SOFTWARE REQUIREMENT SPECIFICATION

Library Management System

Prepared by:

Kaustubh Wade - 160410116050

Naisargi Kothari - 160410116051

Karansinh Matroja - 160410116055

Jeet Meghpara – 160410116056

Academic Year:

2018-2019

Sardar Vallabhbhai Institute of Technology

Contents

1. Introduction

With the increase in the number of readers, better management of libraries system is required. The Library management system focuses on improving the management of libraries in a city or town. “What If you can check whether a book is available in the library through your phone?” or “what if instead of having different library cards for different libraries you can just have one ?” or “you can reserve a book or issue a book from your phone sitting at your home!”. The Integrated Library Management system provides you the ease of issuing, renewing, or reserving a book from a library within your town through your phone. The Integrated Library Management system is developed on the android platform which basically focuses on issuing, renewing and reserving a book.

1.1 Purpose

The purpose of this document is to present a detailed description of the Library management System. It will explain the purpose of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to institutional libraries.

The main purpose of this project is to maintain an easy circulation system between clients and the libraries, to issue books using single library card, also to search and reserve any book from different available libraries and to maintain details about the user (fine, address, phone number).Moreover, the user can check all these features from their home.

1.2 Scope

Manually updating the library system into an android based application so that the user can know the details of the books available and maximum limit on borrowing from their computer and also through their phones. The ILM System provides information's like details of the books, insertion of new books, deletion of lost books, limitation on issuing books, fine on keeping a book more than one month from the issued date. Also, user can provide feedback for adding some new books to the library.

1.3 Definition, Acronyms, Abbreviation:

- User: A general login id assigned to most users
- Client: Intended users for the software
- JAVA: platform independence
- SQL: Structured query Language
- Data Flow Diagram (DFD): Shows data flow between entities.
- Interface: Something used to communicate across different mediums.
- ER: Entity Relationship
- IDE: Integrated Development Environment
- SRS: Software Requirement Specification
- Use Case: A broad level diagram of the project showing a basic overview.
- ISBN: International Standard Book Number

1.4 References

- Wikipedia (http://en.wikipedia.org/)

2. Overall Description

2.1 Product Perspective

The proposed Library Management System which is being developed by Innovative Library Management Solutions team is an on-line Library Management System. This System will provide a search functionality to facilitate the search of resources. This search will be based on various categories viz. book name or the ISBN. Also Advanced Search feature is provided in order to search various categories simultaneously. Further the library staff personnel can add/update/remove the resources and the resource users from the system.

2.2 Product Features

There are two different users who will be using this product:

- Librarian who will be acting as the administrator
- Student of the University who will be accessing the Library online.

The features that are available to the Librarian are:

- A librarian can issue a book to the student
- Can view the different categories of books available in the Library
- Can view the List of books available in each category
- Can take the book returned from students
- Add books and their information of the books to the database
- Edit the information of the existing books.
- Can check the report of the issued Books.
- Can access all the accounts of the students.

The features available to the Students are:

- Can view the different categories of books available in the Library
- Can view the List of books available in each category
- Can own an account in the library
- Can view the books issued to him

- Can put a request for a new book
- Can view the history of books issued to him previously
- Can search for a particular book

2.3 User Classes and Characteristics

There are various kinds of users for the product. Usually web products are visited by various users for different reasons. The users include:

- Students who will be using the above features by accessing the Library online.
- Librarian who will be acting as the controller and he will have all the privileges of an administrator.

2.4 Operating Environment

The product will be operating in any operating system the only requirement to use this online product would be the internet connection.

2.5 Design and Implementation Constraints

The product is developed using JAVA. The backend database for this SQL Server. The product is accomplished with login facility so that specific function is available to specific student.

2.6 User Documentation

The product will include user manual. The user manual will include product overview, complete configuration of the used software (such as SQL server), technical details, backup procedure and contact information which will include email address.

2.7 Assumptions and Dependencies

The assumptions are:

- The coding should be error free
- The system should be user-friendly so that it is easy to use for the users
- The information of all users, books and libraries must be stored in a database that is accessible by the website
- The system should have more storage capacity and provide fast access to the database
- The system should provide search facility and support quick transactions
- The Library System is running 24 hours a day
- Users may access from any computer that has Internet browsing capabilities and an Internet connection

- Users must have their correct usernames and passwords to enter into their online accounts and do actions

The dependencies are:

- The specific hardware and software due to which the product will be run
- On the basis of listing requirements and specification the project will be developed and run
- The end users (admin) should have proper understanding of the product
- The system should have the general report stored
- The information of all the users must be stored in a database that is accessible by the Library System
- Any update regarding the book from the library is to be recorded to the database and the data entered should be correct

3. External Interface Requirement

3.1 User Interfaces

GUI

Describes the graphical user interface if present. This section should include a set of screen dumps or mockups to illustrate user interface features.

1. Description

The user interface must be customizable by the administrator

2. Criticality

This issue is essential to the overall system. All the modules provided with the software must fit into this graphical user interface and accomplish to the standard defined.

3. Technical issues

In order to satisfy this requirement, the design should be simple, and all the different interfaces should follow a standard template. There will be the possibility of changing colors and images, plus switching between interfaces with the minimum impact for the users.

4. Risks

To reduce the circumstances under which this requirement might not able to be satisfied, all the designers must have been developed web sites previously and they must be aware of html restriction and cross browsers implementations before starting the designing. In order to reduce the probability of this occurrence the entire design team will be trained in basic html development and macromedia fireworks; this tool will be used instead of Photoshop.

5. Dependencies with other requirements

All user interfaces should be able to interact with the user management module and a part of the interface must be dedicated to the login/logout module

3.2 Login Interface:

In case the user is not yet registered, he can enter the details and register to create his account. Once his account is created he can 'Login' which asks the user to type his username and password. If the user entered either his username or password incorrectly then an error message appears.

3.3 Search:

The member or librarian can enter the type of book he is looking for and the title he is interested in, then he can search for the required book by entering the book name.

3.4 Categories View:

Categories view shows the categories of books available and provides ability to the librarian to add/edit or delete category from the list.

3.5 Librarian's Control Panel:

This control panel will allow librarian to add/remove users; add, edit, or remove a resource. And manage lending options.

4. System Features

4.1 Description and Priority

Proposed Database is intended to store, retrieve, update, and manipulate information related to university which include

- Books availability
- Staff information
- Student details
- My Account
- Calculation of fines

4.2 Stimulus / Response Sequences

The administrator can Login and Logout. When the administrator logs into the library system. The system will check for validity of login .If the Login and password are valid, the response to this action is the administrator will be able to modify, view, add, deleting and all other functions that can be performed on the database.

4.3 Functional Requirements

This section gives the list of Functional and nonfunctional requirements which are applicable to the Library Management System.

5. Other Non-functional Requirements

5.1 Performance Requirement

The proposed system that we are going to develop will be used as the Chief performance system within the different campuses of the university which interacts with the university staff and students. Therefore, it is expected that the database would perform functionally all the requirements that are specified by the university.

- The performance of the system should be fast and accurate
- Library Management System shall handle expected and non-expected errors in ways that prevent loss in information and long downtime period. Thus, it should have inbuilt error testing to identify invalid username/password
- The system should be able to handle large amount of data. Thus, it should accommodate high number of books and users without any fault

5.2 Safety Requirement

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup so that the database is not lost. Proper UPS/inverter facility should be there in case of power supply failure.

5.3 Security Requirement

- System will use secured database
- Normal users can just read information, but they cannot edit or modify anything except their personal and some other information.
- System will have different types of users and every user has access constraints
- Proper user authentication should be provided
- No one should be able to hack users' password
- There should be separate accounts for admin and members such that no member can access the database and only admin has the rights to update the database.

5.4 Software Quality attributes

- There may be multiple admins creating the project, all of them will have the right to create changes to the system. But the members or other users cannot do changes

- The project should be open source
- The Quality of the database is maintained in such a way so that it can be very user friendly to all the users of the database
- The user be able to easily download and install the system

5.5 Business Rules

A business rule is anything that captures and implements business policies and practices. A rule can enforce business policy, decide, or infer new data from existing data. This includes the rules and regulations that the System users should abide by. This includes the cost of the project and the discount offers provided. The users should avoid illegal rules and protocols. Neither admin nor member should cross the rules and regulations.

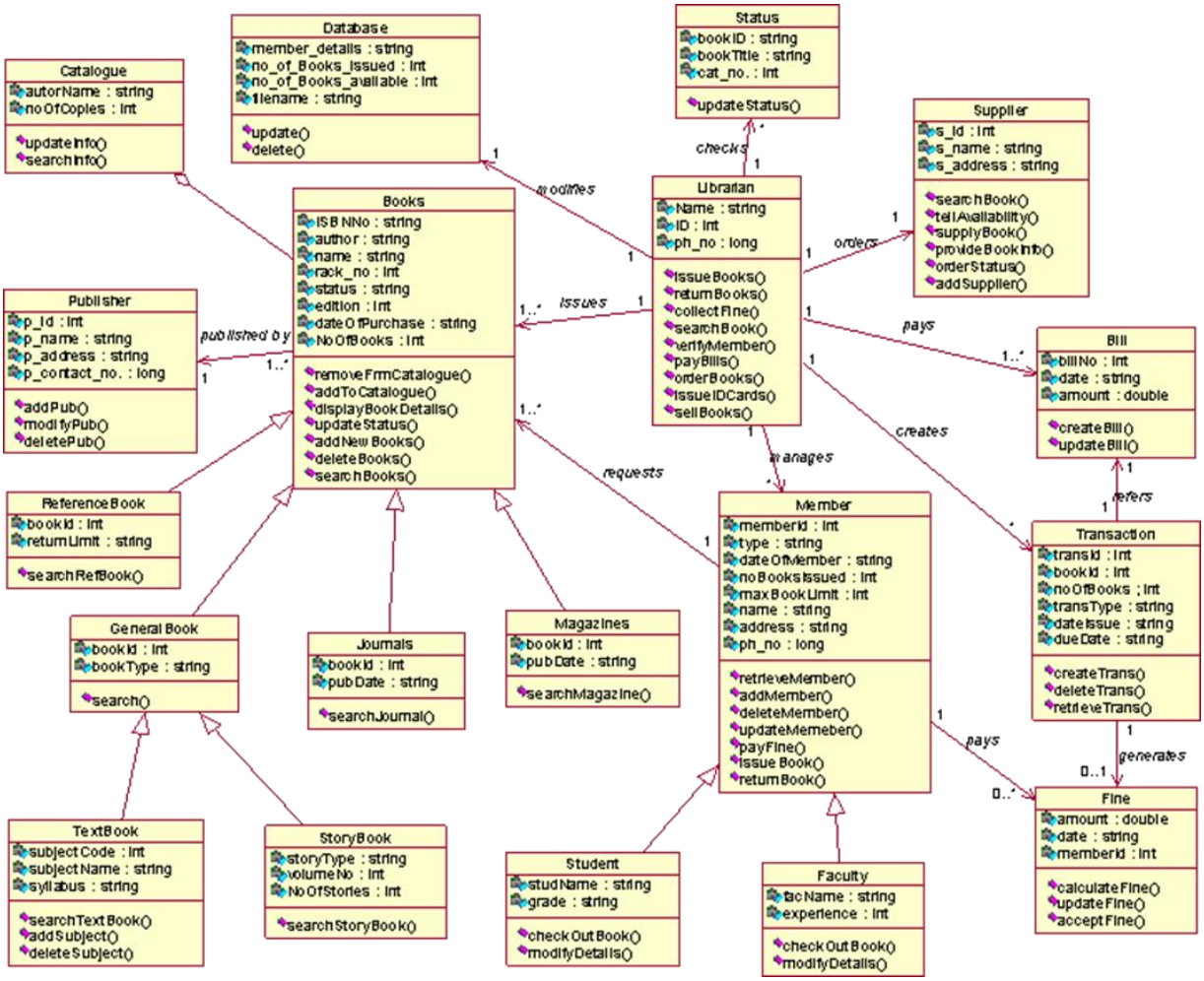
6. Other Requirements

6.1 Data and Category Requirement

There are different categories of users namely teaching staff, Librarian, Admin, students etc. Depending upon the category of user the access rights are decided. It means if the user is an administrator then he can be able to modify the data, delete, append etc. Similarly, there will be different categories of books available. According to the categories of books their relevant data should be displayed. The categories and the data related to each category should be coded in the particular format.

