|  |
| --- |
|  |
| **Micro Finance Management System** |
| Synopsis |
|  |
| **Shovan Saha** |
|  |

|  |
| --- |
|  |

Contents

[**Micro Finance Management System** 0](#_Toc351660798)

[1. Introduction And Objective 4](#_Toc351660799)

[1.1 Introduction 4](#_Toc351660800)

[1.2 Objective 4](#_Toc351660801)

[2. System analysis 5](#_Toc351660802)

[2.1 identification of need 5](#_Toc351660803)

[2.2 Preliminary Investigation 6](#_Toc351660804)

[2.3 Feasibility Study 6](#_Toc351660805)

[2.4 Project Planning & scheduling 6](#_Toc351660806)

[2.4.1 Gantt chart 6](#_Toc351660807)

[2.4.2 Tracking Gantt 7](#_Toc351660808)

[2.4.3 Pert chart (Network Diagram) 8](#_Toc351660809)

[2.5 software requirment SPECIFICATIONS (srs) 9](#_Toc351660810)

[2.5.1 Functional Requirements 9](#_Toc351660811)

[2.5.1.1 Add a new client for loan or new scheme 9](#_Toc351660812)

[2.5.1.2 monitor accounts details 9](#_Toc351660813)

[2.5.1.3 generate bill 10](#_Toc351660814)

[2.5.1.4 generate monthly report 10](#_Toc351660815)

[2.5.1.5 User login with different authentication level 11](#_Toc351660816)

[2.5.1.6 Business details through mobile applicaiton 11](#_Toc351660817)

[2.5.1.7 Remainder of incaome/expense 11](#_Toc351660818)

[2.5.2 Non-functional Requirements 12](#_Toc351660819)

[2.6 Hardware And Software Specification 13](#_Toc351660820)

[2.6.1 Hardware Requirement 13](#_Toc351660821)

[2.6.2 Software Requirement 13](#_Toc351660822)

[2.7 Tools/Platforms used (Hardware/Software): 13](#_Toc351660823)

[2.7.1 Hardware Used 13](#_Toc351660824)

[2.7.2 Software Used 13](#_Toc351660825)

[2.8 Software Engineering Paradigm applied 14](#_Toc351660826)

[2.9 data modeling 14](#_Toc351660827)

[2.7.1 Context Diagram 14](#_Toc351660828)

[2.7.2 Level 0 DFD 15](#_Toc351660829)

[2.7.3 Level 1 DFD 15](#_Toc351660830)

[2.7.4 Level 2 DFD 16](#_Toc351660831)

[2.10 Control Flow diagrams 17](#_Toc351660832)

[2.11 State Diagrams/Sequence diagrams 17](#_Toc351660833)

[2.12 Entity Relationship Model, 17](#_Toc351660834)

[2.13 Class Diagrams/CRC Models/Collaboration Diagrams/Use-case Diagrams/ 20](#_Toc351660835)

[2.13.1 Class Diagram 21](#_Toc351660836)

[2.13.2 Activity Diagrams 22](#_Toc351660837)

[3. System Design 23](#_Toc351660838)

[3.1 Modularisation details 23](#_Toc351660839)

[3.2 Data integrity and constraints 24](#_Toc351660840)

[3.2.1 Entity integrity 24](#_Toc351660841)

[3.2.2 Referential Integrity 24](#_Toc351660842)

[3.2.3 Domain Integrity 24](#_Toc351660843)

[3.2.4 User Defined Integrity 25](#_Toc351660844)

[3.3 Database design, Procedural Design/Object Oriented Design 25](#_Toc351660845)

[3.4 User Interface Design 31](#_Toc351660846)

[3.5 Test Cases (Unit Test Cases and System Test Cases) 31](#_Toc351660847)

[3.5.1 UNIT TEST CASES 31](#_Toc351660848)

[3.5.2 SYSTEM TEST CASES 31](#_Toc351660849)

[4. Coding 33](#_Toc351660850)

[4.1 Complete Project Coding 33](#_Toc351660851)

[4.2 Comments and Description of Coding segments 33](#_Toc351660852)

[4.3 Standardization of the coding 34](#_Toc351660853)

[4.4 Code Efficiency 34](#_Toc351660854)

[4.5 Error handling 34](#_Toc351660855)

[4.6 Parameters calling/passing 34](#_Toc351660856)

[4.7 Validation checks 34](#_Toc351660857)

[5. Testing 35](#_Toc351660858)

[5.1 Testing techniques and Testing strategies used 35](#_Toc351660859)

[5.2 Testing Plan used 35](#_Toc351660860)

[5.3 Test reports for Unit Test Cases and System Test Cases 35](#_Toc351660861)

[5.3.1 UNIT TEST CASES 35](#_Toc351660862)

[5.3.2 SYSTEM TEST CASES 35](#_Toc351660863)

[5.4 Debugging and Code improvement 36](#_Toc351660864)

[6. System Security measures 36](#_Toc351660865)

[6.1 Database/data security 37](#_Toc351660866)

[6.2 Creation of User profiles and access rights 37](#_Toc351660867)

[7. Cost Estimation of the Project along with Cost Estimation Model 37](#_Toc351660868)

[7.1 Estimation of development effort 38](#_Toc351660869)

[7.2 Estimation of development time 38](#_Toc351660870)

[8. Reports 39](#_Toc351660871)

[9. Future scope and further enhancement of the Project 39](#_Toc351660872)

[10. Bibliography 40](#_Toc351660873)

[2. Tools/Platform, Hardware and Software Specification 40](#_Toc351660874)

[3. Requirements And Analysis 40](#_Toc351660875)

[3.1 Problem Definition 40](#_Toc351660876)

[3.1.1 Existing System 40](#_Toc351660877)

[3.1.2 WORK To Be Done 40](#_Toc351660878)

[3.2 Requirements Specification 41](#_Toc351660879)

[3.3 Technical specification 41](#_Toc351660880)

[4. Scope Of The Solution 42](#_Toc351660881)

[4.1 E-R Diagram 42](#_Toc351660882)

[5. Database And Table Details 42](#_Toc351660883)

[6. Complete Data Structure 42](#_Toc351660884)

[6.1 Module Description 42](#_Toc351660885)

[6.2 Estimation 42](#_Toc351660886)

[6.3 Data Structure 43](#_Toc351660887)

[6.4 Implementation Methodology 43](#_Toc351660888)

[6.5 List of Reports 44](#_Toc351660889)

[7. Security Mechanism 44](#_Toc351660890)

[8. Future Scope And Further Requirements 44](#_Toc351660891)

# Introduction And Objective

## Introduction

Microfinance is considered to be effective tool in alleviating poverty by increasing income of poor households and reducing their vulnerabilities. Today, when the majority of the world’s population is living below subsistence level, many organizations are providing microfinance services to millions of the-world’s poor. Yet most of the poor still have little or no access to financial services. Microfinance institutions (MFIs) have reached a lot of poor people. The task of reaching such a big number is a major challenge. The microfinance posed a big challenge as the existing MFI did not have any functionality related to deposits, remittances and insurance that the organization envisaged to Offer. The challenge will be removed through the following strategy:

* Establishment of a strong IT department
* Short-term solution that involved integration of an off the-shelf application with its existing portfolio management system.
* Long-term plan to build an enterprise MIS with the functionality to deliver all banking and microfinance services, and flexible enough to meet the current and future technology requirements of the bank.

As a result of its strong and flexible computerized system MFI will be able to introduce a wide range of products and services, including loans, deposits, remittances and insurance. The system is used to obtain maximum information to support timely decisions.

## Objective

Micro Finance Management System is versatile and complete end-to-end Micro Finance Management software .Micro Finance Management System is used to enhance the administrative efficiency of educational institutions. It is an interactive platform for all entities viz. Students, Teachers, Management, Parents. It is a simple yet powerful one point integrated platform that connects all the departments of an institution namely office, fee counter, library, hostel, stores, academics, activity center and so on.

# System analysis

## identification of need

The microfinance sector is quite diverse in its use of information systems. But there are lots of disadvantages using existing systems like manual system or semi-automated systems.

1. **Manual System:**

Some MFIs still rely on manual systems, which involve maintenance of records in forms and ledgers. Organizations having manual systems are either small micro-credit programs or NGOs.

1. **Semi-automated System:**

More than 50% of MFIs are operating in a semi automated mode. Within this category, the spreadsheet is the common tool being used either in conjunction with a manual system or with an MIS application that does not fulfill the information requirements of the MFI. The majority of non-regulated MFIs have semi automated systems.

**Disadvantage of manual systems:**

Some of the disadvantages of manual Information Systems are:

* Too laborious and time consuming.
* Prone to Errors.
* Data manipulation and analysis is very difficult.
* Maintenance of large amount of data is almost impossible.
* Data and information is not secured.
* Loosely controlled.
* Highly inflexible (addition of new products and change in business processes cannot be made).
* Business continuity is at risk in case of damage to information due to fire, water or any other disaster.
* Reporting is very cumbersome, time consuming and difficult.

From this I felt that we need a solution that could minimize the drawbacks of existing system.

## Preliminary Investigation

I started talking to my relatives and neighbors who have recently invested money in some microfinance organization or they are in the process of investment. Naturally, they are the ones who know the problems very well. I also spoke with some of microfinance organization administrations in my locality about the problems they faced and what exactly would they want if they are given software like Microfinance Management System just to get an idea on what could the points be if we want to develop a new application to minimize microfinance business managing related problems and difficulties. I obviously then consulted with some IT professionals and software engineers and expressed my view to them. I was really amazed to see their positive response on this matter and I got helps in many ways from them. After completing all this process I decided to develop this application for sure

## Feasibility Study

We all know that the microfinance industry is growing as fast as Information Technology. Developing an application is very easy these days so people would love to use technology ease their tasks. There is need and there is solution so undoubtedly this software is going to be appreciated by the market.

## Project Planning & scheduling

### Gantt chart



### Tracking Gantt



### Pert chart (Network Diagram)



## software requirment SPECIFICATIONS (srs)

### 2.5.1 Functional Requirements

#### 2.5.1.1 Add a new client for loan or new scheme

**Introduction**

The system should keep all the records of the various clients with their details. User should be able to get those information whenever is needed.

**Input**

Relevant client details like name, address, contact no., scheme type should be provided to the system.

**Processing**

After getting relevant data the system must automatically generate a unique id for the client and save all the details in the database.

**Output**

MFMS generates the required details and save them in the database. As output they are shown to the user.

#### 2.5.1.2 monitor accounts details

**Introduction**

User should be able to see all types of accounts activities like income and expense along with the details of them.

**Input**

Salaries are paid to the employees using MFMS system and income information should be added to the accounts as well.

**Processing**

System automatically calculates the total income and expense and computes the final account balance and profit loss.

**Output**

All the accounts related details and calculations are saved in database along with the loss/profit details and provided to user as output whenever is asked.

#### 2.5.1.3 generate bill

**Introduction**

MFMS should provide a bill printing option on payment by the clients.

**Input**

User provides all the bill details in the billing GUI of the software.

**Processing**

System calculates the amount total, fetches customer details using his using id and merges them all in a single page for printing.

**Output**

As output a bill gets printed that contains all the details of the transaction.

#### 2.5.1.4 generate monthly report

Introduction

System must monitor monthly business details and provide a monthly/ weekly/ daily report to the user.

Input

User uses the application regularly for business. Generates bill, pays salary etc.

Processing

System tracks all the income, expense and other details of the company and gathers them into a single sheet for generating a report.

Output

User gets a business report whenever he wants and can track the progress of his business.

#### 2.5.1.5 User login with different authentication level

Introduction

The system should be secure by a password. It should provide a login window to the user and according to his job role he can login as admin, clerk, officer etc.

Input

Predefined username and password and account type is provided to the system.

Processing

System checks the login database, matches the password and replies to the user accordingly.

Output

After providing a valid username and password a user can access the contents of the software as per his permission level.

#### 2.5.1.6 Business details through mobile applicaiton

Introduction

A mobile application of the software should be developed to provide user his business details via mobile even when he is away from office.

Input

User provides predefined password in the mobile application installed in his java and internet enabled mobile.

Processing

As per user requirement, the mobile application searches for the relevant data inside the cloud based database for showing to the user.

Output

User gets to see his business details from the mobile application.

#### 2.5.1.7 Remainder of incaome/expense

**Introduction**

The system should automatically provide a remainder to the user whenever a client’s deadline is close enough for paying money to the company.

**Input**

**Processing**

System keeps track of each employee’s payment deadlines regularly and notifies the user when the date is close enough.

**Output**

Without providing any input, the user automatically gets the deadline notification.

### 2.5.2 Non-functional Requirements

* The application will be **self-dependent** and no dependency on other parties required.
* There will be a digital **backup** and restore system.
* There will be more **opportunity** to extend the application in future.
* The response time will be low and the system will **response** fast.
* It will be very **user friendly** and **usable** by any person with minimal computer knowledge.
* In terms of **security** unauthorized access will be denied and register user will be able to change as necessary.
* It will be **efficient** as it reduces manual labor and searching.
* **MMS** will have user manual and help **documents**.
* It is designed such a way that it can be **maintained** with minimal effort.

