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

|  |
| --- |
|  |

Contents

[**Micro Finance Management System** 0](#_Toc351491806)

[1. Introduction And Objective 4](#_Toc351491807)

[1.1 Introduction 4](#_Toc351491808)

[1.2 Objective 4](#_Toc351491809)

[2. System analysis 4](#_Toc351491810)

[2.1 identification of need 4](#_Toc351491811)

[2.2 Preliminary Investigation 4](#_Toc351491812)

[2.3 Feasibility Study 5](#_Toc351491813)

[2.4 Project Planning & scheduling 5](#_Toc351491814)

[2.4.1 Gantt chart 5](#_Toc351491815)

[2.4.2 Tracking Gantt 6](#_Toc351491816)

[2.4.3 Pert chart (Network Diagram) 6](#_Toc351491817)

[2.5 software requirment SPECIFICATIONS (srs) 7](#_Toc351491818)

[2.5.1 Functional Requirements 7](#_Toc351491819)

[2.5.1.1 Add a new client for loan or new scheme 7](#_Toc351491820)

[2.5.1.2 monitor accounts details 7](#_Toc351491821)

[2.5.1.3 generate bill 8](#_Toc351491822)

[2.5.1.4 generate monthly report 8](#_Toc351491823)

[2.5.1.5 User login with different authentication level 9](#_Toc351491824)

[2.5.1.6 Business details through mobile applicaiton 9](#_Toc351491825)

[2.5.1.7 Remainder of incaome/expense 9](#_Toc351491826)

[2.5.2 Non-functional Requirements 10](#_Toc351491827)

[2.6 Software Engineering Paradigm applied 11](#_Toc351491828)

[2.7 data modeling 11](#_Toc351491829)

[2.7.1 Context Diagram 11](#_Toc351491830)

[2.7.2 Level 0 DFD 12](#_Toc351491831)

[2.7.3 Level 1 DFD 12](#_Toc351491832)

[2.7.4 Level 2 DFD 13](#_Toc351491833)

[2.8 Control Flow diagrams 14](#_Toc351491834)

[2.9 State Diagrams/Sequence diagrams 14](#_Toc351491835)

[2.10 Entity Relationship Model, 14](#_Toc351491836)

[2.11 Class Diagrams/CRC Models/Collaboration Diagrams/Use-case Diagrams/ 16](#_Toc351491837)

[2.11.1 Class Diagram 16](#_Toc351491838)

[2.11.2 Activity Diagrams 18](#_Toc351491839)

[3. System Design 18](#_Toc351491840)

[3.1 Modularisation details 18](#_Toc351491841)

[3.2 Data integrity and constraints 19](#_Toc351491842)

[3.3 Database design, Procedural Design/Object Oriented Design 19](#_Toc351491843)

[3.4 User Interface Design 23](#_Toc351491844)

[3.5 Test Cases (Unit Test Cases and System Test Cases) 24](#_Toc351491845)

[4. Coding 24](#_Toc351491846)

[4.1 Complete Project Coding 24](#_Toc351491847)

[4.2 Comments and Description of Coding segments 24](#_Toc351491848)

[4.3 Standardization of the coding 24](#_Toc351491849)

[4.4 Code Efficiency 24](#_Toc351491850)

[4.5 Error handling 24](#_Toc351491851)

[4.6 Parameters calling/passing 24](#_Toc351491852)

[4.7 Validation checks 24](#_Toc351491853)

[5. Testing 25](#_Toc351491854)

[5.1 Testing techniques and Testing strategies used 25](#_Toc351491855)

[5.2 Testing Plan used 25](#_Toc351491856)

[5.3 Test reports for Unit Test Cases and System Test Cases 25](#_Toc351491857)

[5.4 Debugging and Code improvement 25](#_Toc351491858)

[6. System Security measures 25](#_Toc351491859)

[6.1 Database/data security 25](#_Toc351491860)

[6.2 Creation of User profiles and access rights 25](#_Toc351491861)

[7. Cost Estimation of the Project along with Cost Estimation Model 26](#_Toc351491862)

[7.1 Estimation of development effort 26](#_Toc351491863)

[8. Reports 26](#_Toc351491864)

[9. Future scope and further enhancement of the Project 26](#_Toc351491865)

[10. Bibliography 27](#_Toc351491866)

[11. Tools/Platform, Hardware and Software Specification 28](#_Toc351491867)

[11.1 Hardware And Software Specification 28](#_Toc351491868)

[11.1.1 Hardware Requirement 28](#_Toc351491869)

[11.1.2 Software Requirement 28](#_Toc351491870)

[11.2 Tools/Platforms used (Hardware/Software): 28](#_Toc351491871)

[11.2.1 Hardware Used 28](#_Toc351491872)

[11.2.2 Software Used 28](#_Toc351491873)

[12. Requirements And Analysis 29](#_Toc351491874)

[12.1 Problem Definition 29](#_Toc351491875)

[12.1.1 Existing System 29](#_Toc351491876)

[12.1.2 WORK To Be Done 29](#_Toc351491877)

[12.2 Requirements Specification 30](#_Toc351491878)

[12.3 Technical specification 30](#_Toc351491879)

[13. Scope Of The Solution 30](#_Toc351491880)

[13.1 E-R Diagram 31](#_Toc351491881)

[14. Database And Table Details 31](#_Toc351491882)

