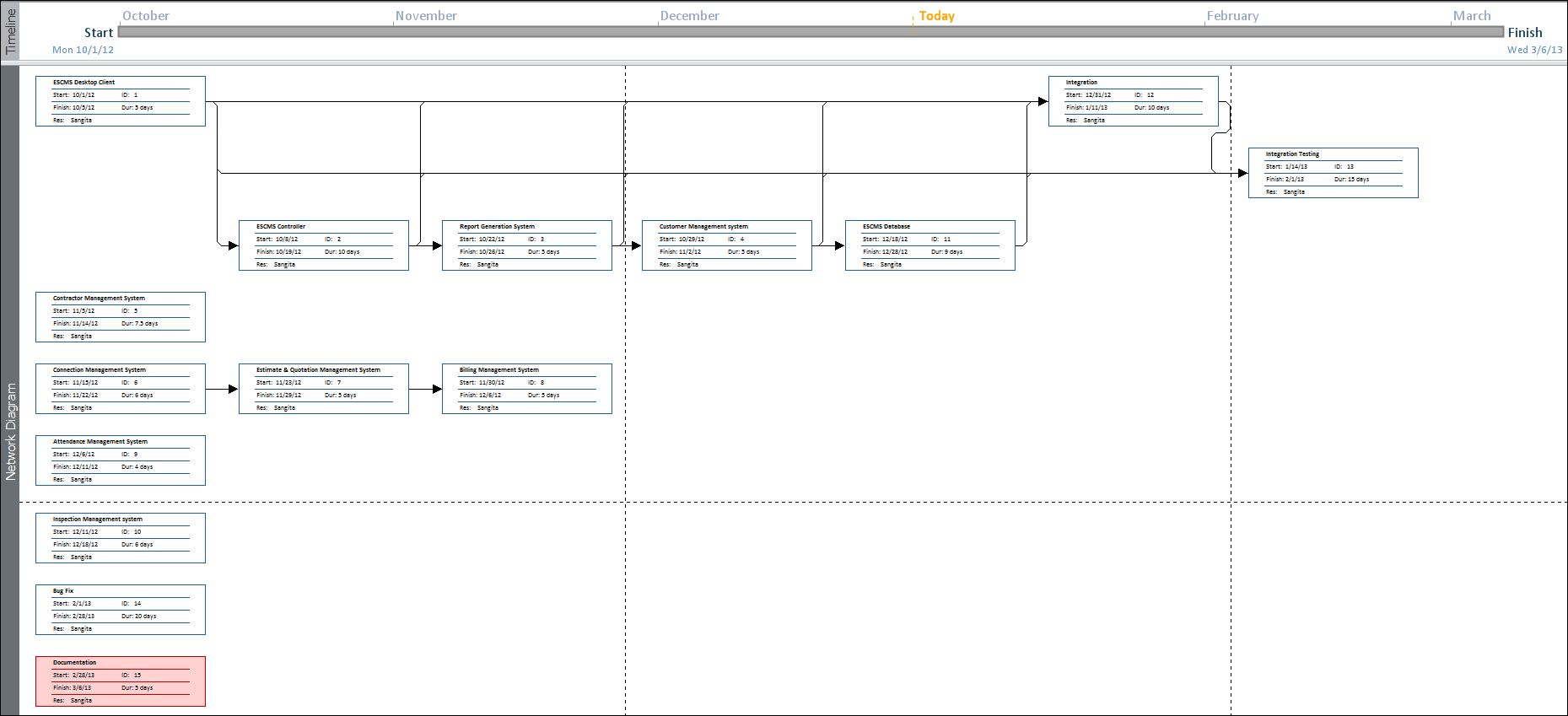
### Gantt chart



### Tracking GantT



### Pert chart (Network Diagram)



Electric Supply Customer Management System

# Software requirement specifications (SRS):

## 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.

# Software Engineering Paradigm applied

We have followed agile version of Model Driven Development (MDD). As the name implies, AMDD is the agile version of Model Driven Development (MDD). MDD is an approach to software development where extensive models are created before source code is written. A primary example of MDD is the Object Management Group (OMG)’s Model Driven Architecture (MDA) standard. With MDD a serial approach to development is often taken, MDD is quite popular with traditionalists, although as the RUP/EUP shows it is possible to take an iterative approach with MDD. The difference with AMDD is that instead of creating extensive models before writing source code you instead create agile models which are just barely good enough that drive your overall development efforts. AMDD is a critical strategy for scaling agile software development beyond the small, co-located team approach that we saw during the first stage of agile adoption.



Figure 1The AMDD lifecycle: Modeling activities throughout the lifecycle of a project

Above Figure depicts a high-level lifecycle for AMDD for the release of a system. First, let’s start with how to read the diagram. Each box represents a development activity. The envisioning includes two main sub-activities, initial requirements envisioning and initial architecture envisioning. These are done during iteration 0, iteration being another term for cycle or sprint. “Iteration 0” is a common term for the first iteration before you start into development iterations, which are iterations one and beyond (for that release). The other activities – iteration modeling, model storming, reviews, and implementation – potentially occur during any iteration, including iteration 0. The time indicated in each box represents the length of an average session: perhaps you’ll model for a few minutes then code for several hours. I’ll discuss timing issues in more detail below..



Figure 2AMDD Through the Agile Development Lifecycle.

Above Figure depicts how the AMDD activities fit into the various iterations of the agile software development lifecycle.  It's simply another way to show that an agile project begins with some initial modelling and that modelling still occurs in each construction’s iteration.

## Data models

### Data Flow Diagram

#### CONTEXT DIAGRAM



#### 1-Level DFD



#### 2-Level DFD



## 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

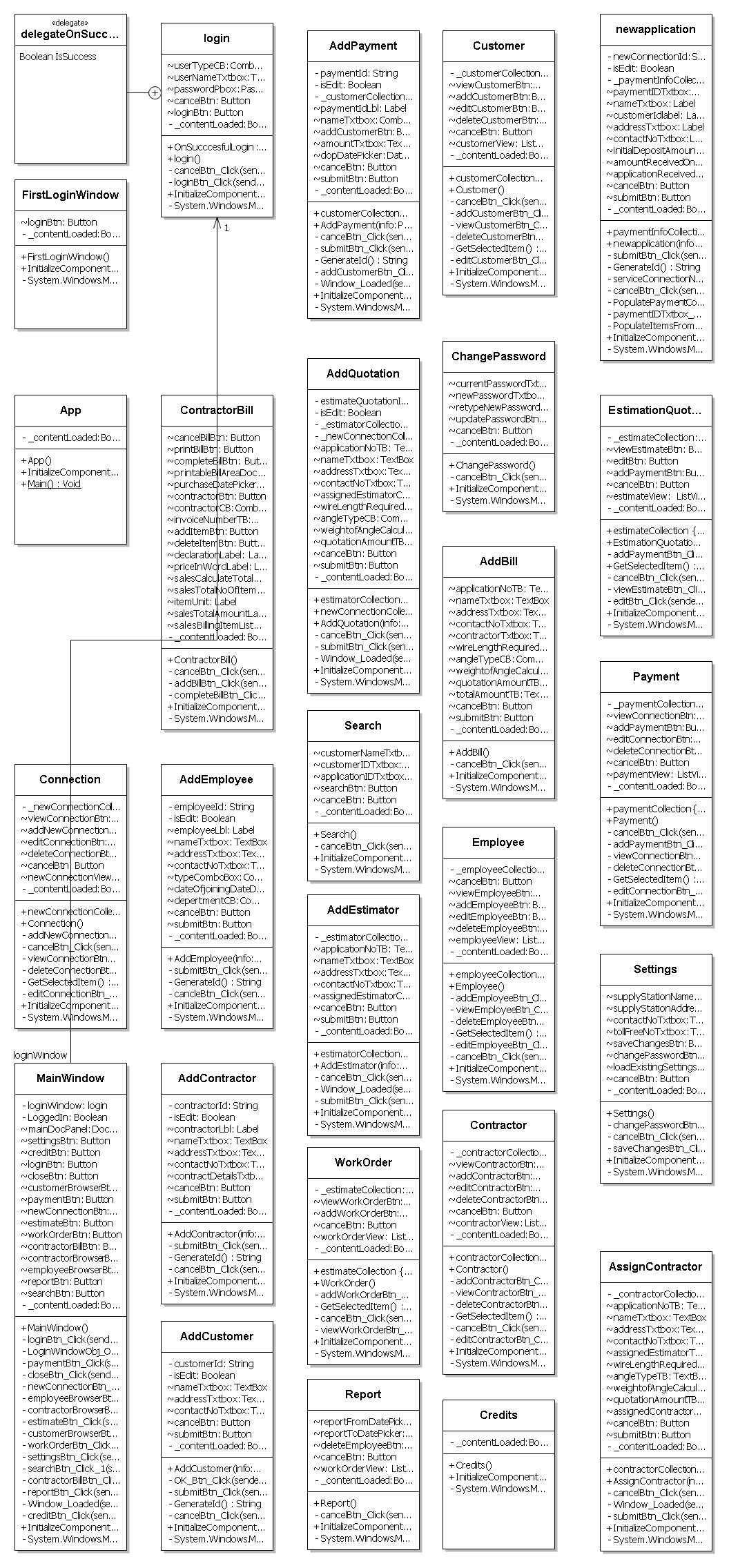
User uses Service Connection 1 : N

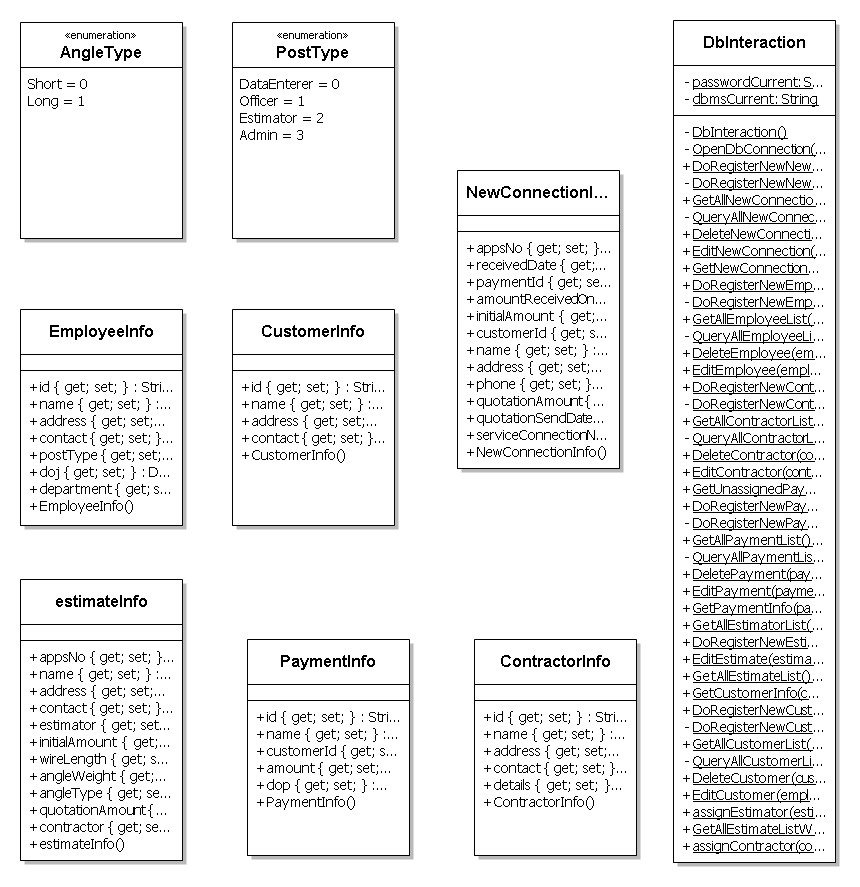
Employees provides Estimates M : N





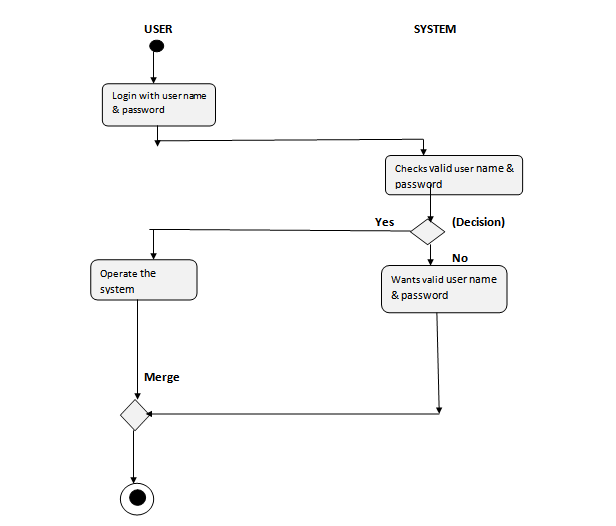
## Class Diagram



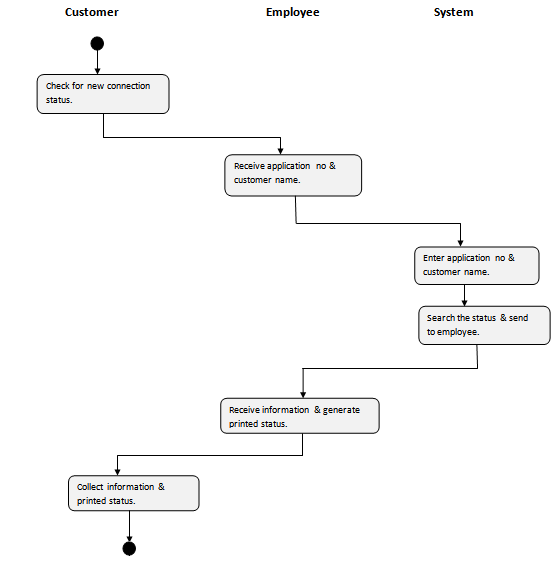


## Control Flow Diagrams

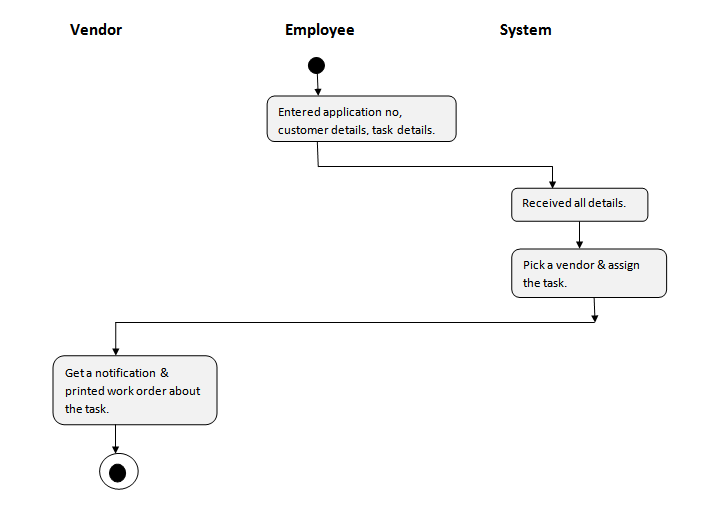
### LOGIN



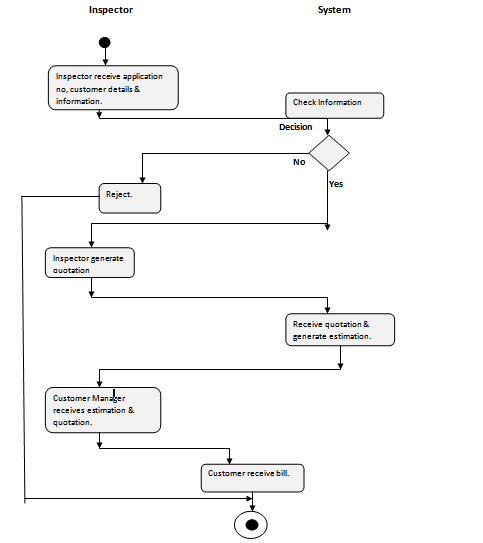
### For Connection Request Status

****

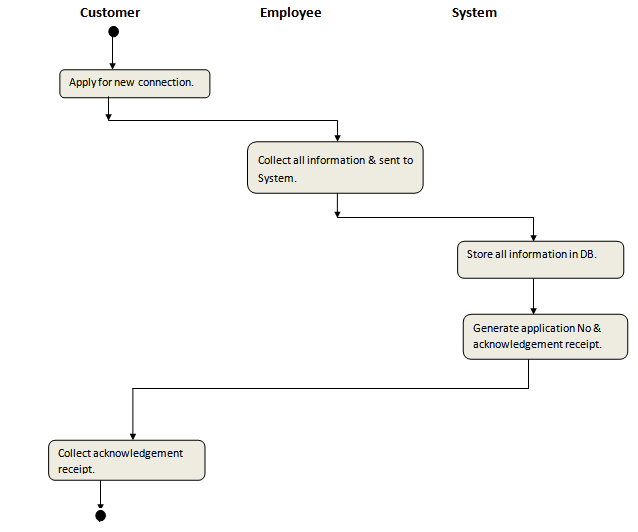
### For Create a Vendor Task

****

### For Create an Inspector Task

****

### For Applying New Connection



## Class Diagram

### Sequence Diagram for apply new connection from Electric Supply Office

Payment Bill Bill]

Register

[Report]

Sequence Diagram for apply new connection from Electric Supply Office

: Employee

: System

: Inspector

Contractor

Submit All DuePayment

Connection Complete &Request for Payment

Submit Meter

Report

Report

Task Details

Report

Request Payment

Report

Inspection

Send Customer Details

Request for new connection

: Register Customer for New Connection

# 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

### 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

1. We have used Integrity constraints in **DNBSN** to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. There are many types of integrity constraints in **DNBSN** that play a role in referential integrity.
2. Codd initially defined two sets of constraints but, in his second version of the relational model, he came up with four integrity constraints:

#### Entity integrity

1. In **EMS** we used various type of primary key and consciously we set the primary key property as not null. The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation. Having null value for the primary key implies that we cannot identify some tuples.This also specifies that there may not be any duplicate entries in primary key column key row.

#### Referential Integrity

1. The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. It is a rule that maintains consistency among the rows of the two relations.

#### Domain Integrity

1. **EMS** has various type of data field with set by default value of Null because if the value is not provided by the user, the vale will be set as null. The domain integrity states that every element from a relation should respect the type and restrictions of its corresponding attribute. A type can have a variable length which needs to be respected. Restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.