## Software Engineering Paradigm applied

## data modeling

### 2.7.1 Context Diagram



### 2.7.2 Level 0 DFD



### 2.7.3 Level 1 DFD





### 2.7.4 Level 2 DFD



## Control Flow diagrams

## State Diagrams/Sequence diagrams

## Entity Relationship Model,

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

|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| Employees | **employeeId**, employeeName, employeeAddress, employeeJoinDate, employeeQualification, employeeContactDetails, employeeContactNumber, employeeEmail, employeeDOB |
| Clients | **clientId**, clientName, clientEmail, clientContactNumber, clientAddress, clientType, businessDetails |
| Accounts | **AmountId,** amountDescription**,** amountType, amountValue, accountBalance |
| Schemes | **schemeId,** schemeName, schemeDescription, schemeType, schemeDuration |
| Loans | **loanId,** loanName, loanDescription, loanType, loanDuration |

**Relationship between Entities:**

* Micro Finance Management System has employees 1 : N
* Micro Finance Management System has clients 1 : N
* Micro Finance Management System manages accounts 1 : 1
* Micro Finance Management System has schemes 1 : N
* Micro Finance Management provides loans 1 : N
* employees receive from accounts N:1
* clients pay to accounts N:1
* clients take loans N:N
* clients take schemes N:N

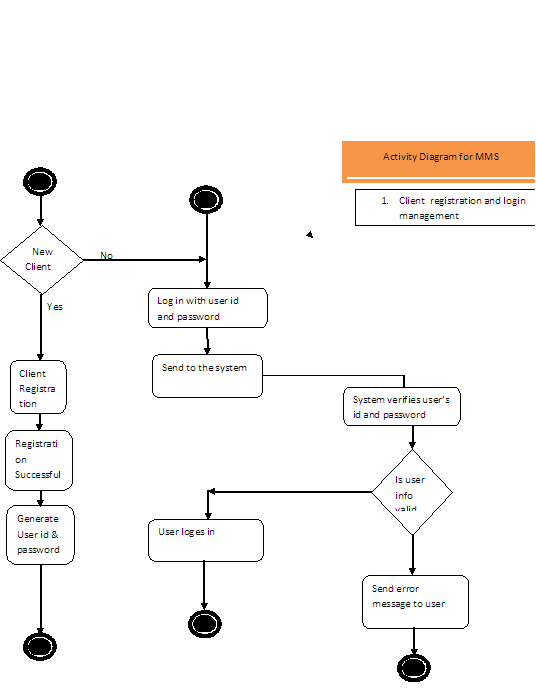


## Class Diagrams/CRC Models/Collaboration Diagrams/Use-case Diagrams/

### Class Diagram



### Activity Diagrams



# System Design

## Modularisation details

All the modules of Micro Finance Management System are divided into 5 different modules. The above picture represents those modules and details of the modules are written below:

**MFMS Engine:** It controls all the logical parts of microfinance management system. It controls the entire system. Takes the data from one module to another and does all the mathematical calculations as well.

**MFMS GUI:** This module works as the bridge between the application and the user. User provides input through it and gets required output through it. To be more precise, it is the graphical representation of the application.

****

**MFMS Storage:** This module holds all the data provided by user as input. The MFMS engine takes the relevant data from different modules and sends them to this module for storing. The MFMS Storage returns it whenever a user asks for a saved data.

**MFMS Web Application:** This module controls the entire process of the desktop application’s online data storage facility. It takes relevant input from user through the GUI; process them according to the instruction of the user and finally send them to the cloud based database.

**MFMS Cloud Storage:** This module acts as an online storage for the data of MFMS. It gets input data to be stored from MFMS Web Application and returns the relevant output to hit whenever is asked. It also sends the data to the mobile application module.

**MFMS Mobile Application:** This module shows the data stored in the cloud based database through a mobile GUI to the user. It gets the input from user via MFMS engine, takes relevant output from online storage and provides them to the user via mobile app’s GUI.

## Data integrity and constraints

We have used Integrity constraints in **MMS** 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 **MMS** that play a role in referential integrity.

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

In **MMS** 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

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

**MMS** 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.

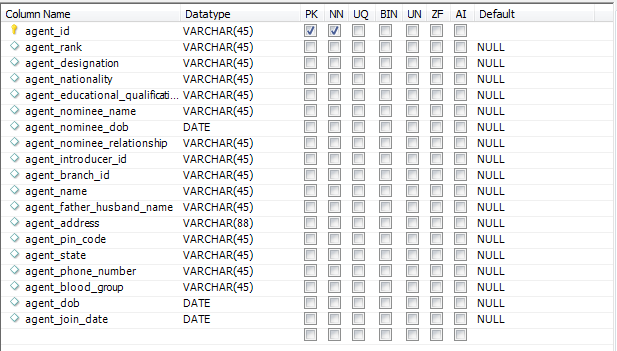
### User Defined Integrity

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.

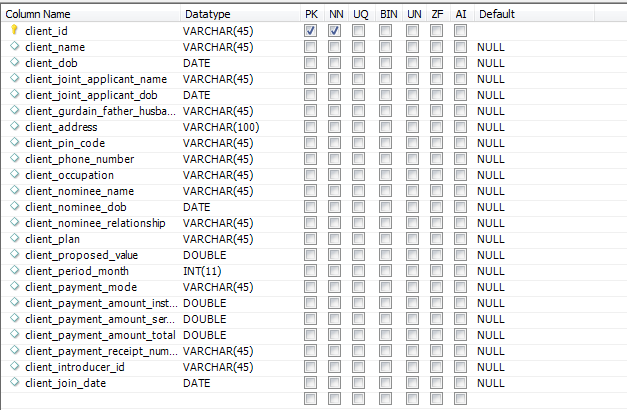
## Database design, Procedural Design/Object Oriented Design

The database used for this software is called **mfmsdb**. Database tables and corresponding keys are shown in tabular form. It shows the tables and its columns. A key in **Bold** is the primary key.

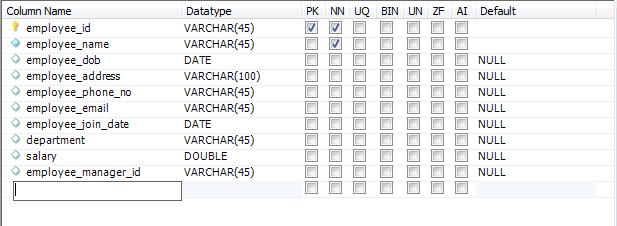
**Agent Table**



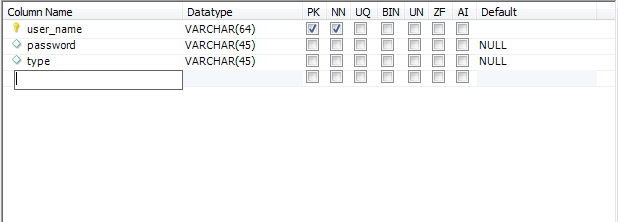
**Client Table**



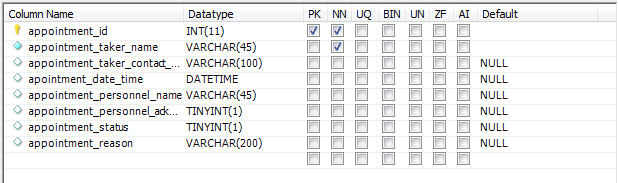
**Employee Table**



**Login Details Table**



**Appointment Table**



|  |  |
| --- | --- |
| **Entities** | **Attributes** |
| Employees | **employeeId**, employeeName, employeeAddress, employeeJoinDate, employeeQualification, employeeContactDetails, employeeContactNumber, employeeEmail, employeeDOB |
| Clients | **clientId**, clientName, clientEmail, clientContactNumber, clientAddress, clientType, businessDetails |
| Accounts | **AmountId,** amountDescription**,** amountType, amountValue, accountBalance |
| Schemes | **schemeId,** schemeName, schemeDescription, schemeType, schemeDuration |
| Loans | **loanId,** loanName, loanDescription, loanType, loanDuration |

|  |
| --- |
| **EmployeeDetails** |
| public class EmployeeDetails  {  public string employeeId { get; set; }  public string employeeName { get; set; }  public DateTime employeeDob { get; set; }  public string employeeAddress { get; set; }  public string employeePhoneNumber { get; set; }  public string employeeEmail { get; set; }  public DateTime employeeJoinDate { get; set; }  public string employeeDepartment { get; set; }  public double employeeSalary { get; set; }  public string employeeManagerId { get; set; }  } |

|  |
| --- |
| AccountInfo |
| public class AccountInfo  {  public string id { get; set; }  public ClientDetails client;  public double balance { get; set; }  public List<AmountInfo> amounts { get; set; }  } |

|  |
| --- |
| SchemeInfo |
| public class SchemeInfo  {  public string id { get; set; }  public string name { get; set; }  public string Type { get; set; }  public DateTime Duration { get; set; }  public DateTime dateOfStart { get; set; }  public string description { get; set; }  public AmountInfo amountPremium { get; set; }  public double incentivePercentage;  } |

|  |
| --- |
| LoanInfo |
| public class LoanInfo  {  public string id { get; set; }  public string name { get; set; }  public string Type { get; set; }  public DateTime Duration { get; set; }  public DateTime dateOfStart { get; set; }  public string description { get; set; }  public AmountInfo amountPrincipal { get; set; }  public double interestPercentage;  } |

|  |
| --- |
| ViewOrEdit |
| public enum ViewOrEdit  {  View,  Edit  } |

|  |
| --- |
| AmountType |
| public enum AmountType  {  Debit,  Credit,  Overdraft  } |

|  |
| --- |
| AmountInfo |
| public class AmountInfo  {  public string id { get; set; }  public double amount { get; set; }  public string description { get; set; }  public AmountType type { get; set; }  } |

|  |
| --- |
| AccountInfo |
| public class AccountInfo  {  public string id { get; set; }  public ClientDetails client;  public double balance { get; set; }  public List<AmountInfo> amounts { get; set; }  } |

|  |
| --- |
| SchemeController |
| public class SchemeController  {  public List<SchemeInfo> schemes;  } |

|  |
| --- |
| LoanController |
| public class LoanController  {  public List<LoanInfo> loans;  } |

|  |
| --- |
| ClientController |
| public class ClientController  {  public List<ClientDetails> clients;  } |

|  |
| --- |
| AgentsController |
| public class AgentsController  {  public List<AgentDetails> agents;  } |

|  |
| --- |
| AcountsController |
| public class AcountsController  {  public List<AccountInfo> accounts;  } |

|  |
| --- |
| MFMSController |
| public class MFMSController  {  public AgentsController agentsController;  public ClientController clientController;  public SchemeController SchemeController;  public LoanController loanController;  public AcountsController AcountsController;  } |

|  |
| --- |
| UserTypeEnum |
| public enum UserTypeEnum  {  user,  manager,  owner  } |

|  |
| --- |
| AgentDetails |
| public class AgentDetails  {  public string agentId { get; set; }  public string agentTitle { get; set; }  public string agentName { get; set; }  public DateTime joinDate { get; set; }  public int rank { get; set; }  public string designation { get; set; }  public string agentFatherHusbandName { get; set; }  public string agentAddress { get; set; }  public int agentPinCode { get; set; }  public string agentState { get; set; }  public string agentPhoneNumber { get; set; }  public string agentBloodGroup { get; set; }  public DateTime agentDateOfBirth { get; set; }  public string agentNationality { get; set; }  public string agentQualification { get; set; }  public string agentNomineeName { get; set; }  public DateTime agentNomineeDob { get; set; }  public string agentNomineeRelationship { get; set; }  public string agentIntroducerId { get; set; }  public string agentBranchId { get; set; }  public List<ClientDetails> clients;  } |

|  |
| --- |
| LoginData |
| public class LoginData  {  public string userName;  public UserTypeEnum type;  public string password;  } |

|  |
| --- |
| ClientDetails |
| public class ClientDetails  {  public string clientId { get; set; }  public string clientName { get; set; }  public DateTime joinDate { get; set; }  public DateTime clientDateOfBirth { get; set; }  public string clientJointApplicantName { get; set; }  public DateTime clientJointApplicantDateOfBirth { get; set; }  public string clientGuardianFatherName { get; set; }  public string clientAddress { get; set; }  public int clientPinCode { get; set; }  public string clientPhoneNumber { get; set; }  public string clientOccupation { get; set; }  public string clientNomineeName { get; set; }  public DateTime clientNomineeDateOfBirth { get; set; }  public string clientNomineeRelationship { get; set; }  public string clientPlan { get; set; }  public double clientProposedValue { get; set; }  public int clientPeriodMonth { get; set; }  public string clientPaymentMode { get; set; }  public double clientPaymentAmountInstallment { get; set; }  public double clientPaymentAmountServiceCharge { get; set; }  public double clientPaymentAmountTotal { get; set; }  public string clientPaymentReciptNumber { get; set; }  public string clientIntroducerId { get; set; }  public List<SchemeInfo> schemes;  public List<LoanInfo> loans;  public List<AmountInfo> payments;  } |