6.2 Class Diagram

A class is an abstract, user-defined description of a type of data. It identifies the attributes of the data and the operations that can be performed on instances (i.e. objects) of the data. A class of data has a name, a set of attributes that describes its characteristics, and a set of operations that can be performed on the objects of that class. The classes' structure and their relationships to each other frozen in time represent the static model. In this project there are certain main classes which are related to other classes required for their working. There are different kinds of relationships between the classes as shown in the diagram like normal association, aggregation, and generalization. The relationships are depicted using a role name and multiplicities. Here 'Librarian', 'Member' and 'Books' are the most important classes which are related to other classes.


Database
member_details : string
no_of_Books_Issued : int
no_of_Books_available : int
filename : string
update()
delete()
Status
bookID : string
bookTitle : string
cat_no. : int
updateStatus()
Supplier
s_id : int
s_name : string
s_address : string
searchBook()
tellAvailability()
supplyBook()
provideBookInfo()
orderStatus()
addSupplier()
Catalogue
autorName : string
noOfCopies : int
updateInfo()
searchInfo()
checks
modifies
orders
Librarian
Name : string
ID : int
ph_no : long
issueBooks()
returnBooks()
collectFine()
searchBook()
verifyMember()
payBills()
orderBooks()
issueIDCards()
sellBooks()
Books
ISBNNo : string
author : string
name : string
rack_no : int
status : string
edition : int
dateOfPurchase : string
NoOfBooks : int
removeFrmCatalogue()
addToCatalogue()
displayBookDetails()
updateStatus()
addNewBooks()
deleteBooks()
searchBooks()
issues
Publisher
p_Id : int
p_name : string
p_address : string
p_contact_no. : long
addPub()
modifyPub()
deletePub()
published by
pays
Bill
billNo : int
date : string
amount : double
createBill()
updateBill()
creates
manages
requests
refers
ReferenceBook
bookId : int
returnLimit : string
searchRefBook()
Member
memberId : int
type : string
dateOfMember : string
noBooksIssued : int
maxBookLimit : int
name : string
address : string
ph_no : long
retrieveMember()
addMember()
deleteMember()
updateMemeber()
payFine()
issueBook()
returnBook()
Transaction
transId : int
bookId : int
noOfBooks : int
transType : string
dateIssue : string
dueDate : string
createTrans()
deleteTrans()
retrieveTrans()
pays
generates
GeneralBook
bookId : int
bookType : string
search()
Journals
bookId : int
pubDate : string
searchJournal()
Magazines
bookId : int
pubDate : string
searchMagazine()
Fine
amount : double
date : string
memberId : int
calculateFine()
updateFine()
acceptFine()
TextBook
subjectCode : int
subjectName : string
syllabus : string
searchTextBook()
addSubject()
deleteSubject()
StoryBook
storyType : string
volumeNo : int
NoOfStories : int
searchStoryBook()
Student
studName : string
grade : string
checkOutBook()
modifyDetails()
Faculty
facName : string
experience : int
checkOutBook()
modifyDetails()

Practical 4

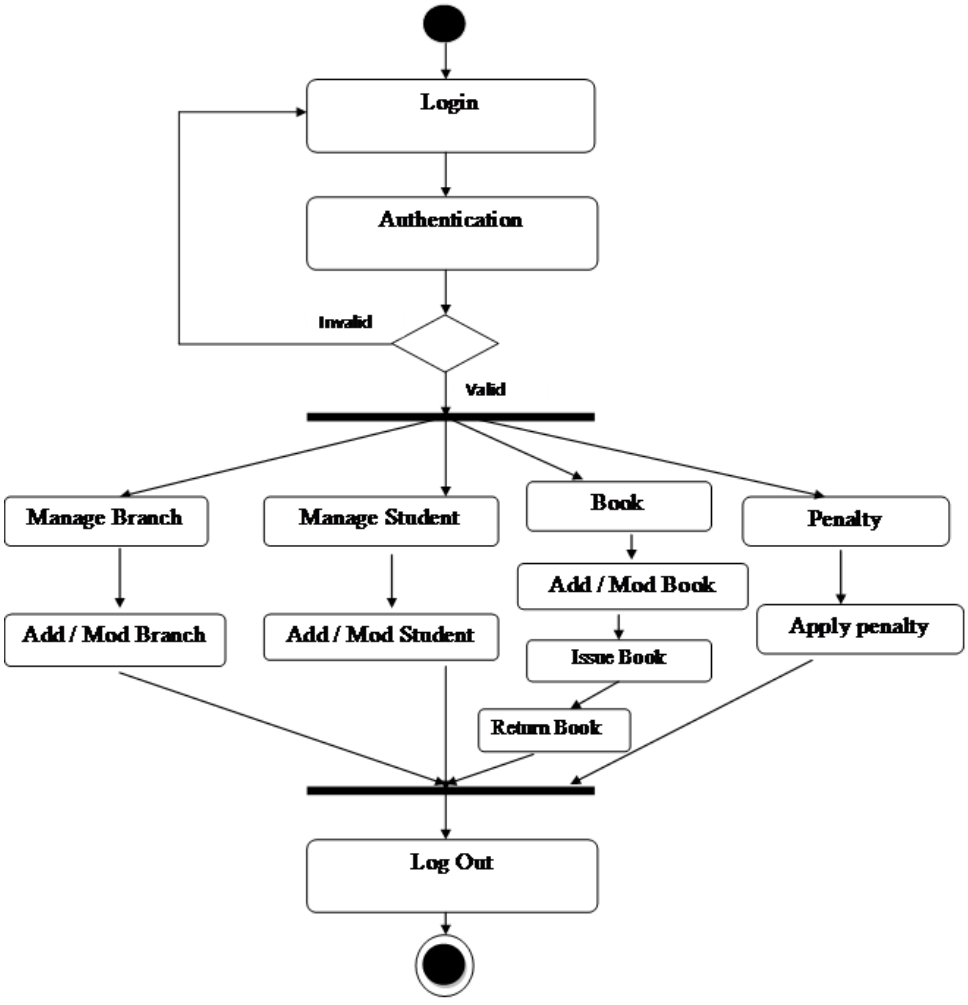
Aim: Documentation of system Design using procedural approach or object-oriented approach

Description:

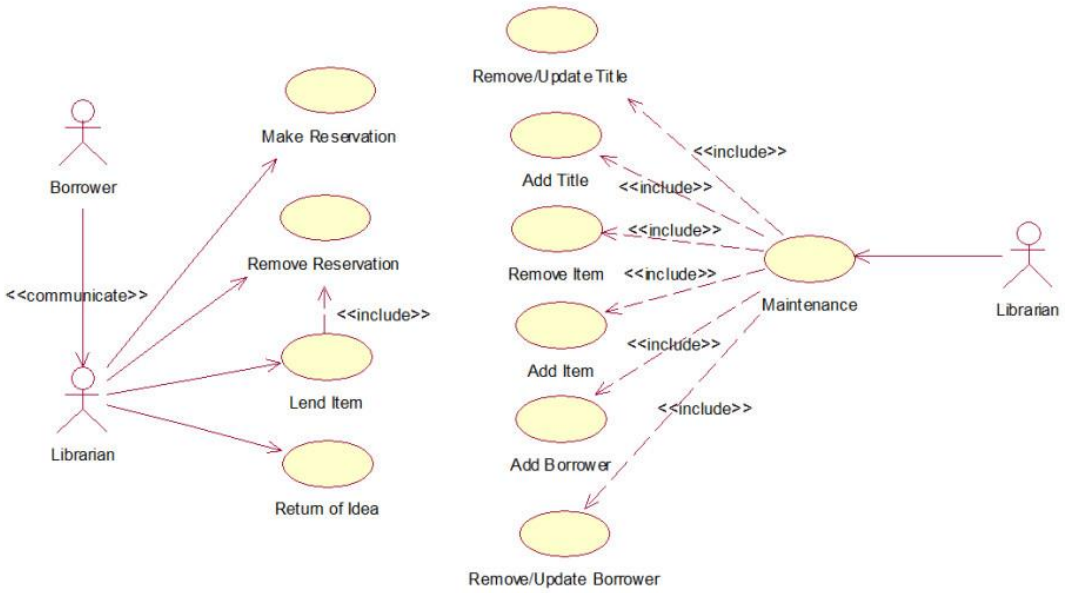
A college library management is a project that manages and stores books information electronically according to student's needs. The system helps both students and library manager to keep a constant track of all the books available in the library. It allows both the admin and the student to search for the desired book. It becomes necessary for colleges to keep a continuous check on the books issued and returned and even calculate fine. This task if carried out manually will be tedious and includes chances of mistakes. These errors are avoided by allowing the system to keep track of information such as issue date, last date to return the book and even fine information and thus there is no need to keep manual track of this information which thereby avoids chances of mistakes.

Thus, this system reduces manual work to a great extent allows smooth flow of library activities by removing chances of errors in the details.

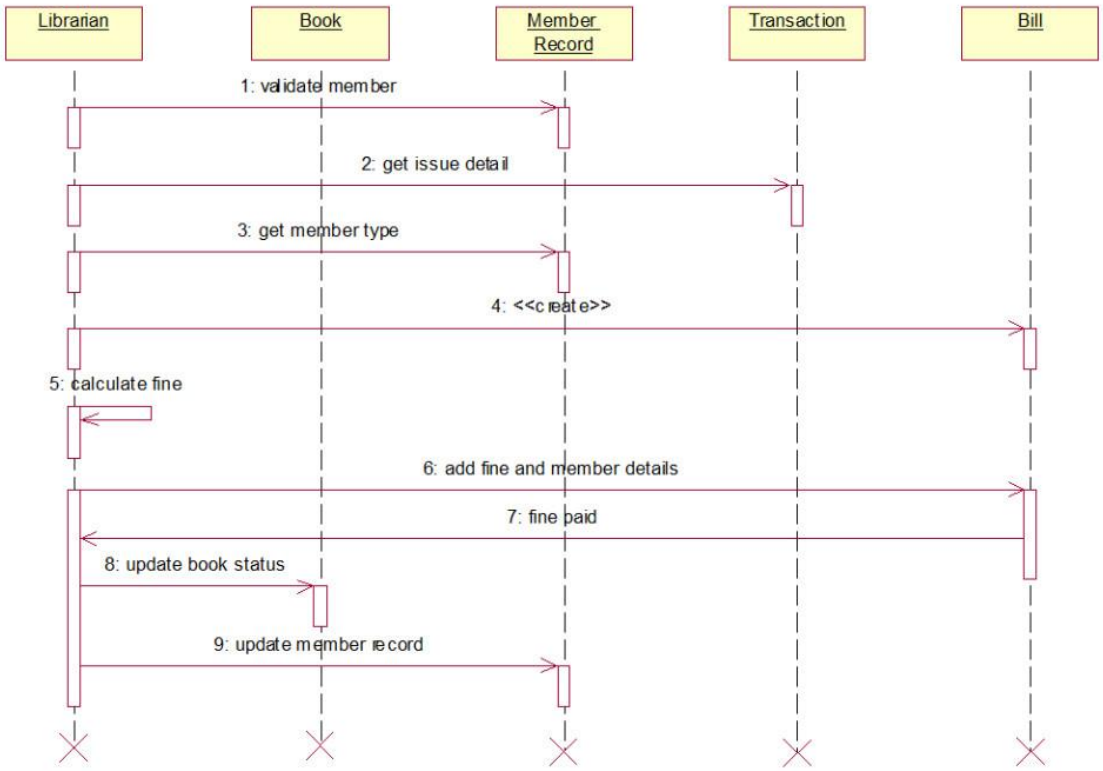
Activity Diagram:



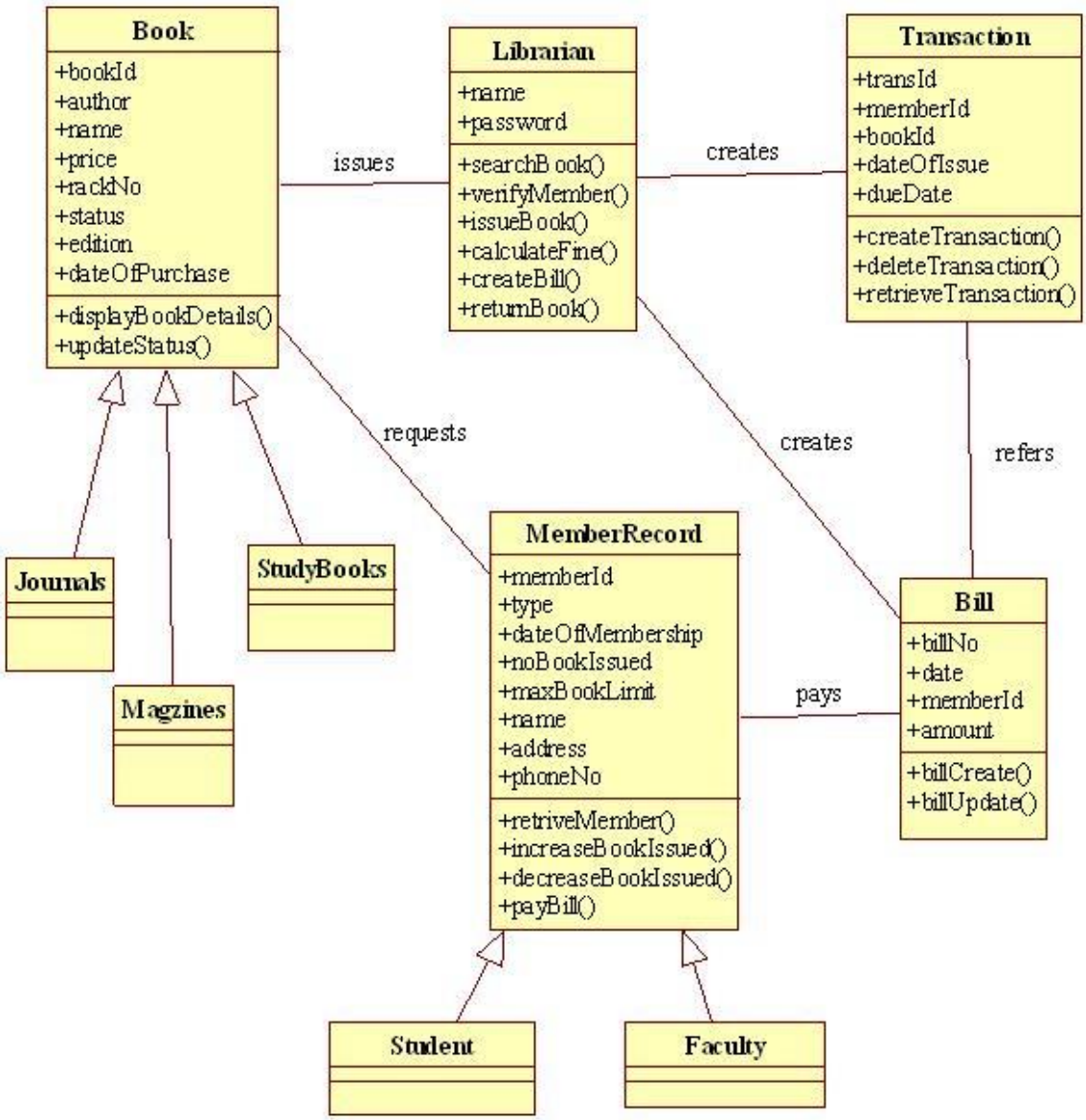
Use Case Diagram:



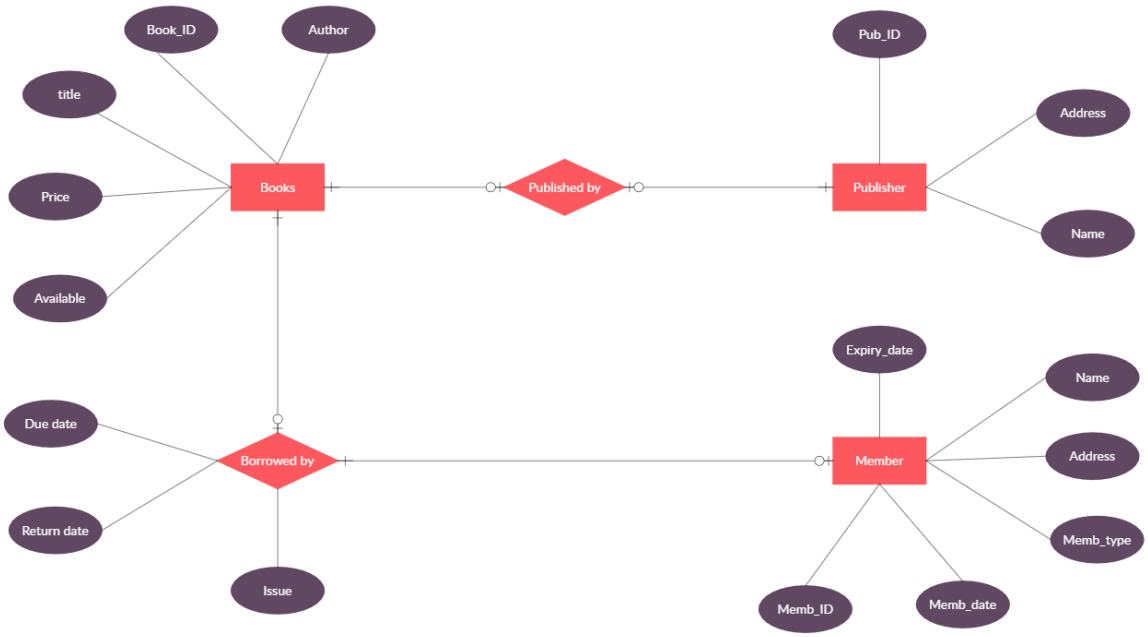
Sequence Diagram:



Class Diagram:



E-R Diagram:


Book_ID
Author
title
Price
Available
Books
Published by
Pub_ID
Address
Name
Publisher
Due date
Return date
Borrowed by
Issue
Expiry_date
Name
Address
Memb_type
Member
Memb_ID
Memb_date

Practical 5

Documentation and Data Dictionary

Description:

Data management describes the persistent data stored by the system and the data management infrastructure required for it. The system will use the MySQL database engine for storing data. This will allow the database to be easily integrated with and accessed by the rest of the system. The database will retain user information (IdNo, name, email address, etc.), event postings and configuration data such as authorized administrator and department. Each of these items will be a separate table. An individual could not have more than one user account.

Data Dictionary for Library management system

Table Name: ActivityLog				
Description : store all the activity done by user in the system				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
LogId	int	Not null	-	-
LogEmp	nchar(10)	Not null	-	-
LogTime	datetime	Not null	-	-
LogActivity	varchar(MAX)	Not null	-	-

Field	**Example Data**
LogId(PK)	113
LogEmp	M0001
LogTime	2011-02-16 16:35:07.000
LogActivity	Information of PublisherID P0003 has been updated

Table 3.4.1 Data Dictionary for ActivityLog table

Table Name: Admin				
Description : store the information of user who used the library system				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
Admin_ID	nvarchar(50)	Not null	-	-
Admin_Name	nvarchar(50)	Not null	-	-
Admin_Level	nvarchar(50)	Not null	-	Format : 1,0
Password	nvarchar(50)	Not null	-	-
Admin_ic	nvarchar(50)	Not null	-	-
admin_contact	nvarchar(50)	Not null	-	-
admin_email	nvarchar(50)	Not null	-	-
admin_address	nvarchar(MAX)	Not null	-	-

Field	**Example Data**
Admin_ID	M0001
Admin_Name	Tan Chaur Chuan
Admin_Level	1
Password	12341234
Admin_ic	880704-35-5263
admin_contact	04-3984851
admin_email	sdfsdf@hotmail.com
admin_address	30, lintang perai 2, taman chai leng, 34567 pahang.

Table 3.4.2 Data Dictionary for Admin table

Table Name: Book				
Description : store the information of the books				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
ISBN	nvarchar(50)	Not null	-	-
BookTitle	nvarchar(50)	Not null	-	-
Author	nvarchar(50)	Not null	-	-
PublisherID	nvarchar(50)	Not null	-	-
Language	nvarchar(50)	Not null	-	-
Category	nvarchar(50)	Not null	-	-
Description	nvarchar(MAX)	Not null	-	-
BookCover	nvarchar(MAX)	Not null	-	-

Field	**Example Data**
ISBN(PK)	9781587132049
BookTitle	Handphone King
Author	C.Y
PublisherID	P0001
Language	Chinese
Category	Technology
Description	A book which show the latest information of all brand handphone
BookCover	Handphone.JPG

Table 3.4.3 Data Dictionary for Book table

Table Name: BookComment				
Description : to store the comment for particular book				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
CID	Int	Not null	-	-
ISBN	Nvarchar(50)	Not null	-	-
UserID	Nvarchar(max)	Not null	-	-
Comment	Nvarchar(max)	Not null	-	-

Field	**Example Data**
CID	13
ISBN	1234567891234
UserID	M0001
Comment	This book is very interesting…thanks

Table 3.4.4 Data Dictionary for BookComment table

Table Name: BookCopy				
Description : to store the quantities of books and the detail of each book				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
BarcodeID	nvarchar(50)	Not null	-	-
ISBN	nvarchar(50)	Not null	-	-
Status	nvarchar(50)	Not null	-	Format : L, A, N
PurchasePrice	Money	Not null	-	-
purchaseDate	Datetime	Not null	-	-

Field	**Example Data**
barcodeID	978158713204901
ISBN	9781587132049
Status	L
PurchasePrice	200.0000
PurchaseDate	2011-02-16 00:00:00.000

Table 3.4.5 Data Dictionary for BookCopy table

Table Name: News				
Description : to post the latest news at web site's main page				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
ID	Int	Not null	-	-
Date	Date	Not null	-	-
[content]	Nvarchar(50)	Not null	-	-

Field	**Example Data**
ID	9781587132049
Date	2011-01-13
[content]	The popular book 'The Lord of the Ring' is now available !!!