#### User Defined Integrity

1. A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business.

### data structures

/\* ContractorInfo.cs \*/

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespace**ESCMS**Data

{

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;

namespace**ESCMS**Data

{

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;

namespace**ESCMS**Data

{

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;

namespace**ESCMS**Data

{

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;

namespace**ESCMS**Data

{

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;

namespace**ESCMS**Data

{

publicclassPaymentInfo

{

publicstring id { get; set; }

publicstring name { get; set; }

publicstringcustomerId { get; set; }

publicdouble amount { get; set; }

publicDateTimedop { get; set; }

}

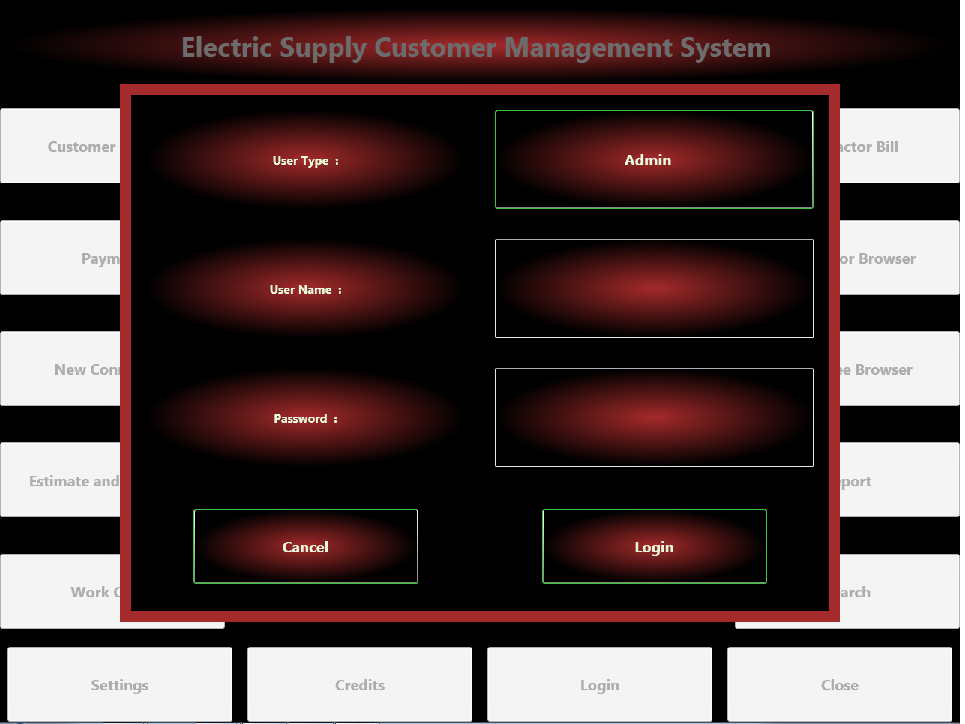
}

### user interface design

#### main window



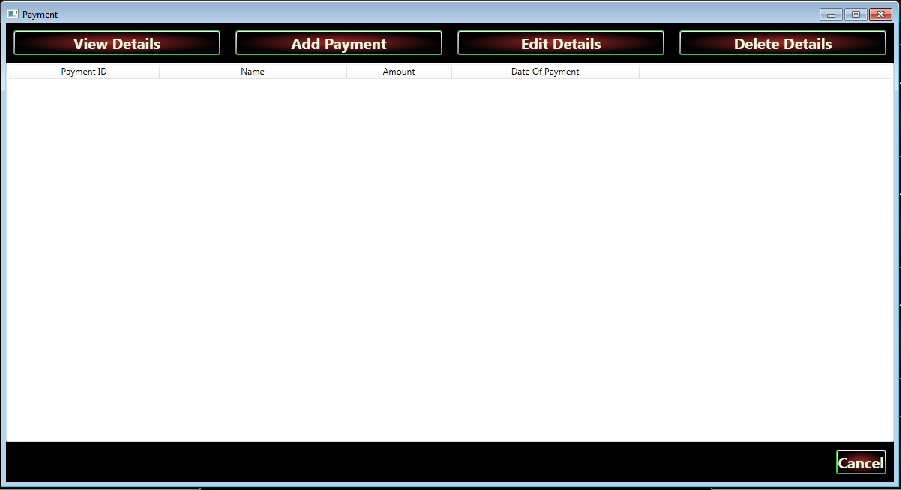
#### login window



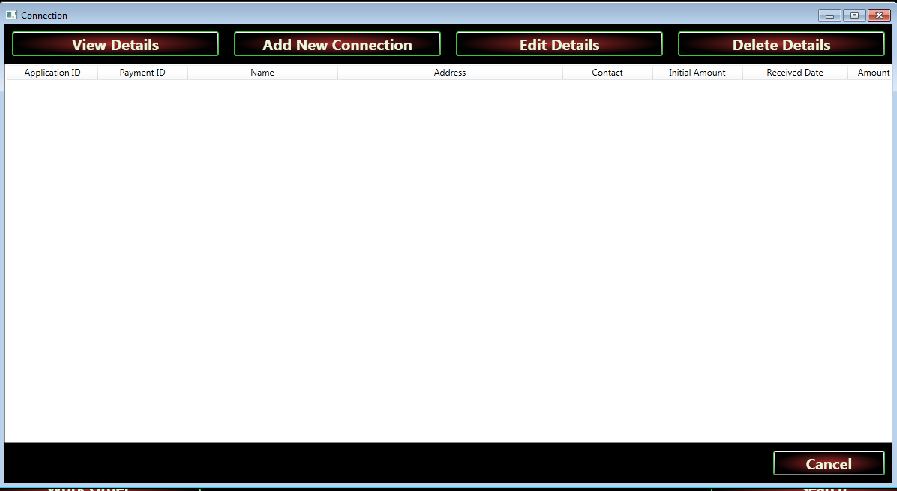
#### customer browser window



#### payment window



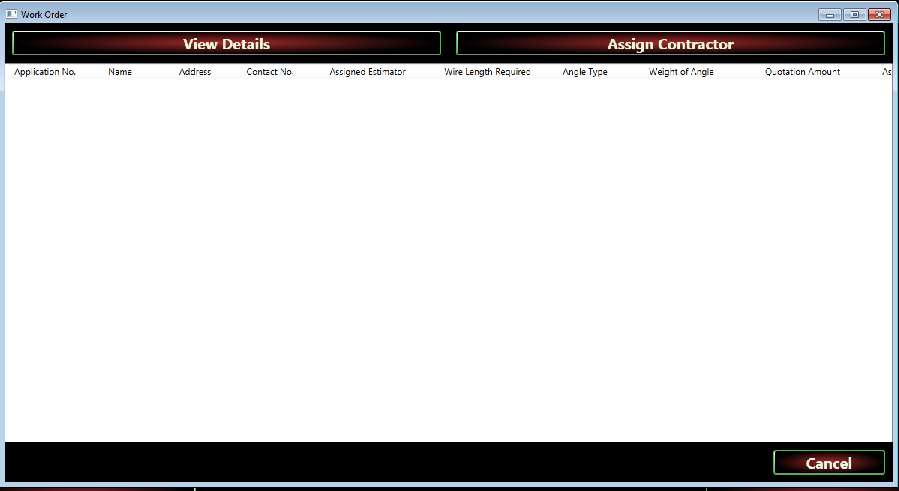
#### connection window



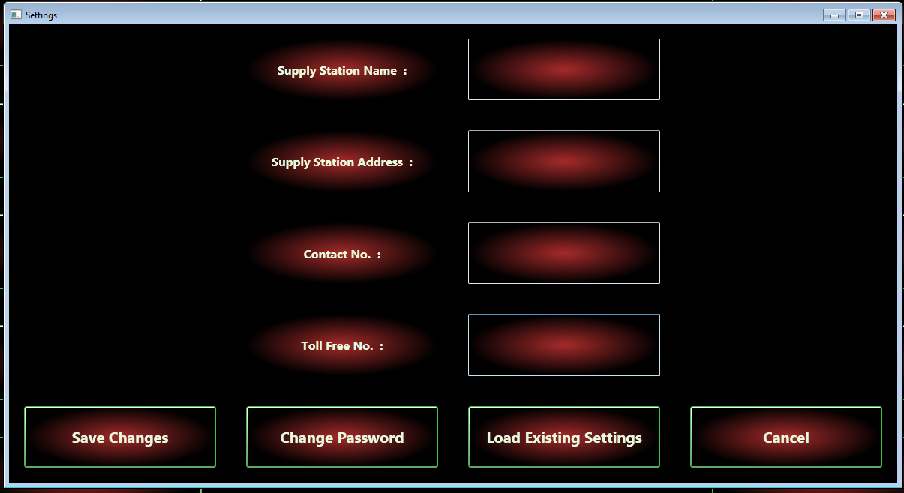
#### estimate quotation window



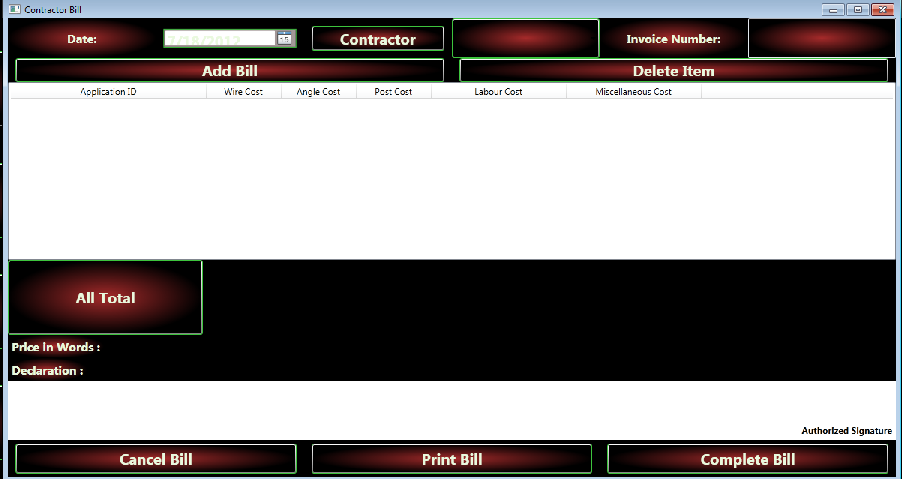
#### work order window



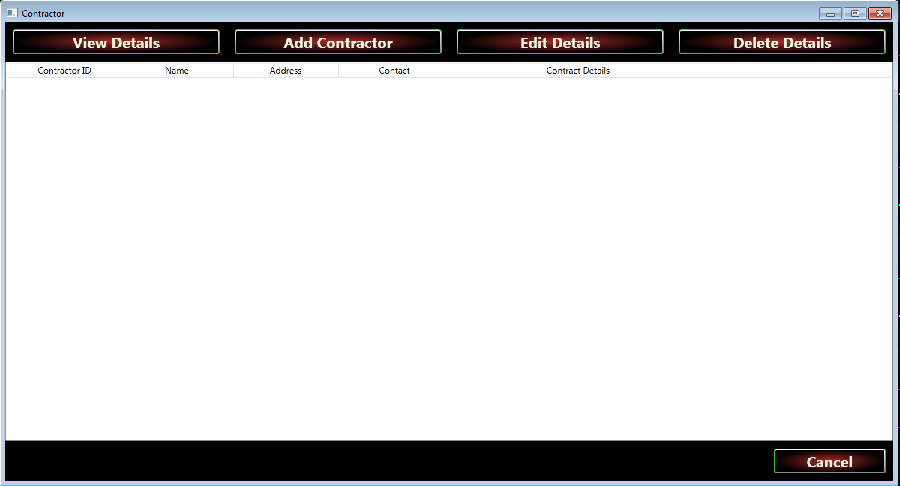
#### settings window



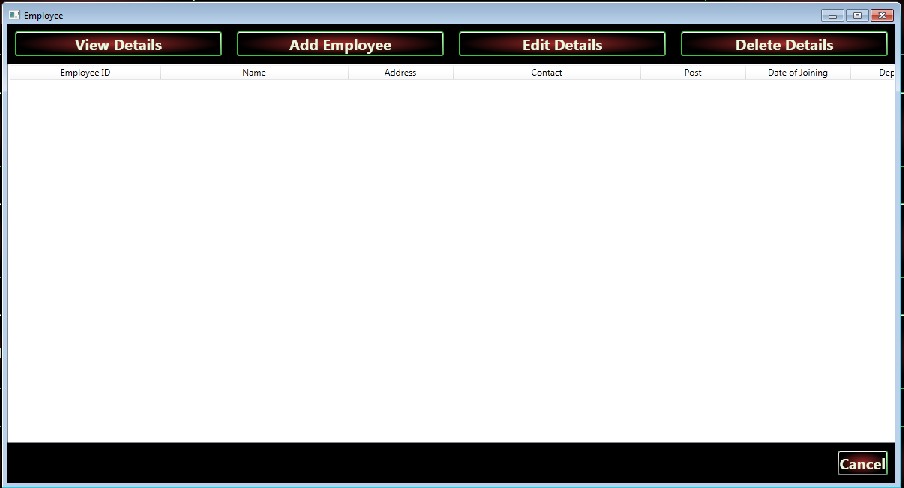
#### contractor billl window



#### contractor browser window



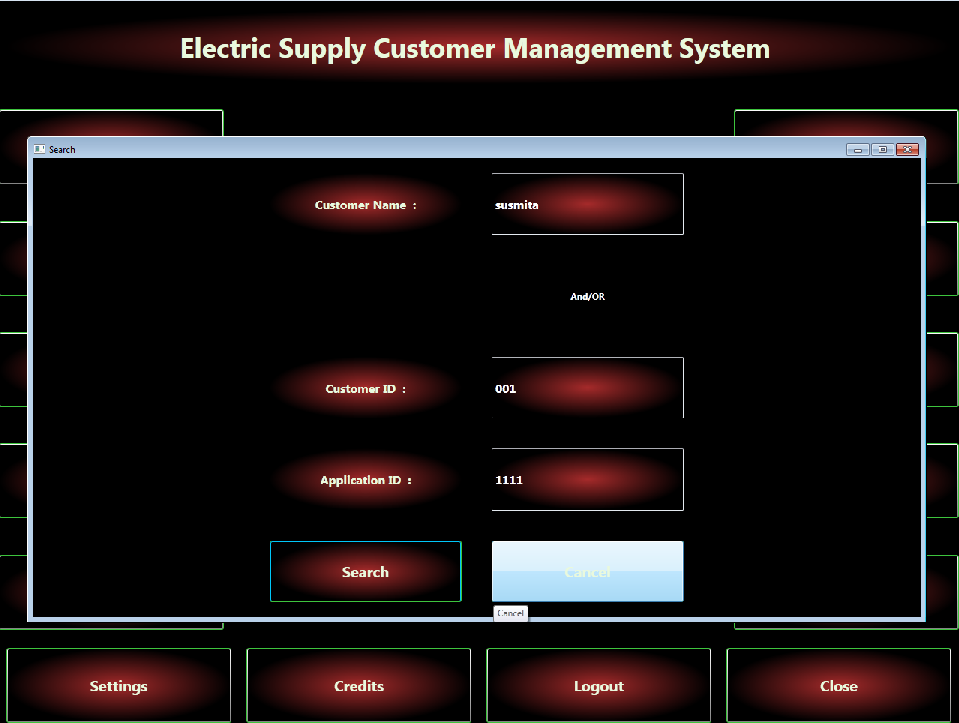
#### employeee browser window



#### report window



#### search window



### security issues

#### Database/data security:

* + This software requires a valid password to login and then it allows using any of its features.
  + The login password will be saved in encrypted format in database.
  + This software will use Google open-id authentication for web interface.
  + A backup and restore feature has been used in case of loss of data due to database crash and other problems.

#### Creation of User profiles and access rights

* A user first must create a new account to use this software.
* A predefined password will be present for a first time user and he/she must immediately change that predefined password and add his own to make his data completely secure.

### 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. |

# Coding

## Complete Project Coding

### ESCMSGUI

|  |
| --- |
| MainWindow.xaml |
| <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>  <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="Log in" Click="loginBtn\_Click" Style="{StaticResource ControlBtnStyle}"></Button>  <Button Name="closeBtn" ToolTip="Close This Application" Click="closeBtn\_Click" Style="{StaticResource ControlBtnStyle}">Exit</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> |

|  |
| --- |
| MainWindow.xaml.cs |
| 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>  publicpartialclassMainWindow : Window  {  login loginWindow;  public MainWindow()  {  InitializeComponent();  loginWindow = newlogin();  loginWindow.OnSucccesfulLogin += new ECMS.login.delegateOnSucccesfulLogin(LoginWindowObj\_OnSucccesfulLogin);  }  privatevoid loginBtn\_Click(object sender, RoutedEventArgs e)  {  if (LoggedIn == false)  {  //ECMS.login LoginWindowObj = new ECMS.login();  //LoginWindowObj.OnSucccesfulLogin += new ECMS.login.delegateOnSucccesfulLogin(LoginWindowObj\_OnSucccesfulLogin);  //LoginWindowObj.ShowDialog();  }  else  {  //customerBrowserBtn.IsEnabled = false;  //paymentBtn.IsEnabled = false;  //newConnectionBtn.IsEnabled = false;  //employeeBrowserBtn.IsEnabled = false;  //estimateBtn.IsEnabled = false;  //workOrderBtn.IsEnabled = false;  //contractorBillBtn.IsEnabled = false;  //contractorBrowserBtn.IsEnabled = false;  //reportBtn.IsEnabled = false;  //searchBtn.IsEnabled = false;  //settingsBtn.IsEnabled = false;  //helpBtn.IsEnabled = false;  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();  }  }  privatevoid paymentBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Payment PaymentObj = new ECMS.Payment();  PaymentObj.Show();  }  privatevoid closeBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid newConnectionBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Connection ConnectionObj = new ECMS.Connection();  ConnectionObj.ShowDialog();  }  privatevoid employeeBrowserBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Employee EmployeeObj = new ECMS.Employee();  EmployeeObj.Show();  }  privatevoid contractorBrowserBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Contractor ContractorObj = new ECMS.Contractor();  ContractorObj.Show();  }  privatevoid estimateBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.EstimationQuotationDisplay EstimationQuatationDisplayObj = new ECMS.EstimationQuotationDisplay();  EstimationQuatationDisplayObj.Show();  }  privatevoid customerBrowserBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Customer CustomerObj = new ECMS.Customer();  CustomerObj.Show();  }  privatevoid workOrderBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.WorkOrder WorkOrderObj = new ECMS.WorkOrder();  WorkOrderObj.Show();  }  privatevoid settingsBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Settings SettingsObj = new ECMS.Settings();  SettingsObj.Show();  }  privatevoid searchBtn\_Click\_1(object sender, RoutedEventArgs e)  {  ECMS.Search SearchObj = new ECMS.Search();  SearchObj.Show();  }  privatevoid contractorBillBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.ContractorBill ContractorBillObj = new ECMS.ContractorBill();  ContractorBillObj.Show();  }  privatevoid reportBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Report ReportObj = new ECMS.Report();  ReportObj.Show();  }  privatevoid Window\_Loaded(object sender, RoutedEventArgs e)  {  if (!LoggedIn)  {  loginWindow.ShowDialog();  }  }  privatevoid creditBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.Credits HelpObj = new ECMS.Credits();  HelpObj.Show();  }  }  } |

|  |
| --- |
| login.xaml |
| <Window x:Class="ECMS.login"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Login"  VerticalContentAlignment="Center"  FontStretch="UltraExpanded"  AllowsTransparency="True" WindowStyle="None" Background="Transparent" WindowStartupLocation="CenterScreen" Topmost="True"  >  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <Border BorderThickness="15" BorderBrush="Brown">  <DockPanel Background="Black">  <UniformGrid DockPanel.Dock="Bottom" Columns="2">  <Label Content="User Type :" Margin="20" Style="{StaticResource labelStyle}"/>  <ComboBox Name="userTypeCB" SelectedIndex="0" Style="{StaticResource CBStyle}">  <ComboBoxItem Style="{StaticResource CBItemStyle}">Admin</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Customer</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Contractor</ComboBoxItem>  </ComboBox>  <Label Content="User Name :" Margin="20" Style="{StaticResource labelStyle}"/>  <TextBox Name="userNameTxtbox" Margin="20" Style="{StaticResource textboxStyle}"></TextBox>  <Label Content="Password :" Margin="20" Style="{StaticResource labelStyle}" />  <PasswordBox Name="passwordPbox" Margin="20" Style="{StaticResource PasswordBoxStyle}"></PasswordBox>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource ControlBtnStyle}" Click="cancelBtn\_Click"></Button>  <Button Content="Login" ToolTip="Login" Name="loginBtn" Style="{StaticResource ControlBtnStyle}" Click="loginBtn\_Click"></Button>  </UniformGrid>  </DockPanel>  </Border>  </Window> |

|  |
| --- |
| login.xaml.cs |
| 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 login.xaml  ///</summary>  publicpartialclasslogin : Window  {  publicdelegatevoiddelegateOnSucccesfulLogin(bool IsSuccess);  publiceventdelegateOnSucccesfulLogin OnSucccesfulLogin;  public login()  {  InitializeComponent();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  if (OnSucccesfulLogin != null)  OnSucccesfulLogin(false);  }  privatevoid loginBtn\_Click(object sender, RoutedEventArgs e)  {  if ((userNameTxtbox.Text.Equals("1")) && (passwordPbox.Password.Equals("1")))  {  if (OnSucccesfulLogin != null)  OnSucccesfulLogin(true);  this.Hide();  }  else  MessageBox.Show("Wrong User Name or Password.");  userNameTxtbox.Text = String.Empty;  passwordPbox.Password = String.Empty;  }  }  } |

|  |
| --- |
| AssignContractor.xaml |
| <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> |

|  |
| --- |
| AssignContractor.xaml.cs |
| 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 AssignContractor.xaml  ///</summary>  publicpartialclassAssignContractor : Window  {  ObservableCollection<ContractorInfo> \_contractorCollection = newObservableCollection<ContractorInfo>();  publicObservableCollection<ContractorInfo> contractorCollection  {  get  {  return \_contractorCollection;  }  }  public AssignContractor(estimateInfo info)  {  InitializeComponent();  applicationNoTB.Text = info.appsNo;  nameTxtbox.Text = info.name;  addressTxtbox.Text = info.address;  contactNoTxtbox.Text = info.contact;  assignedEstimatorTxtbox.Text = info.estimator;  wireLengthRequiredTB.Text =Convert.ToString(info.wireLength);  angleTypeTB.Text = Convert.ToString(info.angleType);  weightofAngleCalculationTB.Text = Convert.ToString(info.angleWeight);  quotationAmountTB.Text =Convert.ToString(info.quotationAmount);  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid Window\_Loaded(object sender, RoutedEventArgs e)  {  List<ContractorInfo> contractors = ESCMSStorage.DbInteraction.GetAllContractorList();  \_contractorCollection.Clear();  foreach (ContractorInfo contractor in contractors)  {  \_contractorCollection.Add(contractor);  }  }  privatevoid submitBtn\_Click(object sender, RoutedEventArgs e)  {  estimateInfo estimateData = new ESCMSData.estimateInfo();  estimateData.appsNo = applicationNoTB.Text;  estimateData.contractor = assignedContractorCB.Text;  ESCMSStorage.DbInteraction.assignContractor(estimateData);  this.Close();  }  }  } |

|  |
| --- |
| WorkOrder.xaml |
| <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> |

|  |
| --- |
| WorkOrder.xaml.cs |
| 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 WorkOrder.xaml  ///</summary>  publicpartialclassWorkOrder : Window  {  ObservableCollection<estimateInfo> \_estimateCollection = newObservableCollection<estimateInfo>();  publicObservableCollection<estimateInfo> estimateCollection  {  get  {  return \_estimateCollection;  }  }  public WorkOrder()  {  InitializeComponent();  }  privatevoid addWorkOrderBtn\_Click(object sender, RoutedEventArgs e)  {  estimateInfo estimateToEdit = GetSelectedItem();  if (estimateToEdit != null)  {  ECMS.AssignContractor AddWorkOrderObj = new ECMS.AssignContractor(estimateToEdit);  AddWorkOrderObj.Show();  }  }  privateestimateInfo GetSelectedItem()  {  estimateInfo estimateToDelete = null;  if (workOrderView.SelectedIndex == -1)  MessageBox.Show("Please Select an Item");  else  {  estimateInfo i = (estimateInfo)workOrderView.SelectedItem;  estimateToDelete = \_estimateCollection.Where(item => item.appsNo.Equals(i.appsNo)).First();  }  return estimateToDelete;  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid viewWorkOrderBtn\_Click(object sender, RoutedEventArgs e)  {  List<estimateInfo> estimates = ESCMSStorage.DbInteraction.GetAllEstimateListWithContractor();  \_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);  }  }  }  } |

