BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

PROGRAMMING PROJECT: BIT 3205

COOPERATIVE SOCIETY MANAGEMENT SYSTEM FOR ZEMA COOPERATIVE SOCIETY

BY

CHIRCHIR BENARD KIPKEMOI

SYSTEM DESIGN SPECIFICATION DOCUMENTATION SUBMITTED IN PARTIAL FULFILMENT FOR THE REQUIREMENTS FOR THE AWARD OF A DEGREE IN INFORMATION TECHNOLOGY

PRESENTED TO: DR. LUCY MBURU

Declaration

I declare that this project is my original work and has not been presented in any other college or University for the award of a Diploma or Degree, and has been duly signed by my supervisor.

Student

Name……. Date…….

Signature……..

Supervisor

Name…….. Date……

Signature……

Table of Contents

Abstract

This project was done for the purpose of automating operations of the Cooperative society. The cooperative is currently using a manual system which is inefficient because of too many errors and hence poor service to members. The new system will streamline all the operations of the cooperative this include processing loans, membership, monthly contribution, repayments etc , keeping members information safe and secure, generating quarterly statements to members.

Data collected during system analysis was first edited and for completeness and consistency. I used statistical package for social sciences as a tool for analysis.

The research found out that automating the operations of the cooperative society will results to efficiency and high productivity. The initial cost of buying computers and software was high but the end it was worth it.

1.0 Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, class diagrams, activity diagrams and other supporting requirement information.

This Software Design Specification provides the design details of Cooperative Management System (CMS) for ZEMA Re cooperative society. It can be traced to a member's requirements and at the same time assessed for quality against a set of predefined criteria for 'good' design.

In systems design we are concerned with producing appropriate design which results in a good-quality information system which is:-.

- Easy to use;
- Provide the correct function for end-users;
- Rapid in retrieving data and moving between different screen views of the data;
- Reliable
- Secure;
- Well integrated with other system

In the software engineering context, design focuses on four major areas of concern, data, architecture, interfaces, and components.

1.1 Purpose

The purpose of the Software Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding

of what is to be built and how it is expected to built. The Software Design Document provides information necessary to provide description of the details for the software and system to be built.

1.2 Scope

This Software Design Document is for a base level system which will work as a proof of concept for the use of building a system that provides a base level of functionality to show feasibility for large scale production use. This Software Design is focused on the base level system and critical parts of the system. For this particular Software Design Document, the focus is placed on generation of the documents and modification of the documents. The system will be used in conjunction with other pre-existing systems.

This document contains a complete description of the design of CMS. The basic architecture is Application server from a client/server paradigm. The basic forms will be in Oracle and coded using PL-SQL.
The designated cooperative staff in charge of the CMS will have full access to make changes, as he/she deems necessary. The changes could include, but not limited to, changing the entry into the system, data collected from members, and the ability of one user to make changes to the system.

1.3 System Design Constraints

The system design is directly constrained by the user requirements specification, which is has been produced as a result of system analysis. This will describe the functions that are required by the user which must be implemented as part of the design. As well as the environmental constraints on design which are a result of the hardware and software environment of implementation. These include:

- A hardware limitation such as the system requires an increase in memory to the currently used machines.

- Integrating data from the existing system to the new system
- Interfacing with other application may be difficult to implement.
- Oracle database is to be used which is expensive to buy and requires mature database performance techniques.
- The system is going to be tested internally and the users may not know what is required of them during testing phase.

1.4 Design Goals and Objectives

The goal of the CMS is to design a system which delivers the functions required by ZEMA cooperative society to support the business of the cooperative.

The importance of software design can be stated with a single word – ***quality***.

- Design is the place where quality is fostered in software development.
- Design provides us with representations of software that can be assessed for quality.
- Design is the only way that we can accurately translate a customer's requirements into a finished software product or system.
- Software design serves as the foundation for all software engineering and software maintenance steps that follow.
- Without design, we risk building an unstable system – one that will fail when small changes are made; one that may be difficult to test; one whose quality cannot be assessed until late in the software engineering process, when time is short and many shillings have already been spent.

Objectives

Flexibility - The system design should enable future requirements of the cooperative society to be incorporated without too much difficulty.

Ease of Use - The system design should enable implementation of system that is user friendly and easy to use.

Secure - The system design should incorporate methods that will restrict access to authorized users only.

1.5 Overview of Document

The Software Design Document is divided into sections with various subsections. The sections of the Software Design Document are:

- Section 2 is the Architectural Design that specifies the design entities that collaborate to perform all the functions included in the system. Each of these entities has an Abstract description concerning the services that it provides to the rest of the system. In turn, each design entity is expanded into a set of lower-level design operations that collaborate to perform its services.
- Section 3 concerns the File and Data Structure Design.
- Section 4 contains the Normalization which is a design activity that is used to optimize the logical storage of data within a given database.
- Section 5 discusses the Conclusion of the System Design Document which shows how the system will be able to fulfill user requirements.

2.0 System Architecture

2.1 Introduction

Software architecture is the blueprint of a software system. It provides an overview of the composition and functionality of the given software system. Just like a structural architect, a software architect needs to analyze the requirements, determine components that should be used to build the system, and support the project by guiding and solving problems all along the execution cycle. Architecting software is like planning for war. Past experiences are very useful here. Strategizing gives you an edge. Understanding of the domain provides you with capability to analyze the requirements and envisioning a solution. Exposure to various tools and technologies imparts in you the ability to choose among the various solutions available, and anticipate and solve technical problems.

2.2 The Client-Server model

The client-server architectural model is a distributed system model which shows how data and processing is distributed across a range of processors. The major components of this model are:

i. A set of stand-alone servers which offer services to other sub-systems. Examples of servers are print servers which offer printing services, file servers which offer file management services and a compile server which offers language translation services.

ii. A set of clients that call on the services offered by servers. These are normally sub-systems in their own right. There may be several instances of a client program executing concurrently.

iii. A network which allows the clients to access these services. In principle, this is not really necessary as both the clients and the servers could run on a single machine. In practice, however, this model would not be used in such a situation.

Clients must know the names of the available servers and the services that they provide. However, servers need not know whether the identity of clients or how many clients there are. Clients access the services provided by as server through remote procedure calls.

2.3 Design Approach

Software architecture embodies modularity; that is, software is divided into separately named and addressable components called modules that are integrated to satisfy problem requirements.

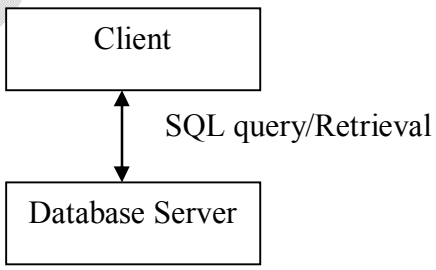
Monolithic software (i.e., a large program, comprised of a single module) cannot be easily grasped by a reader. The number of control paths, span of reference, number of variables, and overall complexity would make understanding close to impossible.

This leads to a "divide and conquer" conclusion-it's easier to solve a complex problem when you break it into manageable pieces. The top-down approach design which involves specifying the overall control architecture of the application before designing the individual modules will be used.

