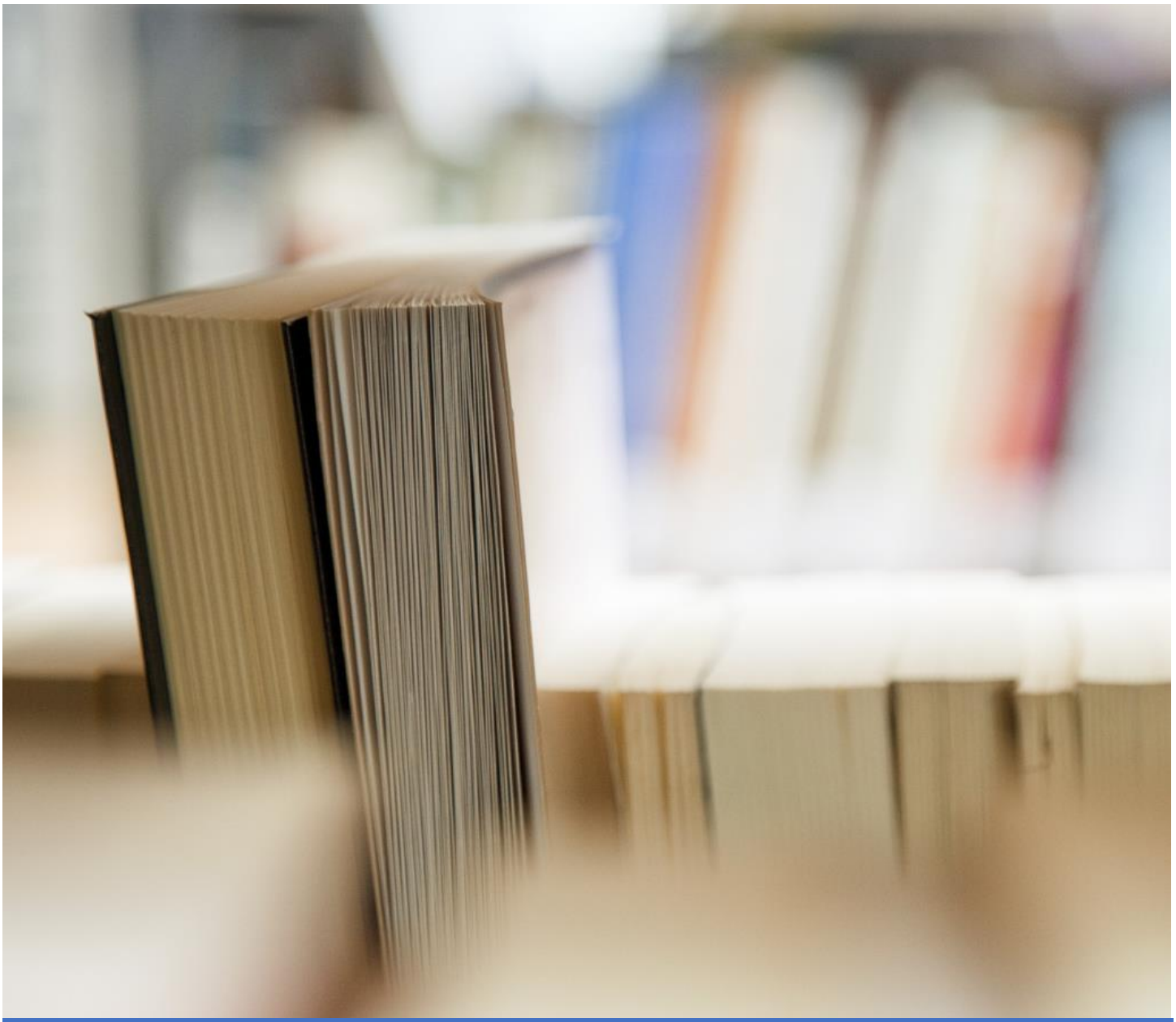


Library Management System

Software Design and Architecture

Akash Chandra Debnath
(Roll-37)

[02/03/2022]

Lab

CONTENTS


Introduction

1. INTRODUCTION

Library Management System consists of list of records about the management of the details of the students and the issues going on and also about some books and all. This is a web-based application. The project has two modules namely- User (Student), Librarian (Admin). According to the Modules the Distributor and Sub Distributors can manage and do their activities in easy manner. As the modern organizations are automated and computers are working as per the instructions, it becomes essential for the co-ordination of human beings, commodity and computers in a modern organization. The main purpose of this project is to maintain easy circulation system using computers and to provide different reports. The Library System is a package to be used by Libraries to improve the efficiency of Librarians, Library employees and users. The system provides books catalogue and information to members and helps them decide on the books to borrow from the library. The Librarian can keep the books catalogue updated all the time so that the members get the updated information all the time.

Online Library Management System (OLMS) is basically updating the manual library system into an internet-based application so that the users can know the details of their accounts, availability of books and remaining time for borrowing.

1.1 OVERVIEW

Library is place where all kind of books are available. Intranet Library Management system is a web based application. This system contains list of all the books and can be accessed by remote users concurrently from anywhere in the campus. But for that users must be registered user. This system is three tier architecture.

Admin sends requests, on receiving the request the server processes it and extracts the data from database and sends the result back to the users. This system provides separate interface and login for librarian and students. Librarian can modify database.

Users can search for books and renewal books online. They can recommend for new books by just sending messages to the librarian from anywhere in the university. They can view the issue and return dates of any book and due they have to pay. This system generates reports that can be used in analyzing the library performance. Thus the management can take appropriate steps to improve the facilities.


Analysis

2.1 SYSTEM ANALYSIS

A. Existing System

Various problems of physical system are described below: -

- ✓ If one is not very careful then there is a possibility of issuing more than one book to a user.
- ✓ There is a possibility of issuing a book to a user, whose membership is not there.
- ✓ When a user requests for a book, one has to physically check for the presence of a book in the library
- ✓ Answering management query is a time consuming process.
- ✓ Daily keeping a manual record of changes taking place in the library such as book being issued, book being returned etc. can become cumbersome if the Library size is bigger.

B. Proposed System

The ONLINE LIBRARY MANAGEMENT SYSTEM is a software application which avoids more manual hours in taking the book, that need to spend in record keeping and generating reports. Maintaining of user details is complex in manual system in terms of agreements, royalty and activities. This all have to be maintained in ledgers or books. Co-coordinators need to verify each record for small information also.