## User Interface Design

## Test Cases (Unit Test Cases and System Test Cases)

### UNIT TEST CASES

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Type | Github ID | Subject | Test Name | Test Description | Step Name | Description | Expected Result |
| **MMS**-001 | Manual | 6d7342686f5117712e6a15885cd54a075c5013a0 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check Successful Login for **MMS** | The purpose of this test is to verify whether login feature is working or not. | Step1 | Insert wrong User Id and Password. And Click on Login Button. | **MMS** will display error message. And Failed to Login. |
| **MMS**-002 |  |  |  |  |  | Step2 | Insert Wrong User Id and valid Password. And Click on Login Button. | **MMS** will display error message. And Failed to Login. |
| **MMS**-003 |  |  |  |  |  | Step3 | Insert Valid User Id and Wrong Password. And Click on Login Button. | **MMS** will display error message. And Failed to Login. |
| **MMS**-004 |  |  |  |  |  | Step4 | Insert Nothing in User Id and Password fields. And Click on Login Button. | **MMS** will display error message. And Failed to Login. |
| **MMS**-005 |  |  |  |  |  | Step5 | Insert Nothing in User Id and insert Valid Password fields. And Click on Login Button. | **MMS** will display error message. And Failed to Login. |
| **MMS**-006 |  |  |  |  |  | Step6 | Insert Nothing in Password and insert Valid User Id fields. And Click on Login Button. | **MMS** will display error message. And Failed to Login. |
| **MMS**-007 |  |  |  |  |  | Step7 | Insert Nothing in User Id and insert invalid Password fields. And Click on Login Button. | **MMS** will display error message. And Failed to Login. |
| **MMS**-008 |  |  |  |  |  | Step8 | Insert Nothing in Password and insert invalid User Id fields. And Click on Login Button. | **MMS** will display error message. And Failed to Login. |
| **MMS**-009 |  |  |  |  |  | Step9 | Insert valid User Id and Password. And Click on Login Button. | Successfully login. |
| **MMS**-010 | Manual | 26c285df4a66e3a74228bac1ab23efe1dc2cf497 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check successful client registration to **MMS** | The purpose of this test is to verify that a whether client Registration is happening or not. | Step1 | Click on client link. | New Account creation area is opened. |
| **MMS**-011 |  |  |  |  |  | Step2 | Click on add client button. | Client registration page will be opened. |
| **MMS**-012 |  |  |  |  |  | Step3 | Provide all the data to register new client and press register. | A message will be shown that a new client has been registered. |
| **MMS**-013 |  |  |  |  |  | Step4 | Press register without providing any data | **MMS** will display error message |
| **MMS**-014 |  |  |  |  |  | Step5 | Press register without providing some selected fields. | **MMS** will display error message. |
| **MMS**-015 |  |  |  |  |  | Step6 | Provide invalid date in the date fields and press register | **MMS** will display error message. |
| **MMS**-016 |  |  |  |  |  | Step7 | Provide email address without proper format and press register. | **MMS** will display error message. |
| **MMS**-017 |  |  |  |  |  | Step8 | Provide phone number without proper format. | **MMS** will show error message. |
| **MMS**-018 | Manual | 26c285df4a66e3a74228bac1ab23efe1dc2cf497 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check successful employee details viewing and sorting. | The purpose of this test is to check whether client details are showing or not. | Step1 | Click on view clients’ screen. | Shows all the related data in the list view with proper formatting. |
| **MMS**-019 |  |  |  |  |  | Step2 | Click on the name column in the list view | The window gets sorted alphabetically |
| **MMS**-020 |  |  |  |  |  | Step3 | Click on the name column on the list view again | The window gets sorted alphabetically but in opposite direction |
| **MMS**-021 |  |  |  |  |  | Step4 | Click on id column on the list view | The window gets sorted numerically |
| **MMS**-022 |  |  |  |  |  | Step5 | Click on the id column on the list view again | The window gets sorted numerically but in opposite direction. |
| **MMS**-023 |  |  |  |  |  | Step6 | Click on address column on the list view | The window gets sorted alphabetically. |
| **MMS**-024 |  |  |  |  |  | Step7 | Click on the address column on the list view again | The window gets sorted alphabetically but in opposite direction. |
| **MMS**-025 | Manual | 26c285df4a66e3a74228bac1ab23efe1dc2cf497 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check Successful client editing & deleting | The purpose of this test is to verify whether client details could be modified or not | Step1 | Click on edit client button | A new window gets opened with all the details of clients |
| **MMS**-026 |  |  |  |  |  | Step2 | Select on e client details in the list view and click on edit details button. | A new window gets opened with all the data of the selected row. |
| **MMS**-027 |  |  |  |  |  | Step3 | Provide some new information on the newly opened window and press update. | The window gets closed; a message is shown that says information has been updated. |
| **MMS**-028 |  |  |  |  |  | Step4 | Click on the edit button without selecting any list view details. | A error message is displayed |
| **MMS**-029 |  |  |  |  |  | Step5 | At the edit window, new information are added without providing valid data, like a number in an alphabetical text field. Then press update | A error message is displayed |
| **MMS**-030 |  |  |  |  |  | Step6 | In the clients window, delete button is pressed | A new window gets opened with all the clients details in a list view. |
| **MMS**-031 |  |  |  |  |  | Step7 | Delete information button is clicked without selecting any row in the list view. | A error message is displayed |
| **MMS**-032 |  |  |  |  |  | Step8 | Delete information button is clicked after selecting any row in the list view. | The selected row gets deleted |
| **MMS**-033 | Manual | 6c285df4a66e3a74228bac1ab23efe1dc2cf497 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check Report window functionality | These steps are for checking whether report window is working properly or not. | Step1 | Click on the report window of the application | A new window gets opened with all the valid data with it. |
| **MMS**-034 |  |  |  |  |  | Step2 | Click on the view monthly report button | All the data are shown in moth view |
| **MMS**-035 |  |  |  |  |  | Step3 | Click on the view daily report button | All the data are shown in day view |
| **MMS**-036 |  |  |  |  |  | Step4 | Click on the view yearly report button | All the data are shown in yearly view |
| **MMS**-037 |  |  |  |  |  | Step5 | Click on the view weekly report button | All the data are shown in weekly view |
| **MMS**-038 |  |  |  |  |  | Step6 | Provide valid dates in the date filed and click on show date wise report button | All the data between the provided dates are shown. |
| **MMS**-039 | Manual | b8a65899fa408a4d99a8ee8cabbe2ac0b54226b0 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check sign out feature | These steps are done to check whether the sign out feature are working or not | Step1 | Sign out button is clicked before logging in | It is not available as the user has not signed in yet |
| **MMS**-040 |  |  |  |  |  | Step2 | Sign out button is clicked after signing in | It becomes unavailable again so as all the buttons in the main window except sign in button. |
| **MMS**-041 | Manual | 6c53de9297011864bd6c07bdcce83415a76e8bd4 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check everything about contact information. | It is to check that agent window works properly. | Step1 | Click on agent tab. | Displays all saved data. |
| **MMS**-042 |  |  |  |  |  | Step2 | Click on Add new agent button | A new window gets opened that asks for all the necessary information of an agent |
| **MMS**-043 |  |  |  |  |  | Step3 | Provide all the information and press register. | Successfully Added agent Information. And display in View area. |
| **MMS**-044 |  |  |  |  |  | Step4 | Click on register button without providing details. | Display error message. |
| **MMS**-045 |  |  |  |  |  | Step5 | Provide invalid data as information | Display error message. |
| **MMS**-046 |  |  |  |  |  | Step6 | Provide invalid email address and press register | Display error message. |
| **MMS**-047 |  |  |  |  |  | Step7 | Provide invalid phone number and press register | Display error message. |
| **MMS**-048 | Manual | 90330b92328d862892fc77436539081cc2b7f70d | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check everything about settings information. | It is to check that settings works properly. | Step1 | Click on Password tab. | Password is asking for login password to access all information. |
| **MMS**-049 |  |  |  |  |  | Step2 | Insert wrong User Name and Password. And Click on Login Button. | settings of **MMS** will display error message. And Failed to Login. |
| **MMS**-050 |  |  |  |  |  | Step3 | Insert correct User Name and Password. And Click on Login Button. | Settings login successfully. And display all Password related information. |
| **MMS**-051 |  |  |  |  |  | Step4 | Click on Add Info Expender and enter incorrect data or empty the mandatory field. | Failed to store Info information in database and display the error message. |
| **MMS**-052 |  |  |  |  |  | Step5 | Click on Add Info Expender and enter correct data in the mandatory field. | Successfully Added Info Information. And display in View area. |
| **MMS**-053 |  |  |  |  |  | Step6 | Click on delete button without selecting details. | Display error message. |
| **MMS**-054 |  |  |  |  |  | Step7 | Select an information and Click on Delete button. | Deleted successfully. |
| **MMS**-055 |  |  |  |  |  | Step8 | Click on Edit button without selecting details. | Display error message. |
| **MMS**-056 |  |  |  |  |  | Step9 | Select an information and Click on Edit button. | Expender Opened and asking to edit and click on Save to successfully update data. |
| **MMS**-057 |  |  |  |  |  | Step10 | Click on logout button to logout from password section. | Successfully Logout from Password. |

### SYSTEM TEST CASES

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Type | Github ID | Subject | Test Name | Test Description | Step Name | Description | Expected Result |
| **MMS**-058 | Manual | f3563be0a9c431104f52839039e86043cf640cf1 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check Log in. | It is to check that Login works properly. | Step1 | Click on Login button after inserting invalid User id and password from **MMS**. | Login failed to **MMS**. And can’t able to use the feature. |
| **MMS**-059 |  |  |  |  |  | Step2 | Click on Login button after inserting valid User id and password from **MMS**. | Successfully Login to **MMS**. And can able to use the feature. |
| **MMS**-060 | Manual | d01197ee4cd3bee9245874b5937ba740019fd131 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check Successful Registration for New **MMS** employee. | The purpose of this test is to verify that the all new connection could be creating new Account By Registration. | Step1 | Click on Registration link. | New Account creation area is opened. |
| **MMS**-061 |  |  |  |  |  | Step2 | Click on Registration button after inserting invalid information from **MMS**. | Registration failed to **MMS**. And can’t able to use the feature. |
| **MMS**-062 |  |  |  |  |  | Step3 | Click on Registration button after inserting valid information from **MMS**. | Registration Successfully done to **MMS**. |
| **MMS**-063 |  |  |  |  |  | Step4 | Click on Login button after inserting newly created valid User id and password from **MMS**. | Successfully Login to by new User Id And password **MMS**. And can able to use the feature. |
| **MMS**-064 | Manual | f0657bbdf47e26ec481fa172b0fa76f9becb2681 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check Successful work flow of settings page. | The purpose of this test is to check Successful work flow of settings page. | Step1 | Check the password updation procedure | Working properly |
| **MMS**-065 |  |  |  |  |  | Step2 | Check the user info addition and updation procedure | Working properly |
| **MMS**-066 |  |  |  |  |  | Step3 | Check whether the user information and T & C provided in settings page are coming in the billing window or not | Working properly |
| **MMS**-067 |  |  |  |  |  | Step4 | Check whether the terms and conditions details could be modified or not | Working properly |
| **MMS**-068 | Manual | 47fb570f63ffec837a49e235f629c49cc55a70f0 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check report window work flow | It is to check report window work flow | Step1 | Provide some income and expense information into the application | The report window calculates them and provides profit loss details |
| **MMS**-069 |  |  |  |  |  | Step2 | Provide some income and expense info for a few days back to back | The report window calculates them and provides profit loss details on daily basis |
| **MMS**-070 |  |  |  |  |  | Step3 | Provide some income and expense info for a few weeks back to back | The report window calculates them and provides profit loss details on weekly basis. |
| **MMS**-071 |  |  | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager |  |  | Step4 | Provide some income and expense info for a few days back to back | The date wise report window shows all the info between the provided dates |
| **MMS**-072 | Manual | b8a65899fa408a4d99a8ee8cabbe2ac0b54226b0 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check database security | It is to check that the database password is working properly or not | Step1 | Updated the mysql password and deleted the old one | Shows error as password has been removed |
| **MMS**-073 |  |  |  |  |  | Step2 | Visited the database directly by providing password | As the data are encrypted, information are not visible in a normal language, but in a modified format |
| **MMS**-074 | Manual | 6c53de9297011864bd6c07bdcce83415a76e8bd4 | C:\Users\Amit\Documents\GitHub\MoneyMarkettingManager | Check everything about contact information. | It is to check that contact works properly. | Step1 | Click on Contact tab. Compose contact information and click on Save Button. | Contact display all saved data. And created contact successfully. |
| **MMS**-075 |  |  |  |  |  | Step2 | Select Contact from list view area .And click on Delete Event. | Successfully Deleted Contact from list view area. |