|  |
| --- |
| ChangePassword.xaml |
| <Window x:Class="ECMS.ChangePassword"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="ChangePassword" 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" Columns="4">  <Label></Label>  <Label Content="Current Password :" Margin="20" Style="{StaticResource labelStyle}"/>  <PasswordBox Name="currentPasswordTxtbox" Margin="20" Style="{StaticResource PasswordBoxStyle}"></PasswordBox>  <Label></Label>  <Label></Label>  <Label Content="New Password :" Margin="20" Style="{StaticResource labelStyle}"/>  <PasswordBox Name="newPasswordTxtbox" Margin="20" Style="{StaticResource PasswordBoxStyle}"></PasswordBox>  <Label></Label>  <Label></Label>  <Label Content="Retype New Password :" Margin="20" Style="{StaticResource labelStyle}" />  <PasswordBox Name="retypeNewPasswordTxtbox" Margin="20" Style="{StaticResource PasswordBoxStyle}"></PasswordBox>  <Label></Label>  <Label></Label>  <Button Content="Update Password" ToolTip="Update Password" Margin="20" Name="updatePasswordBtn" Style="{StaticResource smallControlBtnStyle}"></Button>  <Button Content="Cancel" ToolTip="Cancel" Margin="20" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  </UniformGrid>  </DockPanel>  </Window> |

|  |
| --- |
| ChangePassword.xaml.cs |
| 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 ChangePassword.xaml  ///</summary>  publicpartialclassChangePassword : Window  {  public ChangePassword()  {  InitializeComponent();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  }  } |

|  |
| --- |
| Settings.xaml |
| <Window x:Class="ECMS.Settings"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Settings"  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" Columns="4">  <Label></Label>  <Label Content="Supply Station Name :" Margin="20" Style="{StaticResource labelStyle}"/>  <TextBox Name="supplyStationNameTxtbox" Margin="20" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Supply Station Address :" Margin="20" Style="{StaticResource labelStyle}"/>  <TextBox Name="supplyStationAddressTxtbox" Margin="20" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Contact No. :" Margin="20" Style="{StaticResource labelStyle}" />  <TextBox Name="contactNoTxtbox" Margin="20" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Label></Label>  <Label Content="Toll Free No. :" Margin="20" Style="{StaticResource labelStyle}" />  <TextBox Name="tollFreeNoTxtbox" Margin="20" Style="{StaticResource textboxStyle}"></TextBox>  <Label></Label>  <Button Content="Save Changes" ToolTip="Save Changes" Margin="20" Name="saveChangesBtn" Style="{StaticResource smallControlBtnStyle}" Click="saveChangesBtn\_Click"></Button>  <Button Content="Change Password" ToolTip="Change Password" Margin="20" Name="changePasswordBtn" Style="{StaticResource smallControlBtnStyle}" Click="changePasswordBtn\_Click"></Button>  <Button Content="Load Existing Settings" ToolTip="Load Existing Settings" Margin="20" Name="loadExistingSettingsBtn" Style="{StaticResource smallControlBtnStyle}"></Button>  <Button Content="Cancel" ToolTip="Cancel" Margin="20" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  </UniformGrid>  </DockPanel>  </Window> |

|  |
| --- |
| Settings.xaml.cs |
| 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 Settings.xaml  ///</summary>  publicpartialclassSettings : Window  {  public Settings()  {  InitializeComponent();  }  privatevoid changePasswordBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.ChangePassword ChangePasswordObj = new ECMS.ChangePassword();  ChangePasswordObj.Show();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid saveChangesBtn\_Click(object sender, RoutedEventArgs e)  {  }  }  } |

|  |
| --- |
| Search.xaml |
| <Window x:Class="ECMS.Search"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  DataContext="{Binding RelativeSource={RelativeSource Self}}"  Title="Search 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="Bottom" Rows="1" Columns="7">  <Label HorizontalContentAlignment="Center" Style= "{StaticResource labelStyle}" >Search by Id:</Label>  <TextBox Name="searchTxtBlck" Style="{StaticResource textboxStyle}"></TextBox>  <Button Content="Search" ToolTip="Serach Customer Database" Name="searchBtn" Style="{StaticResource smallControlBtnStyle}" Click="searchBtn\_Click"></Button>  <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="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> |

|  |
| --- |
| AddPayment.xaml |
| <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> |

|  |
| --- |
| AddPayment.xaml.cs |
| 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 AddPayment.xaml  ///</summary>  publicpartialclassAddPayment : Window  {  string paymentId;  bool isEdit = false;  ObservableCollection<CustomerInfo> \_customerCollection = newObservableCollection<CustomerInfo>();  publicObservableCollection<CustomerInfo> customerCollection  {  get  {  return \_customerCollection;  }  }  public AddPayment(ESCMSData.PaymentInfo info)  {  InitializeComponent();  if (info != null)  {  isEdit = true;  nameTxtbox.Text = info.customerId;  amountTxtbox.Text = info.amount.ToString();  dopDatePicker.Text = info.dop.ToString();  paymentId = info.id;  }  else  {  paymentIdLbl.Content = GenerateId();  dopDatePicker.SelectedDate = DateTime.Now;  }  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid submitBtn\_Click(object sender, RoutedEventArgs e)  {  ESCMSData.PaymentInfo PaymentData = new ESCMSData.PaymentInfo();  PaymentData.id = paymentIdLbl.Content.ToString();  PaymentData.customerId = nameTxtbox.SelectedValue.ToString();  PaymentData.amount = Convert.ToDouble(amountTxtbox.Text);  PaymentData.dop = dopDatePicker.SelectedDate.Value;  if (isEdit == false)  {  PaymentData.id = GenerateId();  ESCMSStorage.DbInteraction.DoRegisterNewPayment(PaymentData);  }  else  {  PaymentData.id = paymentId;  ESCMSStorage.DbInteraction.EditPayment(PaymentData);  }  this.Close();  }  privatestring GenerateId()  {  returnDateTime.Now.ToOADate().ToString();  }  privatevoid addCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.AddCustomer AddCustomerObj = new ECMS.AddCustomer(null);  AddCustomerObj.Show();  }  privatevoid Window\_Loaded(object sender, RoutedEventArgs e)  {  List<CustomerInfo> customers = ESCMSStorage.DbInteraction.GetAllCustomerList();  \_customerCollection.Clear();  foreach (CustomerInfo customer in customers)  {  \_customerCollection.Add(customer);  }  }  }  } |

|  |
| --- |
| Payment.xaml |
| <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> |

|  |
| --- |
| Payment.xaml.cs |
| 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 Payment.xaml  ///</summary>  publicpartialclassPayment : Window  {  ObservableCollection<PaymentInfo> \_paymentCollection = newObservableCollection<PaymentInfo>();  publicObservableCollection<PaymentInfo> paymentCollection  {  get  {  return \_paymentCollection;  }  }  public Payment()  {  InitializeComponent();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid addPaymentBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.AddPayment AddPaymentObj = new ECMS.AddPayment(null);  AddPaymentObj.Show();  }  privatevoid viewConnectionBtn\_Click(object sender, RoutedEventArgs e)  {  List<PaymentInfo> payments = ESCMSStorage.DbInteraction.GetAllPaymentList();  \_paymentCollection.Clear();  foreach (PaymentInfo payment in payments)  {  ESCMSData.CustomerInfo cusInfo = ESCMSStorage.DbInteraction.GetCustomerInfo(payment.customerId);  payment.name = cusInfo.name;  \_paymentCollection.Add(payment);  }  }  privatevoid deleteConnectionBtn\_Click(object sender, RoutedEventArgs e)  {  PaymentInfo paymentToDelete = GetSelectedItem();  if (paymentToDelete != null)  {  \_paymentCollection.Remove(paymentToDelete);  ESCMSStorage.DbInteraction.DeletePayment(paymentToDelete.id);  }  }  privatePaymentInfo GetSelectedItem()  {  PaymentInfo paymentToDelete = null;  if (paymentView.SelectedIndex == -1)  MessageBox.Show("Please Select an Item");  else  {  PaymentInfo i = (PaymentInfo)paymentView.SelectedItem;  paymentToDelete = \_paymentCollection.Where(item => item.id.Equals(i.id)).First();  }  return paymentToDelete;  }  privatevoid editConnectionBtn\_Click(object sender, RoutedEventArgs e)  {  PaymentInfo paymentToEdit = GetSelectedItem();  if (paymentToEdit != null)  {  ECMS.AddPayment AddPaymentObj = new ECMS.AddPayment(paymentToEdit);  AddPaymentObj.Show();  }  }  }  } |

|  |
| --- |
| AddEstimator.xaml |
| <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>  <Label Content="Assigned Contractor :" Style="{StaticResource labelStyle}"/>  <ComboBox Name="assignedContractorCB" Style="{StaticResource CBStyle}" SelectedIndex="0">  <ComboBoxItem Style="{StaticResource CBItemStyle}">CommingSoon</ComboBoxItem>  </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> |

|  |
| --- |
| AddEstimator.xaml.cs |
| 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>  publicpartialclassAddEstimator : Window  {  ObservableCollection<EmployeeInfo> \_estimatorCollection = newObservableCollection<EmployeeInfo>();  publicObservableCollection<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);  }  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid Window\_Loaded(object sender, RoutedEventArgs e)  {  List<EmployeeInfo> estimators = ESCMSStorage.DbInteraction.GetAllEmployeeList();  \_estimatorCollection.Clear();  foreach (EmployeeInfo estimator in estimators)  {  \_estimatorCollection.Add(estimator);  }  }  privatevoid 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();  }  }  } |

|  |
| --- |
| AddQuotation.xaml |
| <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> |

|  |
| --- |
| AddQuotation.xaml.cs |
| 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>  publicpartialclassAddQuotation : Window  {  string estimateQuotationId;  bool isEdit = false;  ObservableCollection<EmployeeInfo> \_estimatorCollection = newObservableCollection<EmployeeInfo>();  publicObservableCollection<EmployeeInfo> estimatorCollection  {  get  {  return \_estimatorCollection;  }  }  ObservableCollection<NewConnectionInfo> \_newConnectionCollection = newObservableCollection<NewConnectionInfo>();  publicObservableCollection<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);  }  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid 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();  }  privatevoid Window\_Loaded(object sender, RoutedEventArgs e)  {  List<EmployeeInfo> estimators = ESCMSStorage.DbInteraction.GetAllEmployeeList();  \_estimatorCollection.Clear();  foreach (EmployeeInfo estimator in estimators)  {  \_estimatorCollection.Add(estimator);  }  }  }  } |

|  |
| --- |
| EstimationQuotationDisplay.xaml |
| <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> |

|  |
| --- |
| EstimationQuotationDisplay.xaml.cs |
| 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>  publicpartialclassEstimationQuotationDisplay : Window  {  ObservableCollection<estimateInfo> \_estimateCollection = newObservableCollection<estimateInfo>();  publicObservableCollection<estimateInfo> estimateCollection  {  get  {  return \_estimateCollection;  }  }  public EstimationQuotationDisplay()  {  InitializeComponent();  }  privatevoid addPaymentBtn\_Click(object sender, RoutedEventArgs e)  {  estimateInfo estimateToEdit = GetSelectedItem();  if (estimateToEdit != null)  {  ECMS.AddQuotation EstimateQuotationObj = new ECMS.AddQuotation(estimateToEdit);  EstimateQuotationObj.Show();  }  }  publicestimateInfo 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;  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid 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);  }  }  privatevoid editBtn\_Click(object sender, RoutedEventArgs e)  {  estimateInfo estimateToEdit = GetSelectedItem();  if (estimateToEdit != null)  {  // ECMS.AddQuotation EstimateQuotationObj = new ECMS.AddQuotation(estimateToEdit);  ECMS.AddEstimator AddEstimatorObj = new ECMS.AddEstimator(estimateToEdit);  AddEstimatorObj.Show();  }  }  }  } |

|  |
| --- |
| Employee.xaml |
| <Window x:Class="ECMS.Employee"  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="Employee"  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 LastChildFill="True">  <Button DockPanel.Dock="Bottom" HorizontalAlignment="Right" Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  <UniformGrid DockPanel.Dock="top" Rows="1">  <Label HorizontalContentAlignment="Center" Style="{StaticResource labelStyle}">Search Employee:</Label>  <TextBox Name="searchTxtBlck" Style="{StaticResource textboxStyle}"></TextBox>  <Button Content="Search" ToolTip="Serach Employee Database" Name="searchBtn" Style="{StaticResource smallControlBtnStyle}" Click="searchBtn\_Click"></Button>  <Button Content="View Info" ToolTip="View or Refresh" Name="viewEmployeeBtn" Style="{StaticResource smallControlBtnStyle}" Click="viewEmployeeBtn\_Click"></Button>  <Button Content="Add Info" ToolTip="Click To Add New Employee" Name="addEmployeeBtn" Style="{StaticResource smallControlBtnStyle}"Click="addEmployeeBtn\_Click"></Button>  <Button Content="Edit Info" ToolTip="Edit Details" Name="editEmployeeBtn" Style="{StaticResource smallControlBtnStyle}" Click="editEmployeeBtn\_Click"></Button>  <Button Content="Delete Info" ToolTip="Delete Selected Details" Name="deleteEmployeeBtn" Style="{StaticResource smallControlBtnStyle}" Click="deleteEmployeeBtn\_Click" Width="159"></Button>  </UniformGrid>  <ListView DockPanel.Dock="top" Name="employeeView" ItemsSource="{Binding employeeCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="200" Header="Employee ID" DisplayMemberBinding="{Binding id}" />  <GridViewColumn Width="250" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="140" Header="Address" DisplayMemberBinding="{Binding address}" />  <GridViewColumn Width="250" Header="Contact" DisplayMemberBinding="{Binding contact}" />  <GridViewColumn Width="140" Header="Post" DisplayMemberBinding="{Binding postType}" />  <GridViewColumn Width="140" Header="Date of Joining" DisplayMemberBinding="{Binding doj}" />  <GridViewColumn Width="140" Header="Department" DisplayMemberBinding="{Binding department}" />  </GridView>  </ListView.View>  </ListView>  </DockPanel>  </Window> |

|  |
| --- |
| Employee.xaml.cs |
| 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 Employee.xaml  ///</summary>  publicpartialclassEmployee : Window  {  ObservableCollection<EmployeeInfo> \_employeeCollection = newObservableCollection<EmployeeInfo>();  publicObservableCollection<EmployeeInfo> employeeCollection  {  get  {  return \_employeeCollection;  }  }  public Employee()  {  InitializeComponent();  }  privatevoid addEmployeeBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.AddEmployee AddEmployeeObj = new ECMS.AddEmployee(null);  AddEmployeeObj.Show();  }  privatevoid viewEmployeeBtn\_Click(object sender, RoutedEventArgs e)  {  List<EmployeeInfo> employees = ESCMSStorage.DbInteraction.GetAllEmployeeList();  \_employeeCollection.Clear();  foreach (EmployeeInfo employee in employees)  {  \_employeeCollection.Add(employee);  }  }  privatevoid deleteEmployeeBtn\_Click(object sender, RoutedEventArgs e)  {  EmployeeInfo employeeToDelete = GetSelectedItem();  if (employeeToDelete != null)  {  \_employeeCollection.Remove(employeeToDelete);  ESCMSStorage.DbInteraction.DeleteEmployee(employeeToDelete.id);  }  }  privateEmployeeInfo GetSelectedItem()  {  EmployeeInfo employeeToDelete = null;  if (employeeView.SelectedIndex == -1)  MessageBox.Show("Please Select an Item");  else  {  EmployeeInfo i = (EmployeeInfo)employeeView.SelectedItem;  employeeToDelete = \_employeeCollection.Where(item => item.id.Equals(i.id)).First();  }  return employeeToDelete;  }  privatevoid editEmployeeBtn\_Click(object sender, RoutedEventArgs e)  {  EmployeeInfo employeeToEdit = GetSelectedItem();  if (employeeToEdit != null)  {  ECMS.AddEmployee AddEmployeeObj = new ECMS.AddEmployee(employeeToEdit);  AddEmployeeObj.Show();  }  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid searchBtn\_Click(object sender, RoutedEventArgs e)  {  EmployeeInfo empInfoObj = newEmployeeInfo();  empInfoObj.name = searchTxtBlck.Text;  List<EmployeeInfo> employees = ESCMSStorage.DbInteraction.SearchAllEmployeeList(empInfoObj);  \_employeeCollection.Clear();  foreach (EmployeeInfo employee in employees)  {  \_employeeCollection.Add(employee);  }  }  }  } |

|  |
| --- |
| AddEmployee.xaml |
| <Window x:Class="ECMS.AddEmployee"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Add Employee" 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="Employee ID :" Style="{StaticResource labelStyle}"/>  <Label Content="" Name="employeeLbl" Style="{StaticResource labelStyle}"></Label>  <Label></Label>  <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>  <Label Content="Post Type :" Style="{StaticResource labelStyle}"/>  <ComboBox Name="typeComboBox" SelectedIndex="0" Style="{StaticResource CBStyle}">  <ComboBoxItem Style="{StaticResource CBItemStyle}">DataEnterer</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Officer</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Admin</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Estimator</ComboBoxItem>  </ComboBox>  <Label></Label>  <Label></Label>  <Label Content="Date of joining :" Style="{StaticResource labelStyle}"/>  <DatePicker Name="dateOfjoiningDateDatePicker" Style="{StaticResource DPStyle}"></DatePicker>  <Label></Label>  <Label></Label>  <Label Content="Department :" Style="{StaticResource labelStyle}"/>  <ComboBox Name="depertmentCB" SelectedIndex="2" Style="{StaticResource CBStyle}">  <ComboBoxItem Style="{StaticResource CBItemStyle}">Customer Care</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Billing</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Accountant</ComboBoxItem>  <ComboBoxItem Style="{StaticResource CBItemStyle}">Other</ComboBoxItem>  </ComboBox>  <Label></Label>  <Label></Label>  <Button Content="Cancel" ToolTip="Cancel" Name="cancelBtn" Style="{StaticResource smallControlBtnStyle}" Click="cancleBtn\_Click"></Button>  <Button Content="Submit" ToolTip="Submit" Name="submitBtn" Style="{StaticResource smallControlBtnStyle}" Click="submitBtn\_Click"></Button>  <Label></Label>  </UniformGrid>  </DockPanel>  </Window> |

|  |
| --- |
| AddEmployee.xaml.cs |
| 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;  namespace ECMS  {  ///<summary>  /// Interaction logic for Employee.xaml  ///</summary>  publicpartialclassAddEmployee : Window  {  string employeeId;  bool isEdit = false;  public AddEmployee(ESCMSData.EmployeeInfo info)  {  InitializeComponent();  if (info != null)  {  isEdit = true;  nameTxtbox.Text = info.name;  addressTxtbox.Text = info.address;  contactNoTxtbox.Text = info.contact;  typeComboBox.Text = info.postType.ToString();  dateOfjoiningDateDatePicker.Text = info.doj.ToString();  depertmentCB.Text = info.department;  employeeId = info.id;  }  }  privatevoid submitBtn\_Click(object sender, RoutedEventArgs e)  {  ESCMSData.EmployeeInfo newEmployee = new ESCMSData.EmployeeInfo();  newEmployee.id = GenerateId();  newEmployee.name = nameTxtbox.Text;  newEmployee.address = addressTxtbox.Text;  newEmployee.contact = contactNoTxtbox.Text;  newEmployee.postType = (PostType)Enum.Parse(typeof(PostType), typeComboBox.Text, true);  newEmployee.doj = dateOfjoiningDateDatePicker.SelectedDate.Value;  newEmployee.department = depertmentCB.Text;  if (isEdit == false)  {  newEmployee.id = GenerateId();  ESCMSStorage.DbInteraction.DoRegisterNewEmployee(newEmployee);  }  else  {  newEmployee.id = employeeId;  ESCMSStorage.DbInteraction.EditEmployee(newEmployee);  }  this.Close();  }  privatestring GenerateId()  {  returnDateTime.Now.ToOADate().ToString();  }  privatevoid cancleBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  }  } |

|  |
| --- |
| AddCustomer.xaml |
| <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> |

|  |
| --- |
| AddCustomer.xaml.cs |
| 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>  publicpartialclassAddCustomer : 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;  }  }  privatevoid OK\_Btn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid 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();  }  privatestring GenerateId()  {  returnDateTime.Now.ToOADate().ToString();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  }  } |

|  |
| --- |
| Customer.xaml |
| <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 HorizontalContentAlignment="Center" Style= "{StaticResource labelStyle}" >Search Customer:</Label>  <TextBox Name="searchTxtBlck" Style="{StaticResource textboxStyle}"></TextBox>  <Button Content="Search" ToolTip="Serach Customer Database" Name="searchBtn" Style="{StaticResource smallControlBtnStyle}" Click="searchBtn\_Click"></Button>  <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> |