- Easy search of book in the online library.
- Avoid the manual work.
- User need not go to the library for Issue any kind of book, he can renewal the book online.

C. Objective of the System

The goal of the system is to bring down the work load with the increased efficiency and to speed up the activities. With this it is very easy to process course fee that is collected time to time from students who are registered and studying at franchisees.

2.2 SYSTEM SPECIFICATIONS

Hardware Requirements: -

- **Hard disk :** 1TB
- **RAM :** 4GB
- **Processor :** intel(R) core-i5-CPU

Software Requirements: -

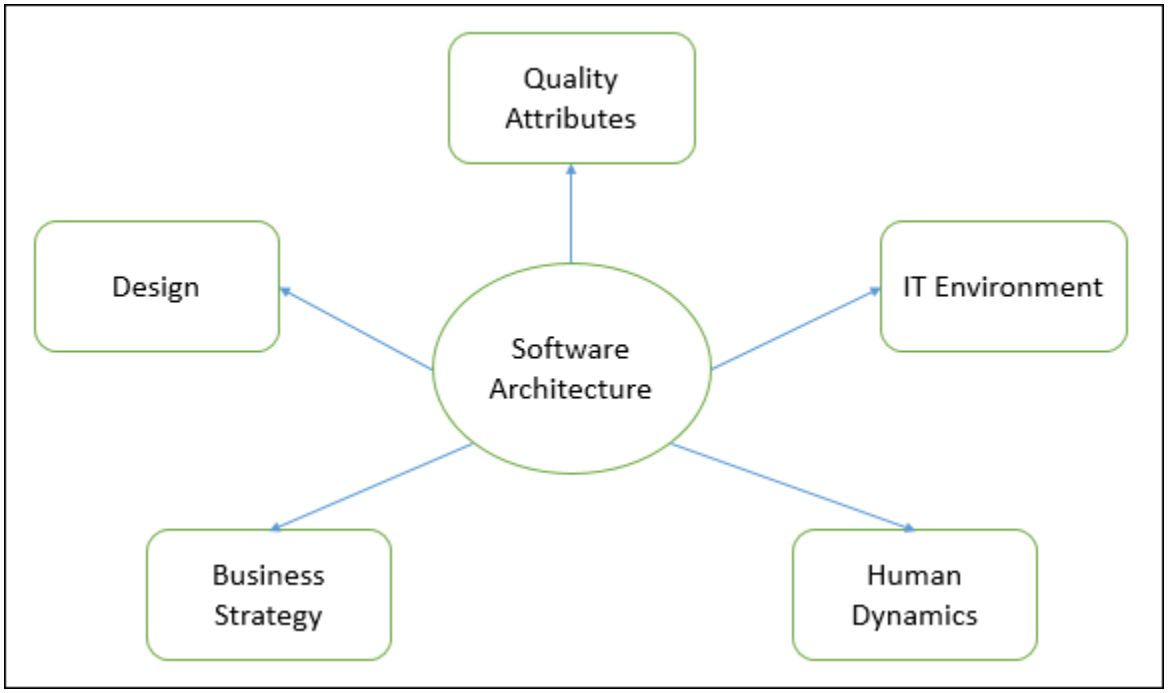
- **Operating System :** Windows 95/98/XP with MS-office
- **Programming language :** Sublime Text
- **Web-Technology :** PHP (raw)
- **Web Server :** Xampp
- **Database :** MySQL


Architecture

3.1 INTRODUCTION

"Software architecture refers to the fundamental structures of a software system and the discipline of creating such structures and systems"

The structure comprises software elements, relations among them, and properties of both elements and relations. Software architecture is about making fundamental structural choices that are costly to change once implemented. Software architecture choices include specific structural options from possibilities in the design of the software.



Architecture serves as a blueprint for a system. It provides an abstraction to manage the system complexity and establish a communication and coordination mechanism among components.

- It defines a structured solution to meet all the technical and operational requirements, while optimizing the common quality attributes like performance and security.

- Further, it involves a set of significant decisions about the organization related to software development and each of these decisions can have a considerable impact on quality, maintainability, performance, and the overall success of the final product.

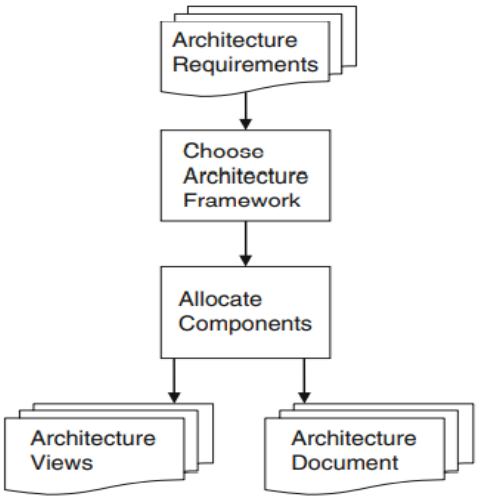


Fig. 7.3 Inputs and outputs of architecture design

3.2 GOALS OF ARCHITECTURE

The primary goal of the architecture is to identify requirements that affect the structure of the application. A well-laid architecture reduces the business risks associated with building a technical solution and builds a bridge between business and technical requirements. Some of the other goals are as follows −

- Expose the structure of the system, but hide its implementation details.
- Realize all the use-cases and scenarios.
- Try to address the requirements of various stakeholders.
- Handle both functional and quality requirements.
- Reduce the goal of ownership and improve the organization's market position.
- Improve quality and functionality offered by the system.
- Improve external confidence in either the organization or system.

So as architectural theory, we firstly fixed the structure of OLMS and flow of execution through the components. Then applied quality attributes to the system functionality.

3.3 LIMITATIONS

- Lack of tools and standardized ways to represent architecture.
- Lack of analysis methods to predict whether architecture will result in an implementation that meets the requirements.
- Lack of awareness of the importance of architectural design to software development.
- Lack of understanding of the role of software architect and poor communication among stakeholders.
- Lack of understanding of the design process, design experience and evaluation of design.