# Coding

## Complete Project Coding

|  |
| --- |
| MainWindow.xaml |
| <Window x:Class="**MMS**Demo.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Money marketting manager" Height="540" Width="1326" WindowState="Maximized" Background="WhiteSmoke">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>    <DockPanel LastChildFill="True">  <Image DockPanel.Dock="Top" Name="companyLogoImage" StretchDirection="DownOnly" Source="/**MMS**Demo;component/Images/technicise.logo-header-medium.png">  <Image.Style>  <Style>  <Style.Triggers>  <EventTrigger RoutedEvent="Image.MouseEnter">  <BeginStoryboard>  <Storyboard>  <DoubleAnimation Storyboard.TargetProperty="(Image.Opacity)"  BeginTime="0:0:0" Duration="0:0:2"  From="1.0" To="0.0" RepeatBehavior="Forever" AutoReverse="True"/>  </Storyboard>  </BeginStoryboard>  </EventTrigger>  </Style.Triggers>  </Style>  </Image.Style>  </Image>  <Image DockPanel.Dock="Top" Name="companyNameImage" StretchDirection="Both" Stretch="Uniform" HorizontalAlignment="Stretch" VerticalAlignment="Top" Source="/**MMS**Demo;component/Images/name.jpg" />  <UniformGrid Width="300" DockPanel.Dock="Left" HorizontalAlignment="Left" Columns="1">  <Button Content="Sign In" Name="loginBtn" Style="{StaticResource MainBtnStyle}" Click="loginBtn\_Click"></Button>  <Button Content="Employees" Name="employeesBtn" Click="employeesBtn\_Click" Style="{StaticResource MainBtnStyle}"></Button>  <Button Content="Agents" Name="agentsBtn" Click="agentsBtn\_Click" Style="{StaticResource MainBtnStyle}"></Button>  <Button Content="Clients" Name="clientsBtn" Click="clientsBtn\_Click" Style="{StaticResource MainBtnStyle}"></Button>  </UniformGrid>  <UniformGrid Width="300" DockPanel.Dock="right" HorizontalAlignment="Right" Columns="1">  <Button Content="Report" Name="reportsBtn" Style="{StaticResource MainBtnStyle}"></Button>  <Button Content="Settings" Name="settingsBtn" Style="{StaticResource MainBtnStyle}"></Button>  <Button Content="Help" Name="helpBtn" Style="{StaticResource MainBtnStyle}"></Button>  <Button Content="Close" Name="closeBtn" Style="{StaticResource MainBtnStyle}" Click="closeBtn\_Click"></Button>  </UniformGrid>  <TabControl Name="controlsTab" DockPanel.Dock="Bottom" TabStripPlacement="Bottom" Background="#37000000" SelectionChanged="controlsTab\_SelectionChanged">  <TabItem Name="homeTab" >  <Image Source="/**MMS**Demo;component/Images/banner1.jpg"></Image>  </TabItem>  <TabItem Name="loginTab">  <UniformGrid Columns="2">  <Label Style="{StaticResource labelStyle}">User Name</Label>  <TextBox Name="userNameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">User Type</Label>  <ComboBox Name="userTypeCombobox" Style="{StaticResource comboboxStyle}">  <ComboBoxItem>User</ComboBoxItem>  <ComboBoxItem>Manager</ComboBoxItem>  <ComboBoxItem>Owner</ComboBoxItem>  </ComboBox>  <Label Style="{StaticResource labelStyle}">Enter Password</Label>  <PasswordBox Name="passwordBox" Margin="10" Height="60" VerticalContentAlignment="Center"></PasswordBox>  <Button Name="dologoutBtn" IsEnabled="False" Margin="10" Click="dologoutBtn\_Click" Style="{StaticResource ControlBtnStyle}">Sign Out</Button>  <Button Name="dologinBtn" Margin="10" Click="dologinBtn\_Click" Style="{StaticResource ControlBtnStyle}">Sign In</Button>    </UniformGrid>  </TabItem>  <TabItem Name="empTab">  <UniformGrid>  <Button Name="addEmployeeBtn" Content="Add Employee" Style="{StaticResource ControlBtnStyle}" Click="addEmployeeBtn\_Click"></Button>  <Button Name="viewEmployeeBtn" Style="{StaticResource ControlBtnStyle}" Click="viewEmployeeBtn\_Click">View Employee</Button>  <Button Name="editEmployeeBtn" Style="{StaticResource ControlBtnStyle}" Click="editEmployeeBtn\_Click">Edit Employee</Button>  <Button Name="closeEmpBtn" Style="{StaticResource ControlBtnStyle}" Click="closeEmpBtn\_Click">Close</Button>  </UniformGrid>  </TabItem>  <TabItem Name="agentsTab">  <UniformGrid>  <Button Name="addAgentsBtn" Style="{StaticResource ControlBtnStyle}" Click="addAgentsBtn\_Click">Add Agents</Button>  <Button Name="viewAgentsBtn" Style="{StaticResource ControlBtnStyle}" Click="viewAgentsBtn\_Click">View Agents</Button>  <Button Name="editAgentsBtn" Style="{StaticResource ControlBtnStyle}" Click="editAgentsBtn\_Click">Edit Agents</Button>  <Button Name="closeAgentsBtn" Click="closeEmpBtn\_Click" Style="{StaticResource ControlBtnStyle}">Close</Button>  </UniformGrid>  </TabItem>  <TabItem Name="clientsTab">  <UniformGrid>  <Button Name="addClientsBtn" Style="{StaticResource ControlBtnStyle}" Click="addClientsBtn\_Click">Add Clients</Button>  <Button Name="viewClientsBtn" Style="{StaticResource ControlBtnStyle}" Click="viewClientsBtn\_Click">View Clients</Button>  <Button Name="editClientsBtn" Style="{StaticResource ControlBtnStyle}" Click="editClientsBtn\_Click">Edit Clients</Button>  <Button Name="closeClientsBtn" Click="closeEmpBtn\_Click" Style="{StaticResource ControlBtnStyle}">Close</Button>  </UniformGrid>  </TabItem>  </TabControl>  </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;  using **MMS**Data;  namespace **MMS**Demo  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  public MainWindow()  {  InitializeComponent();  }  private void employeesBtn\_Click(object sender, RoutedEventArgs e)  {  controlsTab.SelectedIndex = 2;  }    private void agentsBtn\_Click(object sender, RoutedEventArgs e)  {  controlsTab.SelectedIndex = 3;  }  private void clientsBtn\_Click(object sender, RoutedEventArgs e)  {  controlsTab.SelectedIndex = 4;  }  private void closeEmpBtn\_Click(object sender, RoutedEventArgs e)  {  controlsTab.SelectedIndex = 0;  }  private void dologinBtn\_Click(object sender, RoutedEventArgs e)  {  LoginData loginData = new LoginData();  loginData.userName = userNameTxtbox.Text;  loginData.password = passwordBox.Password;  loginData.type = (UserTypeEnum)Enum.Parse(typeof(UserTypeEnum), userTypeCombobox.Text, true);  bool loginStatus = **MMS**Db.DbInteraction.Do**MMS**Login(loginData);  dologinBtn.IsEnabled = !loginStatus;  dologoutBtn.IsEnabled = loginStatus;  }  private void dologoutBtn\_Click(object sender, RoutedEventArgs e)  {  dologinBtn.IsEnabled = true;  dologoutBtn.IsEnabled = false;  }  private void loginBtn\_Click(object sender, RoutedEventArgs e)  {  controlsTab.SelectedIndex = 1;  }  private void addAgentsBtn\_Click(object sender, RoutedEventArgs e)  {  AddNewAgent addAgent = new AddNewAgent();  addAgent.ShowDialog();  }  private void viewAgentsBtn\_Click(object sender, RoutedEventArgs e)  {  Agents.ViewAgent viewAgentWindow = new Agents.ViewAgent();  viewAgentWindow.ShowDialog();  }  private void editAgentsBtn\_Click(object sender, RoutedEventArgs e)  {  Agents.EditAgent editAgentWindow = new Agents.EditAgent();  editAgentWindow.ShowDialog();  }  private void addClientsBtn\_Click(object sender, RoutedEventArgs e)  {  Clients.AddNewClient addClient = new Clients.AddNewClient();  addClient.ShowDialog();  }  private void addEmployeeBtn\_Click(object sender, RoutedEventArgs e)  {  Employees.AddNewEmployee addEmployee = new Employees.AddNewEmployee();  addEmployee.ShowDialog();  }  private void viewClientsBtn\_Click(object sender, RoutedEventArgs e)  {  Clients.ViewCient viewClientWindow = new Clients.ViewCient();  viewClientWindow.ShowDialog();  }  private void closeBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void viewEmployeeBtn\_Click(object sender, RoutedEventArgs e)  {  Employees.ViewEmployee viewEmp = new Employees.ViewEmployee();  viewEmp.ShowDialog();  }  private void editEmployeeBtn\_Click(object sender, RoutedEventArgs e)  {  Employees.EditEmployee editEmployee = new Employees.EditEmployee();  editEmployee.ShowDialog();  }  private void controlsTab\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  }  private void editClientsBtn\_Click(object sender, RoutedEventArgs e)  {  Clients.EditClient editClient = new Clients.EditClient();  editClient.ShowDialog();  }  }  } |

|  |
| --- |
| AddNewAgent.xaml |
| <Window xmlns:my="clr-namespace:**MMS**Demo" x:Class="**MMS**Demo.AddNewAgent"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Add New Agent" Height="1225" Width="690">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Name="registerNewAgentBtn" DockPanel.Dock="Bottom" Style="{StaticResource ControlBtnStyle}" Click="registerNewAgentBtn\_Click">Register</Button>  <my:AgentInfo x:Name="agentInfoControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| AddNewAgent.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 **MMS**Demo  {  /// <summary>  /// Interaction logic for AgentInfo.xaml  /// </summary>  public partial class AddNewAgent : Window  {  public AddNewAgent()  {  InitializeComponent();  }  private void registerNewAgentBtn\_Click(object sender, RoutedEventArgs e)  {  **MMS**Data.AgentDetails agentDetails = agentInfoControl.GetDetails();    agentDetails.agentId = **MMS**Data.IdGenerator.GetAgentUniqueId();  if (**MMS**Db.DbInteraction.DoRegisterNewAgent(agentDetails) == 1)  {  this.Close();  }  else  {  MessageBox.Show("Agent registration failed... Try Again..");  }  }  }  } |

|  |
| --- |
| AgentInfo.xaml |
| <UserControl x:Class="**MMS**Demo.AgentInfo"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  mc:Ignorable="d"  d:DesignHeight="1157" d:DesignWidth="662">  <UserControl.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </UserControl.Resources>  <ScrollViewer HorizontalAlignment="Stretch" VerticalAlignment="Stretch">  <DockPanel Background="#FF256D25" LastChildFill="True">  <UniformGrid Columns="2">  <!--<Label Style="{StaticResource labelStyle}">Title</Label>  <ComboBox Name="titleNameCombobox" Style="{StaticResource comboboxStyle}">  <ComboBoxItem>Mr.</ComboBoxItem>  <ComboBoxItem>Mrs.</ComboBoxItem>  <ComboBoxItem>Ms.</ComboBoxItem>  </ComboBox>-->  <Label Style="{StaticResource labelStyle}">Name</Label>  <TextBox Name="nameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Join date</Label>  <DatePicker Name="joinDatePicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Rank</Label>  <TextBox Name="rankTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Id</Label>  <TextBox Name="agentIdTxtbox" IsReadOnly="True" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Designation</Label>  <TextBox Name="designationTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Father / Husband Name</Label>  <TextBox Name="fatherHusbandNameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Address</Label>  <TextBox Name="addressTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Pincode</Label>  <TextBox Name="pincodeTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">State</Label>  <TextBox Name="stateTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Phone Number</Label>  <TextBox Name="phoneNoTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Blood Group</Label>  <TextBox Name="bloodGroupTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Date of Birth</Label>  <DatePicker Name="dobDatepicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Nationality</Label>  <TextBox Name="nationalityTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Qualification</Label>  <TextBox Name="qualificationTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Nominee Name</Label>  <TextBox Name="nomineeNameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Nominee Dob</Label>  <DatePicker Name="nomineeAgeDatepicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Nominee Relationship</Label>  <TextBox Name="nomineeRelationshipTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Introducer Id</Label>  <TextBox Name="introducerIdTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Branch</Label>  <TextBox Name="branchTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  </UniformGrid>  </DockPanel>  </ScrollViewer>  </UserControl> |

|  |
| --- |
| AgentInfo.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;  using **MMS**Data;  namespace **MMS**Demo  {  /// <summary>  /// Interaction logic for AgentInfo.xaml  /// </summary>  public partial class AgentInfo : UserControl  {  public AgentInfo()  {  InitializeComponent();  }  public AgentDetails GetDetails()  {  AgentDetails agentData = new AgentDetails();  agentData.agentAddress = addressTxtbox.Text;  agentData.agentBloodGroup = bloodGroupTxtbox.Text;  agentData.agentBranchId = branchTxtbox.Text;  agentData.agentDateOfBirth = dobDatepicker.SelectedDate.Value;  agentData.joinDate = joinDatePicker.SelectedDate.Value;  agentData.agentFatherHusbandName = fatherHusbandNameTxtbox.Text;  agentData.agentIntroducerId = introducerIdTxtbox.Text;  agentData.agentName = nameTxtbox.Text;  agentData.agentNationality = nationalityTxtbox.Text;  agentData.agentNomineeDob = nomineeAgeDatepicker.SelectedDate.Value;  agentData.agentNomineeName = nomineeNameTxtbox.Text;  agentData.agentNomineeRelationship = nomineeRelationshipTxtbox.Text;  agentData.agentPhoneNumber = phoneNoTxtbox.Text;  agentData.agentPinCode = Convert.ToInt32(pincodeTxtbox.Text);  agentData.agentQualification = qualificationTxtbox.Text;  agentData.agentState = stateTxtbox.Text;  //agentData.agentTitle = titleNameCombobox.Text;  agentData.designation = designationTxtbox.Text;  agentData.rank = Convert.ToInt32(rankTxtbox.Text);  return agentData;  }  internal void SetDetails(AgentDetails agentDetails)  {  agentIdTxtbox.Text = agentDetails.agentId;  addressTxtbox.Text = agentDetails.agentAddress;  bloodGroupTxtbox.Text = agentDetails.agentBloodGroup;  branchTxtbox.Text = agentDetails.agentBranchId;  dobDatepicker.SelectedDate = agentDetails.agentDateOfBirth;  joinDatePicker.SelectedDate = agentDetails.joinDate;  fatherHusbandNameTxtbox.Text = agentDetails.agentFatherHusbandName;  introducerIdTxtbox.Text = agentDetails.agentIntroducerId;  nameTxtbox.Text = agentDetails.agentName;  nationalityTxtbox.Text = agentDetails.agentNationality ;  nomineeAgeDatepicker.SelectedDate = agentDetails.agentNomineeDob;  nomineeNameTxtbox.Text = agentDetails.agentNomineeName ;  nomineeRelationshipTxtbox.Text = agentDetails.agentNomineeRelationship;  phoneNoTxtbox.Text= agentDetails.agentPhoneNumber ;  pincodeTxtbox.Text= agentDetails.agentPinCode.ToString();  qualificationTxtbox.Text = agentDetails.agentQualification;  stateTxtbox.Text = agentDetails.agentState;  //titleNameCombobox.Text = agentDetails.agentTitle;  designationTxtbox.Text=agentDetails.designation;  rankTxtbox.Text = agentDetails.rank.ToString();  }  }  } |