2.4 Architectural Design

Architectural design moulds program structure and data structure, defining interfaces that enable data to flow throughout the program.

The proposed system will be developed using two-tier architectural design. The application running on the workstation will be a large program containing all the application logic and display code. It retrieves data from a separate database server.



2.5 Logical Design

The logical design describes the functions required of a system, that is, what is to be done, not how it will be done. Logical design is not concerned with hardware and software requirements but rather with the processes to be performed.

2.6 Use Case Diagrams

A use case diagram is a diagram that shows a set of use cases and actors and their relationships.

2.6.1 Actors

An actor is a role that a user plays with respect to the system. There are several actors in this system mainly:

2.6.2 Member

A member is a user who applies for loans and repays them based on agreed parameters.

2.6.3 Accountant

The accountant will be charged with the role of processing membership requests and loans.

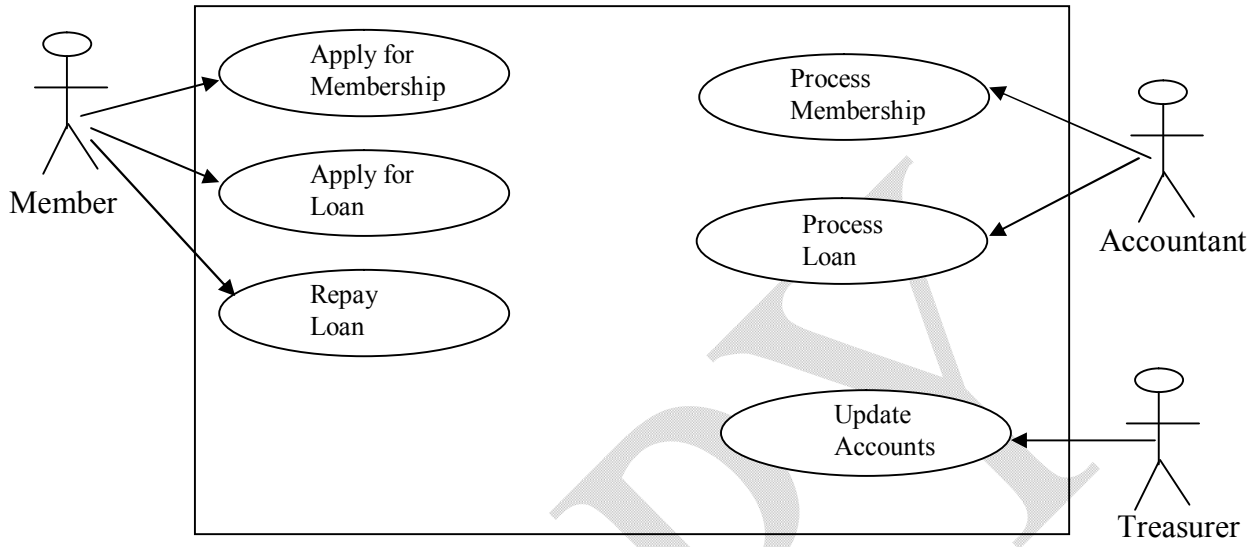
2.6.4 Treasurer

The treasurer is a user who is entrusted with the role of checking what the accountant is doing and also updating the accounts in the system.

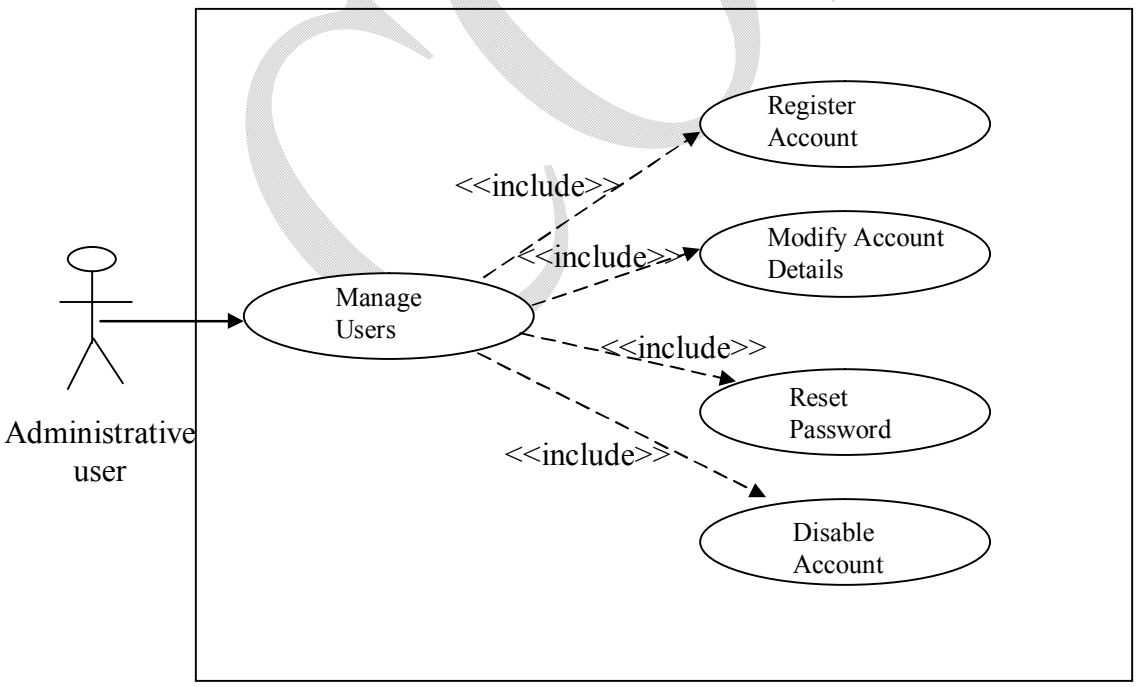
2.6.5 Administrative user

The administrative user is a user who administers the system by overseeing accounts creation and administration.

The use case diagram below depicts the Member, Accountant and Treasure actors and their roles