3.4 ROLE OF SOFTWARE ARCHITECT

A Software Architect provides a solution that the technical team can create and design for the entire application. So architect need to be –

✓ Design expertise
✓ Domain expertise
✓ Technology expertise

3.5 QUALITY ATTRIBUTES

Quality attribute is property of a work product by which its quality will be judged by stakeholders. It is a measure of excellence or the state of being free from deficiencies or defects. Quality attribute requirements are part of an application's nonfunctional requirements, which capture the many facets of how the functional requirements of an application are achieved.

Category	*Quality Attribute*	*Description*	*OLMS*
Design Qualities	Conceptual Integrity	Defines the consistency and co-herence of the overall design. This includes the way components or modules are designed.	Design of profile, book-list, approve - request, manage student, register module etc.
	Maintainability	Ability of the system to undergo changes with a degree of ease.	Keep the flow of the system and each module activity.

	Reusability	Defines the capability for components and subsystems to be suitable for use in other applications.	The modules of OLMS will be decoupled for re-usability.
Run-time Qualities	Interoperability	Ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties.	Third party modules or system should com-municate and exchange information easily with-out any complexity.
	Manageability	Defines how easy it is for system administrators to manage the application.	It should not be a com-plex system that libra-rians or students need more technical know-ledge.
	Reliability	Ability of a system to remain operational over time.	Librarians or students operation should not terminate inconsistently.
	Scalability	Ability of a system to either handle the load increase without impacting the performance of the system or the ability to be readily enlarged.	System should manage many request simul-taneously without in-crease waiting time.
	Security	Capability of a system to prevent malicious or accidental actions outside of the designed usages.	SQLi, XSS, database hack or any other attack issue have to consider.
	Performance	Indication of the responsiveness of a system to execute any action within a given time interval.	Load time or request / approval should be smooth.
	Availability	Defines the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period.	The system should be available 24 hours to authenticated users and admin.
System Qualities	Supportability	Ability of the system to provide information helpful for identifying and resolving issues when it fails to work correctly.	System should informed exact reason for any inconsistent situation.
	Testability	Measure of how easy it is to create test criteria for the system and its components.	Each components need to be testable for gets a bug free system.
User Qualities	Usability	Defines how well the application	Students and librarians

		meets the requirements of the user and consumer by being intuitive.	should be happy and get relatable actions.
Architecture Quality	Correctness	Accountability for satisfying all the requirements of the system.	System should correctly render.
Non-runtime Quality	Portability	Ability of the system to run under different computing environment.	OLMS should be run on desktop and also in mobile devices
	Integrality	Ability to make separately developed components of the system work correctly together.	Modules/components should not conflict each other.
	Modifiability	Ease with which each software system can accommodate changes to its software.	System will be modifi-able for future extension and new features.
Business quality attributes	Cost and schedule	Cost of the system with respect to time to market, expected project lifetime & utilization of legacy.	Cost will depends on modules and complexity of these.
	Marketability	Use of system with respect to market competition.	The system may publish for business demand.

3.6 DESIGN TRADE-OFF

Quality attributes are not orthogonal. They interact in subtle ways, meaning a design that satisfies one quality attribute requirement may have a detrimental effect on another. For example, a highly secure system may be difficult or impossible to integrate in an open environment. A highly available application may trade-off lower performance for greater availability. An application that requires high performance may be tied to a particular platform, and hence not be easily portable. Understanding trade-offs between quality attribute requirements, and designing a solution that makes sensible compromises is one of the toughest parts of the architect role.

3.7 MIDDLEWARE ARCHITECTURE

"Middleware is a layer of software that enables interaction and transmission of information between assorted applications and services"

3.7.1 TYPES OF MIDDLEWARE

The word 'middleware' is a broad term covering different types of middleware based on the various functionalities they provide. Some common types of middleware include:

1. **Database middleware:** This is the most common middleware that enables access and interaction with different database gateways.

2. **Message oriented middleware:** It allows software applications across multiple operating systems and networking protocols to receive and send messages to each other.

3. **Portals:** Enterprises use portal middleware to facilitate interactions between client-facing systems and backend systems.

4. **Enterprise application integration:** This is a virtual layer that connects applications, data, processes, and services, irrespective of where they are located and what technology they are based on.

5. **Transactional middleware**: This type of middleware ensures that transactions between heterogeneous components go through all the necessary steps to reach completion.

6. **Content middleware:** This is used to abstract specific content that is similar to publish and subscribe models.

7. **Remote procedure call (RPC) middleware:** This allows one application to trigger a function in another application, whether they reside in the same network.

8. **Device middleware:** This type contains a specific set of capabilities to integrate with or build applications on specific devices. It is usually used to build mobile applications.

9. **Object request broker (ORB) middleware**: This broker handles requests from one application to another without letting the applications worry about where each is hosted.

10. **Platform middleware:** This kind of middleware allows business logic to be seated in any platform (OS or hardware), including web servers, application servers, hosting environments, or containers.

11. **Application server middleware**: This framework allows the creation, deployment, and maintenance of enterprise applications within a system.

12. **Web middleware:** This allows companies to integrate more easily with individual backend systems, just like the ecommerce example mentioned earlier.

13. **Robotics middleware**: This simplifies the integration of robotic hardware, firmware, and software, irrespective of manufacturer and location.

14. **Cloud middleware:** Cloud middleware sits between the operating systems behind the cloud and the cloud users, providing a remote platform to create, maintain, and communicate with the hosted applications and data.

We will use portals, transactional and Common Object Request Breaker Architecture (CORBA) middleware for online library management system.

3.8 WEB SERVICES

"Web services are a set of integration technology standards that were designed specifically to meet the requirements arising from service-oriented architectures and systems"

All application integration technologies, including Web services, really only provide four basic functions that let developers (and programs) do the following:

* Find suitable services -UDDI
* Find out about a service -WSDL
* Ask a service to do something -SOAP
* Make use of services such as security -WS-* standards)

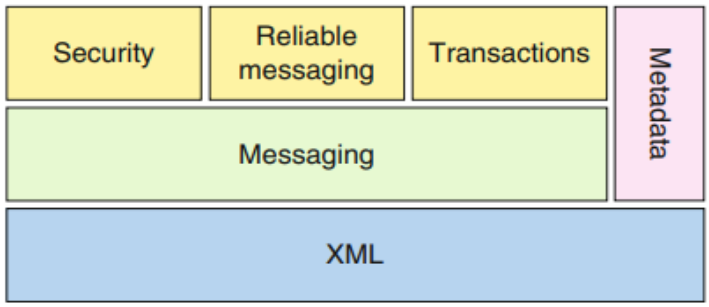


Fig. 5.2 Overview of Web services standards

SOAP, WSDL, and UDDI were the first Web services standards to be published, but they only meet the most basic requirements for application integration. They lack support for security, transactions, reliability, and many other important func tions. This gap is being progressively filled by a series of standards (commonly called "WS-*").