[15. Complete Data Structure 31](#_Toc351491883)

[15.1 Module Description 31](#_Toc351491884)

[15.2 Estimation 31](#_Toc351491885)

[15.3 Data Structure 32](#_Toc351491886)

[15.4 Implementation Methodology 32](#_Toc351491887)

[15.5 List of Reports 33](#_Toc351491888)

[16. Security Mechanism 33](#_Toc351491889)

[17. Future Scope And Further Requirements 33](#_Toc351491890)

# 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

## Preliminary Investigation

## Feasibility Study

## 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 want 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 employees’ payment deadlines regularly and notifies the user when the date is clos 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

## Database design, Procedural Design/Object Oriented Design

|  |
| --- |
| **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)

# Coding

## Complete Project Coding

## Comments and Description of Coding segments

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

//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

## 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

## Parameters calling/passing

## Validation checks

# Testing

## Testing techniques and Testing strategies used

## Testing Plan used

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

## Debugging and Code improvement

# System Security measures

## Database/data security

## 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.

### 7.1 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 DNBSN: Tdev = 2.5\*(Effort)0.35 months

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

Assume that the size of our

The basic COCOMO estimation formula for MMS semidetached software:

Our Effort = 3.0 \* (3.2)1.12 PM

= 11 PM

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

= 6 months

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

= Rs. 120,000

# Reports

# Future scope and further enhancement of the Project

Micro Finance Management System is a complete solution for a micro finance organization. It does most things automatically; a user only needs to provide relevant data. It manages client and employee details, it keeps track of the account, keeps track of all the major to minor transactions of the company to generate a report about the progress of the business. Bill generating facility is also available in this application. For security, it provides a login feature for user with different permission level.

Not only that, it provides remainder about clients’ payment deadline automatically. Using the mobile version of this application a user could keep track of his business even when he is away from office. Both the applications are contain very user friendly and attractive GUI. In short, MFMS is a all in one solution for a micro finance business.

# Bibliography

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

/////////////////////////////////////////////////////////////////////////////////////

# Tools/Platform, Hardware and Software Specification

## Hardware And Software Specification

### Hardware Requirement

* **Disc capacity :** 10 MB of available hard disk space
* **RAM :** 1 GB (32 Bit) or 2 GB (64 Bit)
* **Processor :** 1.6GHz or faster
* DVD-ROM Drive / USB **Port**

### Software Requirement

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

## Tools/Platforms used (Hardware/Software):

### Hardware Used

* Laptop with 2GHZ processor
* 2 GB RAM
* 320 GB hard disk (NTFS File System)

### Software Used

* Windows XP (x86) with Service Pack 3 / Windows Vista (x86) with Service Pack 2 / Windows 7 (x86)
* Microsoft .NET 4.0
* Windows Presentation Framework(WPF)
* Windows Communication Framework(WCF)
* Visual Studio 2010 Express Edition (IDE)
* MySQL Workbench
* Dia for Diagram Drawing & Modeling.

# Requirements And Analysis

## Problem Definition

### Existing System

The microfinance sector is also quite diverse in its use of information systems. Generally there are the following two types:

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.

### WORK To Be Done

The primary roles of the MFMS are to capture information, create new information, store information, and convey information to the user. MFMS can add substantial value in achieving all the objectives of MFI:

* A major advantage of MIS is that it provides easy access to accurate and up-to-date information.
* Detailed information is captured on customers and their activities that can then be used to assess client business to assess impact. It is also useful in tracking historical information of clients.
* Activities, such as disbursements, repayments, deposits, withdrawals and money transfers are completed faster, better controlled and with minimum opportunity for errors.
* Information is produced in user-required formats, which facilitates better understanding, setting priorities, objectives and strategy.
* Key performance indicators provide an overview of the organization’s performance, efficiency and effectiveness of business procedures so that timely adjustments can be made.
* Use of ICT helps make MFI services more interactive, accessible and transparent.
* In terms of innovation, ICT provides full flexibility to structure products and services to the needs of its target group.
* Efficiency and productivity of staff is increased, as they are able to manage more products, customers, and transactions in less time.
* To meet target market needs, introduction of new products and setting procedures is easy and can be quickly applied throughout the branch network.
* It can also provide the flexibility to integrate with other applications and delivery mechanisms
* MFMS will provide low transaction cost, increases productivity, reduces risk of failure, and pushes the boundaries beyond bricks and mortar infrastructure to carryout business.

## Requirements Specification

## Technical specification

* **Front End/ GUI Tools:** Windows Presentation Framework (WPF)
* **IDE:** Visual Studio 2010
* **Framework:** Microsoft .NET 4.0
* **Database:** MySQL
* **Database Tool:** MySQL workbench CE
* **Operating Systems**: Windows XP, Windows 7
* **Cloud Technology**: Google Drive, Google forms

# Scope Of The Solution

## E-R Diagram

# Database And Table Details

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.

|  |  |
| --- | --- |
| **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 |

# Complete Data Structure

## Module Description

## Estimation



## Data Structure

## Implementation Methodology

* Object Oriented Programming methodology will be adopted
* User interface development will be done in MVC architecture using Windows Presentation Framework.
* Relational DBMS MySQL will be used to implement & execute SQL query to database.
* Agile Software Development model will be used while developing this software.

## List of 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

# Security Mechanism

* 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.

# Future Scope And Further Requirements

* 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.