And the use case diagram below depicts the Administrative user and his/her roles

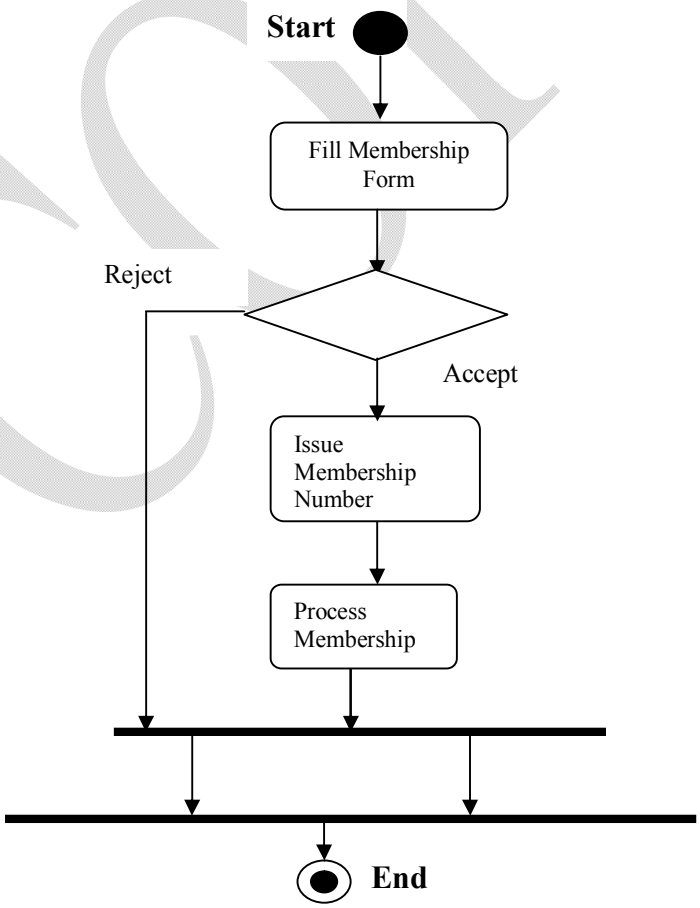


2.7 Activity Diagram

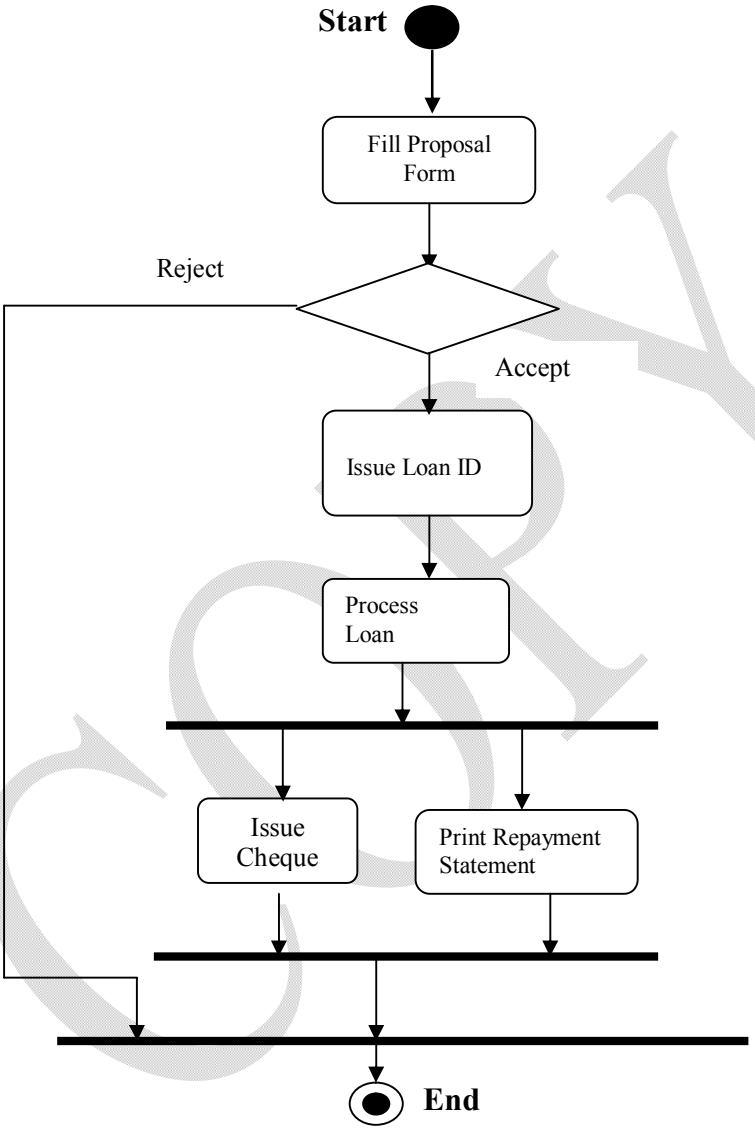
An activity diagram is essentially a flowchart, showing flow of control from activity to activity. The activity diagrams are used to model the dynamic aspects of a system. For the most part, this involves modeling the sequential (and possibly concurrent) steps in a computational process.

Activity diagrams may stand alone to visualize, specify, construct, and document the dynamics of a society of objects, or they may be used to model the flow of control of an operation. Whereas interaction diagrams emphasize the flow of control from object to object, activity diagrams emphasize the flow of control from activity to activity. An activity is an ongoing non atomic execution within a state machine.

The activity diagram below shows the activities involved when a new member requests to join the cooperative society.