So we need SOAP and UDDI for our system implementation and suitability.


Design

4.1 INTRODUCTION

"Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation"

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization. Once the software requirements have been analyzed and specified (SRS) the software design is started.

Software design is the third step in SDLC (Software Development Life Cycle), which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS. The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer's requirements into finished software or a system.

4.2 SOFTWARE DESIGN LEVELS

Software design yields three levels of results.

- **Architectural Design -** The architectural design is the highest abstract version of the system. It identifies the software as a system with many components interacting with each other. At this level, designers get the idea of proposed solution domain.

  The all constraints in conventional library management sytem is get solved by this design proposal. It defined priviledges of student (user) and admin (librarian) and how the OLMS will established.

  *Online Library Management System (OLMS)

- **High-level Design-** The high-level design breaks the 'single entity-multiple component' concept of architectural design into less-abstracted view of sub-systems and modules and depicts their interaction with each other. High-level design focuses on how the system along with all of its components can be implemented in forms of modules. It recognizes modular structure of each sub-system and their relation and interaction among each other.

  User and admin registration, authorization, verification, dashbord, book lists, book request, approving etc all the components depicted in HLD (High Level Design).

- **Detailed Design-** Detailed design deals with the implementation part of what is seen as a system and its sub-systems in the previous two designs. It is more detailed towards modules and their implementations. It defines logical structure of each module and their interfaces to communicate with other modules. This design also known as Low Level Design (LLD).

  It's the last design level where system is decorated logically and programmers get a well purnished document for implement the sytem. This design reached our system to fourth SDLC phase development (coding).

4.3 MODULARIZATION

Modularization is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently. These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently.

Modular design unintentionally follows the rules of 'divide and conquer' problem-solving strategy this is because there are many other benefits attached with the modular design of a software.

Advantage of modularization:

- Smaller components are easier to maintain
- Program can be divided based on functional aspects
- Desired level of abstraction can be brought in the program
- Components with high cohesion can be re-used again
- Concurrent execution can be made possible
- Desired from security aspect

For the **OLMS** we modularized the whole system in small parts. These are – registration module, login module, dashboard module, book request – approve module, mailer module, fine and other functional module.

4.4 COHESION AND COUPLING

When a software program is modularized, its tasks are divided into several modules based on some characteristics. As we know, modules are set of instructions put together in order to achieve some tasks. They are though, considered as single entity but may refer to each other to work together. There are measures by which the quality of a design of modules and their interaction among them can be measured. These measures are called coupling and cohesion.

4.4.1 COHESION

Cohesion is a measure that defines the degree of intra-dependability within elements of a module. The greater the cohesion, the better is the program design.

There are seven types of cohesion, namely –

- **Co-incidental cohesion -** It is unplanned and random cohesion, which might be the result of breaking the program into smaller modules for the sake of modularization. Because it is unplanned, it may serve confusion to the programmers and is generally not-accepted.
- **Logical cohesion -** When logically categorized elements are put together into a module, it is called logical cohesion.
- **Temporal Cohesion -** When elements of module are organized such that they are processed at a similar point in time, it is called temporal cohesion.
- **Procedural cohesion -** When elements of module are grouped together, which are executed sequentially in order to perform a task, it is called procedural cohesion.
- **Communicational cohesion -** When elements of module are grouped together, which are executed sequentially and work on same data (information), it is called communicational cohesion.
- **Sequential cohesion -** When elements of module are grouped because the output of one element serves as input to another and so on, it is called sequential cohesion.

- **Functional cohesion -** It is considered to be the highest degree of cohesion, and it is highly expected. Elements of module in functional cohesion are grouped because they all contribute to a single well-defined function. It can also be reused.

So we are aimed to achive all types of cohesion for high quality system of online library management.

4.4.2 COUPLING

Coupling is a measure that defines the level of inter-dependability among modules of a program. It tells at what level the modules interfere and interact with each other. The lower the coupling, the better the program.

There are five levels of coupling, namely -

- **Content coupling -** When a module can directly access or modify or refer to the content of another module, it is called content level coupling.
- **Common coupling-** When multiple modules have read and write access to some global data, it is called common or global coupling.
- **Control coupling**- Two modules are called control-coupled if one of them decides the function of the other module or changes its flow of execution.
- **Stamp coupling-** When multiple modules share common data structure and work on different part of it, it is called stamp coupling.
- **Data coupling-** Data coupling is when two modules interact with each other by means of passing data (as parameter). If a module passes data structure as parameter, then the receiving module should use all its components.

Ideally, no coupling is considered to be the best. So we have to careful sothat there have minimum coupled modules in our OLMS.

4.5 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed. There are various kinds of methods in software design. They are as follows:

- Use case Diagram
- Sequence Diagram
- Activity Diagram
- Class Diagram

4.5.1 USECASE DIAGRAMS

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what's called an actor. Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can't do.

Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

- The purpose is to show the interactions between the use case and actor.
- To represent the system requirements from user's perspective.
- An actor could be the end-user of the system or an external system.

A Use case is a description of set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship.