|  |
| --- |
| EditAgentDetails.xaml |
| <Window xmlns:my="clr-namespace:**MMS**Demo" x:Class="**MMS**Demo.Agents.EditAgentDetails"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="EditAgentDetails" Height="800" Width="600" Background="#FF256D25">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Width="200" Height="60" DockPanel.Dock="Bottom" Name="updateBtn" Click="updateBtn\_Click" Style="{StaticResource ControlBtnStyle}">Update</Button>  <my:AgentInfo x:Name="agentDetailsControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| EditAgentDetails.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 **MMS**Demo.Agents  {  /// <summary>  /// Interaction logic for EditAgentDetails.xaml  /// </summary>  public partial class EditAgentDetails : Window  {  public EditAgentDetails(**MMS**Data.AgentDetails agentDetails)  {  InitializeComponent();  agentDetailsControl.SetDetails(agentDetails);    }  private void updateBtn\_Click(object sender, RoutedEventArgs e)  {  }  }  } |

|  |
| --- |
| ViewAgentDetails.xaml |
| <Window xmlns:my1="clr-namespace:**MMS**Demo" xmlns:my="clr-namespace:**MMS**Demo.Clients" x:Class="**MMS**Demo.Agents.ViewAgentDetails"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="View Agent Details" Height="800" Width="600" Background="#FF256D25">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Width="200" Height="60" DockPanel.Dock="Bottom" Name="closeBtn" Click="closeBtn\_Click" Style="{StaticResource ControlBtnStyle}">Close</Button>  <my1:AgentInfo x:Name="agentDetailsControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| ViewAgentDetails.xaml.cs |
| <Window xmlns:my1="clr-namespace:**MMS**Demo" xmlns:my="clr-namespace:**MMS**Demo.Clients" x:Class="**MMS**Demo.Agents.ViewAgentDetails"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="View Agent Details" Height="800" Width="600" Background="#FF256D25">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Width="200" Height="60" DockPanel.Dock="Bottom" Name="closeBtn" Click="closeBtn\_Click" Style="{StaticResource ControlBtnStyle}">Close</Button>  <my1:AgentInfo x:Name="agentDetailsControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| AddNewClient.xaml |
| <Window xmlns:my="clr-namespace:**MMS**Demo.Clients" x:Class="**MMS**Demo.Clients.AddNewClient"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Add New Client" Height="900" Width="600">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Name="registerNewClientBtn" DockPanel.Dock="Bottom" Style="{StaticResource ControlBtnStyle}" Click="registerNewClientBtn\_Click">Register</Button>  <my:ClientInfo x:Name="clientInfoControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| AddNewClient.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 **MMS**Demo.Clients  {  /// <summary>  /// Interaction logic for AddNewClient.xaml  /// </summary>  public partial class AddNewClient : Window  {  public AddNewClient()  {  InitializeComponent();  }  private void registerNewClientBtn\_Click(object sender, RoutedEventArgs e)  {  **MMS**Data.ClientDetails clientDetails = clientInfoControl.GetDetails();  clientDetails.clientId = **MMS**Data.IdGenerator.GetClientUniqueId();  if (**MMS**Db.DbInteraction.DoRegisterNewClient(clientDetails) == 1)  {  this.Close();  }  else  {  MessageBox.Show("Client registration failed... Try Again..");  }  }    }  } |

|  |
| --- |
| ClientInfo.xaml |
| <UserControl x:Class="**MMS**Demo.Clients.ClientInfo"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  mc:Ignorable="d"  d:DesignHeight="1516" d:DesignWidth="661">  <UserControl.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </UserControl.Resources>  <ScrollViewer HorizontalAlignment="Stretch" VerticalAlignment="Stretch">  <DockPanel Background="#FF256D25" LastChildFill="True">  <UniformGrid Columns="2">  <!--<Label Style="{StaticResource labelStyle}">Title</Label>  <ComboBox Name="titleNameCombobox" Style="{StaticResource comboboxStyle}">  <ComboBoxItem>Mr.</ComboBoxItem>  <ComboBoxItem>Mrs.</ComboBoxItem>  <ComboBoxItem>Ms.</ComboBoxItem>  </ComboBox>-->  <Label Style="{StaticResource labelStyle}">Name</Label>  <TextBox Name="nameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Join date</Label>  <DatePicker Name="joinDatePicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Date of Birth</Label>  <DatePicker Name="clientDobDatepicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Id</Label>  <TextBox Name="clientIdTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Joint Applicant Name</Label>  <TextBox Name="jointApplicantNameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Joint Applicant DOB</Label>  <DatePicker Name="jointApplicantDobDatepicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Guardian/Father Name</Label>  <TextBox Name="clientGuardianFatherNameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Address</Label>  <TextBox Name="clientAddressTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Pin Code</Label>  <TextBox Name="clientPinCodeTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Phone Number</Label>  <TextBox Name="clientPhoneNumberTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Occupation</Label>  <TextBox Name="clientOccupationTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Nominee Name</Label>  <TextBox Name="clientNomineeNameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Nominee Date of Birth</Label>  <DatePicker Name="clientNomineeDateOfBirthDatepicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Relationship With Client</Label>  <TextBox Name="clientNomineeRelationshipTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Plan</Label>  <TextBox Name="clientPlanTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Propose Value</Label>  <TextBox Name="clientProposedValueTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Month/Period</Label>  <TextBox Name="clientPeriodMonthTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Mode of Payment</Label>  <ComboBox Name="clientPaymentModeCombobox" Style="{StaticResource comboboxStyle}">  <ComboBoxItem>Cash</ComboBoxItem>  <ComboBoxItem>Cheque</ComboBoxItem>  <ComboBoxItem>Demand Draft</ComboBoxItem>  </ComboBox>  <Label Style="{StaticResource labelStyle}">Payment Amount- Installment</Label>  <TextBox Name="clientPaymentAmountInstalmentTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Payment Amount- Service Charge</Label>  <TextBox Name="clientPaymentAmountServiceChargeTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Payment Amount- Total</Label>  <TextBox Name="clientPaymentAmountTotalTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Payment Receipt Number</Label>  <TextBox Name="clientPaymentReceiptNumberTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Introducer Id</Label>  <TextBox Name="clientIntroducerIdTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  </UniformGrid>  </DockPanel>  </ScrollViewer>  </UserControl> |

|  |
| --- |
| ClientInfo.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;  using **MMS**Data;  namespace **MMS**Demo.Clients  {  /// <summary>  /// Interaction logic for ClientInfo.xaml  /// </summary>  public partial class ClientInfo : UserControl  {  public ClientInfo()  {  InitializeComponent();  }  public ClientDetails GetDetails()  {  ClientDetails clientData = new ClientDetails();  clientData.clientName = nameTxtbox.Text;  clientData.joinDate = joinDatePicker.SelectedDate.Value;  clientData.clientDateOfBirth = clientDobDatepicker.SelectedDate.Value;  //clientData.clientId = clientIdTxtbox.Text;  clientData.clientJointApplicantName = jointApplicantNameTxtbox.Text;  clientData.clientJointApplicantDateOfBirth = jointApplicantDobDatepicker.SelectedDate.Value;  clientData.clientGuardianFatherName = clientGuardianFatherNameTxtbox.Text;  clientData.clientAddress = clientAddressTxtbox.Text;  clientData.clientPinCode = Convert.ToInt32(clientPinCodeTxtbox.Text);  clientData.clientPhoneNumber = clientPhoneNumberTxtbox.Text;  clientData.clientOccupation = clientOccupationTxtbox.Text;  clientData.clientNomineeName = clientNomineeNameTxtbox.Text;  clientData.clientNomineeDateOfBirth = clientNomineeDateOfBirthDatepicker.SelectedDate.Value;  clientData.clientNomineeRelationship = clientNomineeRelationshipTxtbox.Text;  clientData.clientPlan = clientPlanTxtbox.Text;  clientData.clientProposedValue = Convert.ToDouble(clientProposedValueTxtbox.Text);  clientData.clientPeriodMonth = Convert.ToInt32(clientPeriodMonthTxtbox.Text);  clientData.clientPaymentMode = clientPaymentModeCombobox.Text;  clientData.clientPaymentAmountInstallment = Convert.ToDouble(clientPaymentAmountInstalmentTxtbox.Text);  clientData.clientPaymentAmountServiceCharge = Convert.ToDouble (clientPaymentAmountServiceChargeTxtbox.Text);  clientData.clientPaymentAmountTotal = Convert.ToDouble(clientPaymentAmountTotalTxtbox.Text);  clientData.clientPaymentReciptNumber = clientPaymentReceiptNumberTxtbox.Text;  clientData.clientIntroducerId = clientIntroducerIdTxtbox.Text;  return clientData;  }  public void SetDetails(ClientDetails clientData)  {  nameTxtbox.Text = clientData.clientName;  joinDatePicker.SelectedDate = clientData.joinDate;  clientDobDatepicker.SelectedDate = clientData.clientDateOfBirth;  clientIdTxtbox.Text = clientData.clientId;  jointApplicantNameTxtbox.Text = clientData.clientJointApplicantName;  jointApplicantDobDatepicker.SelectedDate = clientData.clientJointApplicantDateOfBirth;  clientGuardianFatherNameTxtbox.Text = clientData.clientGuardianFatherName;  clientAddressTxtbox.Text = clientData.clientAddress;  clientPinCodeTxtbox.Text = clientData.clientPinCode.ToString();  clientPhoneNumberTxtbox.Text = clientData.clientPhoneNumber;  clientOccupationTxtbox.Text = clientData.clientOccupation;  clientNomineeNameTxtbox.Text=clientData.clientNomineeName;  clientNomineeDateOfBirthDatepicker.SelectedDate = clientData.clientNomineeDateOfBirth;  clientNomineeRelationshipTxtbox.Text = clientData.clientNomineeRelationship;  clientPlanTxtbox.Text = clientData.clientPlan;  clientProposedValueTxtbox.Text = clientData.clientProposedValue.ToString();  clientPeriodMonthTxtbox.Text=clientData.clientPeriodMonth.ToString();  clientPaymentModeCombobox.Text = clientData.clientPaymentMode;  clientPaymentAmountInstalmentTxtbox.Text = clientData.clientPaymentAmountInstallment.ToString();  clientPaymentAmountServiceChargeTxtbox.Text = clientData.clientPaymentAmountServiceCharge.ToString();  clientPaymentAmountTotalTxtbox.Text = clientData.clientPaymentAmountTotal.ToString();  clientPaymentReceiptNumberTxtbox.Text = clientData.clientPaymentReciptNumber;  clientIntroducerIdTxtbox.Text = clientData.clientIntroducerId;    }  }  } |

|  |
| --- |
| EditClientDetails.xaml |
| <Window xmlns:my="clr-namespace:**MMS**Demo.Clients" x:Class="**MMS**Demo.Clients.EditClientDetails"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Edit Client Details" Height="819" Width="995" Background="#FF256D25">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Width="200" Height="60" DockPanel.Dock="Bottom" Name="updateBtn" Click="updateBtn\_Click" Style="{StaticResource ControlBtnStyle}">Update</Button>  <my:ClientInfo x:Name="clientDetailsControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| ViewClientDetails.xaml |
| <Window xmlns:my="clr-namespace:**MMS**Demo.Clients" x:Class="**MMS**Demo.Clients.ViewClientDetails"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="ViewClientDetails" Height="819" Width="995" Background="#FF256D25">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Width="200" Height="60" DockPanel.Dock="Bottom" Name="closeBtn" Click="closeBtn\_Click" Style="{StaticResource ControlBtnStyle}">Close</Button>  <my:ClientInfo x:Name="clientDetailsControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| AddNewEmployee.xaml |
| <Window xmlns:my="clr-namespace:**MMS**Demo.Employees" x:Class="**MMS**Demo.Employees.AddNewEmployee"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="AddNewEmployee" Height="700" Width="700" Background="#FF256D25">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Name="registerNewClientBtn" DockPanel.Dock="Bottom" Style="{StaticResource ControlBtnStyle}" Click="registerNewClientBtn\_Click">Register</Button>  <my:EmployeeInfo x:Name="employeeInfoControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| AddNewEmployee.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 **MMS**Data;  namespace **MMS**Demo.Employees  {  /// <summary>  /// Interaction logic for AddNewEmployee.xaml  /// </summary>  public partial class AddNewEmployee : Window  {  public AddNewEmployee()  {  InitializeComponent();  }  private void registerNewClientBtn\_Click(object sender, RoutedEventArgs e)  {  **MMS**Data.EmployeeDetails employeeDetails = employeeInfoControl.getDetails();  employeeDetails.employeeId = **MMS**Data.IdGenerator.GetEmployeeUniqueId();  if (**MMS**Db.DbInteraction.DoRegisterNewEmployee(employeeDetails) == 1)  {  this.Close();  }  else  {  MessageBox.Show("do it");  }  }  }  } |