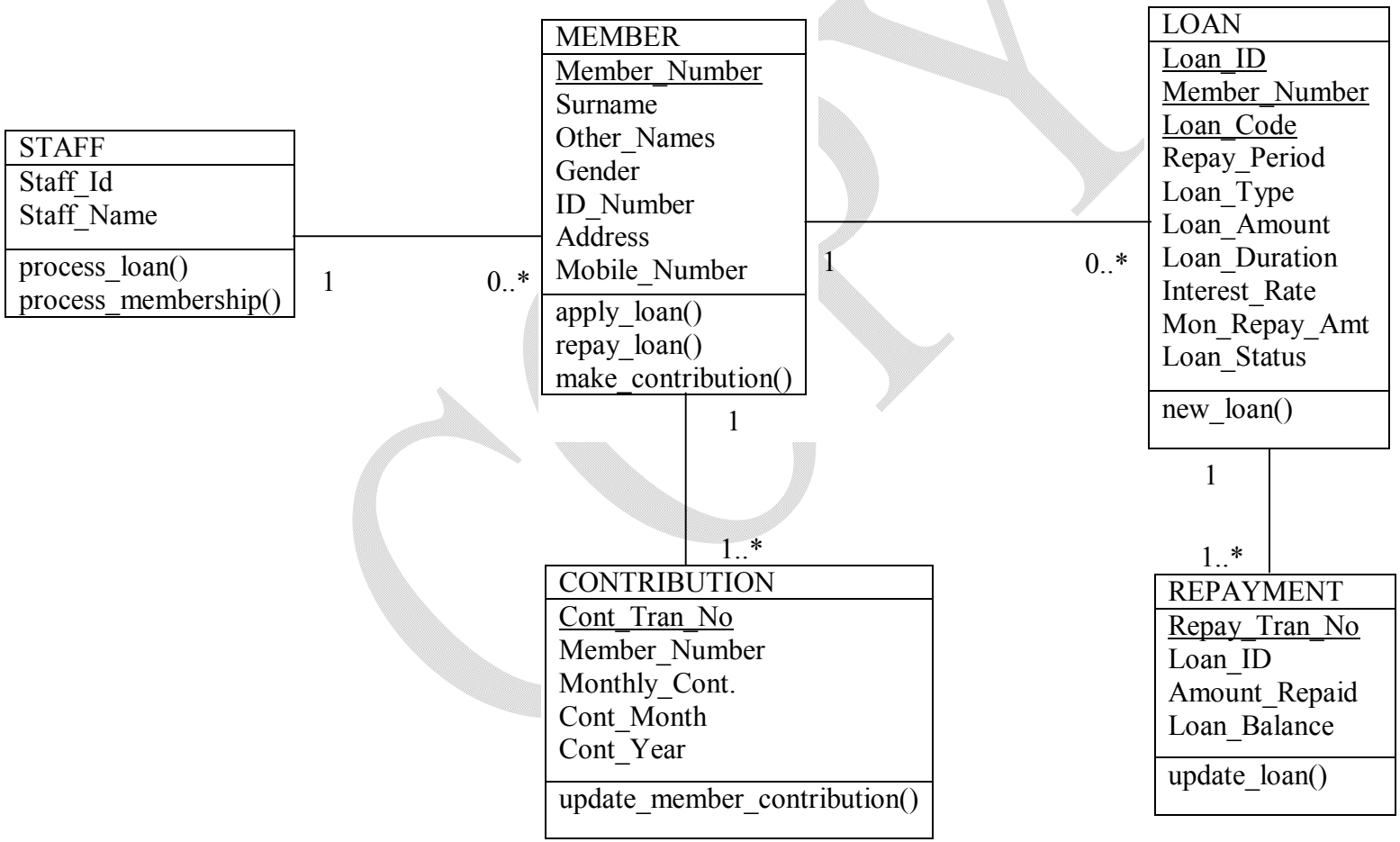


The activity diagram below depicts the activities involved when a member applies for a loan and how it is processed.

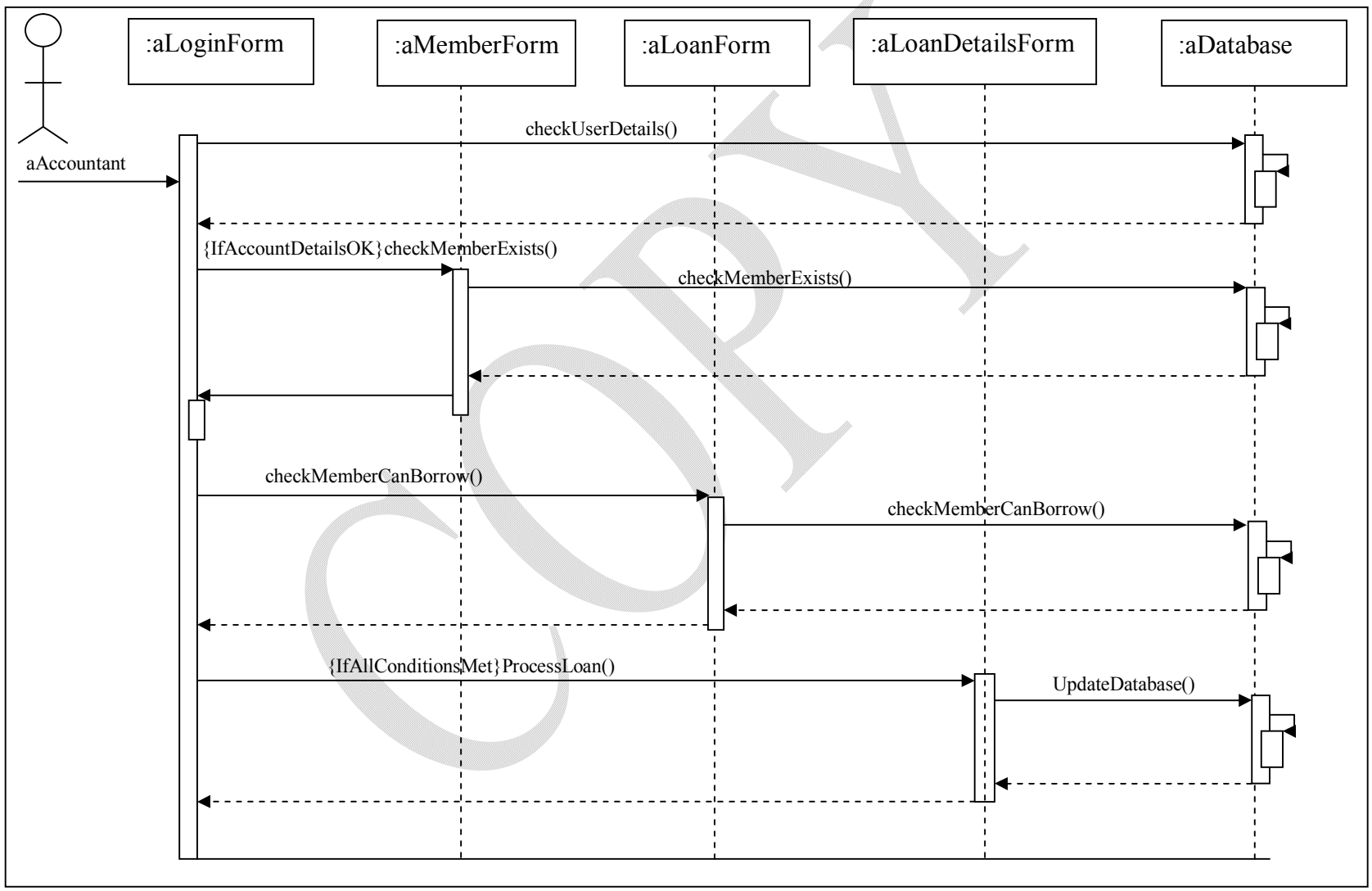


2.8 Class Diagram

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.