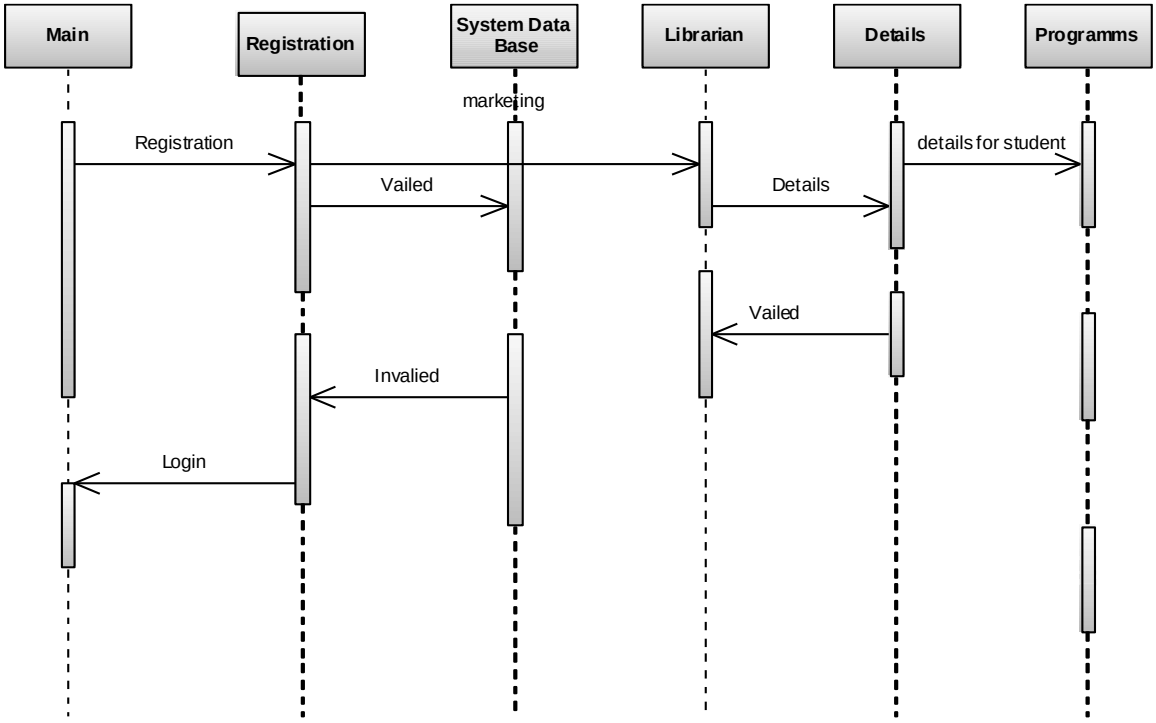

Use Case Diagram For
Library Management System
View And Update Profile
Add and Delete Books
Update Book Info
View Student Info
Register Account
Send Mail to user About Fine
Password Recover
Delete Student
Post Announcement
Fine Details
Approve New Admin
Approve Book Request
Give Feedback
Notify About Fine
View Book List
View Issue Information
Search Book
Book Request
View Announcement
Admin
Student

.

4.5.2 SEQUENCE DIAGRAM

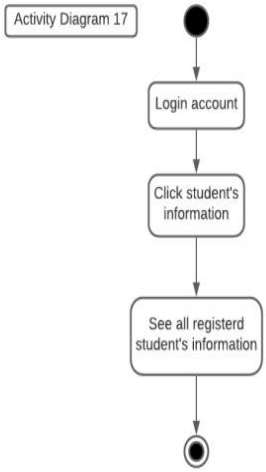
Sequence diagram and collaboration diagram are called interaction diagrams. An interaction diagram shows an interaction, consisting of set of objects and their relationship including the messages that may be dispatched among them.

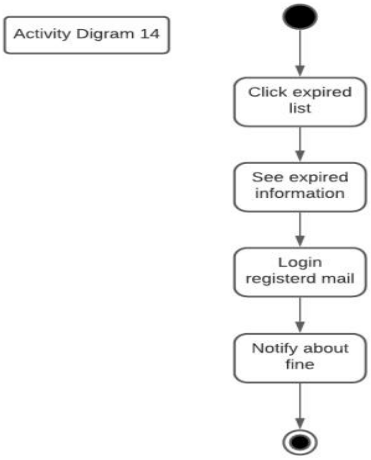
A sequence diagram is an introduction that empathizes the time ordering of messages. Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages ordered in increasing time along the Y-axis



4.5.3 ACTIVITY DIAGRAM

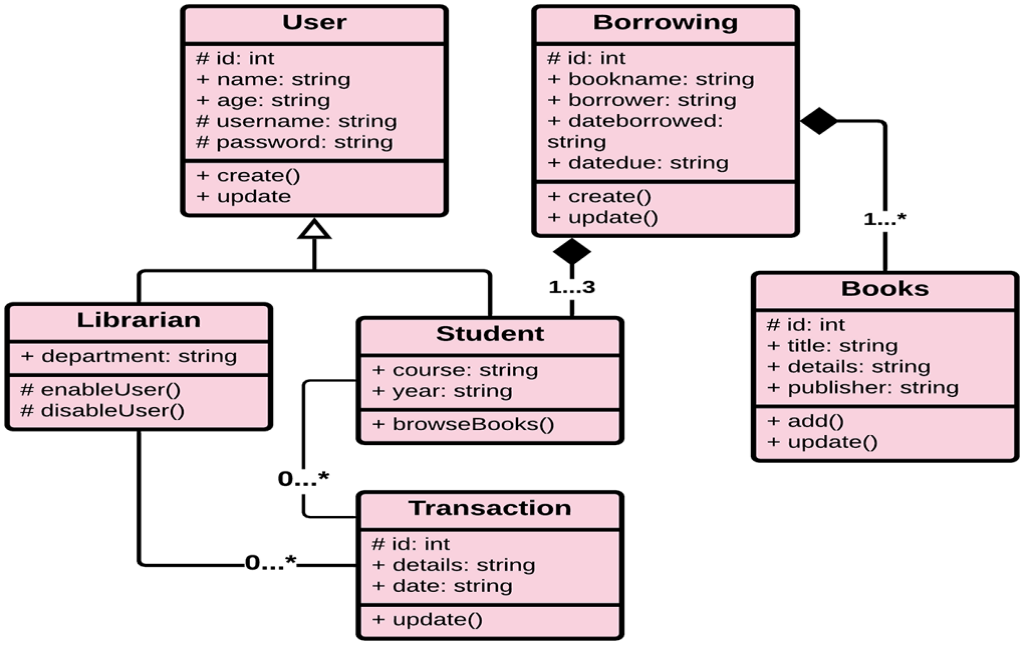
Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.