|  |
| --- |
| Customer.xaml.cs |
| 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>  publicpartialclassCustomer : Window  {  ObservableCollection<CustomerInfo> \_customerCollection = newObservableCollection<CustomerInfo>();  publicObservableCollection<CustomerInfo> customerCollection  {  get  {  return \_customerCollection;  }  }  public Customer()  {  InitializeComponent();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid addCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.AddCustomer AddCustomerObj = new ECMS.AddCustomer(null);  AddCustomerObj.Show();  }  privatevoid viewCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  List<CustomerInfo> customers = ESCMSStorage.DbInteraction.GetAllCustomerList();  \_customerCollection.Clear();  foreach (CustomerInfo customer in customers)  {  \_customerCollection.Add(customer);  }  }  privatevoid deleteCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  CustomerInfo customerToDelete = GetSelectedItem();  if (customerToDelete != null)  {  \_customerCollection.Remove(customerToDelete);  ESCMSStorage.DbInteraction.DeleteCustomer(customerToDelete.id);  }  }  privateCustomerInfo 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;  }  privatevoid editCustomerBtn\_Click(object sender, RoutedEventArgs e)  {  CustomerInfo customerToEdit = GetSelectedItem();  if (customerToEdit != null)  {  ECMS.AddCustomer AddCustomerObj = new ECMS.AddCustomer(customerToEdit);  AddCustomerObj.Show();  }  }  privatevoid searchBtn\_Click(object sender, RoutedEventArgs e)  {  CustomerInfo custinfo = newCustomerInfo();  custinfo.name = searchTxtBlck.Text;  List<CustomerInfo> customers = ESCMSStorage.DbInteraction.searchCustomerList(custinfo);  \_customerCollection.Clear();  foreach (CustomerInfo customer in customers)  {  \_customerCollection.Add(customer);  }  }  }  } |

|  |
| --- |
| AddBill.xaml |
| <Window x:Class="ECMS.AddBill"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Add Bill"  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="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="Contractor :" Style="{StaticResource labelStyle}"/>  <TextBox Name="contractorTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <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>  <Label Content="Total Amount :" Style="{StaticResource labelStyle}"/>  <TextBox Name="totalAmountTB" 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}"></Button>  <Label></Label>  </UniformGrid>  </DockPanel>  </Window> |

|  |
| --- |
| AddBill.xaml.cs |
| 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 AddBill.xaml  ///</summary>  publicpartialclassAddBill : Window  {  public AddBill()  {  InitializeComponent();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  }  } |

|  |
| --- |
| ContractorBill.xaml |
| <Window x:Class="ECMS.ContractorBill"  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="Contractor Bill" Background="Black" Height="650" Width="1200">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/EscmsStyles;component/ControlStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <UniformGrid Height="50" DockPanel.Dock="Bottom" Rows="1">  <Button Margin="10 5" ToolTip="Cancel Bill" Name="cancelBillBtn" KeyboardNavigation.TabIndex="8" Style="{StaticResource smallControlBtnStyle}">Cancel Bill</Button>  <Button Margin="10 5" ToolTip="Print Bill" Name="printBillBtn" KeyboardNavigation.TabIndex="8" Style="{StaticResource smallControlBtnStyle}">Print Bill</Button>  <Button Margin="10 5" ToolTip="Complete Bill" Name="completeBillBtn" KeyboardNavigation.TabIndex="7" Style="{StaticResource smallControlBtnStyle}" Click="completeBillBtn\_Click">Complete Bill</Button>  </UniformGrid>  <DockPanel Name="printableBillAreaDockPanel" DockPanel.Dock="Top" LastChildFill="True">  <UniformGrid DockPanel.Dock="Top" Rows="1">  <Label Content="Date:" Style="{StaticResource labelStyle}"></Label>  <DatePicker Height="25" Name="purchaseDatePicker" Background="#FFA39797" SelectedDate="{x:Static sys:DateTime.Now}" Style="{StaticResource DPStyle}" />  <Button Name="contractorBtn" ToolTip="Contractor" Style="{StaticResource smallControlBtnStyle}">Contractor</Button>  <ComboBox Name="contractorCB" Style="{StaticResource CBStyleForName}"></ComboBox>  <Label Content="Invoice Number:" Style="{StaticResource labelStyle}"></Label>  <TextBox Name="invoiceNumberTB" Style="{StaticResource textboxStyle}" ></TextBox>  </UniformGrid>  <UniformGrid DockPanel.Dock="Top" Rows="1">  <Button Content="Add Bill" ToolTip="Click To Add Bill" Name="addItemBtn" KeyboardNavigation.TabIndex="2" Margin="10 0" Style="{StaticResource smallControlBtnStyle}"></Button>  <Button Content="Delete Item" ToolTip="Delete Selected Details" Margin="10 0" Name="deleteItemBtn" Style="{StaticResource smallControlBtnStyle}"></Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="Bottom" Rows="3">  <Label Background="White"></Label>  <Label Background="White"></Label>  <Label HorizontalContentAlignment="Right" FontWeight="Bold" Background="White">Authorized Signature</Label>  </UniformGrid>  <DockPanel DockPanel.Dock="Bottom" LastChildFill="True">  <Label FontWeight="Bold" Style="{StaticResource labelStyle}">Declaration :</Label>  <Label Name="declarationLabel"></Label>  </DockPanel>  <DockPanel DockPanel.Dock="Bottom" LastChildFill="True">  <Label FontWeight="Bold" Style="{StaticResource labelStyle}">Price in Words :</Label>  <Label Name="priceInWordLabel"></Label>  </DockPanel>  <DockPanel DockPanel.Dock="Bottom">  <Button DockPanel.Dock="Left" ToolTip="Click To Add Total" Name="salesCalculateTotalBtn" KeyboardNavigation.TabIndex="4"HorizontalContentAlignment="Center" Content="All Total" Width="260" FontWeight="Bold" Style="{StaticResource ControlBtnStyle}"></Button>  <Label DockPanel.Dock="Left" Name="salesTotalNoOfItems" Width="100"></Label>  <Label DockPanel.Dock="Left" Name="itemUnit" Width="300" HorizontalAlignment="Left" ></Label>  <Label DockPanel.Dock="Left" Name="salesTotalAmountLabel" Width="100" HorizontalContentAlignment="Right"></Label>  </DockPanel>  <ListView Name="salesBillingItemListView" DockPanel.Dock="Bottom" ItemsSource="{Binding purchaseBillingCollection}">  <ListView.ItemContainerStyle>  <Style TargetType="ListViewItem">  <Setter Property="HorizontalContentAlignment" Value="Right" />  </Style>  </ListView.ItemContainerStyle>  <ListView.View>  <GridView>  <GridViewColumn Width="260" Header="Application ID" DisplayMemberBinding="{Binding productName}" />  <GridViewColumn Width="100" Header="Wire Cost" DisplayMemberBinding="{Binding quantity}" />  <GridViewColumn Width="100" Header="Angle Cost" DisplayMemberBinding="{Binding vat}" />  <GridViewColumn Width="100" Header="Post Cost" DisplayMemberBinding="{Binding rate}" />  <GridViewColumn Width="180" Header="Labour Cost" DisplayMemberBinding="{Binding amount}" />  <GridViewColumn Width="180" Header="Miscellaneous Cost" DisplayMemberBinding="{Binding amount}" />  </GridView>  </ListView.View>  </ListView>  </DockPanel>  </DockPanel>  </Window> |

|  |
| --- |
| ContractorBill.xaml.cs |
| 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 ContractorBill.xaml  ///</summary>  publicpartialclassContractorBill : Window  {  public ContractorBill()  {  InitializeComponent();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid addBillBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.AddBill AddBillObj = new ECMS.AddBill();  AddBillObj.Show();  }  privatevoid completeBillBtn\_Click(object sender, RoutedEventArgs e)  {  }  }  } |

|  |
| --- |
| AddContractor.xaml |
| <Window x:Class="ECMS.AddContractor"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Add Contractor" 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="Contractor ID :" Style="{StaticResource labelStyle}"/>  <Label Content="" Name="contractorLbl" Style="{StaticResource labelStyle}"></Label>  <Label></Label>  <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>  <Label Content="Contract Details :" Style="{StaticResource labelStyle}"/>  <TextBox Name="contractDetailsTxtbox" 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> |

|  |
| --- |
| AddContractor.xaml.cs |
| 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 Contractor.xaml  ///</summary>  publicpartialclassAddContractor : Window  {  string contractorId;  bool isEdit = false;  public AddContractor(ESCMSData.ContractorInfo info)  {  InitializeComponent();  if (info != null)  {  isEdit = true;  nameTxtbox.Text = info.name;  addressTxtbox.Text = info.address;  contactNoTxtbox.Text = info.contact;  contractDetailsTxtbox.Text = info.details;  contractorId = info.id;  }  }  privatevoid submitBtn\_Click(object sender, RoutedEventArgs e)  {  ESCMSData.ContractorInfo newContractor = new ESCMSData.ContractorInfo();  newContractor.id = GenerateId();  newContractor.name = nameTxtbox.Text;  newContractor.address = addressTxtbox.Text;  newContractor.contact = contactNoTxtbox.Text;  newContractor.details = contractDetailsTxtbox.Text;  if (isEdit == false)  {  newContractor.id = GenerateId();  ESCMSStorage.DbInteraction.DoRegisterNewContractor(newContractor);  }  else  {  newContractor.id = contractorId;  ESCMSStorage.DbInteraction.EditContractor(newContractor);  }  this.Close();  }  privatestring GenerateId()  {  returnDateTime.Now.ToOADate().ToString();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  }  } |

|  |
| --- |
| Contractor.xaml |
| <Window x:Class="ECMS.Contractor"  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="Contractor"  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="viewContractorBtn" Style="{StaticResource smallControlBtnStyle}" Click="viewContractorBtn\_Click"></Button>  <Button Content="Add Contractor" ToolTip="Click To Add New Contractor" Name="addContractorBtn" Style="{StaticResource smallControlBtnStyle}" Click="addContractorBtn\_Click"></Button>  <Button Content="Edit Details" ToolTip="Edit Details" Name="editContractorBtn" Style="{StaticResource smallControlBtnStyle}" Click="editContractorBtn\_Click"></Button>  <Button Content="Delete Details" ToolTip="Delete Selected Details" Name="deleteContractorBtn" Style="{StaticResource smallControlBtnStyle}" Click="deleteContractorBtn\_Click"></Button>  </UniformGrid>  <UniformGrid DockPanel.Dock="Bottom" Columns="7">  <Label HorizontalContentAlignment="Center" Style="{StaticResource labelStyle}">Search Contractor:</Label>  <TextBox Name="searchTxtBlck" Style="{StaticResource textboxStyle}"></TextBox>  <Button Content="Search" ToolTip="Serach Contractor Database" Name="searchBtn" Style="{StaticResource smallControlBtnStyle}" Click="searchBtn\_Click"></Button>  <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="contractorView" ItemsSource="{Binding contractorCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="150" Header="Contractor ID" DisplayMemberBinding="{Binding id}" />  <GridViewColumn Width="150" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="140" Header="Address" DisplayMemberBinding="{Binding address}" />  <GridViewColumn Width="150" Header="Contact" DisplayMemberBinding="{Binding contact}" />  <GridViewColumn Width="340" Header="Contract Details" DisplayMemberBinding="{Binding details}" />  </GridView>  </ListView.View>  </ListView>  </UniformGrid>  </DockPanel>  </Window> |

|  |
| --- |
| Contractor.xaml.cs |
| 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 ContractorBrowser.xaml  ///</summary>  publicpartialclassContractor : Window  {  ObservableCollection<ContractorInfo> \_contractorCollection = newObservableCollection<ContractorInfo>();  publicObservableCollection<ContractorInfo> contractorCollection  {  get  {  return \_contractorCollection;  }  }  public Contractor()  {  InitializeComponent();  }  privatevoid addContractorBtn\_Click(object sender, RoutedEventArgs e)  {  ECMS.AddContractor AddContractorObj = new ECMS.AddContractor(null);  AddContractorObj.Show();  }  privatevoid viewContractorBtn\_Click(object sender, RoutedEventArgs e)  {  List<ContractorInfo> contractors = ESCMSStorage.DbInteraction.GetAllContractorList();  \_contractorCollection.Clear();  foreach (ContractorInfo contractor in contractors)  {  \_contractorCollection.Add(contractor);  }  }  privatevoid deleteContractorBtn\_Click(object sender, RoutedEventArgs e)  {  ContractorInfo contractorToDelete = GetSelectedItem();  if (contractorToDelete != null)  {  \_contractorCollection.Remove(contractorToDelete);  ESCMSStorage.DbInteraction.DeleteContractor(contractorToDelete.id);  }  }  privateContractorInfo GetSelectedItem()  {  ContractorInfo contractorToDelete = null;  if (contractorView.SelectedIndex == -1)  MessageBox.Show("Please Select an Item");  else  {  ContractorInfo i = (ContractorInfo)contractorView.SelectedItem;  contractorToDelete = \_contractorCollection.Where(item => item.id.Equals(i.id)).First();  }  return contractorToDelete;  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatevoid editContractorBtn\_Click(object sender, RoutedEventArgs e)  {  ContractorInfo contractorToEdit = GetSelectedItem();  if (contractorToEdit != null)  {  ECMS.AddContractor AddContractorObj = new ECMS.AddContractor(contractorToEdit);  AddContractorObj.Show();  }  }  privatevoid searchBtn\_Click(object sender, RoutedEventArgs e)  {  ContractorInfo conInfoObj = newContractorInfo();  conInfoObj.name = searchTxtBlck.Text;  List<ContractorInfo> contractors = ESCMSStorage.DbInteraction.SearchContractorList(conInfoObj);  \_contractorCollection.Clear();  foreach (ContractorInfo contractor in contractors)  {  \_contractorCollection.Add(contractor);  }  }  }  } |

|  |
| --- |
| Connection.xaml |
| <Window x:Class="ECMS.Connection"  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="Connection"  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 New Connection" ToolTip="Click To Add New Connection" Name="addNewConnectionBtn" Style="{StaticResource smallControlBtnStyle}" Click="addNewConnectionBtn\_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>  <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="newConnectionView" ItemsSource="{Binding newConnectionCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="120" Header="Application ID" DisplayMemberBinding="{Binding appsNo}" />  <GridViewColumn Width="120" Header="Payment ID" DisplayMemberBinding="{Binding paymentId}" />  <GridViewColumn Width="200" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="300" Header="Address" DisplayMemberBinding="{Binding address}" />  <GridViewColumn Width="120" Header="Contact" DisplayMemberBinding="{Binding phone}" />  <GridViewColumn Width="120" Header="Initial Amount" DisplayMemberBinding="{Binding initialAmount}" />  <GridViewColumn Width="140" Header="Received Date" DisplayMemberBinding="{Binding receivedDate}" />  <GridViewColumn Width="140" Header="Amount Received On" DisplayMemberBinding="{Binding amountReceivedOn}" />  </GridView>  </ListView.View>  </ListView>  </UniformGrid>  </DockPanel>  </Window> |

|  |
| --- |
| Connection.xaml.cs |
| <Window x:Class="ECMS.Connection"  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="Connection"  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 New Connection" ToolTip="Click To Add New Connection" Name="addNewConnectionBtn" Style="{StaticResource smallControlBtnStyle}" Click="addNewConnectionBtn\_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>  <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="newConnectionView" ItemsSource="{Binding newConnectionCollection}">  <ListView.View>  <GridView>  <GridViewColumn Width="120" Header="Application ID" DisplayMemberBinding="{Binding appsNo}" />  <GridViewColumn Width="120" Header="Payment ID" DisplayMemberBinding="{Binding paymentId}" />  <GridViewColumn Width="200" Header="Name" DisplayMemberBinding="{Binding name}" />  <GridViewColumn Width="300" Header="Address" DisplayMemberBinding="{Binding address}" />  <GridViewColumn Width="120" Header="Contact" DisplayMemberBinding="{Binding phone}" />  <GridViewColumn Width="120" Header="Initial Amount" DisplayMemberBinding="{Binding initialAmount}" />  <GridViewColumn Width="140" Header="Received Date" DisplayMemberBinding="{Binding receivedDate}" />  <GridViewColumn Width="140" Header="Amount Received On" DisplayMemberBinding="{Binding amountReceivedOn}" />  </GridView>  </ListView.View>  </ListView>  </UniformGrid>  </DockPanel>  </Window> |

|  |
| --- |
| newapplication.xaml |
| <Window x:Class="ECMS.newapplication"  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="New Connection"  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="Payment ID" Style="{StaticResource labelStyle}"/>  <ComboBox Name="paymentIDTxtbox" SelectionChanged="paymentIDTxtbox\_SelectionChanged" Style="{StaticResource CBStyle}" ItemsSource="{Binding paymentInfoCollection}" SelectedValuePath="id" DisplayMemberPath="id"></ComboBox>  <Label></Label>  <Label></Label>  <Label Content="Name :" Style="{StaticResource labelStyle}"/>  <Label Name="nameTxtbox" Style="{StaticResource labelStyle}"></Label>  <Label></Label>  <Label></Label>  <Label Content="Customer Id :" Style="{StaticResource labelStyle}"/>  <Label Name="customerIdlabel" Style="{StaticResource labelStyle}"></Label>  <Label></Label>  <Label></Label>  <Label Content="Address :" Style="{StaticResource labelStyle}" />  <Label Name="addressTxtbox" Style="{StaticResource labelStyle}"/>  <Label></Label>  <Label></Label>  <Label Content="Contact No. :" Style="{StaticResource labelStyle}"/>  <Label Name="contactNoTxtbox" Style="{StaticResource labelStyle}"/>  <Label></Label>  <Label></Label>  <Label Content="Initial Deposit Amount :" Style="{StaticResource labelStyle}"/>  <Label Name="initialDepositAmountTxtbox" Style="{StaticResource labelStyle}"/>  <Label></Label>  <Label></Label>  <Label Content="Amount Received On :" Style="{StaticResource labelStyle}"/>  <DatePicker Name="amountReceivedOnDatePicker" IsEnabled="False" VerticalContentAlignment="Center"></DatePicker>  <Label></Label>  <Label></Label>  <Label Content="Application Received Date :" Style="{StaticResource labelStyle}"/>  <DatePicker Name="applicationReceivedDateDatePicker" VerticalContentAlignment="Center"></DatePicker>  <Label></Label>  <Label></Label>  <Button Content="Cancel" Name="cancelBtn" ToolTip="Cancel" Style="{StaticResource smallControlBtnStyle}" Click="cancelBtn\_Click"></Button>  <Button Content="Submit" Name="submitBtn" ToolTip="Submit" Click="submitBtn\_Click" Style="{StaticResource smallControlBtnStyle}"></Button>  <Label></Label>  </UniformGrid>  </DockPanel>  </Window> |

|  |
| --- |
| newapplication.xaml.cs |
| 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 newapplication.xaml  ///</summary>  publicpartialclassnewapplication : Window  {  string newConnectionId;  bool isEdit = false;  ObservableCollection<PaymentInfo> \_paymentInfoCollection = newObservableCollection<PaymentInfo>();  publicObservableCollection<PaymentInfo> paymentInfoCollection  {  get  {  return \_paymentInfoCollection;  }  }  public newapplication(ESCMSData.NewConnectionInfo info)  {  InitializeComponent();  string idPay = PopulatePaymentComboboxItems();  if (info != null)  {  isEdit = true;  paymentIDTxtbox.Text = info.paymentId;  newConnectionId = info.appsNo;  paymentIDTxtbox.Text = info.paymentId;  applicationReceivedDateDatePicker.SelectedDate = info.receivedDate;  PopulateItemsFromPaymentId(info.paymentId);  }  else  {  applicationReceivedDateDatePicker.SelectedDate = DateTime.Now;  paymentIDTxtbox.Text = idPay;  PopulateItemsFromPaymentId(idPay);  }  }  privatevoid submitBtn\_Click(object sender, RoutedEventArgs e)  {  NewConnectionInfo newConnection = new ESCMSData.NewConnectionInfo();  newConnection.paymentId = paymentIDTxtbox.Text;  newConnection.customerId = customerIdlabel.Content.ToString();  newConnection.receivedDate = applicationReceivedDateDatePicker.SelectedDate.Value;  if (isEdit == false)  {  newConnection.appsNo = GenerateId();  ESCMSStorage.DbInteraction.DoRegisterNewNewConnection(newConnection);  estimateInfo esInfo = newestimateInfo();  esInfo.appsNo = newConnection.appsNo;  ESCMSStorage.DbInteraction.DoRegisterNewEstimate(esInfo);  }  else  {  newConnection.appsNo = newConnectionId;  ESCMSStorage.DbInteraction.EditNewConnection(newConnection);  }  this.Close();  }  privatestring GenerateId()  {  returnDateTime.Now.ToOADate().ToString();  }  privatestring serviceConnectionNo()  {  returnDateTime.Now.ToOADate().ToString();  }  privatevoid cancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  privatestring PopulatePaymentComboboxItems()  {  List<PaymentInfo> payments = ESCMSStorage.DbInteraction.GetUnassignedPaymentList();  \_paymentInfoCollection.Clear();  foreach (PaymentInfo pay in payments)  {  \_paymentInfoCollection.Add(pay);  }  if (\_paymentInfoCollection.Count > 0)  return \_paymentInfoCollection[0].id;  returnnull;  }  privatevoid paymentIDTxtbox\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  string currentPaymentId = (e.AddedItems[0] asPaymentInfo).id;  PopulateItemsFromPaymentId(currentPaymentId);  }  privatevoid PopulateItemsFromPaymentId(string payId)  {  PaymentInfo info = ESCMSStorage.DbInteraction.GetPaymentInfo(payId);  customerIdlabel.Content = info.customerId;  initialDepositAmountTxtbox.Content = info.amount;  amountReceivedOnDatePicker.SelectedDate = info.dop;  ESCMSData.CustomerInfo cusInfo = ESCMSStorage.DbInteraction.GetCustomerInfo(info.customerId);  nameTxtbox.Content = cusInfo.name;  addressTxtbox.Content = cusInfo.address;  contactNoTxtbox.Content = cusInfo.contact;  }  }  } |