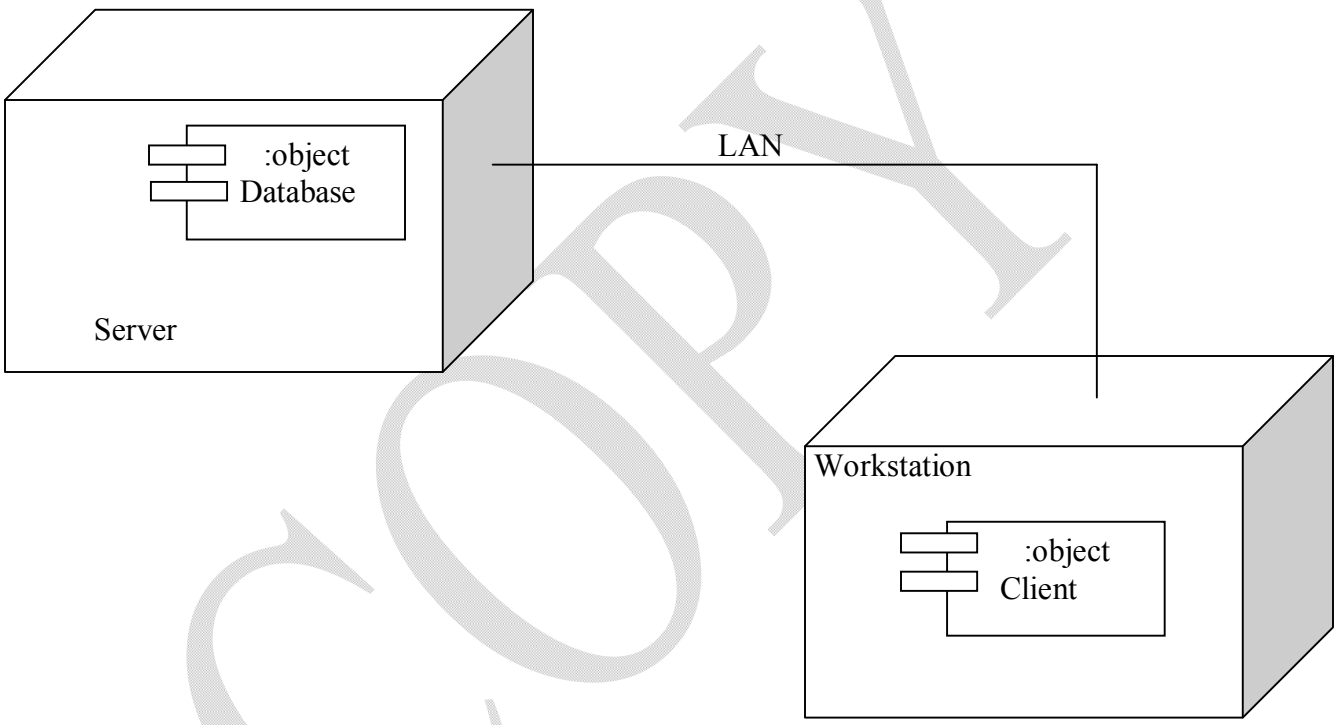
2.9 Sequence Diagram



2.10 Deployment Diagram

A deployment diagram is used to provide a solid blue print for hardware deployment.

The database and the client will communicate using the Local Area Network that is in place. The client application will run from any web browser e.g. Internet explorer, Mozilla etc.
A server will communicate to client as shown below:



3.0 Database Design

3.1 Introduction

A database can be defined as a structured organization of data stored in such a way that there is minimum duplication to provide consistency of the data and also provide access for many users, independent of programs that are used.

The standard user and application program interface to a relational database is SQL. SQL statements are used both for interactive queries for information from a relational database and for gathering data for reports.

In addition to being relatively easy to create and access, a relational database has the important advantage of being easy to extend. After the original database creation, a new data category can be added without requiring that all existing applications be modified.

The database of this project is going to be done in Oracle 9g Relational Database Management System.

Reasons for choice of Oracle Database:-

- Oracle comes with new versions with new features implemented in new version while the features of earlier versions still being maintained.
- Oracle databases tend to be backwards compatible. Also when Oracle releases a new version, their documentation contains a list of all the features new to that version thus makes it user friendly for one to learn the new features.
- Oracle is used for almost all large application and one of the main applications in which oracle takes its major presence is management systems.
- Also with the features available in oracle with the earlier versions in market the oracle company keeps upgrading and releasing new products into market, new

versions releases which serves better than the earlier versions and thus the performance is improved much in later versions and thereby retaining the market growth and thus proves greater satisfaction to the customers using this technology.

- Oracle is a major database which along with its added features passes the ACID test, which is important in insuring the integrity of data. This is very important because data is the heart of any system in organization.

A reliable and adequate database system has the following properties:

Atomicity: That is Results of a transaction's execution are either all committed or all rolled back.

Consistency: The database is transformed from one valid state to another valid state. Illegal transactions aren't allowed and, if an integrity constraint can't be satisfied then the transaction is rolled back.

Isolation: The results of a transaction are invisible to other transactions until the transaction is complete thus increasing the security on data.

Durability: Once committed (completed), the results of a transaction are permanent and survive future system and media failures and thus ensuring maintenance and protection of data.

3.2 Normalization

Normalisation is a design activity that is used to optimize the logical storage of data within a database. It involves simplification of entities and removal of duplication of data. The main purpose of purpose data normalization is to group data items together into database structures of table and records which are simple to understand, accommodate change, contain a minimum of redundant data and are free of insertion, deletion and update anomalies. These anomalies can occur when database is modified, resulting into erroneous data.

The normalization of tables used in this project is shown below. The first tables depicts all the required fields in the un-normalized form.

UNF	UNF LEVEL	1NF	2NF	3NF
Loan_ID	1			
Loan_Code	1			
Repay_Period	1			
Loan_Type	1			
Loan_Amount	1			
Loan_Duration	1			
Interest_Rate	1			
Loan_Status	1			
Repay_Tran_No	1			
Amount_Repaid	1			
Loan_Balance	1			
Cont_Tran_No	1			
Monthly_Cont.	1			
Cont_Month	1			
Cont_Year	1			
Member_Number	2			
Surname	2			
Other_Names	2			
Gender	2			
ID_Number	2			
Address	2			
Mobile_Number	2			

The table below shows the 1NF which is the second stage of normalization after identifying all the fields required.

UNF	UNF LEVEL	1NF	2NF	3NF
Loan_ID	1	Loan_ID		
Loan_Code	1	Loan_Code		
Repay_Period	1	Repay_Period		
Loan_Type	1	Loan_Type		
Loan_Amount	1	Loan_Amount		
Loan_Duration	1	Loan_Duration		
Interest_Rate	1	Interest_Rate		
Loan_Status	1	Loan_Status		
Repay_Tran_No	1	Repay_Tran_No		
Amount_Repaid	1	Amount_Repaid		
Loan_Balance	1	Loan_Balance		
Cont_Tran_No	1	Cont_Tran_No		
Monthly_Cont.	1	Monthly_Cont.		
Cont_Month	1	Cont_Month		
Cont_Year	1	Cont_Year		
Member_Number	2			
Surname	2	Member_Number		
Other_Names	2	Surname		
Gender	2	Other_Names		
ID_Number	2	Gender		
Address	2	ID_Number		
Mobile_Number	2	Address		
		Mobile_Number		

The table below depicts the 2NF.

UNF	UNF LEVEL	1NF	2NF	3NF
Loan_ID	1	Loan_ID	Loan_ID	
Loan_Code	1	Loan_Code	Member_Number	
Repay_Period	1	Repay_Period	Loan_Code	
Loan_Type	1	Loan_Type	Repay_Period	
Loan_Amount	1	Loan_Amount	Loan_Type	
Loan_Duration	1	Loan_Duration	Loan_Amount	
Interest_Rate	1	Interest_Rate	Loan_Duration	
Loan_Status	1	Loan_Status	Interest_Rate	
Repay_Tran_No	1	Repay_Tran_No	Loan_Status	
Amount_Repaid	1	Amount_Repaid	Repay_Tran_No	
Loan_Balance	1	Loan_Balance	Amount_Repaid	
Cont_Tran_No	1	Cont_Tran_No	Loan_Balance	
Monthly_Cont.	1	Monthly_Cont.	Cont_Tran_No	
Cont_Month	1	Cont_Month	Monthly_Cont.	
Cont_Year	1	Cont_Year	Cont_Month	
Member_Number	2		Cont_Year	
Surname	2	Member_Number		
Other_Names	2	Surname	Member_Number	
Gender	2	Other_Names	Surname	
ID_Number	2	Gender	Other_Names	
Address	2	ID_Number	Gender	
Mobile_Number	2	Address	ID_Number	
		Mobile_Number	Address	
			Mobile_Number	

The below shows the 3NF.