|  |
| --- |
| EditEmployeeDetails.xaml |
| <Window xmlns:my="clr-namespace:**MMS**Demo.Employees" x:Class="**MMS**Demo.Employees.EditEmployeeDetails"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="Edit Employee Details" Height="819" Width="995" Background="#FF256D25">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Width="200" Height="60" DockPanel.Dock="Bottom" Name="updateBtn" Click="updateBtn\_Click" Style="{StaticResource ControlBtnStyle}">Update</Button>  <my:EmployeeInfo x:Name="employeeEditControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| EditEmployeeDetails.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 **MMS**Demo.Employees  {  /// <summary>  /// Interaction logic for EditEmployeeDetails.xaml  /// </summary>  public partial class EditEmployeeDetails : Window  {  public EditEmployeeDetails(**MMS**Data.EmployeeDetails employeeDetails)  {  InitializeComponent();  employeeEditControl.SetDetails(employeeDetails);  }  private void updateBtn\_Click(object sender, RoutedEventArgs e)  {  }  }  } |

|  |
| --- |
| EmployeeInfo.xaml |
| <UserControl x:Class="**MMS**Demo.Employees.EmployeeInfo"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  mc:Ignorable="d"  d:DesignHeight="1157" d:DesignWidth="662" >  <UserControl.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </UserControl.Resources>  <ScrollViewer>  <DockPanel Background="#FF256D25" LastChildFill="True">  <UniformGrid Columns="2">  <!--<Label Style="{StaticResource labelStyle}">Title</Label>  <ComboBox Name="titleNameCombobox" Style="{StaticResource comboboxStyle}">  <ComboBoxItem>Mr.</ComboBoxItem>  <ComboBoxItem>Mrs.</ComboBoxItem>  <ComboBoxItem>Ms.</ComboBoxItem>  </ComboBox>-->  <Label Style="{StaticResource labelStyle}">Name</Label>  <TextBox Name="employeeNameTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Join date</Label>  <DatePicker Name="employeeJoinDatePicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Date of Birth</Label>  <DatePicker Name="employeeDobDatepicker" Style="{StaticResource datePickerStyle}"></DatePicker>  <Label Style="{StaticResource labelStyle}">Address</Label>  <TextBox Name="employeeAddressTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Phone Number</Label>  <TextBox Name="employeePhoneNumberTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">E Mail Address</Label>  <TextBox Name="employeeEmailTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Employee Id</Label>  <TextBox Name="employeeIdTxtbox" IsReadOnly="True" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Department</Label>  <TextBox Name="employeeDepartmentTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Salary</Label>  <TextBox Name="employeeSalaryTxtbox" Style="{StaticResource textboxStyle}"></TextBox>  <Label Style="{StaticResource labelStyle}">Manager Id</Label>  <TextBox Name="employeeManagerIdTxtbox" Style="{StaticResource textboxStyle}"></TextBox>      </UniformGrid>  </DockPanel>  </ScrollViewer>  </UserControl> |

|  |
| --- |
| EmployeeInfo.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;  using **MMS**Data;  namespace **MMS**Demo.Employees  {  /// <summary>  /// Interaction logic for EmployeeInfo.xaml  /// </summary>  public partial class EmployeeInfo : UserControl  {  public EmployeeInfo()  {  InitializeComponent();  }  public EmployeeDetails getDetails()  {  EmployeeDetails employeeData = new EmployeeDetails();  employeeData.employeeName = employeeNameTxtbox.Text;  employeeData.employeeDob = employeeDobDatepicker.SelectedDate.Value;  employeeData.employeeAddress = employeeAddressTxtbox.Text;  employeeData.employeePhoneNumber = employeePhoneNumberTxtbox.Text;  employeeData.employeeEmail = employeeEmailTxtbox.Text;  employeeData.employeeJoinDate = employeeJoinDatePicker.SelectedDate.Value;  employeeData.employeeDepartment = employeeDepartmentTxtbox.Text;  employeeData.employeeSalary = Convert.ToDouble(employeeSalaryTxtbox.Text);  employeeData.employeeManagerId = employeeManagerIdTxtbox.Text;  return employeeData;  }  internal void SetDetails(EmployeeDetails employeeDetails)  {  employeeNameTxtbox.Text = employeeDetails.employeeName;  employeeDobDatepicker.SelectedDate = employeeDetails.employeeDob;  employeeAddressTxtbox.Text = employeeDetails.employeeAddress;  employeePhoneNumberTxtbox.Text = employeeDetails.employeePhoneNumber;  employeeEmailTxtbox.Text = employeeDetails.employeeEmail;  employeeJoinDatePicker.SelectedDate = employeeDetails.employeeJoinDate;  employeeDepartmentTxtbox.Text = employeeDetails.employeeDepartment;  employeeSalaryTxtbox.Text = employeeDetails.employeeSalary.ToString();  employeeManagerIdTxtbox.Text = employeeDetails.employeeManagerId;  employeeIdTxtbox.Text = employeeDetails.employeeId;  }  }  } |

|  |
| --- |
| ViewEmployeeDetails.xaml |
| <Window xmlns:my="clr-namespace:**MMS**Demo.Employees" x:Class="**MMS**Demo.Employees.ViewEmployeeDetails"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="View Employee Details" Height="819" Width="995" Background="#FF256D25">  <Window.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="/**MMSM**tyle;component/CommonStyle.xaml"/>  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Window.Resources>  <DockPanel LastChildFill="True">  <Button Width="200" Height="60" DockPanel.Dock="Bottom" Name="closeBtn" Click="closeBtn\_Click" Style="{StaticResource ControlBtnStyle}">Close</Button>  <my:EmployeeInfo x:Name="employeeDetailsControl"/>  </DockPanel>  </Window> |

|  |
| --- |
| ViewEmployeeDetails.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 **MMS**Demo.Employees  {  /// <summary>  /// Interaction logic for ViewEmployeeDetails.xaml  /// </summary>  public partial class ViewEmployeeDetails : Window  {  public ViewEmployeeDetails(**MMS**Data.EmployeeDetails employeeDetails)  {  InitializeComponent();  employeeDetailsControl.SetDetails(employeeDetails);  }  private void closeBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  }  } |

|  |
| --- |
| CommonStyle.xaml |
| <ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">    <Style x:Key="labelStyle" TargetType="Label">  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="16" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="Height" Value="60" />  <Setter Property="VerticalContentAlignment" Value="Center" />  <Setter Property="Margin" Value="20 0 0 0"/>  </Style>  <Style x:Key="textboxStyle" TargetType="TextBox">  <Setter Property="Foreground" Value="Black" />  <Setter Property="FontSize" Value="16" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="Height" Value="60" />  <Setter Property="VerticalContentAlignment" Value="Center" />  <Setter Property="Margin" Value="10"/>  </Style>  <Style x:Key="comboboxStyle" TargetType="ComboBox">  <Setter Property="Foreground" Value="Black" />  <Setter Property="FontSize" Value="16" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="Height" Value="60" />  <Setter Property="VerticalContentAlignment" Value="Center" />  <Setter Property="HorizontalContentAlignment" Value="Center" />  <Setter Property="Margin" Value="10"/>  <Setter Property="SelectedIndex" Value="0"/>  </Style>  <Style x:Key="ControlBtnStyle" TargetType="Button">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Green" Offset="0" />  <GradientStop Color="LightGray" 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" />  </Style>  <Style x:Key="MainBtnStyle" TargetType="Button">  <Setter Property="Background">  <Setter.Value>  <RadialGradientBrush>  <GradientStop Color="Green" Offset="0.119" />  <GradientStop Color="LightGray" Offset="1" />  </RadialGradientBrush>  </Setter.Value>  </Setter>  <Setter Property="Foreground" Value="#FFE9F9DE" />  <Setter Property="FontSize" Value="32" />  <Setter Property="FontWeight" Value="Bold" />  <Setter Property="BorderBrush" Value="#FF001900" />  </Style>  <Style x:Key="datePickerStyle" TargetType="DatePicker">  <Setter Property="VerticalContentAlignment" Value="Center" />  <Setter Property="Margin" Value="10"/>  </Style>  </ResourceDictionary> |