### ESCMSData

|  |
| --- |
| NewConnectionInfo.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace ESCMSData  {  publicclassNewConnectionInfo  {  publicstring appsNo { get; set; }  publicDateTime receivedDate { get; set; }  publicstring paymentId { get; set; }  publicDateTime amountReceivedOn { get; set; }  publicdouble initialAmount { get; set; }  publicstring customerId { get; set; }  publicstring name { get; set; }  publicstring address { get; set; }  publicstring phone { get; set; }  publicdouble quotationAmount { get; set; }  publicDateTime quotationSendDate { get; set; }  publicstring serviceConnectionNo { get; set; }  }  } |

|  |
| --- |
| PaymentInfo.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace ESCMSData  {  publicclassPaymentInfo  {  publicstring id { get; set; }  publicstring name { get; set; }  publicstring customerId { get; set; }  publicdouble amount { get; set; }  publicDateTime dop { get; set; }  }  } |

|  |
| --- |
| estimateInfo.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace ESCMSData  {  publicenumAngleType  {  Short,  Long  }  publicclassestimateInfo  {  publicstring appsNo { get; set; }  publicstring name { get; set; }  publicstring address { get; set; }  publicstring contact { get; set; }  publicstring estimator { get; set; }  publicdouble initialAmount { get; set; }  publicdouble wireLength { get; set; }  publicdouble angleWeight { get; set; }  publicAngleType angleType { get; set; }  publicdouble quotationAmount { get; set; }  publicstring contractor { get; set; }  }  } |

|  |
| --- |
| EmployeeInfo.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace ESCMSData  {  publicenumPostType  {  DataEnterer,  Officer,  Estimator,  Admin  }  publicclassEmployeeInfo  {  publicstring id { get; set; }  publicstring name { get; set; }  publicstring address { get; set; }  publicstring contact { get; set; }  publicPostType postType { get; set; }  publicDateTime doj { get; set; }  publicstring department { get; set; }  }  } |

|  |
| --- |
| CustomerInfo.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace ESCMSData  {  publicclassCustomerInfo  {  publicstring id { get; set; }  publicstring name { get; set; }  publicstring address { get; set; }  publicstring contact { get; set; }  }  } |

|  |
| --- |
| ContractorInfo.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace ESCMSData  {  publicclassContractorInfo  {  publicstring id { get; set; }  publicstring name { get; set; }  publicstring address { get; set; }  publicstring contact { get; set; }  publicstring details { get; set; }  }  } |

### ESCMSStorage

|  |
| --- |
| DbInteraction.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ESCMSData;  namespace ESCMSStorage  {  publicstaticclassDbInteraction  {  staticstring passwordCurrent = "1234";  staticstring dbmsCurrent = "escmsdatabase";  privatestatic 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  publicstaticint DoRegisterNewNewConnection(NewConnectionInfo newConnectionDetails)  {  return DoRegisterNewNewConnectionInDb(newConnectionDetails);  }  privatestaticint DoRegisterNewNewConnectionInDb(NewConnectionInfonewConnectionDetails)  {  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;  }  publicstaticList<NewConnectionInfo> GetAllNewConnectionList()  {  return QueryAllNewConnectionList();  }  privatestaticList<NewConnectionInfo> QueryAllNewConnectionList()  {  List<NewConnectionInfo> NewConnectionList = newList<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 = newNewConnectionInfo();  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;  }  publicstaticvoid 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();  }  }  publicstaticvoid 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();  }  }  publicstaticstring 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  publicstaticint DoRegisterNewEmployee(EmployeeInfo employeeDetails)  {  return DoRegisterNewEmployeeInDb(employeeDetails);  }  privatestaticint 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;  }  publicstaticList<EmployeeInfo> GetAllEmployeeList()  {  return QueryAllEmployeeList();  }  privatestaticList<EmployeeInfo> QueryAllEmployeeList()  {  List<EmployeeInfo> EmployeeList = newList<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 group by name;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  EmployeeInfo Employee = newEmployeeInfo();  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;  }  publicstaticvoid 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();  }  }  publicstaticvoid 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();  }  }  publicstaticList<EmployeeInfo> SearchAllEmployeeList(EmployeeInfo empInfoObj)  {  return SearchAllEmployeeListDetails(empInfoObj);  }  privatestaticList<EmployeeInfo> SearchAllEmployeeListDetails(EmployeeInfo empInfoObj)  {  List<EmployeeInfo> EmployeeList = newList<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 name = @input or address = @input or contact = @input or post\_type = @input or department = @input;";  msqlCommand.Parameters.AddWithValue("@input", empInfoObj.name);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  EmployeeInfo Employee = newEmployeeInfo();  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;  }  #endregion  #region Contractor  publicstaticint DoRegisterNewContractor(ContractorInfo contractorDetails)  {  return DoRegisterNewContractorInDb(contractorDetails);  }  privatestaticint 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;  }  publicstaticList<ContractorInfo> GetAllContractorList()  {  return QueryAllContractorList();  }  privatestaticList<ContractorInfo> QueryAllContractorList()  {  List<ContractorInfo> ContractorList = newList<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 group by name;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  ContractorInfo Contractor = newContractorInfo();  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;  }  publicstaticvoid 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();  }  }  publicstaticvoid 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();  }  }  publicstaticList<ContractorInfo> SearchContractorList(ContractorInfo conInfoObj)  {  return searchAllContractorList(conInfoObj);  }  privatestaticList<ContractorInfo> searchAllContractorList(ContractorInfo coninfoObj)  {  List<ContractorInfo> ContractorList = newList<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 where name = @input or address = @input or contact = @input or contract\_details = @input;";  msqlCommand.Parameters.AddWithValue("@input", coninfoObj.name);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  ContractorInfo Contractor = newContractorInfo();  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;  }  #endregion  #region Payment  publicstaticList<PaymentInfo> GetUnassignedPaymentList()  {  List<PaymentInfo> PaymentList = newList<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 = newPaymentInfo();  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;  }  publicstaticint DoRegisterNewPayment(PaymentInfo peaymentDetails)  {  return DoRegisterNewPaymentInDb(peaymentDetails);  }  privatestaticint 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;  }  publicstaticList<PaymentInfo> GetAllPaymentList()  {  return QueryAllPaymentList();  }  privatestaticList<PaymentInfo> QueryAllPaymentList()  {  List<PaymentInfo> PaymentList = newList<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 = newPaymentInfo();  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;  }  publicstaticvoid 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();  }  }  publicstaticvoid 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();  }  }  publicstaticPaymentInfo GetPaymentInfo(string paymentId)  {  PaymentInfo payment = newPaymentInfo();  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  publicstaticList<EmployeeInfo> GetAllEstimatorList()  {  List<EmployeeInfo> EmployeeList = newList<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 = newEmployeeInfo();  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;  }  publicstaticint DoRegisterNewEstimate(estimateInfo estimateToEdit)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = OpenDbConnection();  try  {  //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = newMySql.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;  }  publicstaticvoid 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();  }  }  publicstaticList<estimateInfo> GetAllEstimateList()  {  List<estimateInfo> estimateList = newList<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 = newestimateInfo();  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  publicstaticCustomerInfo GetCustomerInfo(string cusId)  {  CustomerInfo customer = newCustomerInfo();  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;  }  publicstaticint DoRegisterNewCustomer(CustomerInfo customerDetails)  {  return DoRegisterNewCustomerInDb(customerDetails);  }  privatestaticint 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;  }  publicstaticList<CustomerInfo> GetAllCustomerList()  {  return QueryAllCustomerList();  }  privatestaticList<CustomerInfo> QueryAllCustomerList()  {  List<CustomerInfo> CustomerList = newList<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 group by name;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  CustomerInfo Customer = newCustomerInfo();  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;  }  publicstaticvoid 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();  }  }  publicstaticvoid 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();  }  }  publicstaticList<CustomerInfo> searchCustomerList(CustomerInfo custinfo)  {  return searchAllCustomerList(custinfo);  }  privatestaticList<CustomerInfo> searchAllCustomerList(CustomerInfo custinfo)  {  List<CustomerInfo> CustomerList = newList<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 name = @input or address = @input or contact = @input or id = @input ; ";  msqlCommand.Parameters.AddWithValue("@input", custinfo.name);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  CustomerInfo Customer = newCustomerInfo();  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;  }  #endregion  publicstaticvoid 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();  }  }  publicstaticList<estimateInfo> GetAllEstimateListWithContractor()  {  List<estimateInfo> estimateList = newList<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 = newestimateInfo();  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;  }  publicstaticvoid 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();  }  }  }  } |

### EscmsStyles

|  |
| --- |
| ControlStyle.xaml |
| <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="White" />  <Setter Property="IsEditable" Value="True" />  <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> |

## Comments and Description of Coding segments

|  |  |
| --- | --- |
|  |  |

**ECMS Namespace**

**Classes**

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Public class | [AddBill](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1082937720.htm) | Interaction logic for AddBill.xaml |
| Public class | [AddCustomer](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\184105858.htm) | Interaction logic for AddCustomer.xaml |
| Public class | [AddEstimator](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\898447579.htm) | Interaction logic for AddEstimator.xaml |
| Public class | [AddPayment](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\352280904.htm) | Interaction logic for AddPayment.xaml |
| Public class | [AddQuotation](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1386580595.htm) | Interaction logic for AddQuotation.xaml |
| Public class | [App](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\925390383.htm) | Interaction logic for App.xaml |
| Public class | [AssignContractor](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1784910850.htm) | Interaction logic for AssignContractor.xaml |
| Public class | [ChangePassword](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1603259998.htm) | Interaction logic for ChangePassword.xaml |
| Public class | [Connection](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\2119422741.htm) | Interaction logic for Connection.xaml |
| Public class | [AddContractor](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1646331701.htm) | Interaction logic for Contractor.xaml |
| Public class | [ContractorBill](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1112746571.htm) | Interaction logic for ContractorBill.xaml |
| Public class | [Contractor](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\2017809802.htm) | Interaction logic for ContractorBrowser.xaml |
| Public class | [Customer](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\602678145.htm) | Interaction logic for Customer.xaml |
| Public class | [AddEmployee](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\754136453.htm) | Interaction logic for Employee.xaml |
| Public class | [Employee](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\2138040054.htm) | Interaction logic for Employee.xaml |
| Public class | [EstimationQuotationDisplay](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\2112478242.htm) | Interaction logic for EstimationQuatationDisplay.xaml |
| Public class | [FirstLoginWindow](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\306491472.htm) | Interaction logic for FirstLoginWindow.xaml |
| Public class | [Credits](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1314378784.htm) | Interaction logic for Help.xaml |
| Public class | [login](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\247903233.htm) | Interaction logic for login.xaml |
| Public class | [MainWindow](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\830531009.htm) | Interaction logic for MainWindow.xaml |
| Public class | [newapplication](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1501638206.htm) | Interaction logic for newapplication.xaml |
| Public class | [Payment](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1809366012.htm) | Interaction logic for Payment.xaml |
| Public class | [Report](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\328979272.htm) | Interaction logic for Report.xaml |
| Public class | [Search](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\994627739.htm) | Interaction logic for Search.xaml |
| Public class | [Settings](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1788945748.htm) | Interaction logic for Settings.xaml |
| Public class | [WorkOrder](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\78502787.htm) | Interaction logic for WorkOrder.xaml |

**Delegates**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Delegate | | Description | |
| Public delegate | | [login..::..delegateOnSucccesfulLogin](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\2015326467.htm) | |  | |
|  | |  | |

**ESCMSData Namespace**

**Classes**

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Public class | [ContractorInfo](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1724623368.htm) |  |
| Public class | [CustomerInfo](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1977677038.htm) |  |
| Public class | [EmployeeInfo](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1710459084.htm) |  |
| Public class | [estimateInfo](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1282101193.htm) |  |
| Public class | [NewConnectionInfo](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\496899345.htm) |  |
| Public class | [PaymentInfo](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1106578629.htm) |  |

**Enumerations**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Enumeration | | Description | |
| Public enumeration | | [PostType](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1173755643.htm) | |  | |
| Public enumeration | | [AngleType](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1791310350.htm) | |  | |
|  | |  | |

**ECMS.Properties Namespace**

**Classes**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Class | Description |
| Protected class | | [Settings](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1587494287.htm) |  |
| Protected class | | [Resources](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\630978387.htm) | A strongly-typed resource class, for looking up localized strings, etc. |
|  | |  |

**ESCMSStorage Namespace**

**Classes**

|  |  |  |
| --- | --- | --- |
|  | Class | Description |
| Public class | [DbInteraction](file:///C:\Users\chandra\Documents\GitHub\ElectricSupplyCustomerManagmentSystem\Help\html\1289587633.htm) |  |

### Code Commenting

* All comments have been written in the same language, be grammatically correct, and contain appropriate
* punctuation.
* Used // or /// but never /\* … \*/
* Did not “flowerbox” comment blocks.

Example:

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Comment block

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Always Used inline-comments to explain assumptions, known issues, and algorithm insights.
* Never used inline-comments to explain obvious code. Well written code is self documenting.
* Only used comments for bad code to say “fix this code” – otherwise remove, or rewrite the code!
* Included comments using Task-List keyword flags to allow comment-filtering.

Example:

// TODO: Place Database Code Here

// UNDONE: Removed P\Invoke Call due to errors

// HACK: Temporary fix until able to refactor

* Always applied C# comment-blocks (///) to public, protected, and internal declarations.
* Only used C# comment-blocks for documenting the API.
* Always included <summary> comments. Include <param>, <return>, and <exception> comment
* sections where applicable.
* Included <see cref=””/> and <seeAlso cref=””/> where possible.
* Always added CDATA tags to comments containing code and other embedded markup in order to avoid encoding issues.

Example:

/// <example>

/// Add the following key to the “appSettings” section of your config:

/// <code><![CDATA[

/// <configuration>

/// <appSettings>

/// <add key=”mySetting” value=”myValue”/>

/// </appSettings>

/// </configuration>

/// ]]></code>>

/// </example>

## Standardization of the coding

Coding style causes the most inconsistency and controversy between developers. Each developer has a preference, and

rarely are two the same. However, consistent layout, format, and organization are key to creating maintainable code.

The following sections describe the preferred way to implement C# source code in order to create readable, clear, and

consistent code that is easy to understand and maintain.

#### Formatting

* Never declared more than 1 namespace per file.
* Avoided putting multiple classes in a single file.
* Always placed curly braSMS ({ and }) on a new line.
* Always used curly braSMS ({ and }) in conditional statements.
* Always used a Tab & Indention size of 4.
* Declared each variable independently – not in the same statement.
* Placed namespace “using” statements together at the top of file. Group .NET namespaSMS above custom namespaSMS.
* Grouped internal class implementation by type in the following order:

1. Member variables.
2. Constructors & Finalizers.
3. Nested Enums, Structs, and Classes.
4. Properties
5. Methods

* Sequence declarations within type groups based upon acSMSs modifier and visibility:

1. Public
2. Protected
3. Internal
4. Private

* Segregate interface Implementation by using #region statements.
* Append folder-name to namespace for source files within sub-folders.
* Recursively indent all code blocks contained within braSMS.
* Use white space (CR/LF, Tabs, etc) liberally to separate and organize code.
* Only declare related attribute declarations on a single line, otherwise stack each attribute as a separatedeclaration.

Example:

// Bad!

[Attrbute1, Attrbute2, Attrbute3]

public class MyClass

{…}

// Good!

[Attrbute1, RelatedAttribute2]

[Attrbute3]

[Attrbute4]

public class MyClass

{…}

* Place Assembly scope attribute declarations on a separate line.
* Place Type scope attribute declarations on a separate line.
* Place Method scope attribute declarations on a separate line.
* Place Member scope attribute declarations on a separate line.
* Place Parameter attribute declarations inline with the parameter.
* If in doubt, always err on the side of clarity and consistency.

## 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 studies all the rules of software development life cycle and applied them to keep our application flexible. 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, updating, deletion and fetching of data take place flexibly. You can see the result as a user; our application does all the works very smoothly.

## Error handling

The C# language's exception handling features help us to deal with any unexpected or exceptional situations that occur when a program is running. Exception handling uses the **try**, **catch**, and **finally** keywords to try actions that may not succeed, to handle failures when you decide that it is reasonable to do so, and to clean up resources afterward. Exceptions can be generated by the common language runtime (CLR), by the .NET Framework or any third-party libraries, or by application code. Exceptions are created by using the **throw** keyword.

In many cases, an exception may be thrown not by a method that your code has called directly, but by another method further down in the call stack. When this happens, the CLR will unwind the stack, looking for a method with a **catch** block for the specific exception type, and it will execute the first such **catch** block that if finds. If it finds no appropriate **catch** block anywhere in the call stack, it will terminate the process and display a message to the user.

[Exceptions Overview](javascript:void(0))

Exceptions have the following properties:

Exceptions are types that all ultimately derive from **System.Exception**.

Use a **try** block around the statements that might throw exceptions.

Once an exception occurs in the **try** block, the flow of control jumps to the first associated exception handler that is present anywhere in the call stack. In C#, the**catch** keyword is used to define an exception handler.

If no exception handler for a given exception is present, the program stops executing with an error message.

Do not catch an exception unless you can handle it and leave the application in a known state. If you catch **System.Exception**, rethrow it using the **throw** keyword at the end of the **catch** block.

If a **catch** block defines an exception variable, you can use it to obtain more information about the type of exception that occurred.

Exceptions can be explicitly generated by a program by using the **throw** keyword.

Exception objects contain detailed information about the error, such as the state of the call stack and a text description of the error.

Code in a **finally** block is executed even if an exception is thrown. Use a **finally** block to release resources, for example to close any streams or files that were opened in the **try** block.

Managed exceptions in the .NET Framework are implemented on top of the Win32 structured exception handling mechanism.

## Parameters calling/passing

## Validation checks

We have performed following data validation checks on available data:

#### Allowed character checks

Checks that ascertain that only expected characters are present in a field. For example a numeric field may only allow the digits 0-9, the decimal point and perhaps a minus sign or commas. A text field such as a personal name might disallow characters such as < and >, as they could be evidence of a markup-based security attack. An e-mail address might require exactly one @ sign and various other structural details. Regular expressions are effective ways of implementing such checks. (See also data type checks below)

#### Batch totals

Checks for missing records. Numerical fields may be added together for all records in a batch. The batch total is entered and the computer checks that the total is correct, e.g., add the 'Total Cost' field of a number of transactions together.

#### Cardinality check

Checks that record has a valid number of related records. For example if Contact record classified as a Customer it must have at least one associated Order (Cardinality > 0). If order does not exist for a "customer" record then it must be either changed to "seed" or the order must be created. This type of rule can be complicated by additional conditions. For example if contact record in Payroll database is marked as "former employee", then this record must not have any associated salary payments after the date on which employee left organization (Cardinality = 0).

#### Check digits

Used for numerical data. An extra digit is added to a number which is calculated from the digits. The computer checks this calculation when data are entered. For example the last digit of an ISBN for a book is a check digit calculated modulus 10.[3]

#### Consistency checks

Checks fields to ensure data in these fields corresponds, e.g., If Title = "Mr.", then Gender = "M".

#### Control totals

This is a total done on one or more numeric fields which appears in every record. This is a meaningful total, e.g., add the total payment for a number of Customers.

#### Cross-system consistency checks

Compares data in different systems to ensure it is consistent, e.g., The address for the customer with the same id is the same in both systems. The data may be represented differently in different systems and may need to be transformed to a common format to be compared, e.g., one system may store customer name in a single Name field as 'Doe, John Q', while another in three different fields: First\_Name (John), Last\_Name (Doe) and Middle\_Name (Quality); to compare the two, the validation engine would have to transform data from the second system to match the data from the first, for example, using SQL: Last\_Name || ', ' || First\_Name || substr(Middle\_Name, 1, 1) would convert the data from the second system to look like the data from the first 'Doe, John Q'

#### Data type checks

Checks the data type of the input and give an error message if the input data does not match with the chosen data type, e.g., In an input box accepting numeric data, if the letter 'O' was typed instead of the number zero, an error message would appear.

#### File existence check

Checks that a file with a specified name exists. This check is essential for programs that use file handling.

#### Format or picture check

Checks that the data is in a specified format (template), e.g., dates have to be in the format DD/MM/YYYY.