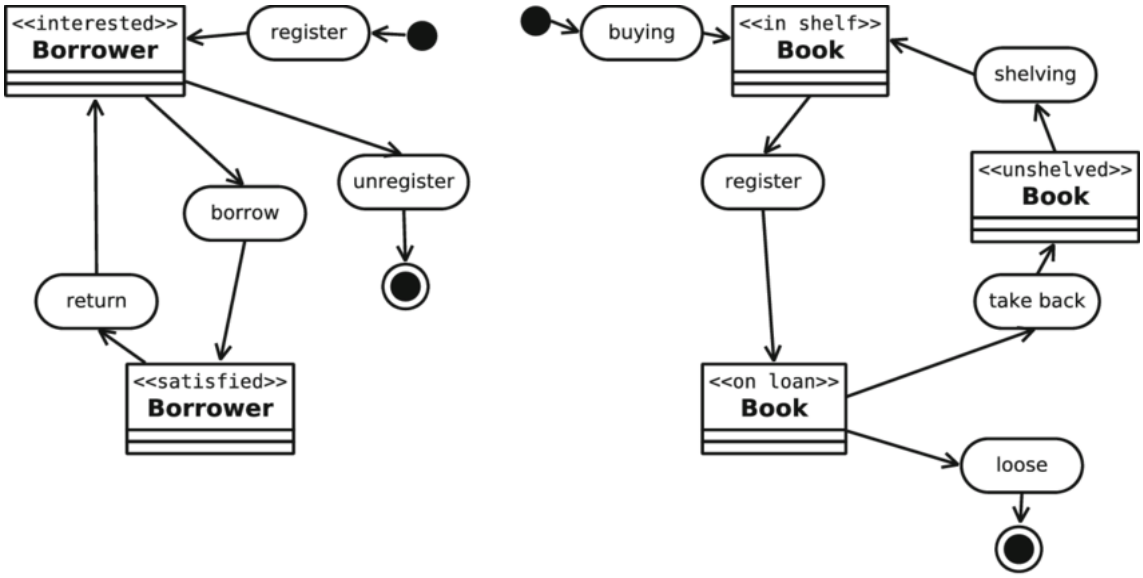
4.5.4 CLASS DIAGRAM

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, operations (or methods), and the relationships among objects.



4.5.5 STATE MACHINE DIAGRAM

State Machine Diagrams (or sometimes referred to as state diagram, state machine or state chart) show the different states of an entity. State machine diagrams can also show how an entity responds to various events by changing from one state to another.



4.6 ARCHITECTURE VIEW

"Architecture views are representations of the overall architecture that are meaningful to one or more stakeholders in the system"

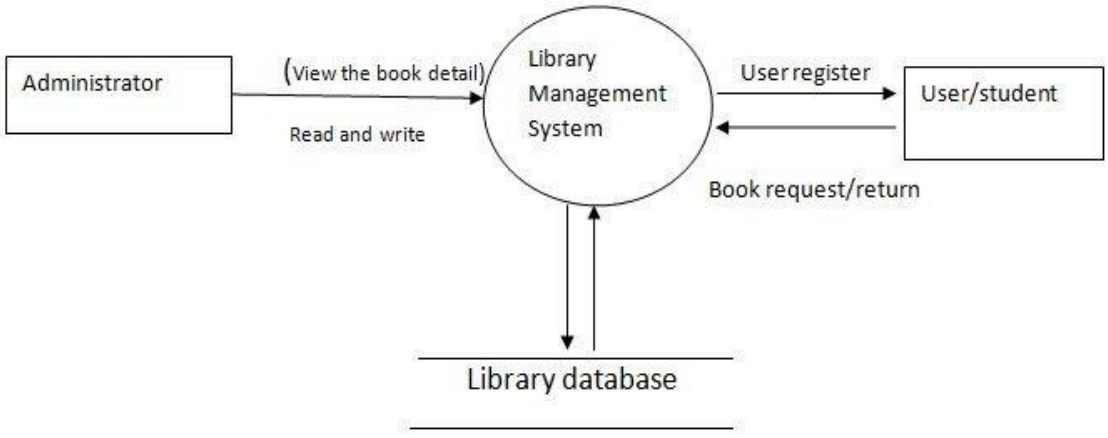


Figure: OLMS Architecture View

4.7 DATA FLOW DIAGRAMS (DFD)

The DFD takes an input-process-output view of a system i.e. data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software. Data objects represented by labeled arrows and transformation are represented by circles also called as bubbles.

A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrow represents data objects or object hierarchy.

4.8 E-R DIAGRAMS

The Entity-Relationship (ER) model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represents data objects.

- It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.
- It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.
- In addition, the model can be used as a design plan by the database developer to implement a data model in a specific database management software.

4.8.1 CONNECTIVITY AND CARDINALITY

The basic types of connectivity for relations are: one-to-one, one-to-many, and many-to-many. A *one-to-one* (1:1) relationship is when at most one instance of a entity A is associated with one instance of entity B. For example, "employees in the company are each assigned their

own office. For each employee there exists a unique office and for each office there exists a unique employee.