Table 3.4.6 Data Dictionary for News table

Table Name: LibraryDetail				
Description : to store the information of the library				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
libno	Char(10)	Not null	-	-
Libname	varchar(50)	Not null	-	-
Libadd1	varchar(50)	Not null	-	-
Libadd2	varchar(50)	Not null	-	-
Libposcode	Char(5)	Not null	-	-
Libstate	varchar(50)	Not null	-	-
Libtown	varchar(50)	Not null	-	-
Libphone	varchar(50)	Not null	-	-
Libfax	varchar(50)	Not null	-	-
Libemail	varchar(50)	Not null	-	Format : abc@abc.abc
Libweb	varchar(50)	Not null	-	-

Field	Example Data
libno	001
Libname	Chen Library
Libadd1	30, lintang perai 5,
Libadd2	Taman putang,
Libposcode	13506
Libstate	Pulau Pinang
Libtown	Butterworth
Libphone	04-3859451
Libfax	04-3225645
Libemail	librarycs@hotmail.com
Libweb	www.google.com

Table 3.4.7 Data Dictionary for LibraryDetail table

Table Name: Reservation				
Description : to record the book reservation for the member				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
UserID	Nvarchar(50)	Not null	-	-
BarCodeId	Nvarchar(50)	Not null	-	-
DateReserve	date	Not null	-	-

Field	**Example Data**
UserID	M0001
BarCodeId	584961352652401

Table 3.4.8 Data Dictionary for Reservation table

Table Name: LostBook				
Description : to keep the information of lost book				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
ID	int	Not null	-	-
Userid	varchar(50)	Not null	-	-
Barcodeid	varchar(50)	Not null	-	-
ISBN	varchar(50)	Not null	-	-
LostDate	Date	Not null	-	-

Field	**Example Data**
ID	3
Userid	M0002
Barcodeid	978158713204901
ISBN	9781587132049
LostDate	2011-02-16

Table 3.4.7 Data Dictionary for LostBook table

Table Name: RetalInfo				
Description : to keep the book transaction detail				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
userID	nvarchar(50)	Not null	-	-
BarCodeID	nvarchar(50)	Not null	-	-
DateReturned	Datetime	-	-	-
DateRented	Datetime	Not null	-	-
DateDue	Datetime	Not null	-	-
Total Fine	money	-	-	-

Field	**Example Data**
userID	M0003
BarCodeID	123456789123401
DateReturned	2011-03-02 00:00:00.000
DateRented	2010-09-09 00:00:00.000
DateDue	2010-10-10 00:00:00.000
Total Fine	200.0000

Table 3.4.7 Data Dictionary for Rental Info table

Table Name: Publisher				
Description : to store the publisher so it is available when register book				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
PublisherId	Nvarchar(50)	Not null	-	-
PublisherName	nvarchar(50)	Not null	-	-

Field	**Example Data**
PublisherId	P0001
PublisherName	Tan Khen Khen

Table 3.4.7 Data Dictionary for Publisher table

Table Name: User				
Description : to store the information of the member				
Fields	**Data Type**	**Null/Not Null**	**Default Value**	**Rules**
UserID	Char(10)	Not null	-	-
Username	varchar(50)	Not null	-	-
Useraddress	varchar(50)	Not null	-	-
UserPhone	varchar(50)	Not null	-	-
UserIC	Char(5)	Not null	-	-
UserRegDate	varchar(50)	Not null	-	-
AvailableBook	varchar(50)	Not null	-	-
Userpass	varchar(50)	Not null	-	-
Userphoto	varchar(50)	Not null	-	-
Useremail	varchar(50)	Not null	-	Format : abc@abc.abc
UserExpiredDate	varchar(50)	Not null	-	-

Field	**Example Data**
UserID	M001
Username	Ooi Yee Neng
Useraddress	30 lintang talang 2, taman perai. 13600 Perai, Penang.
UserPhone	04-3568956
UserIC	880407-35-5266
UserRegDate	2011-02-16 00:00:00.000
AvailableBook	4
Userpass	12345678
Userphoto	Tan_chen_khen.JPG
Useremail	mrtan@hotmail.com
UserExpiredDate	2012-03-05 00:00:00.000

Table 3.4.7 Data Dictionary for User table

SARADAR VALLABHBHAI PATEL INSTITUE OF TECHNOLOGY, VASAD

SOFTWARE ENGINEERING

Software Design Description

Date: 07/04/2019

ONLINE LIBRARY MANAGEMENY SYSTEM

BY:

Kaustubh Wade	(160410116050)
Naisargi Kothari	(160410116051)
Karansinh Matroja	(160410116055)
Jeet Meghapara	(160410116056)

Contents

1.0. Introduction

1.1. Purpose and Scope

The purpose of this Design Document is to present the system design at a level that can be directly traced to the specific system objective along with providing more detailed data, functional, and behavioral requirements. This Design Document will verify that the current design meets all of the explicit requirements contained in the system model as well as the implicit requirements desired by the customer.

1.2. Glossary

- LMS – Library management system
- SRS – Software requirements specification
- PC – Personal Computer
- HDD - Hard Disc Drive
- RAM – Random Access Memory
- ISME – International school of management excellence
- IE – Microsoft Internet Explorer
- SQL – Structured Query Language
- RD – Requirements Documentation
- DD – Design Documentation

1.3. References

www.google.com

www.scribd.com

www.docstok.com

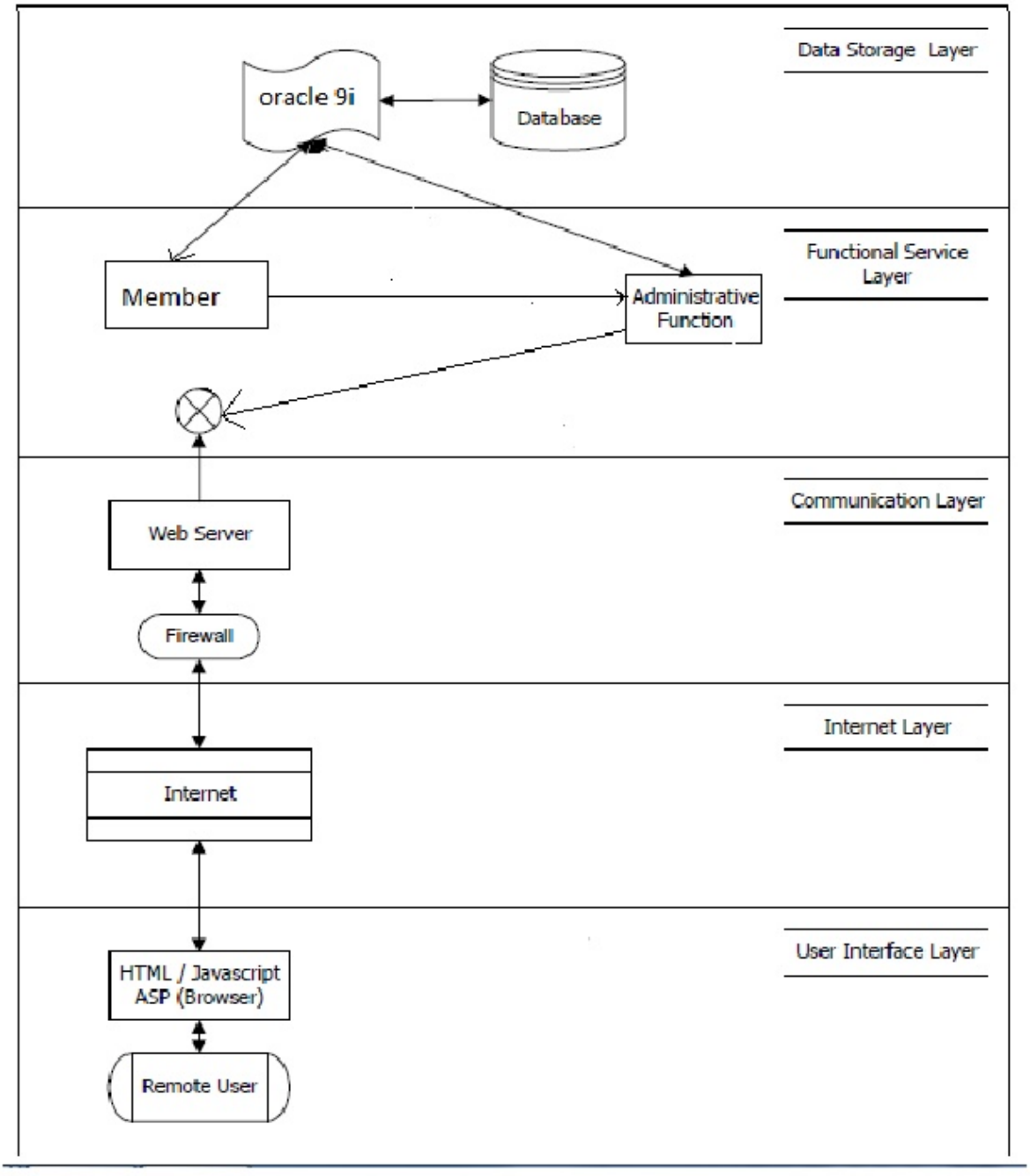
www.Wikipedia.org

1.4. Overview of Document

The overall system design objective is to provide an efficient, modular design that will reduce the system's complexity, facilitate change, and result in an easy implementation. This will be accomplished by designing a strongly cohesion system with minimal coupling. In addition, this document will provide interface design models that are consistent, user friendly, and will provide straightforward transitions through the various system functions.

2.0. System Architecture

The online Library System is a client-server-based system, which contains the following layers: user interface



Internet communication, functional service, and data storage layers.

Data transfers occur in both directions in the system. The users input or data request is sent using either an internet browser or through the windows client. This data then connects to the system either through the internet or, in the case of an onsite connection, through the LAN connection. In the case of an internet connection, the data is required to pass through the system's firewall, for security purposes, prior to connecting to the web server. Local personnel, once validated within the

system, will be connected directly to the application server. In the functional services layer, the data input or request is routed to the appropriate functional module in accordance with the user's login and account type. Through these modules, the users will interact with the database via the SQL server.

3.0.Data Dictionary

3.1. Objects

Member Object:

Description: This object contains information such as the member's full name, email address, member id, etc. The member id serves as a primary key in the database. This LMS is for ISME College so member will include students, staff and faculty. Who will get the book issued.

Usage: This object is used to associate with book and multi-media object when items are checked out or reserved.

Item Object

Description: this object includes Item type, Item name, Isbn. Item can be a book or a CD, that would be issued to member.

Usage: Librarian will issue the item to the member, and member will return the item (book) to the librarian and he will calculate the fine if it is.

Librarian Object:

Description: This object contains information such as the user id, password, email address User id is the primary key.

Usage: This object will issue the books ,get it return and calculate the fine and search the availability of items in inventory.