Regular expressions should be considered for this type of validation.

#### Hash totals

This is just a batch total done on one or more numeric fields which appears in every record. This is a meaningless total, e.g., add the Telephone Numbers together for a number of Customers.

#### Limit check

Unlike range checks, data are checked for one limit only, upper OR lower, e.g., data should not be greater than 2 (<=2).

#### Logic check

Checks that an input does not yield a logical error, e.g., an input value should not be 0 when there will be a number that divides it somewhere in a program.

#### Presence check

Checks that important data are actually present and have not been missed out, e.g., customers may be required to have their telephone numbers listed.

#### Range check

Checks that the data lie within a specified range of values, e.g., the month of a person's date of birth should lie between 1 and 12.

#### Referential integrity

In modern Relational database values in two tables can be linked through foreign key and primary key. If values in the primary key field are not constrained by database internal mechanism,[4] then they should be validated. Validation of the foreign key field checks that referencing table must always refer to a valid row in the referenced table.[5]

#### Spelling and grammar check

Looks for spelling and grammatical errors.

#### Uniqueness check

Checks that each value is unique. This can be applied to several fields (i.e. Address, First Name, Last Name).

#### Table Look Up Check

A table look up check takes the entered data item and compares it to a valid list of entries that are stored in a database table.

# Testing

## Testing techniques and testing strategies used

**ESCMS** application will be tested using following strategies to ensure that the application succeeds to complete all the functional and non functional requirements:

### Database & Data Integrity Testing

The databases and the database processes should be tested as a subsystem within the **ESCMS** Application.These subsystems should be tested with the target-of-test’s User Interface as the interface to the database.

|  |  |
| --- | --- |
| Test Objective: | Ensure that data is stored correctly, audits can be performed, access is controlled |
| Technique: | * SQL queries will be executed in the DB to verify the data content and correctness. |
| Completion Criteria: | * All planned tests have been executed. * All defects that have been identified have been resolved * All resolutions have been implemented. |

### Functional Testing:

Function testing focuses on any requirements for test that can be traced directly to use cases or business functions and business rules. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box techniques; that are verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI) and analyzing the output or results. Identified below is an outline of the function testing recommended for **ESCMS**:

|  |  |
| --- | --- |
| Test Objective: | Ensure proper target-of-test functionality, including business process validation. |
| Technique: | Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following:   * The expected results occur when valid data is used. * The appropriate error or warning messages are displayed when invalid data is used. * Business rules are properly applied. * Black Box end to end testing of configured processes. Manual validation of required and optional fields. |
| Completion Criteria: | * All planned tests have been executed. * All defects that have been identified have been resolved * All resolutions have been implemented. |

### Regression Testing:

Regression testing focuses on software functionality that may have been previously working however through subsequent changes may have been inadvertently impacted. The goals of these tests are to verify that the broader impact of changes has been verified. Identified below is an outline of the regression testing recommended for each application(s)/module(s) of **ESCMS**.

|  |  |
| --- | --- |
| Test Objective: | Ensure that previously passed test cases continue to pass as the new system development is deployed and that surrounding systems that may be impacted by a change are still functioning as expected. |
| Technique: | * Execute previous passed testing suites to ensure the following: * The expected results occur when valid data is used. * The appropriate error or warning messages are displayed when invalid data is used. * Each business rule is properly applied. |
| Completion Criteria: | • All planned regression tests have been executed.  • All identified defects have been resolved. |

### User Interface Testing:

User Interface (UI) testing verifies a user’s interaction with the software. The goal of UI testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the target-of-test. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards. Most of this testing will have been done during functional testing. The areas of focus will be on design, layout and navigation of the screens.

|  |  |
| --- | --- |
| Test Objective: | UI testing will verify the screens and the layouts and navigation |
| Technique: | * Verify the design and layout of the screen. * Identify the integration links. * Test the functioning of the links – that the proper page is displayed and correct messages, pop-ups are shown when they need to be displayed etc * Validation of general navigation |
| Completion Criteria: | * All navigation test cases have been executed. * All screens have been verified as per design and layouts * All defects that have been identified have been resolved. |

### Performance Profiling:

Performance profiling is a performance test in which response times, transaction rates, and other time-sensitive requirements are measured and evaluated. The goal of Performance Profiling is to verify performance requirements have been achieved. Performance profiling is implemented and executed to profile and tune performance behaviours as a function of conditions such as workload or hardware configurations

|  |  |
| --- | --- |
| Test Objective: | The purpose of performance profiling is to ensure the performance of the **ESCMS** application is up to the desired level. |
| Technique: | * Use a subset of Test Procedures developed for Function and Business Cycle Testing. * Modify data files to increase the number of transactions or the scripts to increase the number of iterations each transaction occurs. * This will be done by using Load Runner or Quick Test Professional (QTP). |
| Completion Criteria: | * Single Transaction or single user: Successful completion of the test scripts without any failures and within the expected or required time allocation per transaction. * Results are recorded and a performance baseline is created for the major logical functions within the scenarios listed above. * All performance defects are reviewed and triaged to an acceptable resolution. |

.

### Load Testing:

Load testing is a performance test which subjects the target-of-test to varying workloads to measure and evaluate the performance behaviours and ability of the target-of-test to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly at the expected maximum workload. Additionally, load testing evaluates the performance characteristics, such as response times, transaction rates, and other time sensitive issues.

|  |  |
| --- | --- |
| Test Objective: | The purpose of load testing is to verify performance behaviour time for designated transactions or business cases under varying workload conditions. |
| Technique: | * Use a subset of Test Procedures developed for Function and Business Cycle Testing. * Scripts will be executed to simulate the peak load for 1 hour and report will be generated and analysed. * This will be done using Load Runner. |
| Completion Criteria: | * Multiple transactions or multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation. * Results are recorded and a performance baseline is created for the major business functions within the scenarios listed above. * All load testing defects are reviewed and triaged to an acceptable resolution. |

### Stress Testing:

Stress testing is a type of performance test implemented and executed to find errors due to low resources or competition for resources. Low memory or disk space may reveal defects in the target-of-test that aren't apparent under normal conditions. Other defects might result from competition for shared resources like database locks or network bandwidth. Stress testing can also be used to identify the peak workload the target-of-test can handle, which is often beyond the production workload.

### Volume Testing:

Volume Testing subjects the target-of-test to large amounts of data to determine if limits are reached that cause the software to fail. Volume Testing also identifies the continuous maximum load or volume the target-of-test can handle for a given period. For example, if the target-of-test is processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.

### Security & Access Control Testing:

Security and Access Control Testing focus on following key areas of security:

* Application-level security, including access to the Data or Business Functions

Application-level security ensures the authentication and authorization of a user. Authentication ensures that the user is a valid user of the system and authorization ensures that the user has the proper privileges to perform specific tasks on desired resources of the system. Testing will be conducted to validate the rules by taking into considerations the various roles applicable for the system.

### Failover & Recovery Testing:

Failover and Recovery Testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity.

Failover testing ensures that, for those systems that must be kept running, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without loss of data or transactions.

Recovery testing is an antagonistic test process in which the application or system is exposed to extreme conditions, or simulated conditions, to cause a failure, such as device Input/Output (I/O) failures or invalid database pointers and keys. Recovery processes are invoked and the application or system is monitored and inspected to verify proper application, or system, and data recovery has been achieved.

### Configuration Testing:

Configuration testing verifies the operation of the target-of-test on different software and hardware configurations. In most production environments, the particular hardware specifications for the client workstations, network connections and database servers vary. Client workstations may have different software loaded⎯for example, applications, drivers, and so on⎯and at any one time, many different combinations may be active using different resources.

### Installation/Deploy & Back out Testing:

Installation testing has two purposes. The first is to ensure that the software can be installed under different conditions⎯such as a new installation, an upgrade and a complete or custom installation⎯under normal and abnormal conditions. Abnormal conditions include insufficient disk space, lack of privilege to create directories, and so on. The second purpose is to verify that, once installed; the software operates correctly and can be backed out successfully. This usually means running a number of the tests that were developed for Function testing before and after the back out.

### Post Production Testing:

The purpose of Post production testing is to verify that, once deployed, the software operates correctly. This usually means running a number of the tests that were developed for Function Testing ensuring that no data is changed/modified in production. Typically, the business SME’s assist with Post production testing.

### Unit Testing:

Unit testing will take place within the construction phase of the project. After application module has been built to meet design specifications, each component (screen, view, interface, conversion program, etc.) will be tested individually to help confirm that it functions properly as an individual unit. Basic performance testing will also be completed during unit test to resolve obvious issues with performance prior to the System Testing.

The resource responsible for development will conduct testing of their module using the unit test conditions defined by the developer based on detailed design documents. The final step of unit test will be a review by the team lead to obtain their signoff on the component test checklist.

### Smoke Testing:

|  |  |
| --- | --- |
| Test Objective: | Verifies the major functionality at high level in order to determine if further testing is possible. |
| Technique: | * After initial deployment to the test environment validate all critical components of the application prior to proceeding with testing. |
| Completion Criteria: | * Navigation through the application at high level is possible, testing can continue. |

### Data Migration Testing:

This is the process of testing to verify whether or not the data migration (or conversion) has been successfully completed. The testing process will be carried out by running SQL scripts on both the source and destination databases.

The fields which are present in the newdata Model in the Destination DB(s) will be migrated from the existing systemssource DB(s) to Destination DB(s).

|  |  |
| --- | --- |
| Test Objective: | The objective of this test is to verify that data migration is successful from source DB(s) to destination DB(s). |
| Technique: | * The Team is notified before the data migration. * Team runs queries on the source DB and fetches the data. * Data Migration is done. * Mapped data needs to be determined. * Team runs the queries on the Destination DB and fetches the data. * Cross verification of the data is done to see that data fetched from the old database is same as the data fetched from the new database. * Verification of the table structure. * Verification of record counts. * Verification of the data formatting. |
| Completion Criteria: | * Data fetched from the Source DB(s) and Destination DB(s) matches. * The record count in the Source and the Destination databases should be equal. * No data are truncated. * Data formatting is proper (if required at any instance). * All defects that have been identified have been resolved. |

## Test reports for Unit Test Cases and System Test Cases

### Test reports for Unit Test Cases

|  |  |  |
| --- | --- | --- |
| 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 |
| **ESCMS**-026 | NA | PASS |
| **ESCMS**-027 | NA | PASS |
| **ESCMS**-028 | NA | PASS |
| **ESCMS**-029 | NA | PASS |
| **ESCMS**-030 | NA | PASS |
| **ESCMS**-031 | NA | PASS |
| **ESCMS**-032 | NA | PASS |
| **ESCMS**-033 | NA | PASS |
| **ESCMS**-034 | NA | PASS |
| **ESCMS**-035 | NA | PASS |
| **ESCMS**-036 | NA | PASS |
| **ESCMS**-037 | NA | PASS |
| **ESCMS**-038 | NA | PASS |
| **ESCMS**-039 | NA | PASS |
| **ESCMS**-040 | NA | PASS |
| **ESCMS**-041 | NA | PASS |
| **ESCMS**-042 | NA | PASS |
| **ESCMS**-043 | NA | PASS |
| **ESCMS**-044 | NA | PASS |
| **ESCMS**-045 | NA | PASS |
| **ESCMS**-046 | NA | PASS |
| **ESCMS**-047 | NA | PASS |
|  |  |  |
| **ESCMS**-048 | NA | PASS |
| **ESCMS**-049 | NA | PASS |
| **ESCMS**-050 | NA | PASS |
| **ESCMS**-051 | NA | PASS |
| **ESCMS**-052 | NA | PASS |
| **ESCMS**-053 | NA | PASS |
| **ESCMS**-054 | NA | PASS |
| **ESCMS**-055 | NA | PASS |
| **ESCMS**-056 | NA | PASS |
| **ESCMS**-057 | NA | PASS |
| **ESCMS**-058 | NA | PASS |
| **ESCMS**-059 | NA | PASS |
| **ESCMS**-060 | NA | PASS |
| **ESCMS**-061 | NA | PASS |
| **ESCMS**-062 | NA | PASS |
| **ESCMS**-063 | NA | PASS |
| **ESCMS**-064 | NA | PASS |
| **ESCMS**-065 | NA | PASS |
| **ESCMS**-066 | NA | PASS |

### Test reports for System Test Cases

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Test Case Id | Comment | Status |
| **ESCMS**-067 | NA | PASS |
| **ESCMS**-068 | NA | PASS |
| **ESCMS**-069 | NA | PASS |
| **ESCMS**-070 | NA | PASS |
| **ESCMS**-071 | NA | PASS |
| **ESCMS**-072 | NA | PASS |
| **ESCMS**-073 | NA | PASS |
| **ESCMS**-074 | NA | PASS |
| **ESCMS**-075 | NA | PASS |
| **ESCMS**-076 | NA | PASS |
| **ESCMS**-077 | NA | PASS |
| **ESCMS**-078 | NA | PASS |
| **ESCMS**-079 | NA | PASS |
| **ESCMS**-080 | NA | PASS |
| **ESCMS**-081 | NA | PASS |
| **ESCMS**-082 | NA | PASS |
| **ESCMS**-083 | NA | PASS |
| **ESCMS**-084 | NA | PASS |
| **ESCMS**-085 | NA | PASS |
| **ESCMS**-086 | NA | PASS |
| **ESCMS**-087 | NA | PASS |
| **ESCMS**-089 | NA | PASS |
| **ESCMS**-090 | NA | PASS |
| **ESCMS**-091 | NA | PASS |
| **ESCMS**-092 | NA | PASS |
| **ESCMS**-093 | NA | PASS |

## Debugging and Code improvement:

# System Security measures:

## Database/data security:

It encrypts the data stored in the database so that even if someone succeeds to hack the database still not much harm could be done.

The application will use Google open-id authentication for web interface.

## Creation of User profiles and access rights

This software requires a valid password to login and then it allows using any of its features. It allows admin to create various types of accounts with different permission levels like clerk, officer, contractor etc. so that user can see relevant data only.

The login password will be saved in encrypted format in database.

This software will use Google open-id authentication for web interface.

# Cost Estimation of the Project along with Cost Estimation Model

We used the basic COCOMO model, which gives an approximate estimate of our **ESCMS** project parameters. The basic COCOMO estimation model is given by the following expressions:

Effort = a1 \* (KLOC)a2 PM

Tdev = b1 \* (Effort)b2months

Where

KLOC is the estimated size of the software product expressed in Kilo Lines of Code a1, a2, b1, b2 are constants for each category of software products.

Tdev is the estimated time to develop the software, expressed in months.

Effort is the total effort required to develop the software product, expressed in person-month (PM).

Our project is semidetached type, because the development team consists of a mixture of experienced and inexperienced staff like my guide and me. Team members may have limited experience on related system but may be unfamiliar with aspects of the system being developed.

## Estimation of development effort

For our Semi-detached class software product **ESCMS**, the formula for estimating the effort based on the code size is shown below:

Semi-detached **ESCMS**: Tdev = 3.0\*(KLOC)1.12 PM

## Estimation of development time

For our Semi-detached class software product **ESCMS**, the formula for estimating the development time based on the effort is given below:

Semi-detached **ESCMS**: Tdev = 2.5\*(Effort)0.35 months

Assume that the size of a Semi-detached **ESCMS**product has been estimated to be 3,200 lines of source code. Assume that the average salary of software engineer(me) is Rs. 20,000 per month.

Assume that the size of our

The basic COCOMO estimation formula for **ESCMS** semidetached software:

Our Effort =3.0\*(3.2)1.12PM

= 11 PM

Normal Development time = 2.5\*(11)0.35months

=6 months

Cost required to develop the product = Rs. 6 \* 20000

= Rs. 120,000

# Reports

List of reports that are likely to be generated in this software are given below:

* Locality wise connection report can be generated
* List of New connections can be generated
* List of Customers can be generated
* List of Vendors can be generated
* Connection details can be generated
* Fund details can be generated
* Yearly report can be generated
* Salary slips can be created

# 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.

# 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.

# Bibliography

## Website

* <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)
* http://www.mysql.com/support/
* <http://dev.mysql.com/>
* http://dev.mysql.com/support/
* <http://www.homeandlearn.co.uk/csharp/csharp.html>
* <http://www.wpftutorial.net/Home.html>
* <http://www.youtube.com/?gl=IN>

## Books

* Fundamentals of software engineering by Rajib Mall
* Pro C# 2010 and the .NET 4.0 Platform by Andrew Troselen
* C# Programming by Rob Miles
* **Programming C#** - E. R. Balaguruswamy
* And Electric Supply Professionals

# Appendices

## IDE Used:

### Visual Studio 2010

visual_studio_logo

Microsoft Visual Studio is a powerful IDE that ensures quality code throughout the entire application lifecycle, from design to deployment. Whether we are developing applications for SharePoint, the web, Windows, Windows Phone, and beyond, Visual Studio is the ultimate all-in-one solution. Visual Studio includes a [code editor](http://en.wikipedia.org/wiki/Code_editor) supporting [IntelliSense](http://en.wikipedia.org/wiki/IntelliSense) as well as [code refactoring](http://en.wikipedia.org/wiki/Code_refactoring). The integrated [debugger](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building [GUI](http://en.wikipedia.org/wiki/GUI) applications, web designer, [class](http://en.wikipedia.org/wiki/Class_(computing)) designer, and [database schema](http://en.wikipedia.org/wiki/Database_schema) designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for [source-control](http://en.wikipedia.org/wiki/Source_control) systems (like [Subversion](http://en.wikipedia.org/wiki/Subversion_(software)) and [Visual SourceSafe](http://en.wikipedia.org/wiki/Visual_SourceSafe)) and adding new toolsets like editors and visual designers for [domain-specific languages](http://en.wikipedia.org/wiki/Domain-specific_language) or toolsets for other aspects of the [software development lifecycle](http://en.wikipedia.org/wiki/Software_development_lifecycle) (like the [Team Foundation Server](http://en.wikipedia.org/wiki/Team_Foundation_Server) client: Team Explorer).

#### Standout Features

* User interface built on Windows Presentation Foundation (WPF)
* Improved Start page
* Improved code editor
* Improved IntelliSense
* Call Hierarchy Viewer

#### What problems does it solve?

The newly designed user experience is refreshing for an application showing its age. The user interface is built on WPF and no longer relies on the limited MDI interface in previous versions; this allows for better multi-monitor support with fly-out windows. The first thing you might notice when opening Visual Studio 2010 is the new Start page. As an xaml file, this page is completely customizable and includes the ability to remove and pin project files in the Recent Projects section.

The code editor has a number of enhancements. You can scale the font by holding down [Ctrl] while scrolling the mouse wheel. In previous versions of Visual Studio, users had to set the font size through a dialog and exit to see if the changes were correct.

In Visual Studio 2010, Box Selection is enhanced to allow for zero-length boxes and improved pasting.

The feature that will see the most use (by accident if not design) is Highlight References. By selecting any symbol, such as a variable or a property, all references to the symbol are highlighted. The symbols can then be navigated by holding down [Ctrl][Shift] and pressing the up/down keys.

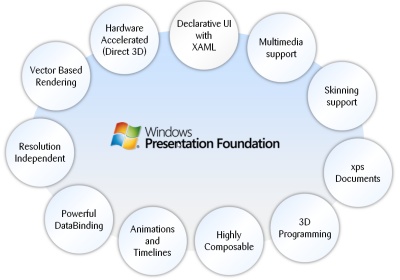
IntelliSense has been improved to allow for acronyms based on Pascal casing. For example, typing String.INOE and then a non-alphanumeric character will convert the call toString.IsNullOrEmpty. This still doesn’t prevent IntelliSense from interfering when you’re writing code that doesn’t exist, as you would with a unit test.

The Suggestion Completion mode allows you to type freely without IntelliSense changing the text you typed. You can toggle between Standard and Suggestion Completion modes by pressing [Ctrl][Alt]space.

IntelliSense for JavaScript has seen the most improvement, as it is now able to determine the correct structure of a variable even after the structure is changed.

In the past, I would use .NET Reflector or another tool to analyze a user’s call hierarchy; now that functionality is built-in. Right-click the user and choose View Call Hierarchy, and calls to and from the user will be available for browsing.

## Front End - WPF (Windows Presentation Framework)



Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications.

Windows Presentation Foundation (WPF) provides developers with a unified programming model for building rich Windows smart client user experiences that incorporate UI, media, and documents. Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications. The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typography. WPF is included in the Microsoft .NET Framework, so you can build applications that incorporate other elements of the .NET Framework class library.

The core of WPF is a resolution-independent and vector-based rendering engine that is built to take advantage of modern graphics hardware. WPF extends the core with a comprehensive set of application-development features that include Extensible Application Markup Language (XAML), controls, data binding, layout, 2-D and 3-D graphics, animation, styles, templates, documents, media, text, and typography. WPF is included in the Microsoft .NET Framework, so you can build applications that incorporate other elements of the .NET Framework class library.

#### Programming with wpf

WPF exists as a subset of .NET Framework types that are for the most part located in the [System.Windows](http://msdn.microsoft.com/en-IN/library/system.windows.aspx) namespace. If you have previously built applications with .NET Framework using managed technologies like ASP.NET and Windows Forms, the fundamental WPF programming experience should be familiar; you instantiate classes, set properties, call methods, and handle events, all using your favorite .NET Framework programming language, such as C# or Visual Basic.