|  |
| --- |
| DbInteraction.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using **MMS**Data;  namespace **MMS**Db  {  public static class DbInteraction  {  #region Login  public static bool Do**MMS**Login(LoginData loginData)  {  return FetcheLoginData(loginData);  }  private static bool FetcheLoginData(LoginData loginData)  {  bool returnVal = false;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=technicise;database=**MMS**\_mb;persist security info=False");  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlConnection.Open();  msqlCommand.CommandText = "Select password,type from login\_details where user\_name = @userName;";  msqlCommand.Parameters.AddWithValue("@userName", loginData.userName);  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  msqlReader.Read();  string password = msqlReader.GetString("password");  string type = msqlReader.GetString("type");  if (loginData.password.Equals(password) && loginData.type.ToString().Equals(type))  returnVal = true;  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  #endregion  #region Agents  public static int DoRegisterNewAgent(AgentDetails agentDetails)  {  return RegisterNewAgentInDb(agentDetails);  }  private static int RegisterNewAgentInDb(AgentDetails agentDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=technicise;database=**MMS**\_mb;persist security info=False");  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;  //open the connection  if (msqlConnection.State != System.Data.ConnectionState.Open)  msqlConnection.Open();  msqlCommand.CommandText = "INSERT INTO agents(agent\_id,agent\_join\_date,agent\_rank,agent\_designation,agent\_nationality,agent\_educational\_qualification,agent\_nominee\_name,agent\_nominee\_dob,agent\_nominee\_relationship,agent\_introducer\_id,agent\_branch\_id,agent\_name,agent\_father\_husband\_name,agent\_address,agent\_pin\_code,agent\_state,agent\_phone\_number,agent\_blood\_group,agent\_dob) "  + "VALUES(@agent\_id,@agent\_join\_date,@agent\_rank,@agent\_designation,@agent\_nationality,@agent\_educational\_qualification,@agent\_nominee\_name,@agent\_nominee\_dob,@agent\_nominee\_relationship,@agent\_introducer\_id,@agent\_branch\_id,@agent\_name,@agent\_father\_husband\_name,@agent\_address,@agent\_pin\_code,@agent\_state,@agent\_phone\_number,@agent\_blood\_group,@agent\_dob)";  msqlCommand.Parameters.AddWithValue("@agent\_id", agentDetails.agentId);  msqlCommand.Parameters.AddWithValue("@agent\_join\_date", agentDetails.joinDate);  msqlCommand.Parameters.AddWithValue("@agent\_rank", agentDetails.rank);  msqlCommand.Parameters.AddWithValue("@agent\_designation", agentDetails.designation);  msqlCommand.Parameters.AddWithValue("@agent\_nationality", agentDetails.agentNationality);  msqlCommand.Parameters.AddWithValue("@agent\_educational\_qualification", agentDetails.agentQualification);  msqlCommand.Parameters.AddWithValue("@agent\_nominee\_name", agentDetails.agentNomineeName);  msqlCommand.Parameters.AddWithValue("@agent\_nominee\_dob", agentDetails.agentNomineeDob);  msqlCommand.Parameters.AddWithValue("@agent\_nominee\_relationship", agentDetails.agentNomineeRelationship);  msqlCommand.Parameters.AddWithValue("@agent\_introducer\_id", agentDetails.agentIntroducerId);  msqlCommand.Parameters.AddWithValue("@agent\_branch\_id", agentDetails.agentBranchId);  msqlCommand.Parameters.AddWithValue("@agent\_name", agentDetails.agentName);  msqlCommand.Parameters.AddWithValue("@agent\_father\_husband\_name", agentDetails.agentFatherHusbandName);  msqlCommand.Parameters.AddWithValue("@agent\_address", agentDetails.agentAddress);  msqlCommand.Parameters.AddWithValue("@agent\_pin\_code", agentDetails.agentPinCode);  msqlCommand.Parameters.AddWithValue("@agent\_state", agentDetails.agentState);  msqlCommand.Parameters.AddWithValue("@agent\_phone\_number", agentDetails.agentPhoneNumber);  msqlCommand.Parameters.AddWithValue("@agent\_blood\_group", agentDetails.agentBloodGroup);  msqlCommand.Parameters.AddWithValue("@agent\_dob", agentDetails.agentDateOfBirth);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<AgentDetails> GetAllAgentList()  {  return QueryAllAgentList();  }  private static List<AgentDetails> QueryAllAgentList()  {  List<AgentDetails> agentList = new List<AgentDetails>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=technicise;database=**MMS**\_mb;persist security info=False");  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlConnection.Open();  msqlCommand.CommandText = "Select \* From agents;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  AgentDetails agent = new AgentDetails();  agent.agentId = msqlReader.GetString("agent\_id");  agent.rank = msqlReader.GetInt32("agent\_rank");  agent.designation = msqlReader.GetString("agent\_designation");  agent.agentNationality = msqlReader.GetString("agent\_nationality");  agent.agentQualification = msqlReader.GetString("agent\_educational\_qualification");  agent.agentNomineeName = msqlReader.GetString("agent\_nominee\_name");  agent.agentNomineeDob = msqlReader.GetDateTime("agent\_nominee\_dob");  agent.joinDate = msqlReader.GetDateTime("agent\_join\_date");  agent.agentNomineeRelationship = msqlReader.GetString("agent\_nominee\_relationship");  agent.agentIntroducerId = msqlReader.GetString("agent\_introducer\_id");  agent.agentBranchId = msqlReader.GetString("agent\_branch\_id");  agent.agentName = msqlReader.GetString("agent\_name");  agent.agentFatherHusbandName = msqlReader.GetString("agent\_father\_husband\_name");  agent.agentAddress = msqlReader.GetString("agent\_address");  agent.agentPinCode = msqlReader.GetInt32("agent\_pin\_code");  agent.agentState = msqlReader.GetString("agent\_state");  agent.agentPhoneNumber = msqlReader.GetString("agent\_phone\_number");  agent.agentBloodGroup = msqlReader.GetString("agent\_blood\_group");  agent.agentDateOfBirth = msqlReader.GetDateTime("agent\_dob");  agentList.Add(agent);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return agentList;  }  #endregion  #region Clients  public static int DoRegisterNewClient(ClientDetails clientDetails)  {  return RegisterNewClientInDb(clientDetails);  }  private static int RegisterNewClientInDb(ClientDetails clientDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=technicise;database=**MMS**\_mb;persist security info=False");  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;  //open the connection  if (msqlConnection.State != System.Data.ConnectionState.Open)  msqlConnection.Open();  msqlCommand.CommandText = "INSERT INTO clients(client\_id,client\_name,client\_dob,client\_joint\_applicant\_name,client\_joint\_applicant\_dob,client\_gurdain\_father\_husband\_name,client\_address,client\_pin\_code,client\_phone\_number,client\_occupation,client\_nominee\_name,client\_nominee\_dob,client\_nominee\_relationship,client\_plan,client\_proposed\_value,client\_period\_month,client\_payment\_mode,client\_payment\_amount\_instalment,client\_payment\_amount\_service\_charge,client\_payment\_amount\_total,client\_payment\_receipt\_number,client\_introducer\_id,client\_join\_date) "  + "VALUES(@client\_id,@client\_name,@client\_dob,@client\_joint\_applicant\_name,@client\_joint\_applicant\_dob,@client\_gurdain\_father\_husband\_name,@client\_address,@client\_pin\_code,@client\_phone\_number,@client\_occupation,@client\_nominee\_name,@client\_nominee\_dob,@client\_nominee\_relationship,@client\_plan,@client\_proposed\_value,@client\_period\_month,@client\_payment\_mode,@client\_payment\_amount\_instalment,@client\_payment\_amount\_service\_charge,@client\_payment\_amount\_total,@client\_payment\_receipt\_number,@client\_introducer\_id,@client\_join\_date)";  msqlCommand.Parameters.AddWithValue("@client\_id", clientDetails.clientId);  msqlCommand.Parameters.AddWithValue("@client\_name", clientDetails.clientName);  msqlCommand.Parameters.AddWithValue("@client\_dob", clientDetails.clientDateOfBirth);  msqlCommand.Parameters.AddWithValue("@client\_joint\_applicant\_name", clientDetails.clientJointApplicantName);  msqlCommand.Parameters.AddWithValue("@client\_joint\_applicant\_dob", clientDetails.clientJointApplicantDateOfBirth);  msqlCommand.Parameters.AddWithValue("@client\_gurdain\_father\_husband\_name", clientDetails.clientGuardianFatherName);  msqlCommand.Parameters.AddWithValue("@client\_address", clientDetails.clientAddress);  msqlCommand.Parameters.AddWithValue("@client\_pin\_code", clientDetails.clientPinCode);  msqlCommand.Parameters.AddWithValue("@client\_phone\_number", clientDetails.clientPhoneNumber);  msqlCommand.Parameters.AddWithValue("@client\_occupation", clientDetails.clientOccupation);  msqlCommand.Parameters.AddWithValue("@client\_nominee\_name", clientDetails.clientNomineeName);  msqlCommand.Parameters.AddWithValue("@client\_nominee\_dob", clientDetails.clientNomineeDateOfBirth);  msqlCommand.Parameters.AddWithValue("@client\_nominee\_relationship", clientDetails.clientNomineeRelationship);  msqlCommand.Parameters.AddWithValue("@client\_plan", clientDetails.clientPlan);  msqlCommand.Parameters.AddWithValue("@client\_proposed\_value", clientDetails.clientProposedValue);  msqlCommand.Parameters.AddWithValue("@client\_period\_month", clientDetails.clientPeriodMonth);  msqlCommand.Parameters.AddWithValue("@client\_payment\_mode", clientDetails.clientPaymentMode);  msqlCommand.Parameters.AddWithValue("@client\_payment\_amount\_instalment", clientDetails.clientPaymentAmountInstallment);  msqlCommand.Parameters.AddWithValue("@client\_payment\_amount\_service\_charge", clientDetails.clientPaymentAmountServiceCharge);  msqlCommand.Parameters.AddWithValue("@client\_payment\_amount\_total", clientDetails.clientPaymentAmountTotal);  msqlCommand.Parameters.AddWithValue("@client\_payment\_receipt\_number", clientDetails.clientPaymentReciptNumber);  msqlCommand.Parameters.AddWithValue("@client\_introducer\_id", clientDetails.clientIntroducerId);  msqlCommand.Parameters.AddWithValue("@client\_join\_date", clientDetails.joinDate);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<ClientDetails> GetAllClientList()  {  return QueryAllClientList();  }  private static List<ClientDetails> QueryAllClientList()  {  List<ClientDetails> clientList = new List<ClientDetails>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=technicise;database=**MMS**\_mb;persist security info=False");  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlConnection.Open();  msqlCommand.CommandText = "Select \* From clients;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  ClientDetails client = new ClientDetails();  client.clientId = msqlReader.GetString("client\_id");  client.clientName = msqlReader.GetString("client\_name");  client.clientDateOfBirth = msqlReader.GetDateTime("client\_dob");  client.clientJointApplicantName = msqlReader.GetString("client\_joint\_applicant\_name");  client.clientJointApplicantDateOfBirth = msqlReader.GetDateTime("client\_joint\_applicant\_dob");  client.clientGuardianFatherName = msqlReader.GetString("client\_gurdain\_father\_husband\_name");  client.clientAddress = msqlReader.GetString("client\_address");  client.clientPinCode = msqlReader.GetInt32("client\_pin\_code");  client.clientPhoneNumber = msqlReader.GetString("client\_phone\_number");  client.clientOccupation = msqlReader.GetString("client\_occupation");  client.clientNomineeName = msqlReader.GetString("client\_nominee\_name");  client.clientNomineeDateOfBirth = msqlReader.GetDateTime("client\_nominee\_dob");  client.clientNomineeRelationship = msqlReader.GetString("client\_nominee\_relationship");  client.clientPlan = msqlReader.GetString("client\_plan");  client.clientPeriodMonth = msqlReader.GetInt32("client\_period\_month");  client.clientPaymentMode = msqlReader.GetString("client\_payment\_mode");  client.clientPaymentAmountInstallment = msqlReader.GetDouble("client\_payment\_amount\_instalment");  client.clientPaymentAmountServiceCharge = msqlReader.GetDouble("client\_payment\_amount\_service\_charge");  client.clientPaymentAmountTotal = msqlReader.GetDouble("client\_payment\_amount\_total");  client.clientPaymentReciptNumber = msqlReader.GetString("client\_payment\_receipt\_number");  client.clientIntroducerId = msqlReader.GetString("client\_introducer\_id");  client.joinDate = msqlReader.GetDateTime("client\_join\_date");  clientList.Add(client);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return clientList;  }  #endregion  #region Employee  public static int DoRegisterNewEmployee(EmployeeDetails employeeDetails)  {  return DoRegisterNewEmployeeInDb(employeeDetails);  }  private static int DoRegisterNewEmployeeInDb(EmployeeDetails employeeDetails)  {  int returnVal = 0;  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=technicise;database=**MMS**\_mb;persist security info=False");  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;  //open the connection  if (msqlConnection.State != System.Data.ConnectionState.Open)  msqlConnection.Open();  msqlCommand.CommandText = "INSERT INTO employees(employee\_id,employee\_name,employee\_dob,employee\_address,employee\_phone\_no,employee\_email,employee\_join\_date,department,salary,employee\_manager\_id) "  + "VALUES(@employee\_id,@employee\_name,@employee\_dob,@employee\_address,@employee\_phone\_no,@employee\_email,@employee\_join\_date,@department,@salary,@employee\_manager\_id)";  msqlCommand.Parameters.AddWithValue("@employee\_id", employeeDetails.employeeId);  msqlCommand.Parameters.AddWithValue("@employee\_name", employeeDetails.employeeName);  msqlCommand.Parameters.AddWithValue("@employee\_dob", employeeDetails.employeeDob);  msqlCommand.Parameters.AddWithValue("@employee\_address", employeeDetails.employeeAddress);  msqlCommand.Parameters.AddWithValue("@employee\_phone\_no", employeeDetails.employeePhoneNumber);  msqlCommand.Parameters.AddWithValue("@employee\_email", employeeDetails.employeeEmail);  msqlCommand.Parameters.AddWithValue("@employee\_join\_date", employeeDetails.employeeJoinDate);  msqlCommand.Parameters.AddWithValue("@department", employeeDetails.employeeDepartment);  msqlCommand.Parameters.AddWithValue("@salary", employeeDetails.employeeSalary);  msqlCommand.Parameters.AddWithValue("@employee\_manager\_id", employeeDetails.employeeManagerId);  msqlCommand.ExecuteNonQuery();  returnVal = 1;  }  catch (Exception er)  {  returnVal = 0;  }  finally  {  //always close the connection  msqlConnection.Close();  }  return returnVal;  }  public static List<EmployeeDetails> GetAllEmployeeList()  {  return QueryAllEmployeeList();  }  private static List<EmployeeDetails> QueryAllEmployeeList()  {  List<EmployeeDetails> employeeList = new List<EmployeeDetails>();  MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;  msqlConnection = new MySql.Data.MySqlClient.MySqlConnection("server=localhost;user id=root;Password=technicise;database=**MMS**\_mb;persist security info=False");  try  { //define the command reference  MySql.Data.MySqlClient.MySqlCommand msqlCommand = new MySql.Data.MySqlClient.MySqlCommand();  msqlCommand.Connection = msqlConnection;  msqlConnection.Open();  msqlCommand.CommandText = "Select \* From employees;";  MySql.Data.MySqlClient.MySqlDataReader msqlReader = msqlCommand.ExecuteReader();  while (msqlReader.Read())  {  EmployeeDetails employee = new EmployeeDetails();    employee.employeeId = msqlReader.GetString("employee\_id");  employee.employeeName = msqlReader.GetString("employee\_name");  employee.employeeDob = msqlReader.GetDateTime("employee\_dob");  employee.employeeAddress = msqlReader.GetString("employee\_address");  employee.employeeJoinDate = msqlReader.GetDateTime("employee\_join\_date");  employee.employeePhoneNumber = msqlReader.GetString("employee\_phone\_no");  employee.employeeEmail = msqlReader.GetString("employee\_email");  employee.employeeDepartment = msqlReader.GetString("department");  employee.employeeSalary = msqlReader.GetDouble("salary");  employee.employeeManagerId = msqlReader.GetString("employee\_manager\_id");  employeeList.Add(employee);  }  }  catch (Exception er)  {  }  finally  {  //always close the connection  msqlConnection.Close();  }  return employeeList;  }  #endregion  }  } |