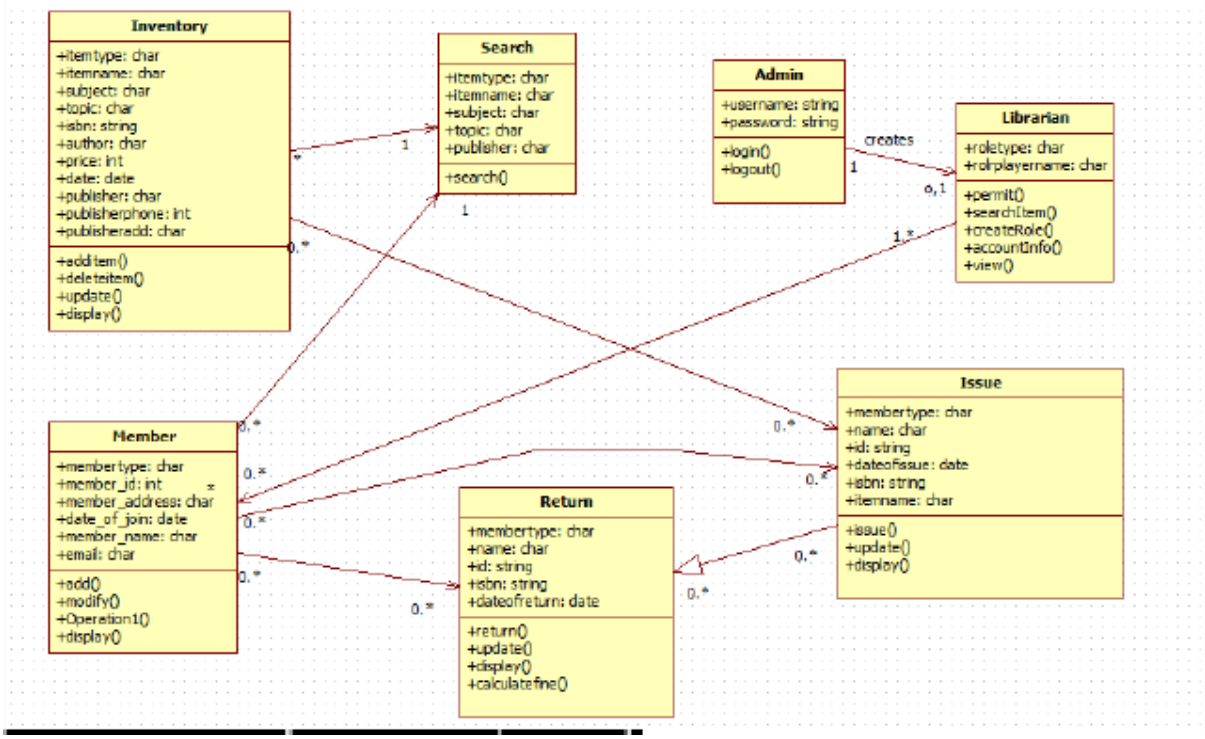
Administrator Object:

Description: This object contains information such as the administrator's full name, username, and email address and password.

Usage: An administrator will assign the roles and job for them. Put the books into the library and adds the members in the system.

3.2. Class Diagram:

In software, a class diagram in the Unified Modeling Language (UML) 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.



4.0.Architectural Design

This software system will do all the process required in library. This can be explained by different steps as below:

1. **Adding a Member**: When a member is added, during this process system will the member that it exist in database or not. If member exist then it show error message and if member does not exist then it will save the record.

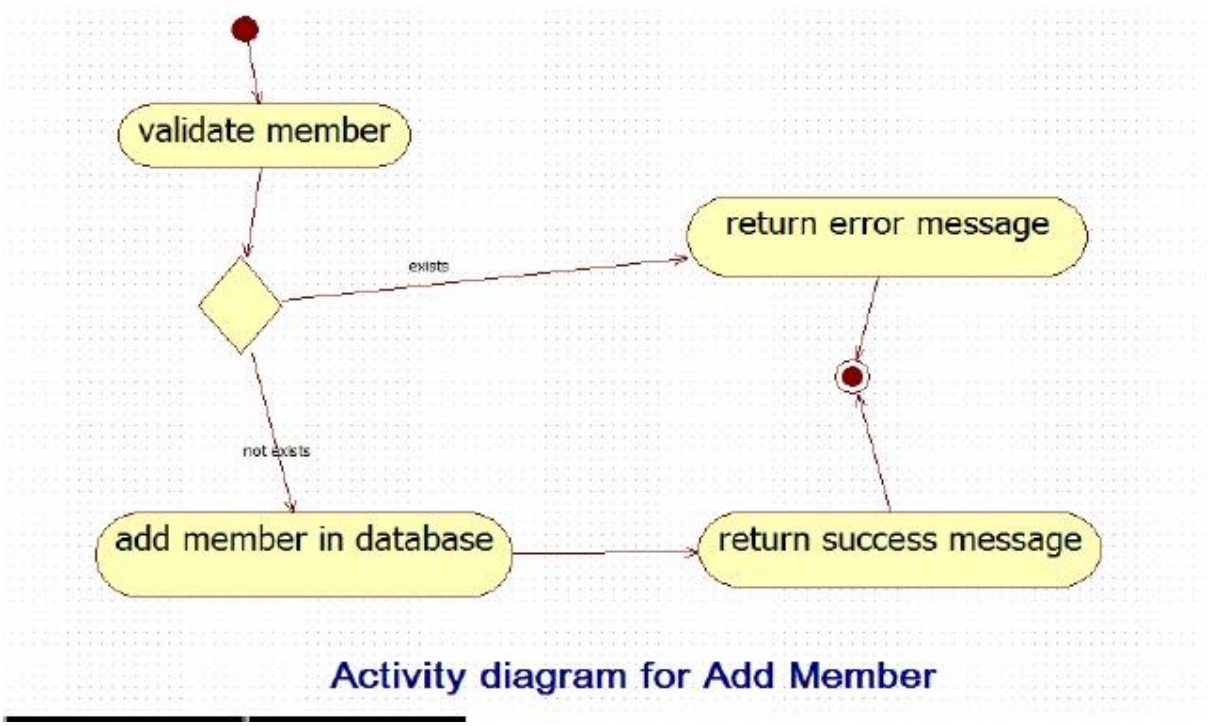
2. **Update:** When there will be some changes in member detail then update operation will occur.

3. **Adding Item:** In this case item consist of books, CDs, newspaper and journals. When a new item will arrive, detail of each item will be inserted & if same is present in database then error message will be displayed else success message will be displayed.

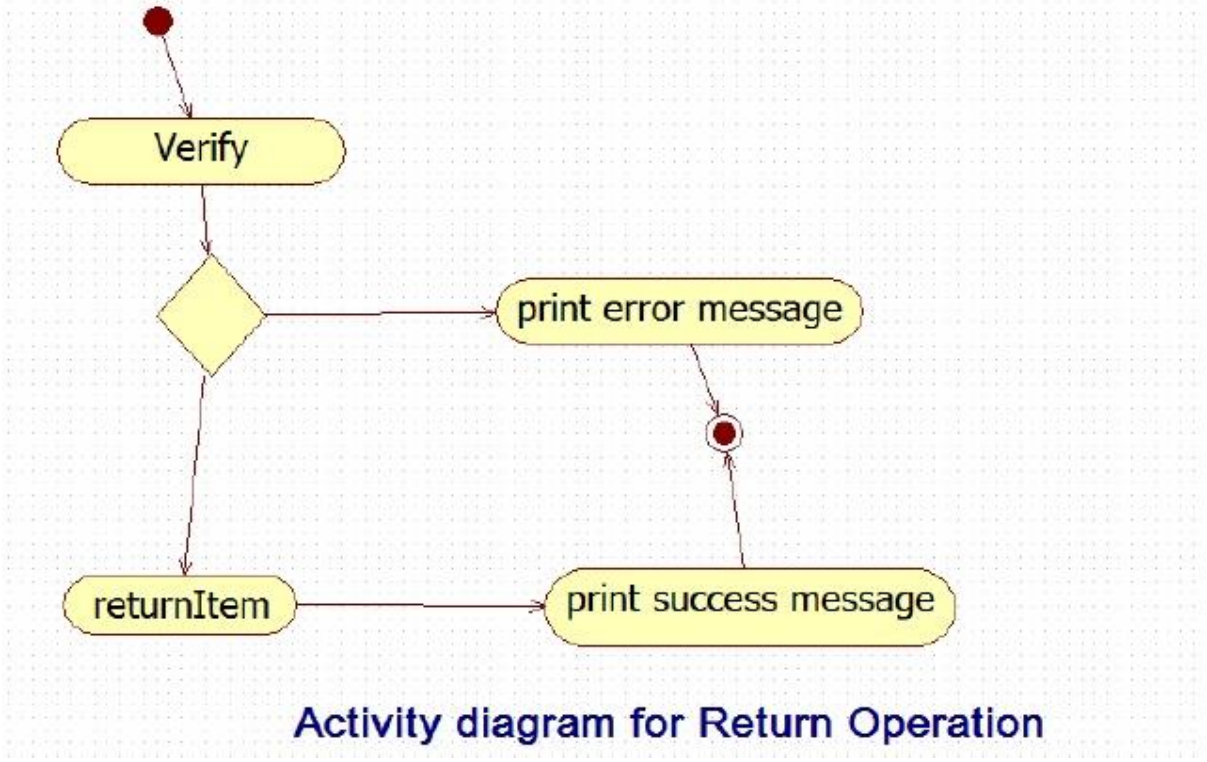
4. **Issue of Item:** During this operation there are two steps. Firstly, system will check the desired item with the database, if it exist then it will go to second step or will display error message. Secondly, system will check validity of user. If user is valid then item is issued with an issued ID.

5. **Return of Item:** When a member will return an item(Books, CDs, newspaper, journals) it will check with the system & calculate fine.