#### Markup & code-behind

WPF offers additional programming enhancements for Windows client application development. One obvious enhancement is the ability to develop an application using both *markup* and *code-behind*, an experience that ASP.NET developers should be familiar with. You generally use Extensible Application Markup Language (XAML) markup to implement the appearance of an application while using managed programming languages (code-behind) to implement its behavior.

#### security

Because XBAPs are hosted in a browser, security is important. In particular, a partial-trust security sandbox is used by XBAPs to enforce restrictions that are less than or equal to the restrictions imposed on HTML-based applications. Furthermore, each HTML feature that is safe to run from XBAPs in partial trust has been tested using a comprehensive security process.

#### controls

The user experiences that are delivered by the application model are constructed controls. In WPF, "control" is an umbrella term that applies to a category of WPF classes that are hosted in either a window or a page, have a user interface (UI), and implement some behavior.

#### Wpf controls by function

The built-in WPF controls are listed here.

* **Buttons**: [Button](http://msdn.microsoft.com/en-IN/library/system.windows.controls.button.aspx) and [RepeatButton](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.repeatbutton.aspx).
* **Data Display**: [DataGrid](http://msdn.microsoft.com/en-IN/library/system.windows.controls.datagrid.aspx), [ListView](http://msdn.microsoft.com/en-IN/library/system.windows.controls.listview.aspx),and [TreeView](http://msdn.microsoft.com/en-IN/library/system.windows.controls.treeview.aspx).
* **Date Display and Selection**: [Calendar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.calendar.aspx) and [DatePicker](http://msdn.microsoft.com/en-IN/library/system.windows.controls.datepicker.aspx).
* **Dialog Boxes**: [OpenFileDialog](http://msdn.microsoft.com/en-IN/library/microsoft.win32.openfiledialog.aspx), [PrintDialog](http://msdn.microsoft.com/en-IN/library/system.windows.controls.printdialog.aspx), and [SaveFileDialog](http://msdn.microsoft.com/en-IN/library/microsoft.win32.savefiledialog.aspx).
* **Digital Ink**: [InkCanvas](http://msdn.microsoft.com/en-IN/library/system.windows.controls.inkcanvas.aspx) and [InkPresenter](http://msdn.microsoft.com/en-IN/library/system.windows.controls.inkpresenter.aspx).
* **Documents**: [DocumentViewer](http://msdn.microsoft.com/en-IN/library/system.windows.controls.documentviewer.aspx), [FlowDocumentPageViewer](http://msdn.microsoft.com/en-IN/library/system.windows.controls.flowdocumentpageviewer.aspx), [FlowDocumentReader](http://msdn.microsoft.com/en-IN/library/system.windows.controls.flowdocumentreader.aspx), [FlowDocumentScrollViewer](http://msdn.microsoft.com/en-IN/library/system.windows.controls.flowdocumentscrollviewer.aspx), and[StickyNoteControl](http://msdn.microsoft.com/en-IN/library/system.windows.controls.stickynotecontrol.aspx).
* **Input**: [TextBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.textbox.aspx), [RichTextBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.richtextbox.aspx), and [PasswordBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.passwordbox.aspx).
* **Layout**: [Border](http://msdn.microsoft.com/en-IN/library/system.windows.controls.border.aspx), [BulletDecorator](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.bulletdecorator.aspx), [Canvas](http://msdn.microsoft.com/en-IN/library/system.windows.controls.canvas.aspx), [DockPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.dockpanel.aspx), [Expander](http://msdn.microsoft.com/en-IN/library/system.windows.controls.expander.aspx), [Grid](http://msdn.microsoft.com/en-IN/library/system.windows.controls.grid.aspx), [GridView](http://msdn.microsoft.com/en-IN/library/system.windows.controls.gridview.aspx), [GridSplitter](http://msdn.microsoft.com/en-IN/library/system.windows.controls.gridsplitter.aspx), [GroupBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.groupbox.aspx), [Panel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.panel.aspx),[ResizeGrip](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.resizegrip.aspx), [Separator](http://msdn.microsoft.com/en-IN/library/system.windows.controls.separator.aspx), [ScrollBar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.scrollbar.aspx), [ScrollViewer](http://msdn.microsoft.com/en-IN/library/system.windows.controls.scrollviewer.aspx), [StackPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.stackpanel.aspx), [Thumb](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.thumb.aspx), [Viewbox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.viewbox.aspx), [VirtualizingStackPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.virtualizingstackpanel.aspx), [Window](http://msdn.microsoft.com/en-IN/library/system.windows.window.aspx), and[WrapPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.wrappanel.aspx).
* **Media**: [Image](http://msdn.microsoft.com/en-IN/library/system.windows.controls.image.aspx), [MediaElement](http://msdn.microsoft.com/en-IN/library/system.windows.controls.mediaelement.aspx), and [SoundPlayerAction](http://msdn.microsoft.com/en-IN/library/system.windows.controls.soundplayeraction.aspx).
* **Menus**: [ContextMenu](http://msdn.microsoft.com/en-IN/library/system.windows.controls.contextmenu.aspx), [Menu](http://msdn.microsoft.com/en-IN/library/system.windows.controls.menu.aspx), and [ToolBar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.toolbar.aspx).
* **Navigation**: [Frame](http://msdn.microsoft.com/en-IN/library/system.windows.controls.frame.aspx), [Hyperlink](http://msdn.microsoft.com/en-IN/library/system.windows.documents.hyperlink.aspx), [Page](http://msdn.microsoft.com/en-IN/library/system.windows.controls.page.aspx), [NavigationWindow](http://msdn.microsoft.com/en-IN/library/system.windows.navigation.navigationwindow.aspx), and [TabControl](http://msdn.microsoft.com/en-IN/library/system.windows.controls.tabcontrol.aspx).
* **Selection**: [CheckBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.checkbox.aspx), [ComboBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.combobox.aspx), [ListBox](http://msdn.microsoft.com/en-IN/library/system.windows.controls.listbox.aspx), [RadioButton](http://msdn.microsoft.com/en-IN/library/system.windows.controls.radiobutton.aspx), and [Slider](http://msdn.microsoft.com/en-IN/library/system.windows.controls.slider.aspx).
* **User Information**: [AccessText](http://msdn.microsoft.com/en-IN/library/system.windows.controls.accesstext.aspx), [Label](http://msdn.microsoft.com/en-IN/library/system.windows.controls.label.aspx), [Popup](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.popup.aspx), [ProgressBar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.progressbar.aspx), [StatusBar](http://msdn.microsoft.com/en-IN/library/system.windows.controls.primitives.statusbar.aspx), [TextBlock](http://msdn.microsoft.com/en-IN/library/system.windows.controls.textblock.aspx), and [ToolTip](http://msdn.microsoft.com/en-IN/library/system.windows.controls.tooltip.aspx).

#### layout

When you create a UI, you arrange your controls by location and size to form a layout. A key requirement of any layout is to adapt to changes in window size and display settings. Rather than forcing you to write the code to adapt a layout in these circumstances, WPF provides a first-class, extensible layout system for you.

The cornerstone of the layout system is relative positioning, which increases the ability to adapt to changing window and display conditions. In addition, the layout system manages the negotiation between controls to determine the layout. The negotiation is a two-step process: first, a control tells its parent what location and size it requires; second, the parent tells the control what space it can have.

The layout system is exposed to child controls through base WPF classes. For common layouts such as grids, stacking, and docking, WPF includes several layout controls:

* [Canvas](http://msdn.microsoft.com/en-IN/library/system.windows.controls.canvas.aspx) : Child controls provide their own layout.
* [DockPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.dockpanel.aspx) : Child controls are aligned to the edges of the panel.
* [Grid](http://msdn.microsoft.com/en-IN/library/system.windows.controls.grid.aspx) : Child controls are positioned by rows and columns.
* [StackPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.stackpanel.aspx) : Child controls are stacked either vertically or horizontally.
* [VirtualizingStackPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.virtualizingstackpanel.aspx) : Child controls are virtualized and arranged on a single line that is either horizontally or vertically oriented.
* [WrapPanel](http://msdn.microsoft.com/en-IN/library/system.windows.controls.wrappanel.aspx) : Child controls are positioned in left-to-right order and wrapped to the next line when there are more controls on the current line than space allows.

#### graphics

WPF introduces an extensive, scalable, and flexible set of graphics features that have the following benefits:

* **Resolution-independent and device-independent graphics**. The basic unit of measurement in the WPF graphics system is the device independent pixel, which is 1/96th of an inch, regardless of actual screen resolution, and provides the foundation for resolution-independent and device-independent rendering. Each device-independent pixel automatically scales to match the dots-per-inch (dpi) setting of the system it renders on.
* **Improved precision**. The WPF coordinate system is measured with double-precision floating-point numbers rather than single-precision. Transformations and opacity values are also expressed as double-precision. WPF also supports a wide color gamut (scRGB) and provides integrated support for managing inputs from different color spaces.
* **Advanced graphics and animation support**. WPF simplifies graphics programming by managing animation scenes for you; there is no need to worry about scene processing, rendering loops, and bilinear interpolation. Additionally, WPF provides hit-testing support and full alpha-compositing support.
* **Hardware acceleration**. The WPF graphics system takes advantage of graphics hardware to minimize CPU usage.

## Extensible application Markup Language (XAML)



XAML stands for Extensible Application Markup Language. Its a simple language based on XML to create and initialize .NET objects with hierarchical relations. Although it was originally invented for WPF it can by used to create any kind of object trees.

Today XAML is used to create user interfaces in WPF, Silverlight, declare workflows in WF and for electronic paper in the XPS standard.

All classes in WPF have parameter less constructors and make excessive usage of properties. That is done to make it perfectly fit for XML languages like XAML.

All you can do in XAML can also be done in code. XAML ist just another way to create and initialize objects. You can use WPF without using XAML. It's up to you if you want to declare it in XAML or write it in code. Declare your UI in XAML has some advantages:

* XAML code is short and clear to read
* Separation of designer code and logic
* Graphical design tools like Expression Blend require XAML as source.
* The separation of XAML and UI logic allows it to clearly separate the roles of designer and developer.

## Programming Framework

### .NET 4.5



The .NET Framework is a development platform for building apps for Windows, Windows Phone, Windows Server, and Windows Azure. It consists of the common language runtime (CLR) and the .NET Framework class library, which includes classes, interfaces, and value types that support an extensive range of technologies. The .NET Framework provides a managed execution environment, simplified development and deployment, and integration with a variety of programming languages, including Visual Basic and Visual C#.

#### .net framework class libraries

The .NET Framework class library is a library of classes, interfaces, and value types that provide access to system functionality. It is the foundation on which .NET Framework applications, components, and controls are built. The namespaces and namespace categories in the class library are listed in the following table and documented in detail in this reference. The namespaces and categories are listed by usage, with the most frequently used namespaces appearing first.

|  |  |
| --- | --- |
| **Namespace** | **Description** |
| [System](http://msdn.microsoft.com/en-us/library/system.aspx) | The [System](http://msdn.microsoft.com/en-us/library/system.aspx) namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions. |
| [System.Activities](http://msdn.microsoft.com/en-us/library/gg145022.aspx) | The System.Activities namespaces contain all the classes necessary to create and work with activities in Window Workflow Foundation. |
| [System.AddIn](http://msdn.microsoft.com/en-us/library/gg145020.aspx) | The System.AddIn namespaces contain types used to identify, register, activate, and control add-ins, and to allow add-ins to communicate with a host application. |
| [System.CodeDom](http://msdn.microsoft.com/en-us/library/gg145034.aspx) | The System.CodeDom namespaces contain classes that represent the elements of a source code document and that support the generation and compilation of source code in supported programming languages. |
| [System.Collections](http://msdn.microsoft.com/en-us/library/gg145035.aspx) | The System.Collections namespaces contain types that define various standard, specialized, and generic collection objects. |
| [System.ComponentModel](http://msdn.microsoft.com/en-us/library/gg145042.aspx) | The System.ComponentModel namespaces contain types that implement the run-time and design-time behavior of components and controls. Child namespaces support the Managed Extensibility Framework (MEF), provide attribute classes that define metadata for ASP.NET Dynamic Data controls, and contain types that let you define the design-time behavior of components and their user interfaces. |
| [System.Configuration](http://msdn.microsoft.com/en-us/library/gg145027.aspx) | The System.Configuration namespaces contain types for handling configuration data, such as data in machine or application configuration files. Child namespaces contain types that are used to configure an assembly, to write custom installers for components, and to support a pluggable model for adding functionality to, or removing functionality from, both client and server applications. |
| [System.Data](http://msdn.microsoft.com/en-us/library/gg145028.aspx) | The System.Data namespaces contain classes for accessing and managing data from diverse sources. The top-level namespace and a number of the child namespaces together form the ADO.NET architecture and ADO.NET data providers. For example, providers are available for SQL Server, Oracle, ODBC, and OleDB. Other child namespaces contain classes used by the ADO.NET Entity Data Model (EDM) and by WCF Data Services. |
| [System.Deployment](http://msdn.microsoft.com/en-us/library/gg145029.aspx) | The System.Deployment namespaces contain types that support deployment of ClickOnce applications. |
| [System.Device.Location](http://msdn.microsoft.com/en-us/library/system.device.location.aspx) | The [System.Device.Location](http://msdn.microsoft.com/en-us/library/system.device.location.aspx) namespace allows application developers to easily access the computer's location by using a single API. Location information may come from multiple providers, such as GPS, Wi-Fi triangulation, and cell phone tower triangulation. The [System.Device.Location](http://msdn.microsoft.com/en-us/library/system.device.location.aspx) classes provide a single API to encapsulate the multiple location providers on a computer and support seamless prioritization and transitioning between them. As a result, application developers who use this API do not need to tailor applications to specific hardware configurations. |
| [System.Diagnostics](http://msdn.microsoft.com/en-us/library/gg145030.aspx) | The System.Diagnostics namespaces contain types that enable you to interact with system processes, event logs, and performance counters. Child namespaces contain types to interact with code analysis tools, to support contracts, to extend design-time support for application monitoring and instrumentation, to log event data using the Event Tracing for Windows (ETW) tracing subsystem, to read to and write from event logs and collect performance data, and to read and write debug symbol information. |
| [System.DirectoryServices](http://msdn.microsoft.com/en-us/library/gg145037.aspx) | The System.DirectoryServices namespaces contain types that provide access to Active Directory from managed code. |
| [System.Drawing](http://msdn.microsoft.com/en-us/library/gg145023.aspx) | The System.Drawing parent namespace contains types that support basic GDI+ graphics functionality. Child namespaces support advanced two-dimensional and vector graphics functionality, advanced imaging functionality, and print-related and typographical services. A child namespace also contains types that extend design-time user-interface logic and drawing. |
| [System.Dynamic](http://msdn.microsoft.com/en-us/library/system.dynamic.aspx) | The [System.Dynamic](http://msdn.microsoft.com/en-us/library/system.dynamic.aspx) namespace provides classes and interfaces that support Dynamic Language Runtime. |
| [System.EnterpriseServices](http://msdn.microsoft.com/en-us/library/gg145047.aspx) | The System.EnterpriseServices namespaces contain types that define the COM+ services architecture, which provides an infrastructure for enterprise applications. A child namespace supports Compensating Resource Manager (CRM), a COM+ service that enables non-transactional objects to be included in Microsoft Distributed Transaction Coordinator (DTC) transactions. Child namespaces are described briefly in the following table and documented in detail in this reference. |
| [System.Globalization](http://msdn.microsoft.com/en-us/library/system.globalization.aspx) | The [System.Globalization](http://msdn.microsoft.com/en-us/library/system.globalization.aspx) namespace contains classes that define culture-related information, including language, country/region, calendars in use, format patterns for dates, currency, and numbers, and sort order for strings. These classes are useful for writing globalized (internationalized) applications. Classes such as [StringInfo](http://msdn.microsoft.com/en-us/library/system.globalization.stringinfo.aspx) and[TextInfo](http://msdn.microsoft.com/en-us/library/system.globalization.textinfo.aspx) provide advanced globalization functionalities, including surrogate support and text element processing. |
| [System.IdentityModel](http://msdn.microsoft.com/en-us/library/gg145031.aspx) | The System.IdentityModel namespaces contain types that are used to provide authentication and authorization for .NET applications. |
| [System.IO](http://msdn.microsoft.com/en-us/library/gg145019.aspx) | The System.IO namespaces contain types that support input and output, including the ability to read and write data to streams either synchronously or asynchronously, to compress data in streams, to create and use isolated stores, to map files to an application's logical address space, to store multiple data objects in a single container, to communicate using anonymous or named pipes, to implement custom logging, and to handle the flow of data to and from serial ports. |
| [System.Linq](http://msdn.microsoft.com/en-us/library/gg145016.aspx) | The System.Linq namespaces contain types that support queries that use Language-Integrated Query (LINQ). This includes types that represent queries as objects in expression trees. |
| [System.Management](http://msdn.microsoft.com/en-us/library/gg145024.aspx) | The System.Management namespaces contain types that provide access to management information and management events about the system, devices, and applications instrumented to the Windows Management Instrumentation (WMI) infrastructure. These namespaces also contain types necessary for instrumenting applications so that they expose their management information and events through WMI to potential customers. |
| [System.Media](http://msdn.microsoft.com/en-us/library/system.media.aspx) | The [System.Media](http://msdn.microsoft.com/en-us/library/system.media.aspx) namespace contains classes for playing sound files and accessing sounds provided by the system. |
| [System.Messaging](http://msdn.microsoft.com/en-us/library/gg145046.aspx) | The System.Messaging namespaces contain types that enable you to connect to, monitor, and administer message queues on the network and to send, receive, or peek messages. A child namespace contains classes that can be used to extend design-time support for messaging classes. |
| [System.Net](http://msdn.microsoft.com/en-us/library/gg145039.aspx) | The System.Net namespaces contain classes that provide a simple programming interface for a number of network protocols, programmatically access and update configuration settings for the System.Net namespaces, define cache policies for web resources, compose and send e-mail, represent Multipurpose Internet Mail Exchange (MIME) headers, access network traffic data and network address information, and access peer-to-peer networking functionality. Additional child namespaces provide a managed implementation of the Windows Sockets (Winsock) interface and provide access to network streams for secure communications between hosts. |
| [System.Numerics](http://msdn.microsoft.com/en-us/library/system.numerics.aspx) | The [System.Numerics](http://msdn.microsoft.com/en-us/library/system.numerics.aspx) namespace contains numeric types that complement the numeric primitives, such as [Byte](http://msdn.microsoft.com/en-us/library/system.byte.aspx), [Double](http://msdn.microsoft.com/en-us/library/system.double.aspx), and [Int32](http://msdn.microsoft.com/en-us/library/system.int32.aspx), that are defined by the .NET Framework. |
| [System.Printing](http://msdn.microsoft.com/en-us/library/gg145044.aspx) | The System.Printing namespaces contain types that support printing, that provide access to the properties of print system objects and enable rapid copying of their property settings to another object of the same type, and that support the interconversion of managed System.PrintTicket objects and unmanaged GDI DEVMODE structures. |
| [System.Reflection](http://msdn.microsoft.com/en-us/library/gg145033.aspx) | The System.Reflection namespaces contain types that provide a managed view of loaded types, methods, and fields, and that can dynamically create and invoke types. A child namespace contains types that enable a compiler or other tool to emit metadata and Microsoft intermediate language (MSIL). |
| [System.Resources](http://msdn.microsoft.com/en-us/library/gg145043.aspx) | The System.Resources namespaces contain types that enable developers to create, store, and manage an application's culture-specific resources. |
| [System.Runtime](http://msdn.microsoft.com/en-us/library/gg145017.aspx) | The System.Runtime namespaces contain types that support an application's interaction with the common language runtime, and types that enable features such as application data caching, advanced exception handling, application activation within application domains, COM interop, distributed applications, serialization and deserialization, and versioning. Additional namespaces enable compiler writers to specify attributes that affect the run-time behavior of the common language runtime, define a contract for reliability between a set of code and other code that takes a dependency on it, and implement a persistence provider for Windows Communication Foundation (WCF). |
| [System.Security](http://msdn.microsoft.com/en-us/library/gg145025.aspx) | The System.Security namespaces contain classes that represent the .NET Framework security system and permissions. Child namespaces provide types that control access to and audit securable objects, allow authentication, provide crytographic services, control access to operations and resources based on policy, and support rights management of application-created content. |
| [System.ServiceModel](http://msdn.microsoft.com/en-us/library/gg145010.aspx) | The System.ServiceModel namespaces contain the types necessary to build Windows Communication Foundation (WCF) service and client applications. |
| [System.ServiceProcess](http://msdn.microsoft.com/en-us/library/gg145038.aspx) | The System.ServiceProcess namespaces contain types that enable you to implement, install, and control Windows service applications and extend design-time support for Windows service applications. |
| [System.Speech](http://msdn.microsoft.com/en-us/library/gg145021.aspx) | The System.Speech namespaces contain types that support speech recognition. |
| [System.Text](http://msdn.microsoft.com/en-us/library/gg145012.aspx) | The System.Text namespaces contain types for character encoding and string manipulation. A child namespace enables you to process text using regular expressions. |
| [System.Threading](http://msdn.microsoft.com/en-us/library/gg145014.aspx) | The System.Threading namespaces contain types that enable multithreaded programming. A child namespace provides types that simplify the work of writing concurrent and asynchronous code. |
| [System.Timers](http://msdn.microsoft.com/en-us/library/system.timers.aspx) | The [System.Timers](http://msdn.microsoft.com/en-us/library/system.timers.aspx) namespace provides the [Timer](http://msdn.microsoft.com/en-us/library/system.timers.timer.aspx) component, which allows you to raise an event on a specified interval. |
| [System.Transactions](http://msdn.microsoft.com/en-us/library/gg145032.aspx) | The System.Transactions namespaces contain types that support transactions with multiple, distributed participants, multiple phase notifications, and durable enlistments. A child namespace contains types that describe the configuration options used by the System.Transactions types. |
| [System.Web](http://msdn.microsoft.com/en-us/library/gg145018.aspx) | The System.Web namespaces contain types that enable browser/server communication. Child namespaces include types that support ASP.NET forms authentication, application services, data caching on the server, ASP.NET application configuration, dynamic data, HTTP handlers, JSON serialization, incorporating AJAX functionality into ASP.NET, ASP.NET security, and web services. |
| [System.Windows](http://msdn.microsoft.com/en-us/library/gg145013.aspx) | The System.Windows namespaces contain types used in Windows Presentation Foundation (WPF) applications, including animation clients, user interface controls, data binding, and type conversion. System.Windows.Forms and its child namespaces are used for developing Windows Forms applications. |
| [System.Workflow](http://msdn.microsoft.com/en-us/library/gg145026.aspx) | The System.Workflow namespaces contain types used to develop applications that use Windows Workflow Foundation. These types provide design time and run-time support for rules and activities, to configure, control, host, and debug the workflow runtime engine. |
| [System.Xaml](http://msdn.microsoft.com/en-us/library/gg145048.aspx) | The System.Xaml namespaces contain types that support parsing and processing the Extensible Application Markup Language (XAML). |
| [System.Xml](http://msdn.microsoft.com/en-us/library/gg145036.aspx) | The System.Xml namespaces contain types for processing XML. Child namespaces support serialization of XML documents or streams, XSD schemas, XQuery 1.0 and XPath 2.0, and LINQ to XML, which is an in-memory XML programming interface that enables easy modification of XML documents. |
| [Accessibility](http://msdn.microsoft.com/en-us/library/accessibility.aspx) | The [Accessibility](http://msdn.microsoft.com/en-us/library/accessibility.aspx) and all of its exposed members are part of a managed wrapper for the Component Object Model (COM) accessibility interface. |
| [Microsoft.Activities](http://msdn.microsoft.com/en-us/library/hh135392.aspx) | The Microsoft.Activities namespaces contain types that support MSBuild and debugger extensions for Windows Workflow Foundation applications. |
| [Microsoft.Aspnet.Snapin](http://msdn.microsoft.com/en-us/library/microsoft.aspnet.snapin.aspx) | The [Microsoft.Aspnet.Snapin](http://msdn.microsoft.com/en-us/library/microsoft.aspnet.snapin.aspx) namespace defines the types necessary for the ASP.NET management console application to interact with Microsoft Management Console (MMC). For more information, see "MMC Programmer's Guide" in the [MSDN Library](http://go.microsoft.com/fwlink/?linkid=37118). |
| [Microsoft.Build](http://msdn.microsoft.com/en-us/library/gg145008.aspx) | The Microsoft.Build namespaces contain types that provide programmatic access to, and control of, the MSBuild engine. |
| [Microsoft.CSharp](http://msdn.microsoft.com/en-us/library/gg145015.aspx) | The Microsoft.CSharp namespaces contain types that support compilation and code generation of source code written in the C# language, and types that support interoperation betwen the dynamic language runtime (DLR) and C#. |
| [Microsoft.Data.Entity.Build.Tasks](http://msdn.microsoft.com/en-us/library/microsoft.data.entity.build.tasks.aspx) | The [Microsoft.Data.Entity.Build.Tasks](http://msdn.microsoft.com/en-us/library/microsoft.data.entity.build.tasks.aspx) namespace contains two MSBuild tasks that are used by the ADO.NET Entity Data Model Designer (Entity Designer). |
| [Microsoft.JScript](http://msdn.microsoft.com/en-us/library/gg145041.aspx) | The Microsoft.JScript namespaces contain classes that support compilation and code generation using the JScript language. |
| [Microsoft.SqlServer.Server](http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.server.aspx) | The [Microsoft.SqlServer.Server](http://msdn.microsoft.com/en-us/library/microsoft.sqlserver.server.aspx) namespace contains classes, interfaces, and enumerations that are specific to the integration of the Microsoft .NET Framework common language runtime (CLR) into Microsoft SQL Server, and the SQL Server database engine process execution environment. |
| [Microsoft.VisualBasic](http://msdn.microsoft.com/en-us/library/gg145009.aspx) | The Microsoft.VisualBasic namespaces contain classes that support compilation and code generation using the Visual Basic language. Child namespaces contain types that provide services to the Visual Basic compiler and types that include support for the Visual Basic application model, the My namespace, lambda expressions, and code conversion. |
| [Microsoft.VisualC](http://msdn.microsoft.com/en-us/library/gg145040.aspx) | The Microsoft.VisualC namespaces contain types that support the Visual C++ compiler and types that implement the STL/CLR Library and the generic interface to the STL/CLR Library. |
| [Microsoft.Win32](http://msdn.microsoft.com/en-us/library/gg145011.aspx) | The Microsoft.Win32 namespaces provide types that handle events raised by the operating system, that manipulate the system registry, and that represent file and operating system handles. |
| [Microsoft.Windows](http://msdn.microsoft.com/en-us/library/hh135393.aspx) | The Microsoft.Windows namespaces contain types that support themes and preview in Windows Presentation Framework (WPF) applications. |
| [UIAutomationClientsideProviders](http://msdn.microsoft.com/en-us/library/uiautomationclientsideproviders.aspx) | Contains a single type that maps client automation providers. |
| [XamlGeneratedNamespace](http://msdn.microsoft.com/en-us/library/xamlgeneratednamespace.aspx) | Contains compiler-generated types that are not intended to be used directly from your code. |