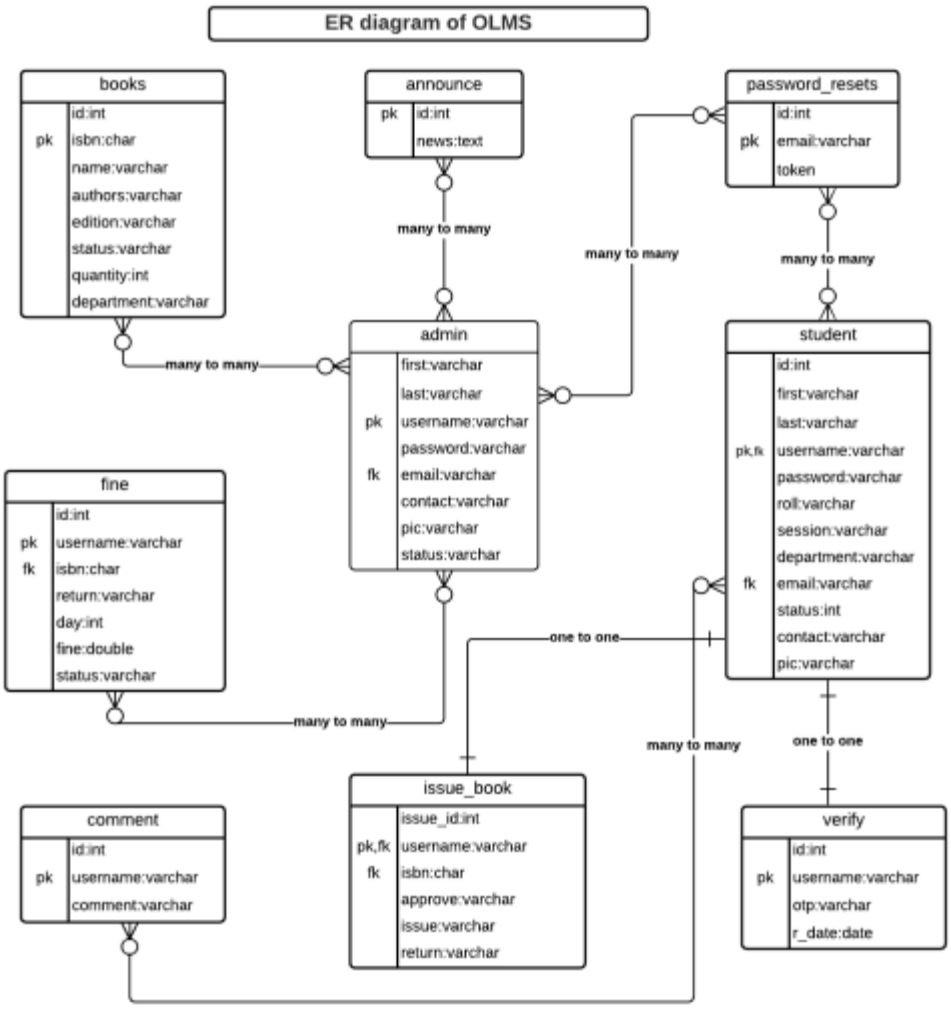
A *one-to-many* (1:N) relationships is when for one instance of entity A, there are zero, one, or many instances of entity B, but for one instance of entity B, there is only one instance of entity A. An example of a 1:N relationships is a department has many employees each employee is assigned to one department.

A *many-to-many* (M:N) relationship, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instances of entity B and for one instance of entity B there are zero, one, or many instances of entity A. The connectivity of a relationship describes the mapping of associated

4.8.2 ER NOTATION

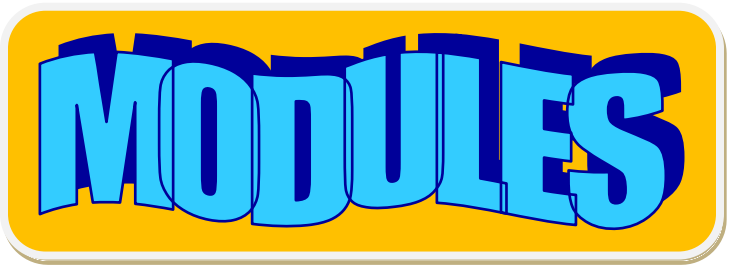
All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The notation used in this document is from Martin. The symbols used for the basic ER constructs are:

- **Entities** are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.
- **Relationships** are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs
- **Attributes**, when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.
- **Cardinality** of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.
- **Existence** is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional



4.9 RISKS OF OLMS

1. The risk of not acquiring materials that are critical to the continued development of the research collections that meet the needs of Congress and the research community;
2. The risk of failing to make the collections available to users in a timely and appropriate fashion;
3. The risk of not preserving the collections from the physical degradation inherent in each of the various media the Library holds, and from deterioration through use; and
4. The risk of exposing the items in the collection to theft, mutilation, or accidental loss.


MODULES

5. MODULES / COMPONENTS

The proposed system categories and follows these modules to implement.

Login component

1. Librarian (ADMIN)
2. Students (USER)

Librarian Component

1. Librarian Dashboard

Student Component

1. Books Details
2. Issue Details

5.1 MODULES DESCRIPSION

User: Using login id and password user can the use Library online where users can search for books and renewal books online. They can recommend for new books by just sending messages to the librarian from anywhere in the college. They can view the issue and return dates of any book and due they have to pay.

Registration: In the Registration module, user has to register himself by supplying his personal information which gets store in data base which are using as backend. By registering himself user will get his login id and Password so that he can access Library online. Separate Register form should be designed for separate user
(Student, Faculty, Librarian) and separate login has to provide for each user. For example, if the users are students then student id should be ASH/BKH********M/F.

Librarian: Librarian is a person who manages the Library. Librarian has the permission that he can access the database. There are some tasks which are performed by the Librarian like:

- Addition of a new book.
- Modification of the book.
- Deletion of the book.
- Searching of the book.
- Managing User

5.2 BACK END TECHNOLOGY

Microsoft SQL Server

Microsoft SQL Server is a Structured Query Language (SQL) based, client/server relational database. Each of these terms describes a fundamental part of the architecture of SQL Server.

Database

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format. A database typically has two components: the files holding the physical database and the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including:

- Maintaining the relationships between data in the database.
- Ensuring that data is stored correctly and that the rules defining data relationships are not violated.
- Recovering all data to a point of known consistency in case of system failures.

Relational Database

There are different ways to organize data in a database but relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to

the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory). When organizing data into tables, we can usually find many different ways to define tables. Relational database theory defines a process, normalization, which ensures that the set of tables you define will organize your data effectively.
We used relational database in our Online Library Management System.

Client/Server:-

In a client/server system, the server is a relatively large computer in a central location that manages a resource used by many people. When individuals need to use the resource, they connect over the network from their computers, or clients, to the server.

Server applications are usually capable of working with several clients at the same time. As OLMS system will handle so many students at a time so we need establish client – server system. That will be infrangible and reliable.

.

Structured Query Language (SQL):-

To work with data in a database, we must use a set of commands and statements (language) defined by the DBMS software. There are several different languages that can be used with relational databases; the most common is SQL.

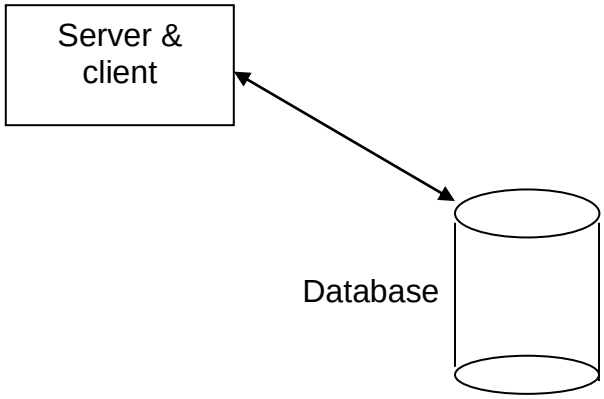
We will implement sql to interact with our relational database. It is convenient and easy for developers.