|  |
| --- |
| DataClasses.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace **MMS**Data  {  public class EmployeeDetails  {  public string employeeId { get; set; }  public string employeeName { get; set; }  public DateTime employeeDob { get; set; }  public string employeeAddress { get; set; }  public string employeePhoneNumber { get; set; }  public string employeeEmail { get; set; }  public DateTime employeeJoinDate { get; set; }  public string employeeDepartment { get; set; }  public double employeeSalary { get; set; }  public string employeeManagerId { get; set; }  }  /\* public class AccountInfo  {  public string id { get; set; }  public ClientDetails client;  public double balance { get; set; }  public List<AmountInfo> amounts { get; set; }  }  public class SchemeInfo  {  public string id { get; set; }  public string name { get; set; }  public string Type { get; set; }  public DateTime Duration { get; set; }  public DateTime dateOfStart { get; set; }  public string description { get; set; }  public AmountInfo amountPremium { get; set; }  public double incentivePercentage;  }  public class LoanInfo  {  public string id { get; set; }  public string name { get; set; }  public string Type { get; set; }  public DateTime Duration { get; set; }  public DateTime dateOfStart { get; set; }  public string description { get; set; }  public AmountInfo amountPrincipal { get; set; }  public double interestPercentage;  }\*/  public enum ViewOrEdit  {  View,  Edit  }  public enum AmountType  {  Debit,  Credit,  Overdraft  }  public class AmountInfo  {  public string id { get; set; }  public double amount { get; set; }  public string description { get; set; }  public AmountType type { get; set; }  }  public class AccountInfo  {  public string id { get; set; }  public ClientDetails client;  public double balance { get; set; }  public List<AmountInfo> amounts { get; set; }  }  public class SchemeInfo  {  public string id { get; set; }  public string name { get; set; }  public string Type { get; set; }  public DateTime Duration { get; set; }  public DateTime dateOfStart { get; set; }  public string description { get; set; }  public AmountInfo amountPremium { get; set; }  public double incentivePercentage;  }  public class LoanInfo  {  public string id { get; set; }  public string name { get; set; }  public string Type { get; set; }  public DateTime Duration { get; set; }  public DateTime dateOfStart { get; set; }  public string description { get; set; }  public AmountInfo amountPrincipal { get; set; }  public double interestPercentage;  }  public class SchemeController  {  public List<SchemeInfo> schemes;  }  public class LoanController  {  public List<LoanInfo> loans;  }  public class ClientController  {  public List<ClientDetails> clients;  }  public class AgentsController  {  public List<AgentDetails> agents;  }  public class AcountsController  {  public List<AccountInfo> accounts;  }  public class MFMSController  {  public AgentsController agentsController;  public ClientController clientController;  public SchemeController SchemeController;  public LoanController loanController;  public AcountsController AcountsController;  }  public enum UserTypeEnum  {  user,  manager,  owner  }  public class AgentDetails  {  public string agentId { get; set; }  public string agentTitle { get; set; }  public string agentName { get; set; }  public DateTime joinDate { get; set; }  public int rank { get; set; }  public string designation { get; set; }  public string agentFatherHusbandName { get; set; }  public string agentAddress { get; set; }  public int agentPinCode { get; set; }  public string agentState { get; set; }  public string agentPhoneNumber { get; set; }  public string agentBloodGroup { get; set; }  public DateTime agentDateOfBirth { get; set; }  public string agentNationality { get; set; }  public string agentQualification { get; set; }  public string agentNomineeName { get; set; }  public DateTime agentNomineeDob { get; set; }  public string agentNomineeRelationship { get; set; }  public string agentIntroducerId { get; set; }  public string agentBranchId { get; set; }  public List<ClientDetails> clients;  }  public class LoginData  {  public string userName;  public UserTypeEnum type;  public string password;  }  public class ClientDetails  {  public string clientId { get; set; }  public string clientName { get; set; }  public DateTime joinDate { get; set; }  public DateTime clientDateOfBirth { get; set; }  public string clientJointApplicantName { get; set; }  public DateTime clientJointApplicantDateOfBirth { get; set; }  public string clientGuardianFatherName { get; set; }  public string clientAddress { get; set; }  public int clientPinCode { get; set; }  public string clientPhoneNumber { get; set; }  public string clientOccupation { get; set; }  public string clientNomineeName { get; set; }  public DateTime clientNomineeDateOfBirth { get; set; }  public string clientNomineeRelationship { get; set; }  public string clientPlan { get; set; }  public double clientProposedValue { get; set; }  public int clientPeriodMonth { get; set; }  public string clientPaymentMode { get; set; }  public double clientPaymentAmountInstallment { get; set; }  public double clientPaymentAmountServiceCharge { get; set; }  public double clientPaymentAmountTotal { get; set; }  public string clientPaymentReciptNumber { get; set; }  public string clientIntroducerId { get; set; }  public List<SchemeInfo> schemes;  public List<LoanInfo> loans;  public List<AmountInfo> payments;  }  } |

|  |
| --- |
| IdGenerator.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace **MMS**Data  {  public static class IdGenerator  {  public static string GetAgentUniqueId()  {  return "**MMS**-"+ DateTime.Now.ToOADate().ToString();  }  public static string GetClientUniqueId()  {  return "**MMS**-" + DateTime.Now.ToOADate().ToString();  }  public static string GetEmployeeUniqueId()  {  return "**MMS**-" + DateTime.Now.ToOADate().ToString();  }  }  } |

## Comments and Description of Coding segments

Various types of comments and description we use in our coding section. Some of them are:

#### 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>

//open the connection

This comment is use at the data interaction section where the code to open the MySql connection.

//define the command reference

To define a command reference in MySql.

//define the connection used by the command object

To define the connection, which is used by the comment object.

//always close the connection

It is indicating to close connection after code is executed.

Manu Unused code in our project we did comment them also like :  
<!--<Condition Property="Password" Value="c" />-->

## 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 bra**BDMS** ({ and }) on a new line.
* Always used curly bra**BDMS** ({ 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 namespa**BDMS** above custom namespa**BDMS**.
* 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 ac**BDMS**s 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 bra**BDMS**.
* 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, updation, deletion and fetching of data take place flexibly. You can see the result as a user; our application does all the works very smoothly.

## 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 resource 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 **MMS**’s and display a message to the user.

## 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

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

### 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 **MMS**.

|  |  |
| --- | --- |
| 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. |

### 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. |

### 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.

### 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 **MMS**:

|  |  |
| --- | --- |
| 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. |

### Database & Data Integrity Testing

The databases and the database processes should be tested as a subsystem within the **MMS** 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. |

### 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 **MMS** 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.

### 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 new data Model in the Destination DB(s) will be migrated from the existing systems source 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. |

## Testing Plan used

### Creation of Test Plan

Early in the deployment planning phase, the testing team creates a test plan. The test plan defines the objectives and scope of the testing effort, and identifies the methodology that our team will use to conduct tests. It also identifies the hardware, software, and tools required for testing and the features and functions that will be tested. A well-rounded test plan notes any risk factors that jeopardize testing and includes a testing schedule.

If our testing team is divided into technology subteams, each subteam should develop a test plan for that team’s specific technology area. For example, the networking team would write a test plan for testing networking features. All members of each subteam should review and approve its team’s test plan before it is integrated into the general test plan.

Figure 2.3 illustrates the tasks we performed to create the test plan.

Figure 2.3   Creating a Test Plan



### Testing Scope and Objectives

In the scope and objectives section of the test plan, the testing team described specifically what we want our testing to accomplish. Also, we needed to define the scope of our testing by identifying what we will test and what we will not. We might limit our testing of client computer hardware to the minimum supported configurations or to the standard configurations.

### Testing Methodology

The general methodology that our team used for testing to testing schema changes might be to configure an isolated domain in the test lab where schema changes can be applied without affecting other lab tests. This section of the test plan addressed the following:

* The domain architecture used for testing
* The tools and techniques used to conduct the tests or to measure results
* Automated techniques we plan to use during testing

### Features and Functions to Test

Included tests that verify or address:

* The functionality of each feature and service that you will implement.
* Interoperability with existing components and systems in the production environment, both during the phase-in period, when there is a mix of old functionality and new Windows Server 2003 functionality, and after the Windows Server 2003 environment has been rolled out.
* Hardware and driver compatibility for every type of client computer that will be running Windows XP Professional.
* Application compatibility for every application that will run on Windows XP Professional.
* Application compatibility for every server application that will run on Windows Server 2003.
* Optimization of configurations, such as those for standardized desktops on client computers.

### Risk Factors

We described the risk factors that could prevent the useful completion of all required tests. We found that the test lab is behind schedule, or that required hardware or software is unavailable, or that testers are working on other projects or need additional training. After we have identified the risk factors, decide what we will do to avoid or mitigate each risk.

### Testing Schedule

We drafted a preliminary schedule that includes each test listed in the test plan. The schedule can help us coordinate test lab use among sub teams. Assign a team member, ideally the test lab manager, if our team has one, to maintain and update the lab schedule. Having an up-to-date schedule is critical when unscheduled lab requests are submitted.

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

### UNIT TEST CASES

|  |  |  |
| --- | --- | --- |
| Test Case Id | Comments | Status |
| MMS – 001  MMS – 001 | NA  NA | PASS  PASS |

### SYSTEM TEST CASES

|  |  |  |
| --- | --- | --- |
| Test Case Id | Comments | Status |
| MMS – 001 | NA | PASS |
| MMS – 002  MMS – 003  MMS – 004  MMS – 005  MMS – 006  MMS – 007  MMS – 008  MMS – 009  MMS – 010  MMS – 011  MMS – 012  MMS – 013  MMS – 014  MMS – 015  MMS – 016  MMS – 017  MMS – 018  MMS – 019  MMS – 020  MMS – 021  MMS – 022  MMS – 023  MMS – 024  MMS – 025  MMS – 026  MMS – 027  MMS – 028  MMS – 029  MMS – 030 | NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA | PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS  PASS |

## Debugging and Code improvement

# System Security measures

* Micro Finance Management System is password protected software. It will be developed such a way that the admin will have complete control on the client’s data.
* Admin can create account with various permission levels, like employee, clients, customer, admin etc. so that the users can see relevant data only.
* The data of the Microfinance will be stored in the database with an encrypted format so even if someone hacks the database somehow still he can make no real harm.
* The software will provide a backup and restore feature in case of loss of data.

## 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

The software requires a predefined username and password to login. It allows a guest login as well which lets a guest user this application with very limited access to the user data.

# Cost Estimation of the Project along with Cost Estimation Model

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

Effort = a1 \* (KLOC)a2 PM

Tdev = b1 \* (Effort)b2 months

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 **MMS**, the formula for estimating the effort based on the code size is shown below:

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

## Estimation of development time

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

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

Assume that the size of a Semi-detached **MMS** product has been estimated to be 3,400 lines of source code. Assume that the average salary of software engineer (me) is Rs. 18,000 per month.

Assume that the size of our

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

Our Effort = 3.0 \* (3.4)1.12 PM

= 12 PM

Normal Development time = 2.5 \* (12)0.35 months

= 6 months

Cost required to develop the product = Rs. 6 \* 18000

= Rs. 1,08,000

# Reports

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

* Annual client’s saving report can be generated
* Hierarchical relationship of client’s report can be generated
* Loan statements of borrower can be generated
* Annual turnover report can be generated
* Updated balance sheet can be generated
* Scheme details can be generated
* Fund details can be generated
* Salary slips can be created

# Future scope and further enhancement of the Project

* It is available for windows operating system only. It could be developed for other OS like Linux as well.
* An online application could be added that would show all the business scheme details and loan offer details through a website. It would be really helpful if a client could access his plan details and other required information after logging in to that website.
* The mobile version of the application has limited features only. Hence a fully functional mobile app could be developed which would be almost as useful as the desktop app itself.
* The mobile version is getting developed for java environment only. We could develop the app such a way that it would support other mobile OS like Symbian, Android etc.

# Bibliography

## Websites

* <http://en.wikipedia.org>
* http://en.wikipedia.org/wiki/Windows\_Presentation\_Foundation
* <http://msdn.microsoft.com/en-us/>
* http://www.c-sharpcorner.com/beginners/
* <http://www.microsoft.com/en-us/default.aspx>
* <http://www.codeplex.com/>
* http://searchitchannel.techtarget.com/
* <http://stackoverflow.com/>
* <http://www.codeguru.com/>
* http://www.roseindia.net/
* http://www.csharpcourse.com/
* <http://www.w3schools.com>
* <http://blogs.technicise.com/>
* http://connect.technicise.com/
* <http://www.mysql.com/>
* 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://netbeans.org/
* https://www.facebook.com/
* <https://developers.facebook.com/>
* <https://developers.facebook.com/apps>
* <https://www.facebook.com/creatormyapp>
* https://www.facebook.com/help/
* https://twitter.com/#
* <https://dev.twitter.com/>
* <https://dev.twitter.com/discussions/3477>
* https://support.twitter.com/
* <https://developers.google.com/>
* <https://maps.google.co.in/>
* http://support.google.com/maps/?hl=en&rd=1
* <http://www.youtube.com/?gl=IN>
* <http://blendinsider.com/>
* https://github.com/
* <https://github.com/anirban-nandy>
* [https://github.com/anirban-nandy/DailyNoteBook](https://github.com/%20anirban-nandy%20/DailyNoteBook)
* <http://learn.github.com/p/intro.html>
* <http://www.vogella.com/articles/Git/article.html>
* <http://try.github.com/levels/1/challenges/1>
* <https://enterprise.github.com/support>
* <https://support.enterprise.github.com/home>

## 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.

# APPENDICES

## IDE:

### 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 to*String.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 development, 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.

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 tyoe 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.

## other softwares used

### 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).

### http://t1.gstatic.com/images?q=tbn:ANd9GcS-CmbHGLD4MH83JH1oNIr_acREqblVhrcFuvQfYZR8HFi1UpaqlgGitHub – repository management tool

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.

### 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 cannot 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

#### OAuth

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

#### applications

* Access Token:https://cacoo.com/oauth/access\_token
* Authorize:https://cacoo.com/oauth/authorize
* Request Token:https://cacoo.com/oauth/request\_token

### 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.*

### GhostDoc

**GhostDoc** is a Visual Studio extension that automatically generates XML documentation comments for methods and properties based on their type, parameters, name, and other contextual information.

When generating documentation for class derived from a base class or for interface implementation (e.g. .NET Framework or your custom framework), GhostDoc will use the documentation that Microsoft or the framework vendor has already written for the base class or interface.

#### Benefits

* Save keystrokes and time
* Simplify documenting your code
* Benefit of the base class documentation
* [New] XML Comment Preview

If you follow good naming conventions in your code, then you will get very decent results on the summary GhostDoc generates.  When I see code that is not documented, it is as simple as hit Ctrl-Shift-D to have GhostDoc document it.

#### Features

New in version 4:

* (Pro) Code Spell Checker
* (Pro) Help File generation
* XML Comment Preview
* Support for C/C++

# glossary

|  |  |
| --- | --- |
| Short Form | Meaning |
| MMS | Microfinance Management System |
| SRS | Software Requirement Specification |
| DFD | Data Flow Diagram |
| ERD | Entity Relationship Diagram |
| GUI | Graphical User Interface |
| DB | Database |
| API | Application Programming Interface |
| COCOMO | Constructive Cost Model |
| SDK | Sweater Development Kit |
| WPF | Windows Presentation Framework |
| XAML | Extensible application Markup Language |
| IDE | Integrated Development Environment |
| HTML | Hyper Text Markup Language |
| www | World Wide Web |
| DBMS | Database Management System |
| Sync | Synchronization |
| C# | C Sharp |