## Database/backend:

### 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

#### Detailed features of mysql

The following list shows the most important properties of MySQL. This section is directed to the reader who already has some knowledge of relational databases. We will use some terminology from the relational database world without defining our terms exactly. On the other hand, the explanations should make it possible for database novices to understand to some extent what we are talking about.

**Relational Database System:** Like almost all other database systems on the market, MySQL is a relational database system.

**Client/Server Architecture:** MySQL is a client/server system. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc. The clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).

|  |  |
| --- | --- |
|  | |
|  |  |
|  | |

Almost all of the familiar large database systems (Oracle, Microsoft SQL Server, etc.) are client/server systems. These are in contrast to the file-server systems, which include Microsoft Access, dBase and FoxPro. The decisive drawback to file-server systems is that when run over a network, they become extremely inefficient as the number of users grows.

**SQL compatibility:** MySQL supports as its database language -- as its name suggests – SQL (Structured Query Language). SQL is a standardized language for querying and updating data and for the administration of a database. There are several SQL dialects (about as many as there are database systems). MySQL adheres to the current SQL standard (at the moment SQL:2003), although with significant restrictions and a large number of extensions.

Through the configuration setting sql-mode you can make the MySQL server behave for the most part compatibly with various database systems. Among these are IBM DB/2 and Oracle. (The setting sql-mode changes some of the syntax conventions, and performs no miracles.

**SubSELECTs:** Since version 4.1, MySQL is capable of processing a query in the form SELECT \* FROM table1 WHERE x IN (SELECT y FROM table2) (There are also numerous syntax variants for subSELECTs.)

**Views:** Put simply, views relate to an SQL query that is viewed as a distinct database object and makes possible a particular view of the database. MySQL has supported views since version 5.0.

**Stored procedures:** Here we are dealing with SQL code that is stored in the database system.

Stored procedures (SPs for short) are generally used to simplify certain steps, such as inserting or deleting a data record. For client programmers this has the advantage that they do not have to process the tables directly, but can rely on SPs. Like views, SPs help in the administration of large database projects. SPs can also increase efficiency. MySQL has supported SPs since version 5.0.

**Triggers:** Triggers are SQL commands that are automatically executed by the server in certain database operations (INSERT, UPDATE, and DELETE). MySQL has supported triggers in a limited form from version 5.0, and additional functionality is promised for version 5.1.

**Unicode:** MySQL has supported all conceivable character sets since version 4.1, including Latin-1, Latin-2, and Unicode (either in the variant UTF8 or UCS2).

**User interface:** There are a number of convenient user interfaces for administering a MySQL server.

**Full-text search:** Full-text search simplifies and accelerates the search for words that are located within a text field. If you employ MySQL for storing text (such as in an Internet discussion group), you can use full-text search to implement simply an efficient search function.

**Replication:** Replication allows the contents of a database to be copied (replicated) onto a number of computers. In practice, this is done for two reasons: to increase protection against system failure (so that if one computer goes down, another can be put into service) and to improve the speed of database queries.

**Transactions:** In the context of a database system, a transaction means the execution of several database operations as a block. The database system ensures that either all of the operations are correctly executed or none of them. This holds even if in the middle of a transaction there is a power failure, the computer crashes, or some other disaster occurs. Thus, for example, it cannot occur that a sum of money is withdrawn from account A but fails to be deposited in account B due to some type of system error.

Transactions also give programmers the possibility of interrupting a series of already executed commands (a sort of revocation). In many situations this leads to a considerable simplification of the programming process. In spite of popular opinion, MySQL has supported transactions for a long time. One should note here that MySQL can store tables in a variety of formats. The default table format is called MyISAM, and this format does not support transactions. But there are a number of additional formats that do support transactions. The most popular of these is InnoDB, which will be described extensively in this book.

**Foreign key constraints:** These are rules that ensure that there are no cross references in linked tables that lead to nowhere. MySQL supports foreign key constraints for InnoDB tables.

**GIS functions:** Since version 4.1, MySQL has supported the storing and processing of two-dimensional geographical data. Thus MySQL is well suited for GIS (geographic information systems) applications.

**Programming languages:** There are quite a number of APIs (application programming interfaces) and libraries for the development of MySQL applications. For client programming you can use, among others, the languages C, C++, Java, Perl, PHP, Python, and Tcl.

**ODBC:** MySQL supports the ODBC interface [Connector/ODBC](http://searchenterpriselinux.techtarget.com/definition/MySQL-Connector-ODBC). This allows MySQL to be addressed by all the usual programming languages that run under Microsoft Windows (Delphi, Visual Basic, etc.). The ODBC interface can also be implemented under Unix, though that is seldom necessary.

Windows programmers who have migrated to Microsoft's new .NET platform can, if they wish, use the ODBC provider or the .NET interface Connector/NET.

**Platform independence:** It is not only client applications that run under a variety of operating systems; MySQL itself (that is, the server) can be executed under a number of operating systems. The most important are Apple Macintosh OS X, Linux, Microsoft Windows, and the countless Unix variants, such as AIX, BSDI, FreeBSD, HP-UX, OpenBSD, Net BSD, SGI Iris, and Sun Solaris.

**Speed:** MySQL is considered a very fast database program. This speed has been backed up by a large number of benchmark.

## ide for Database

### MySQL workbench

MySQL Workbench is a visual database design tool that integrates SQL evelopment,administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system. It is the successor to DBDesigner 4 from fabFORCE.net, and replaces the previous package of software,MySQL GUI Tools Bundle.[MySQL Workbench](http://www.mysql.com/products/workbench/) enables a DBA, developer, or data architect to visually design, generate, and manage all types of databases including Web, OLTP, and data warehouse databases. It includes everything a data modeler needs for creating complex ER models, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort. MySQL Workbench is available on Windows, Linux and Mac OS.

#### benefits

* Simplifies database design and maintenance
* Automates time-consuming and error-prone tasks
* Enables data architects to visualize requirements, communicate with stakeholders, and resolve design issues before a major investment of time and resources is made
* Enables model-driven database design—the most efficient methodology for creating valid and well-performing databases—while providing the flexibility to respond to evolving business requirements
* Provides capabilities to forward-engineer physical database designs and reverse-engineer existing databases
* Allows you to import SQL scripts to build models and export models to DDL scripts that can be run at a later time
* Enables you to compare two live databases or a model and a live database, visually see the differences, and perform a synchronization between a model and a live database or vice versa
* Simplifies the documentation of database designs, providing a point-and-click process that delivers documentation in HTML or plain-text format

#### tools

The three main tools of MySQL Workbench are:

* SQL Development
* Data Modelling
* Server Administration

## Programming Language

### C# - C sharp



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](http://en.wikipedia.org/wiki/Multi-paradigm_programming_language) encompassing [imperative](http://en.wikipedia.org/wiki/Imperative_programming), [declarative](http://en.wikipedia.org/wiki/Declarative_programming), [functional](http://en.wikipedia.org/wiki/Functional_programming), [generic](http://en.wikipedia.org/wiki/Generic_programming), [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming)([class-based](http://en.wikipedia.org/wiki/Class_(computer_science))), and [component-oriented](http://en.wikipedia.org/wiki/Component-based_software_engineering) programming disciplines. It was developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) within the [.NET](http://en.wikipedia.org/wiki/.NET_Framework) initiative and later approved as a standard by [Ecma](http://en.wikipedia.org/wiki/Ecma_International) (ECMA-334) and [ISO](http://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO/IEC 23270). C# is one of the programming languages designed for the [Common Language Infrastructure](http://en.wikipedia.org/wiki/Common_Language_Infrastructure).

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

#### Characteristics of C#:

C# was developed to bring rapid development to C++ without sacrificing the power and control of C and C++. C# provides various characteristics, which are:  
Simple:  
C# eliminates the use of tedious operators such as -->, and pointers. C# treats inter and Boolean as two different data types, which enable the compiler   
to recognize the use of = in place of = = with if statement.  
  
**Consistent:-**  
C# supports only one integer type and there is no limitation of range.  
**Modern:-**  
C# contains various features necessary to develop web applications. Following are the features of C#:  
It provides automatic garbage collection.  
It provides robust security model.  
It provides decimal data type for financial application.  
It provides modern approach for debugging.  
It provides a rich intrinsic model for error handling.  
**Object Oriented:-**  
C# supports all the features of object oriented language such as encapsulation, inheritance and polymorphism. It treats everything as an object and there are no global   
functions, variables and constants in C#.  
**Type Safe:-**  
C# provides various type safe measures, which are:   
Dynamically allocated objects and arrays are initialized to zero.  
Products an error message while using an uninitialized variable.  
Checks the range of an array and warns when the access goes out of bound.  
Unsafe casts are not allowed.  
Enforces overflow checking in arithmetic operations.  
**Versionable:-**  
C# supports versioning that enables the existing applications to run on different versions with the help of new and override command.  
Compatible:  
C# contains the .NET specifications and therefore, allows inter operation with other .NET languages.  
**Flexible:-**  
C# does not support pointers but you may use pointers to manipulate the data of certain classes and methods by declaring them unsafe.  
Inter-operability:  
C# enables a program to call out any native API. It also allows the use of COM objects written in different languages.

## Dia for Diagram Drawing &Modeling

Dia is free and open source general-purpose diagramming software, developed as part of the GNOME project's office suite and was originally created by Alexander Larsson. Dia uses a controlled single document interface (CSDI) similar to GIMP and Sodipodi.

Dia has a modular design with several shape packages available for different needs: flowchart, network diagrams, circuit diagrams, and more. It does not restrict symbols and connectors from various categories from being placed together.

Dia is a gtk+ based diagram creation program released under the GPL license.

Dia is inspired by the commercial Windows program 'Visio', though more geared towards informal diagrams for casual use. It can be used to draw many different kinds of diagrams. It currently has special objects to help draw entity relationship diagrams, UML diagrams, flowcharts, network diagrams, and many other diagrams. It is also possible to add support for new shapes by writing simple XML files, using a subset of SVG to draw the shape.

It can load and save diagrams to a custom XML format (gzipped by default, to save space), can export diagrams to a number of formats, including EPS, SVG, XFIG, WMF and PNG, and can print diagrams (including ones that span multiple pages).

## Google Spreadsheet Interface:

*With Google Spreadsheets, we can easily create, share, and edit spreadsheets online. Here are a few specific things we can do:*

* *Import and export these file types: .xls, .csv, .txt and .ods. We can also export data to a PDF or an HTML file.*
* *Format cells and edit formulas so we can calculate results and make data look the way we want it.*
* *Chat in real time with others who are editing our spreadsheet.*
* *Embed a spreadsheet, or a section of a spreadsheet, in our blog or website.*

## Cacoo::online drawing tool

 Cacoo is a diagram creation tool that runs in your web browser.Multiple people can work together on the same diagram in real time.Diagrams can be published directly to websites, wikis, and blogs.

### Creating Diagrams

* Elements can be dragged and drop to easily create diagrams.
* Elements can be linked together with connectors.
* Connectors automatically move when elements are repositioned.
* You can use a text box and put text anywhere you like.
* You can upload images from your PC and include them in Diagrams.
* You can take screenshots of your computer from within Cacoo.
* Smart styles can easily be applied to stencils.
* You can have multiple sheets in a diagram and use them as backgrounds or layers.
* When you move the objects on your canvas, they will be snapped at the objects or grids nearby and align automatically.
* Copying, pasting and other functionality of basic drawing software is also built in to Cacoo.
* All actions are stored so there are unlimited levels of undo.
* You can import an image from the other websites by indicating the URL.
* The imported image can be easily trimmed only using your mouse.
* According to your editing status, tips will be shown on the right bottom corner of the canvas.

### Collaboration

* You can invite collaborators to work with you in Cacoo.
* Multiple people can edit a diagram in real time.
* There is a chat function in the editor so people can communicate while creating diagrams.
* People can leave comments about the diagrams.
* Each user can set their own user icon.
* When editing with multiple people, users icons appear on selected objects.
* Sharing diagrams become much smoother. Diagrams in the shared folders can be accessible and editable by people who you have shared the folder with.

### Sharing Diagrams

* If you keep the diagram private then other users can't see it.
* If you make the diagram URL public, then anyone who knows the URL can see it.
* Publishing a diagram to a blog can be useful in various ways.
* You can place code into blogs to create a slideshow
* Published images always display the most recent version.
* Diagrams can be exported to SVG format (Plus Plan users only) and PNG format. (More formats will be available in the future.)
* Diagrams can be posted to Twitter/Facebook/GoogleBuzz
* Diagrams can be displayed in SVG format for printing. (Plus Plan users only. A few browsers are not supported.)

### Managing Diagrams

* Diagrams can be placed into folders.
* Diagrams can be copied.
* Diagrams can be displayed as thumbnails or as a list.

### Languages and Time Zones

* All pages and notification e-mails support English and Japanese
* Users can enter text from almost all languages.
* Dates are displayed relative to your local time zone.

### Security

* Private diagrams can only be seen by users you select.
* URLs which you do not share can not be found by other users or search engines.
* All editing and management is protected by SSL.
* In order to access information about diagrams a Cacoo ID and password are requited.
* User passwords are encrypted on Cacoo's server.

### API

* You can access Cacoo using the API.
* The Cacoo API supports OAuth and an API Key.

By using the Cacoo API you are able to interact with Cacoo from other services and applications.

Authorization Methods

There are two ways to access the Cacoo API.

#### 1. API Key

The API key allows you make requests to the Cacoo API. You can make an API key here.

#### API Key

Append your API key to requests to the API to return data from your account.(Parameter name "apiKey")

Example: https://cacoo.com/api/v1/diagrams.json?apiKey=abcdefghijklmn

#### 2. OAuth

OAuth 1.0a is supported as an authorization method for Cacoo. You can register applications here.

You can get your Access Token from the following links.

#### applications

Access Token:https://cacoo.com/oauth/access\_token

Authorize:https://cacoo.com/oauth/authorize

Request Token:https://cacoo.com/oauth/request\_token

## http://t1.gstatic.com/images?q=tbn:ANd9GcS-CmbHGLD4MH83JH1oNIr_acREqblVhrcFuvQfYZR8HFi1UpaqlgVersion Control System :GitHub

GitHub is a web-based hosting service for software development projects that use the Git revision control system. GitHub offers both paid plans for private repositories, and free accounts for open source projects. As of May 2011, GitHub was the most popular open source code repository site.GitHub Inc. was founded in 2008 and is based in San Francisco, California.

### Description

The site provides social networking functionality such as feeds, followers and the network graph to display how developers work on their versions of a repository.

GitHub also operates other services: a pastebin-style site called Gist that provides wikis for individual repositories and web pages that can be edited through a Git repository, a slide hosting service called Speaker Deck, and a web analytics platform called Gauges.

As of January 2010, GitHub is operated under the name GitHub, Inc.

The software that runs GitHub was written using Ruby on Rails and Erlang by GitHub, Inc. (previously known as Logical Awesome) developers Chris Wanstrath, PJ Hyett, and Tom Preston-Werner.

### Limitations and constraints

According to the terms of service,if an account's bandwidth usage significantly exceeds the average of other GitHub customers, the account's file hosting service may be immediately disabled or throttled until bandwidth consumption is reduced. In addition, while there is no hard limit, the guideline for the maximum size of a repository is one gigabyte.

# Glossary