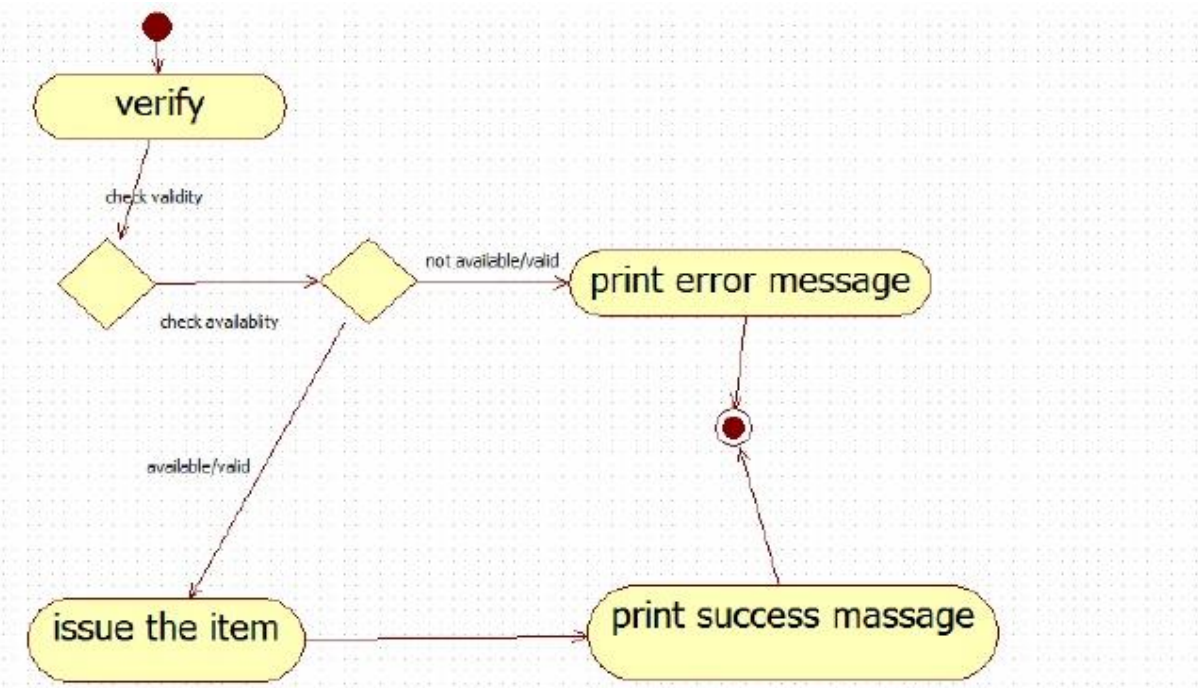
Activity Diagrams



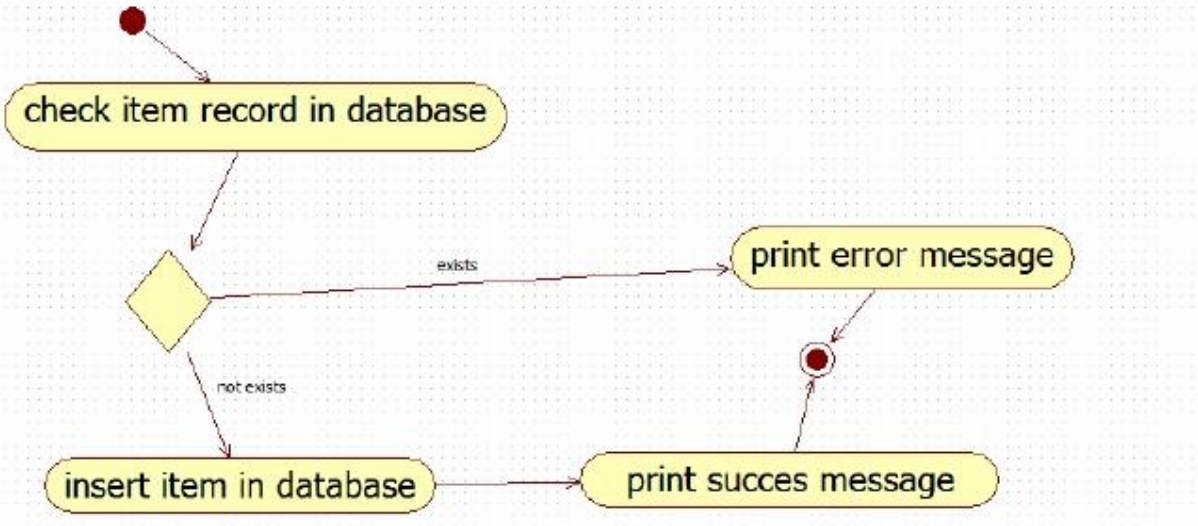
Activity diagram for Add Member



Activity diagram for Return Operation



Activity diagram for Issue Operation

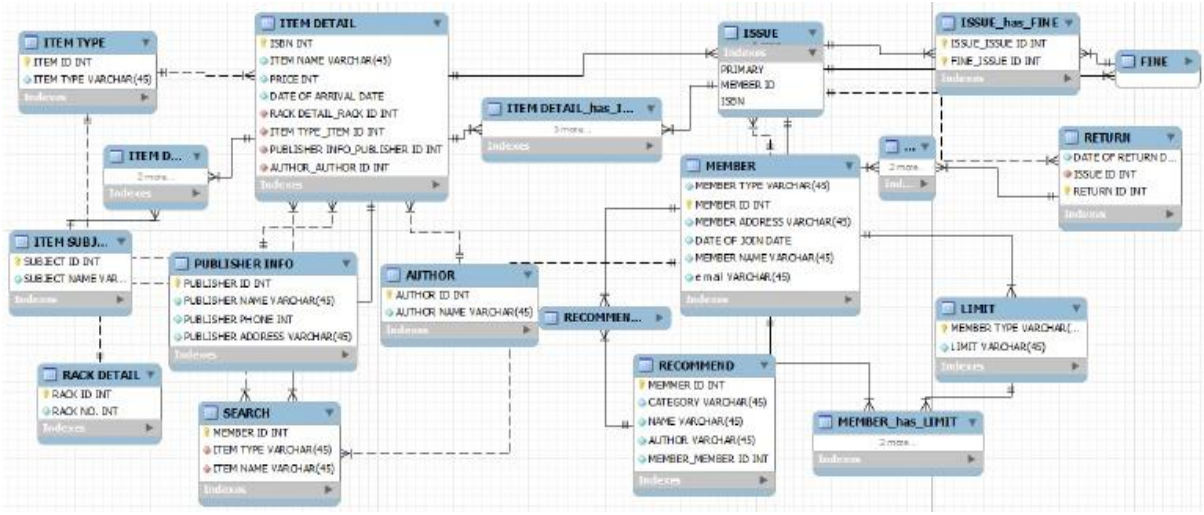


Activity diagram for Add Item

5.0.Data Design

Provide below is a summary of the various data objects that make up the online library system. Included in each table are the attributes of each object, the datatype for each attribute, the number of characters allowed for each field, the default value, and any other information that defines the fields.

ER Diagram:



Item Detail		
Field Name	**Data Type**	**Description**
ISBN	Int	Item library number
Item Name	Char(45)	Name of Item
Price	Int	Item price
Date of Arrival	Date	Date in which item arrived
Rack ID	Int	Rack ID in which items are placed
Item ID	Int	Item unique ID
Publisher ID	Int	Publisher unique ID

Item Type		
Field Name	**Data Type**	**Description**
Item Id	int	Item unique ID
Item Type	Char(45)	Category of Item

Item Subject		
Field Name	**Data Type**	**Description**
Subject ID	int	Subject unique ID
Subject Name	Char(45)	Name of Subject

Author		
Field Name	**Data Type**	**Description**
Author ID	int	Author unique ID
Author Name	Char(45)	Name of Author

Search		
Field name	**Data type**	**Description**
Item type	Char(45)	Category of item
Item name	Char(45)	Name of Item
Member id	int	Unique id for member

Recommend		
Field Name	**Data Type**	**Description**
Member ID	int	Member unique ID
Category	Char(45)	

Member		
Field Name	**Data Type**	**Description**
Member Type	Char(45)	Type of Member
Member ID	int	Member unique ID
Member Address	Char(45)	Address of Member
Date of Join	Date	Joining Date
Member Name	Char(45)	Name of Member
E-mail	Char(45)	Member e-mail address

Return		
Field Name	**Data Type**	**Description**
Date of Return	Date	Date of returinig of item
Issue ID	int	Issue Unique ID
Return ID	int	Return Unique ID

Fine		
Field Name	**Data Type**	**Description**
Amount	int	Amount collected as fine
Issue ID	int	Issue Unique ID

Publisher Info		
Field Name	**Data Type**	**Description**
Publisher ID	int	Publisher unique ID
Punblisher Name	Char(45)	Name of Publisher
Publisher Phone	int	Contact no. of Publisher
Publisher Address	Char(45)	Address of Publisher

Rack Detail

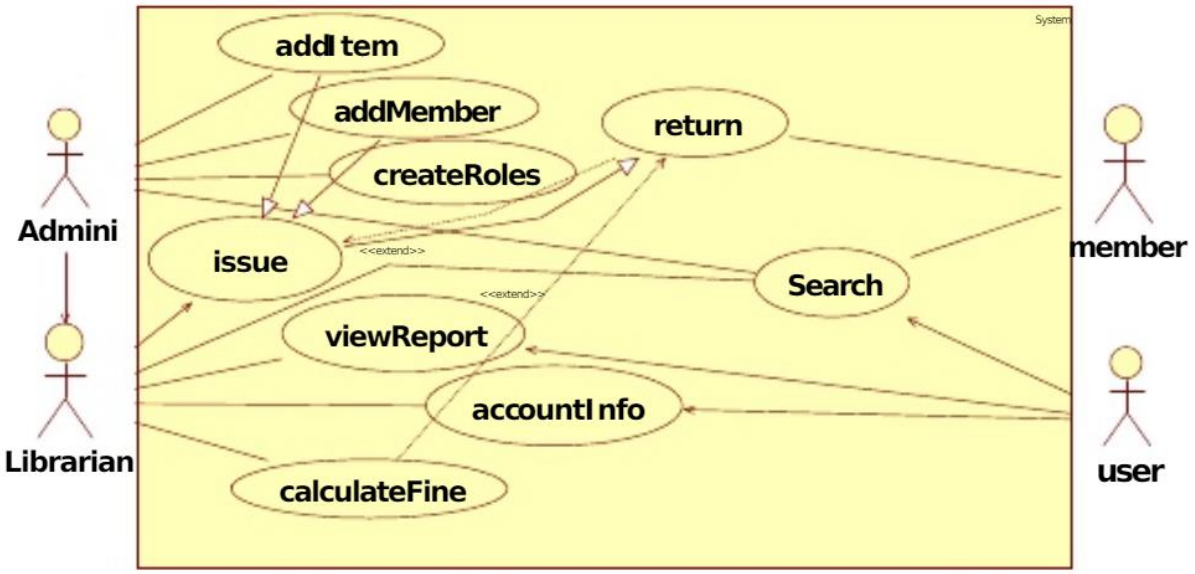
Field Name	Data Type	Description
Rack ID	int	Rack ID in which items are placed
Rack No.	int	Serial no. of rack

Limit		
Field Name	**Data Type**	**Description**
Member Type	Char(45)	Type of Member
Limit	Char(45)	Limit of item issued

Issue		
Field Name	**Data Type**	**Description**
Member ID	int	Member unique ID
Date of Issue	Date	Date of issuing item
Issue ID	int	Issue unique ID
ISBN	int	Item Library number

6.0. Use Case Realizations

A use case diagram in the UML is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals(represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Here the use case of LMS is shown; different actors and their roles are defined:



We talk about return scenario; Member returns the book to librarian, the system will validate the member and the item that is issued, get return the book calculate the fine and update the database. Return is connected to member, means member will return the item(book) to librarian. There is a relationship between issue and return, return and calculate fine. It is an extends relationship. It means member will have to return the book after issuing, but there is a possibility that he/she does not return the book. That is why there is an extend relationship. After returning the book librarian calculates the fine if there is a late submission fine will be added into the account of the member. Here return is specific use case that is generalized by issue use case. It means return is using some properties of issue operation.

Practical 8

AIM: Software Development Tools and Techniques (CASE).

CASE stands for **C**omputer **A**ided **S**oftware **E**ngineering. It means, development and maintenance of software projects with help of various automated software tools.

CASE Tools:

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.

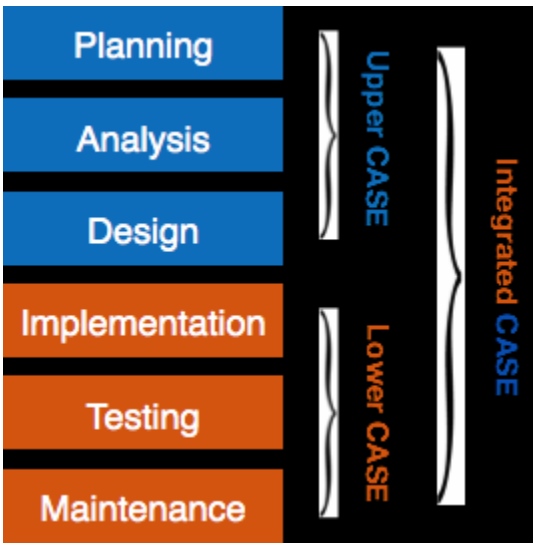
There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

Components of CASE Tools

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

Central Repository - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.



Upper Case Tools - Upper CASE tools are used in planning, analysis and design stages of SDLC.

Lower Case Tools - Lower CASE tools are used in implementation, testing and maintenance.

Integrated Case Tools - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

Case Tools Types:

Diagram tools

These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

Process Modeling Tools

Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers must strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.

Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

Design Tools

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provide detailing of each module and interconnections among modules. For example, Animated Software Design.

Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with –

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV.

Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Scope to search code in C, Eclipse.

Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspects of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, Soap Test, AppsWatch, JMeter.

Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

Practical 9

AIM: Study of Software Testing using Testing Tools.

What is testing?

Software testing is a process of executing a program or application with the intent of finding the software bugs.

It can also be stated as the process of validating and verifying that a software program or application or product:

- Meets the business and technical requirements that guided its design and development
- Works as expected
- Can be implemented with the same characteristic.