UNF	UNF LEVEL	1NF	2NF	3NF
Loan_ID	1	Loan_ID	Loan_ID	Loan_ID
Loan_Code	1	Loan_Code	Member_Number	Member_Number
Repay_Period	1	Repay_Period	Loan_Code	Loan_Code
Loan_Type	1	Loan_Type	Repay_Period	Repay_Period
Loan_Amount	1	Loan_Amount	Loan_Type	Loan_Type
Loan_Duration	1	Loan_Duration	Loan_Amount	Loan_Amount
Interest_Rate	1	Interest_Rate	Loan_Duration	Loan_Duration
Loan_Status	1	Loan_Status	Interest_Rate	Interest_Rate
Mon_Repay_Amt	1	Mon_Repay_Amt	Mon_Repay_Amt	Mon_Repay_Amt
Repay_Tran_No	1	Repay_Tran_No	Loan_Status	Loan_Status
Amount_Repaid	1	Amount_Repaid	Repay_Tran_No	
Loan_Balance	1	Loan_Balance	Amount_Repaid	Loan_Code
Cont_Tran_No	1	Cont_Tran_No	Loan_Balance	Loan_Type
Monthly_Cont.	1	Monthly_Cont.	Cont_Tran_No	Max_Repay_Period
Cont_Month	1	Cont_Month	Monthly_Cont.	
Cont_Year	1	Cont_Year	Cont_Month	Repay_Tran_No
Member_Number	2		Cont_Year	Loan_ID
Surname	2	Member_Number		Amount_Repaid
Other_Names	2	Surname	Member_Number	Loan_Balance
Gender	2	Other_Names	Surname	
ID_Number	2	Gender	Other_Names	Cont_Tran_No
Address	2	ID_Number	Gender	Member_Number
Mobile_Number	2	Address	ID_Number	Monthly_Cont.
		Mobile_Number	Address	Cont_Month
			Mobile_Number	Cont_Year
				Member_Number
				Surname
				Other_Names
				Gender
				ID_Number
				Address
				Mobile_Number

3.3 Database Description

Loan_Details Table			
FIELD NAME	**DATA TYPE**	**SIZE**	**REMARK**
Loan_ID	Number	5	Primary Key
Member_Number	Number	4	Foreign Key
Loan_Code	Varchar2	2	
Repay_Period	Number	2	
Loan_Type	Varchar2	30	
Loan_Amount	Number	7	
Loan_Duration	Number	2	
Interest_Rate	Number	2	
Mon_Repay_Amt	Number	7	
Loan_Status	Varchar2	2	
User_Created	Varchar2	3	
Date_Created	Date		

Loan_Type Table			
FIELD NAME	**DATA TYPE**	**SIZE**	**REMARK**
Loan_Code	Varchar2	3	Primary Key
Loan_Type	Varchar2	20	
Max_Repay_Period	Number	2	
User_Created	Varchar2	3	
Date_Created	Date		

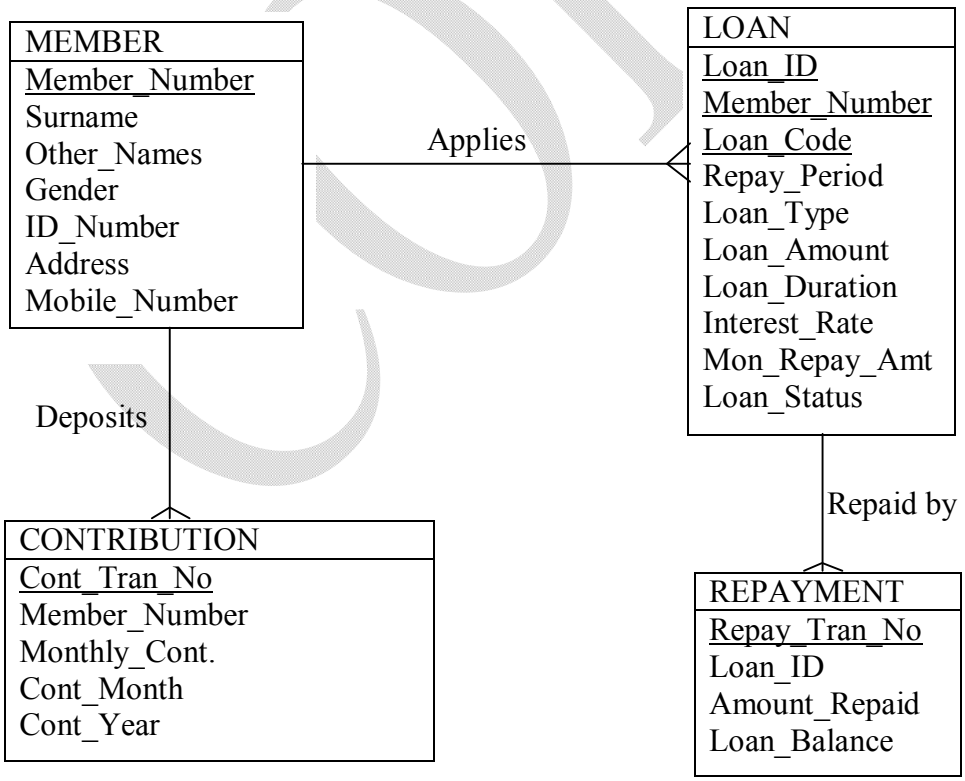
Loan_Repayment Table			
FIELD NAME	**DATA TYPE**	**SIZE**	**REMARK**
Repay_Tran_No	Number	5	Primary Key
Loan_ID	Number	5	Foreign Key
Amount_Repaid	Number	7	
Loan_Balance	Number	7	
User_Created	Varchar2	3	
Date_Created	Date		

Contribution Table			
FIELD NAME	**DATA TYPE**	**SIZE**	**REMARK**
Cont_Tran_No	Number	5	Primary Key
Member_Number	Number	5	Foreign Key
Monthly_Cont.	Number	5	
Cont_Month	Number	2	
Cont_Year	Number	4	
Total_Shares	Number	7	
User_Created	Varchar2	3	
Date_Created	Date		

Member Table			
FIELD NAME	**DATA TYPE**	**SIZE**	**REMARK**
Member_Number	Number	5	Primary Key
Surname	Varchar2	30	
Other_Names	Varchar2	60	
Gender	Varchar2	7	
ID_Number	Number	10	
Address	Varchar2	100	
Mobile_Number	Number	10	
User_Created	Varchar2	3	
Date_Created	Date		

3.4 Entity Relationship Diagram

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database.



4.0 User Interface Design

4.1 Internal machine interfaces

There is an existing client/server network and the system will be installed on the server. The users would access the information through the network on their local machines. Anytime a user updates a record the system is automatically updated on the server. When one person accesses a record the system automatically locks that record until that user releases the record by saving it. This reduces chances of updated a wrong data by any other client

4.2 External system interfaces

Printers and other machines can also be accessed via the network. Key boards, mouse and any other accessory are connected directly to the local machines. The server should be installed with antivirus to protect it from any viral attack. Internet should be installed in to the organization to ensure that there is online antivirus update.

4.3 Human interface

Only the technical staff are allowed to the server room this ensures that no errors that can occur unknowingly. Users are only allowed to access the system through their local machines and this is after they have been rightly authenticated and accepted by the system.

4.4 User interface design

This concept depicts how the user will be able to communicate with the system for it to perform the required tasks. Cooperative management system will use oracle forms that will be launched through a web browser.

4.4.1 Description of the user interface

Command buttons: – which will act as a way of issuing commands to the system to do a certain task such as save, edit, delete, search etc.

Drop down menus:- User commands that are in words and can often be used if need be.

Icons:- Theses are images which represents a certain function e.g. a printer icon to indicate print function

Command line:- The user can write command lines in SQL

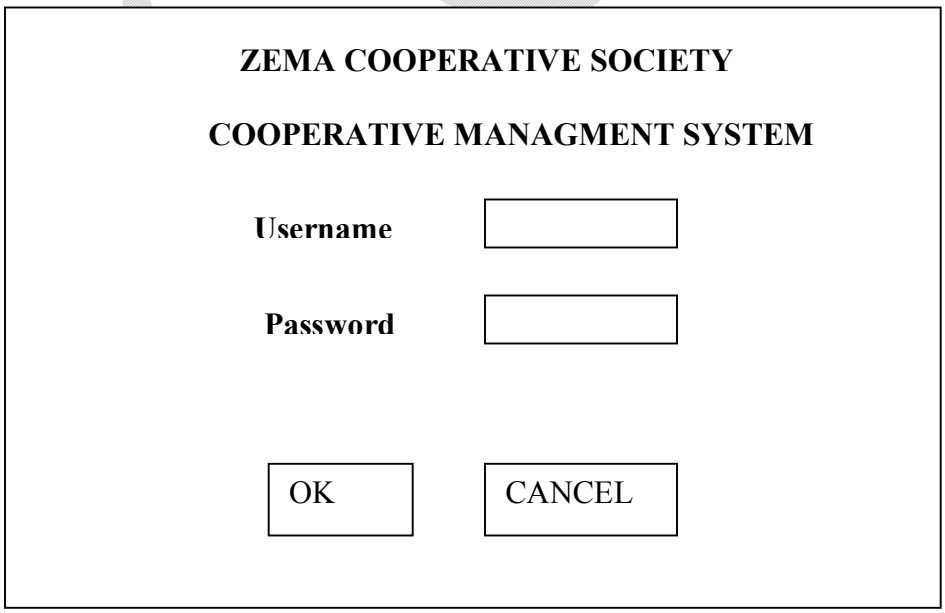
Pop ups – User prompts which pops on the screen as the user uses the system.

4.4.2 Screen Images

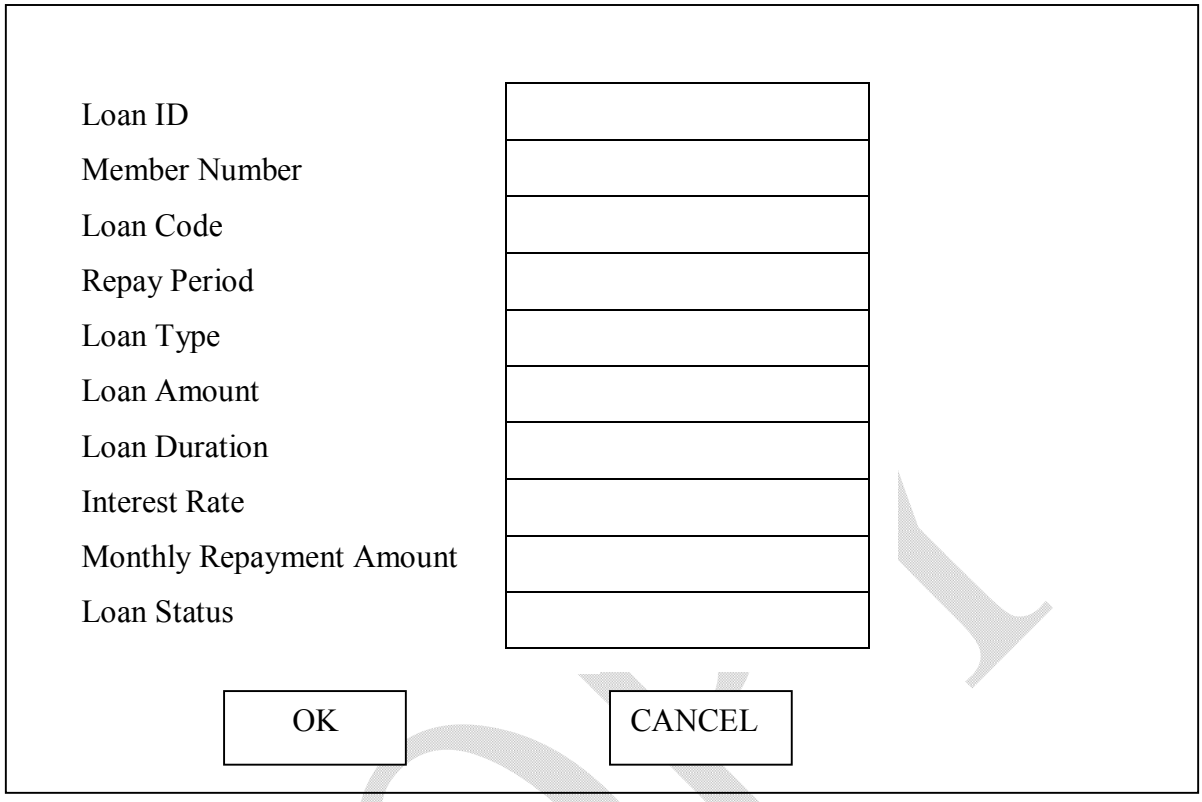
Screen images depicts what the user will see when perform certain takes in the system, for instance he/she can get a pop up window asking him/her to do something.