We can further break the definition of Software testing into the following parts:

1. Process: Testing is a process rather than a single activity.
2. All Life Cycle Activities: Testing is a process that's take place throughout the Software Development Life Cycle (SDLC).

The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it's referred as "verifying the test basis via the design". The test basis includes documents such as the requirements and design specifications.

3. Static Testing: It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost-effective way of testing. For example: reviewing, walkthrough, inspection, etc.
4. Dynamic Testing: In dynamic testing the software code is executed to demonstrate the result of running tests. It's done during validation process. For example: unit testing, integration testing, system testing, etc.
5. Planning: We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.
6. Preparation: We need to choose what testing we will do, by selecting test conditions and designing test cases
7. Evaluation: During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.
8. Software products and related work products: Along with the testing of code the testing of requirement and design specifications and the related documents like operation, user and training material is equally important.

Difference between black box and white box testing

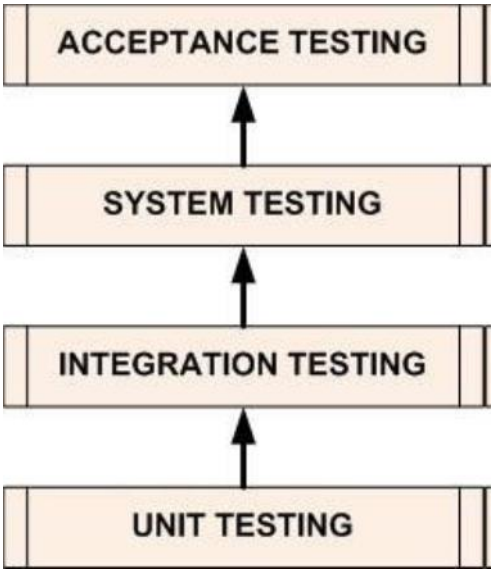
Criteria	Black Box Testing	White Box Testing
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
Levels Applicable To	Mainly applicable to higher levels of testing: Acceptance System Testing	Mainly applicable to lower levels of testing: Unit Integration Testing
Responsibility	Generally, independent Software Testers	Generally, Software Developers
Programming Knowledge	Not Required	Required
Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design

Software Testing levels

Testing levels are basically to identify missing areas and prevent overlap and repetition between the development life cycle phases. In software development lifecycle models, there are defined phases like requirement gathering and analysis, design, coding or implementation, testing and deployment. Each phase goes through the testing. Hence there are various levels of testing.

There are four levels of software testing:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing



Level	Summary
Unit Testing	A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed
Integration Testing	A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Below are few types of integration testing: Big bang integration testing Top down Bottom up Functional incremental
System Testing	A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements

Performance testing:

Performance testing is the general name for tests that check how the system behaves and performs. Performance testing examines responsiveness, stability, scalability, reliability, speed and resource usage of your software and infrastructure. Different types of performance tests provide you with different data, as we will further detail.

Before performance testing, it's important to determine your system's business goals, so you can tell if your system behaves satisfactorily or not according to your customers' needs.

After running performance tests, you can analyze different KPIs, such as the number of virtual users, hits per second, errors per second, response time, latency and bytes per second (throughput), as well as the correlations between them. Through the reports you can identify bottlenecks, bugs and errors, an decide what needs to be done.

When should you use Performance Testing?

When you want to check your website performance and app performance, as well as servers, databases, networks, etc. If you work with the waterfall methodology, then at least each time you release a version. If you're shifting left and going agile, you should test continuously.

Load Testing?

Load testing is testing that checks how systems function under a heavy number of concurrent virtual users performing transactions over a certain period. Or in other words, how systems handle heavy load volumes. There are a few types of open-source load testing tools, with JMeter being the most popular one.

When should you use Load Testing?

When you want to determine how many users your system can handle. You can determine different user scenarios that let you focus on different parts of your system, like the checkout webpage on your website or app for web load testing. You can also determine how the load behaves, like the geo-location users come from or how the load builds and sustains in the system. Basically, load testing is something you should do all the time, to ensure your system is always on point. That's why it should be integrated into your Continuous Integration cycles, with tools like Jenkins and Taurus.

Stress Testing?

Stress testing is testing that checks the upper limits of your system by testing it under extreme loads. The testing examines how the system behaves under intense loads, and how it recovers when going back to normal usage, i.e. are the KPIs like throughput and response time the same as before? In addition to load testing KPIs, stress testing also examines memory leaks, slowness, security issues and data corruption.

Stress testing can be conducted through load testing tools, by defining a test case with a very high number of concurrent virtual users. If your stress test includes a sudden ramp-up in the number of virtual users, it is called a Spike Test. If you stress test for a long period of time to check the system's sustainability over time with a slow ramp-up, it's called a Soak Test.

When Should You Use Stress Testing?

Website stress tests and app stress tests are important before major events, like Black Friday, ticket selling for a popular concert with high demand or the elections. But we recommend you stress test occasionally, so you know your system's endurance capabilities. This ensures you're always prepared for unexpected traffic spikes and gives you more time and resources to fix your bottlenecks.

WinRunner & load runner tools

Win Runner is the most used Automated Software Testing Tool. Main Features of Win Runner are

Developed by Mercury Interactive

Functionality testing tool

Supports C/s and web technologies such as (VB, VC++, D2K, Java, HTML, Power Builder, Delphe, Cibell (ERP))

To Support .net, xml, SAP, Peoplesoft, Oracle applications, Multimedia we can use QTP.

Winrunner run on Windows only.

Xrunner run only UNIX and Linux.

Tool developed in C on VC++ environment.

To automate our manual test win runner used TSL (Test Script language like c)

The main Testing Process in Win Runner is

1. Learning: Recognition of objects and windows in our application by WinRunner is called learning. Winrunner 7.0 follows Auto learning.
2. Recording: Winrunner records over manual business operation in TSL
3. Edit Script: Depends on corresponding manual test, test engineer inserts check points in to that record script.
4. Run Script: During test script execution, WinRunner compare tester given expected values and application actual values and returns results.
5. Analyze Results: Tester analyzes the tool given results to concentrate on defect tracking if required.

HP LoadRunner software allows you to prevent application performance problems by detecting bottlenecks before a new system or upgrade is deployed. The testing solution LoadRunner enables you to test rich Internet applications, Web 2.0 technologies, ERP and CRM applications, and legacy applications. It gives you a picture of end-to-end system performance before going live so that you can verify that new or upgraded applications meet performance requirements. And it reduces hardware and software costs by accurately predicting application scalability and capacity.

How it works?

Load Runner is divided up into 3 smaller applications:

The Virtual User Generator allows to determine what actions you would like the Vusers, or virtual users, to perform under stress within the application. You create scripts that generate a series of actions, such as logging on, navigating through the application, and exiting the program.

The Controller takes the scripts that you have made and runs them through a schedule that you set up. Tell the Controller how many Vusers to activate, when to activate them, and how to group the Vusers and keep track of them.

The Results and Analysis program gives you all the results of the load test in various forms. It allows to see summaries of data, as well as the details of the load test for pinpointing problems or bottlenecks.

LoadRunner can emulate hundreds or thousands of concurrent users to put the application through the rigors of real-life user loads, while collecting information from key infrastructure components (Web servers, database servers etc.). The results can then be analyzed in detail, to explore the reasons for behavior.

The LoadRunner Testing Process

- Planning the Test
- Creating the Vuser scripts
- Creating the Scenario
- Running the Scenario
- Analyzing Test Results

Verification and validation in software engineering

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle. In simple terms, software verification is: "Assuming we should build X, does our software achieve its goals without any bugs or gaps?" On the other hand, software validation is: "Was X what we should have built? Does X meet the high-level requirements?"

Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements. This is done through dynamic testing and other forms of review.

Verification and validation are not the same thing, although they are often confused. Boehm [1] succinctly expressed the difference between

- Validation: Are we building the right product?
- Verification: Are we building the product right According to the Capability Maturity Model (CMMI-SW v1.1),
- Software Verification: The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. [IEEE-STD-610]

- Software Validation: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. [IEEE-STD-610]

In other words, software verification is ensuring that the product has been built according to the requirements and design specifications, while software validation ensures that the product meets the user's needs, and that the specifications were correct in the first place. Software verification ensures that "you built it right". Software validation ensures that "you built the right thing". Software validation confirms that the product, as provided, will fulfill its intended use.

Advantages or benefits of using testing tools

There are many benefits that can be gained by using tools to support testing. They are:

Reduction of repetitive work: Repetitive work is very boring if it is done manually. People tend to make mistakes when doing the same task over and over. Examples of this type of repetitive work include running regression tests, entering the same test data again and again (can be done by a test execution tool), checking against coding standards (which can be done by a static analysis tool) or creating a specific test database (which can be done by a test data preparation tool).

Greater consistency and repeatability: People have tendency to do the same task in a slightly different way even when they think they are repeating something exactly. A tool will exactly reproduce what it did before, so each time it is run the result is consistent.

Objective assessment: If a person calculates a value from the software or incident reports, by mistake they may omit something, or their own one-sided preconceived judgments or convictions may lead them to interpret that data incorrectly. Using a tool means that subjective preconceived notion is removed, and the assessment is more repeatable and consistently calculated. Examples include assessing the cyclomatic complexity or nesting levels of a component (which can be done by a static analysis tool), coverage (coverage measurement tool), system behavior (monitoring tools) and incident statistics (test management tool).

Ease of access to information about tests or testing: Information presented visually is much easier for the human mind to understand and interpret. For example, a chart or graph is a better way to show information than a long list of numbers – therefore charts and graphs in spreadsheets are so useful. Special purpose tools give these features directly for the information they process. Examples include statistics and graphs about test progress (test execution or test management tool), incident rates (incident management or test management tool) and performance (performance testing tool).

Disadvantages or risks of using testing tools

Although there are many benefits that can be achieved by using tools to support testing activities, but there are also many risks that are associated with it when tool support for testing is introduced and used.

Risks include:

Unrealistic expectations from the tool: Unrealistic expectations may be one of the greatest risks to success with tools. The tools are just software and we all know that there are many problems associated with any kind of software. It is very important to have clear and realistic objectives for what the tool can do.

People often make mistakes by underestimating the time, cost and effort for the initial introduction of a tool: Introducing something new into an organization is hardly straightforward. Once you purchase a tool, you want to have several people being able to use the tool in a way that will be beneficial. There will be some technical issues to overcome, but there will also be resistance from other people – both need to be handled in such a way that the tool will be of great success.

People frequently miscalculate the time and effort needed to achieve significant and continuing benefits from the tool: Mostly in the initial phase when the tool is new to the people, they miscalculate the time and effort needed to achieve significant and continuing benefits from the tool. Just think back to the last time you tried something new for the very first time (learning to drive, riding a bike, skiing). Your first attempts were unlikely to be very good but with more experience and practice you became much better. Using a testing tool for the first time will not be your best use of the tool either. It takes time to develop ways of using the tool in order to achieve what is expected.

Mostly people underestimate the effort required to maintain the test assets generated by the tool: Generally, people underestimate the effort required to maintain the test assets generated by the tool. Because of the insufficient planning for maintenance of the assets that the tool produces there are chances that the tool might end up as ‘shelf-ware’, along with the previously listed risks.

People depend on the tool a lot (over-reliance on the tool): Since there are many benefits that can be gained by using tools to support testing like reduction of repetitive work, greater consistency and repeatability, etc. people started to depend on the tool a lot. But the tools are just a software they can do only what they have been designed to do (at least a good quality tool can), but they cannot do everything. A tool can help, but it cannot replace the intelligence needed to know how best to use it, and how to evaluate current and future uses of the tool. For example, a test execution tool does not replace the need for good test design and should not be used for every test – some tests are still better executed manually. A test that takes a very long time to automate and will not be run very often is better done manually.