Below are some form designs that will be used in the system

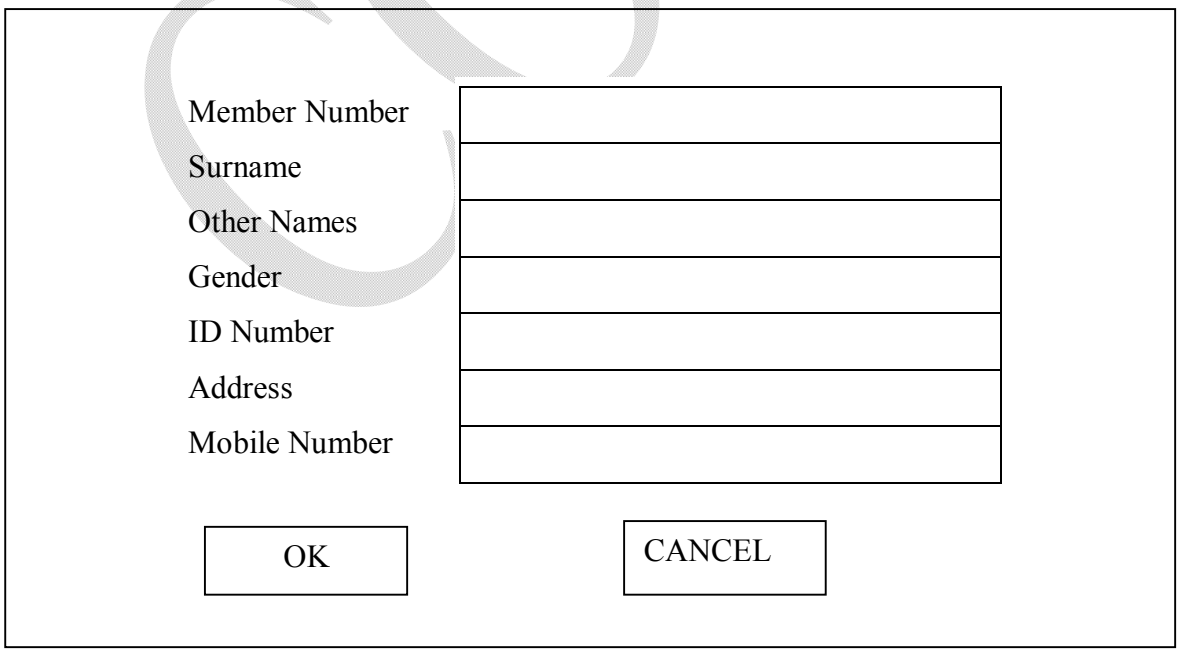
4.4.2.1 Login Form



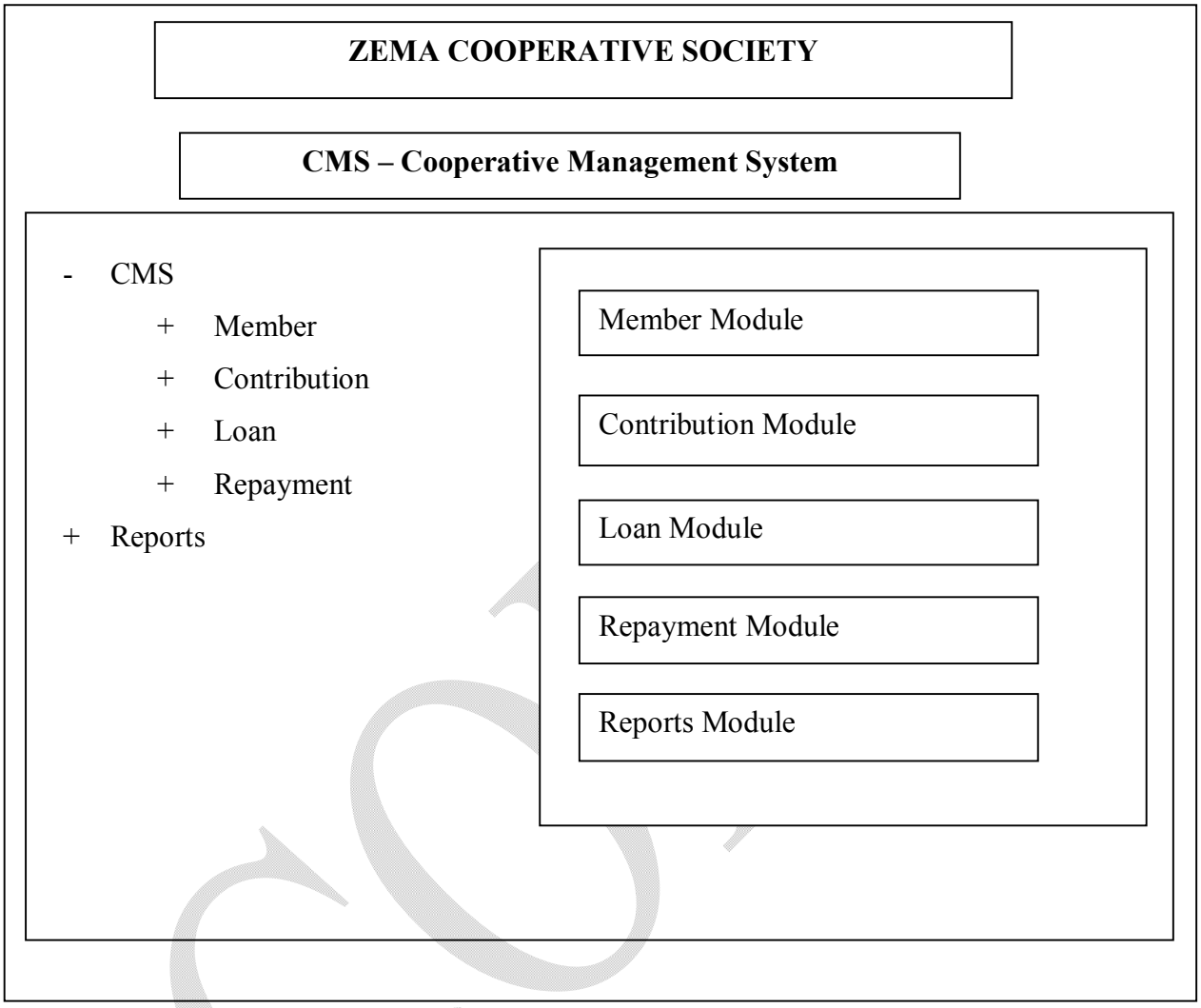
4.4.2.2 Loan Form



4.4.2.3 Member Form



4.4.2.3 Main Form



4.4.2 Components Available

Save button – used to save edited or entered data

Delete- this is used to erase unwanted entries

Move to first- when this button is clicked the system moves to the first record

Move to last- when clicked the system moves to the last record.

Move to next- when clicked the system moves to the next record to be viewed.

Move to previous – when clicked the system moves to the previous entry viewed.

Search – this button is used to search for an entry in the system.

Update – these button is used to update an edited record.

4.4.4 Software Context

The system will include several modules (i.e. Member, Loan, Contribution, Repayment and Reports module) , a relationship to connect all the tables so that they can relate and share the data to accomplish all the tasks the system is expected to do.

4.4.5 Expected software response

The software is expected to respond normally with no or minimal errors because the system designer has done through testing before handing over to team of testers. These errors which might come up should be corrected before the system id delivered to the client.

4.4.6 Packaging and installation issues

The system should be packaged in a copyrighted compact disk and accompanied by its documentation.

5.0. Conclusion

The system requirement specification was an important input to the design specification. This is because for the system to be accepted by the users, it must be designed in relation to their proposed needs.

From the design it is concluded that the system will be able to fulfill the described user requirements and tests plan document will be designed to help in verification of this argument.

6.0 References

1. Scott W. Ambler (2009) UML 2 Class Diagrams.
2. Ulric J. Gelinas, Richard B. Dull (2009) Accounting Information Systems
3. Frank B. Watts (2004) Engineering documentation control handbook