PRACTICAL 10

AIM: Study of Software Quality Assurance (SQA) Standards.

What is Quality assurance?

Software quality assurance (SQA) is a process that ensures that developed software meets and complies with defined or standardized quality specifications. SQA is an ongoing process within the software development life cycle (SDLC) that routinely checks the developed software to ensure it meets desired quality measures.

SQA helps ensure the development of high-quality software. SQA practices are implemented in most types of software development, regardless of the underlying software development model being used. In a broader sense, SQA incorporates and implements software testing methodologies to test software. Rather than checking for quality after completion, SQA processes tests for quality in each phase of development until the software is complete.

What is software reliability?

Software Reliability is the probability of failure-free software operation for a specified period in a specified environment. Software Reliability is also an important factor affecting system reliability. Software Reliability is not a function of time - although researchers have come up with models relating the two. The modeling technique for Software Reliability is reaching its prosperity, but before using the technique, we must carefully select the appropriate model that can best suit our case.

Software Reliability is an important to attribute of software quality, together with functionality, usability, performance, serviceability, capability, install ability, maintainability, and documentation. Software Reliability is hard to achieve, because the complexity of software tends to be high.

S/w versus H/w reliability

HARDWARE RELIABILITY	SOFTWARE RELIABILITY
The hardware reliability is based on the measure of its own failures, depending on its environment.	For software, there is no random failures, the software does not fail. Bugs come from coding error.
To have a high hardware reliability, we need to increase robustness or to adapt its environment (air cooling...)	To have a high software reliability we need to have a hard process and tests (depending on the need)

Failures versus faults

Fault	Failure
Fault is a software defect that causes a failure.	The inability of a software to perform its required functions within specified performance requirements.

A manifestation of an error in software. A fault, if encountered may cause a failure.	Deviation of the software from its expected delivery or service.
An error becomes a fault when it is included in any of the developed software products.	Failures occur when a software system does not behave as desired, which reveals a fault in the software.
A fault is the cause of an error. A software fault lies in software, a hardware fault lies in hardware An incorrect step, process, or data definition in a computer program.	A failure occurs when the observed behavior differs from the expected one. The inability of a system to perform its required functions within specified performance requirements.

What is Six-Sigma?

Six Sigma is a set of techniques and tools for process improvement. It seeks to improve the quality of the output of a process by identifying and removing the causes of defects and minimizing variability in manufacturing and business processes.

It uses a set of quality management methods, mainly empirical, statistical methods, and creates a special infrastructure of people within the organization who are experts in these methods.

Six sigma levels

"Six Sigma" management has several levels of certification, they are: Champion, Yellow/Green Belt, Brown/Black Belt, and Master Black Belt. Each level of certification is described below.

A Six Sigma Champion is the most basic form of Six Sigma certification. A Champion understands the theory of Six Sigma management but does not yet have the quantitative skills to function as an active Six Sigma project team member.

A Six Sigma Green Belt is an individual who works on projects part-time (25%), either as a team member for complex projects, or as a project leader for simpler projects. Green belts are the "work horses" of Six Sigma projects.

A Six Sigma Yellow Belt is an individual who has passed the Green Belt certification examination but has not yet completed a Six Sigma project.

A Six Sigma Black Belt is a full-time change agent and improvement leader who may not be an expert in the process under study. A black belt is a full-time quality professional who is mentored by a master black belt, but may report to a manager, for his or her tour of duty as a black belt.

A Six Sigma Brown Belt is a Six Sigma Green Belt who has passed the Black Belt certification examination but has not yet completed their second Six Sigma project.

A Six Sigma Master Black Belt takes on a leadership role as keeper of the Six Sigma process, advisor to executives or business unit managers, and leverages, his/her skills with projects that are led by black belts and green belts. Frequently, master black belts report directly to senior executives or business unit managers.

Strength & Weakness of Six Sigma Strengths:

One of the biggest advantages of using Six Sigma lies in the methodology statements which assert that “no project shall be approved if a bottom-line impact has not been clearly defined”. With goals being unmistakably defined, there is less vagueness to deal with and decisions are implemented that are derived from statistical data and research, not haphazard assumptions.

Other benefits include:

- Emphasis on achieving attainable goals
- Implementing projects that will produce results
- Effective use of scientific techniques and precise tools
- Infuses upper management with passion and dedication
- Integrated concepts benefiting employees and customers
- Using information that has real world meaning

Weakness:

While Six Sigma is rapidly spreading throughout a variety of industries and organizations, some limitations can be said to exist within its procedures and measurements. Projects which are directed are selected by organizations subjectively rather than objectively, which means that goals may be mistakenly thought of as attainable and favorable when in fact they may eventually be a waste of resources and time.

Also, researchers investigating the trend have noticed that some individuals calling themselves “experts” in Six Sigma methodology do not comprehend the techniques and complex tools necessary to effectively implement the quality control process in an organization. Thus, these companies hiring “experts” are being treated to a substandard version of the principles which will do nothing to help their company and only lend a warped perspective of what it is supposed to do.

What is CMM?

The Capability Maturity Model (CMM) is a development model created after study of data collected from organizations that contracted with the U.S. Department of Defense, who funded the research. The term "maturity" relates to the degree of formality and optimization of processes, from *ad hoc*

practices, to formally defined steps, to managed result metrics, to active optimization of the processes.

The model's aim is to improve existing software development processes, but it can also be applied to other processes.

Structure

The model involves five aspects:

Maturity Levels: a 5-level process maturity continuum - where the uppermost (5th) level is a notional ideal state where processes would be systematically managed by a combination of process optimization and continuous process improvement.

Key Process Areas: A Key Process Area identifies a cluster of related activities that, when performed together, achieve a set of goals considered important.

Goals: the goals of a key process area summarize the states that must exist for that key process area to have been implemented in an effective and lasting way. The goals signify the scope, boundaries, and intent of each key process area.

Common Features: Common features include practices that implement and institutionalize a key process area. There are five types of common features: commitment to perform, ability to perform, activities performed, measurement and analysis, and verifying implementation.

Key Practices: The key practices describe the elements of infrastructure and practice that contribute most effectively to the implementation and institutionalization of the area.

CMM Levels

There are five levels defined along the continuum of the model and, according to the SEI:

1. Initial (chaotic, ad hoc, individual heroics) - the starting point for use of a new or undocumented repeat process.

2. Repeatable - the process is at least documented sufficiently such that repeating the same steps may be attempted.

3. Defined - the process is defined / confirmed as a standard business process.

4. Managed - the process is quantitatively managed in accordance with agreed-upon metrics.

5. Optimizing - process management includes deliberate process optimization/improvement.

Strength & weakness of CMM

The CMM model's application in software development has sometimes been problematic. Applying multiple models that are not integrated within and across an organization could be costly in training, appraisals, and improvement activities.

The Capability Maturity Model (CMM) project was formed to sort out the problem of using multiple models for software development processes, thus the CMMI model has superseded the CMM model, though the CMM model continues to be a general theoretical process capability model used in the public domain.

The model was originally intended to evaluate the ability of government contractors to perform a software project. It has been used for and may be suited to that purpose, but critic pointed out that

process maturity according to the CMM was not necessarily mandatory for successful software development.

Similarities & differences between CMM & Six sigma Similarities:

CMM and Six Sigma should not be in competition. Simultaneous implementation of

these concepts in an organization produces a synergy that helps in the successful accomplishment of company goals in a faster, better, and cheaper way.

A comparison of CMMI vs Six Sigma shows that while both CMMI and Six Sigma are improvement-oriented initiatives with many overlaps, there exists fundamental differences between these two project management concepts.

Six Sigma advocates two methodologies; one for projects aimed at improving existing business programs and another aimed at creating new product or processes. I will describe the latter methodology here:

1) Define design goals that are consistent with customer demands and the enterprise strategy.

2) Measure and identify characteristics that are critical to quality

3) Analyze to develop and design alternatives, create a high-level design and evaluate design capability to select the best design.

4) Design details optimize the design, and plan for design verification. This phase may require simulations.

5) Verify the design, set up pilot runs, implement the production process and hand it over to the process owners.

There are five levels of an organization's maturity as defined by CMMI from highly reactive organizations being the least mature to organizations focusing on process improvement being the most mature. In addition, there are 16 process areas in CMMI that can be tailored to meet the needs of an organization. Eight of these areas are:

1) Requirements Management

2) Project Monitoring and Control

3) Project Planning Project

4) Configuration Management

5) Measurement and Analysis Support

6) Process and Product Quality Assurance

7) Organizational Process Definition Process

Differences:

Six Sigma: Is an overall enterprise improvement methodology that uses data to monitor, control, and improve operational performance by eliminating and preventing ‘defects' in products and

associated processes. Six Sigma emphasizes producing better, faster, and lower cost product and services than the competition and stresses breakthrough improvement, for improved bottom line results.

CMM: Capability Maturity Model is unpredictable and often chaotic because the s/w process is constantly changed as the work progresses. The process is, essentially, ad hoc and generally Undisciplined, making the organization an unstable environment for developing s/w.

What is ISO 9000?

The **ISO 9000** family of quality management systems standards is designed to help organizations ensure that they meet the needs of customers and other stakeholders while meeting statutory and regulatory requirements related to a product or program. ISO 9000 deals with the fundamentals of quality management systems, including the seven quality management principles upon which the family of standards is based.

Objectives of ISO 9000

The Objectives of ISO 9000 are as below:

Gives businesses with useful, globally recognized models for operating a quality management system.

Achieve, maintain and aim to regularly enhance product quality (the standards define “product” as the output of any process. Therefore, this word will also apply to “services,” whether internal or external to the business).

Primary objective of getting these standards is to boost the goodwill of organization. Customer can compare the quality of two companies, one is with ISO standard and other is without ISO standard. Goodwill could be in form of rise in sale or more promotion of product of company.

To create a compliance standard which is followed 24 hours-a-day, 7 days-a week, 52 weeks-a-year.

Offer confidence to internal management as well as other workers that requirements for quality are being fulfilled and maintained, and that quality improvement is taking place.

Benefits of ISO 9000

The benefits associated with ISO 9000 certification are numerous, as both business analysts and business owners will attest. These benefits, which can impact nearly all corners of a company, range from increased stature to bottom-line operational savings. They include:

Increased marketability—Nearly all observers agree that ISO 9000 registration provides businesses with markedly heightened credibility with current and prospective clients alike. Basically, it proves that the company provides quality to its customers, which is no small advantage whether the company is negotiating with a long-time customer or endeavoring to pry a potentially lucrative customer away from a competitor. This benefit manifests itself not only in increased customer retention, but also in increased customer acquisition and heightened ability to enter into new

markets; indeed, ISO 9000 registration has been cited as being of value for small and mid-sized businesses hoping to establish a presence in international markets.

Improved internal communication—The ISO 9000 certification process's emphasis on self-analysis and operations management issues encourages various internal areas or departments of companies to interact with one another in hopes of gaining a more complete understanding of the needs and desires of their internal customers.

Improved customer service—The process of securing ISO 9000 registration often serves to refocus company priorities on pleasing their customers in all respects, including customer service areas. It also helps heighten awareness of quality issues among employees.

Reduction of product-liability risks—Many business experts contend that companies that achieve ISO 9000 certification are less likely to be hit with product liability lawsuits, etc., because of the quality of their processes.

Attractiveness to investors—Business consultants and small business owners alike agree that ISO-9000 certification can be a potent tool in securing funding from venture capital firms.