5.3 DATABASE MODELS

There are single tier, two tier and multi-tier database model. We can use single or two tier model in our Online Library Management System.

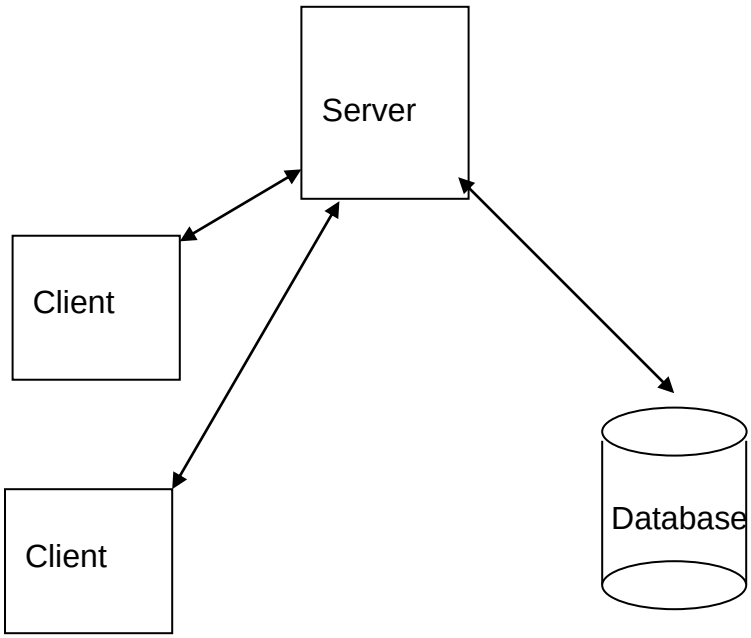
Single Tier / 1-Tier

In a single tier the server and client are the same in the sense that a client program that needs information (client) and the source of this type of architecture is also possible in java, in case flat files are used to store the data. However this is useful only in case of small applications. The advantage with this is the simplicity and portability of the application developed.



Two Tier (client-server)

In two tier architecture the database resides in one machine and client in different machine they are connected through the network. In this type of architecture a database management takes control of the database and provides access to clients in a network. This software bundle is also called as the server. Software in different machines, requesting for information are called as the clients.



Any of the two models we can implement in our OLMS.

5.4 INTERNET AND INTRANET

Internet is network of computers. It is not just a network, but a network of networks. Internet is global communication system of diverse, inter-connected computer network for the exchange of information of virtually every remote area.

Intranet is a local or restricted communications network, especially a private network created for any specific organization for their secrecy. It is bounded network through the target organization.

If we want to use Online Library Management System for a university then we can set up intranet network and if we want world wide access then it need to set up in internet.

5.5 DATABASE TABLES

Table: insertbook

BOOKID	INT,
TITLE	VARCHAR(150),
AUTHOR	VARCHAR(100),
YEARPUBLISHED	VARCHAR(20),
CATEGORY	VARCHAR(100),
PUBLISHER	VARCHAR(100),
EDITION	VARCHAR(50),
COST	VARCHAR(20),
DATEOFENTRY	DATETIME,
LIBRARYCOPY	INT,
STATUS	VARCHAR(20),
CONDITION	VARCHAR(20)

Table: librarybooks

BOOKID	INT NULL,
TITLE	VARCHAR(150) NULL,
AUTHOR	VARCHAR(100) NULL,
YEARPUBLISHED	VARCHAR(20) NULL,
CATEGORY	VARCHAR(100) NULL,
PUBLISHER	VARCHAR(100) NULL,
EDITION	VARCHAR(50) NULL,
COST	VARCHAR(20) NULL,
DATEOFENTRY	DATETIME NULL,
LIBRARYCOPY	INT NULL,
STATUS	VARCHAR(20) NULL,
CONDITION	VARCHAR(20) NULL

Table: issuebook

TRNSID	INT NULL,
ROLLNUMBER	INT NULL,
NAME	VARCHAR(150) NULL,
AGE	INT NULL,
GENDER	VARCHAR(10) NULL,
BRANCH	VARCHAR(10) NULL,
YEAR	INT NULL,
BARTYPE	VARCHAR(15) NULL,
BOOKID	INT NULL,
TITLE	VARCHAR(150) NULL,
AUTHOR	VARCHAR(150) NULL,
CATEGORY	VARCHAR(100) NULL,
EDITION	VARCHAR(20) NULL,
PUBLISHER	VARCHAR(50) NULL,
LIBRARYCOPY	INT NULL,
ISSUEDATE	DATETIME NULL,
DUEBACKBY	DATETIME NULL

5.6 FEASIBILITY STUDY

Feasibility study is conducted once the problem is clearly understood. Feasibility study is a high level capsule version of the entire system analysis and design process. The objective is to determine quickly at a minimum expense how to solve a problem. The purpose of feasibility is not to solve the problem but to determine if the problem is worth solving.

The system has been tested for feasibility in the following points.

1. Technical Feasibility
2. Economical Feasibility
3. Operational Feasibility.

1. Technical Feasibility

The project entitles "Courier Service System" is technically feasibility because of the below mentioned feature. The project was developed in Java which Graphical User Interface.

It provides the high level of reliability, availability and compatibility. All these make Java an appropriate language for this project. Thus the existing software Java is a powerful language.

2. Economical Feasibility

The computerized system will help in automate the selection leading the profits and details of the organization. With this software, the machine and manpower utilization are expected to go up by 80-90% approximately. The costs incurred of not creating the system are set to be great, because precious time can be wanted by manually.

3. Operational Feasibility

In this project, the management will know the details of each project where he may be presented and the data will be maintained as decentralized and if any inquires for that particular contract can be known as per their requirements and necessaries.

5.7 MAINTENANCE AND ENVIRONMENT

The maintenance phase focuses on change that is associated with error correction, adaptations required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements. Only about 20 percent of all maintenance work are spent "fixing mistakes". The remaining 80 percent are spent adapting existing systems to changes in their external environment, making enhancements requested by users, and reengineering an application for use. Four types of changes are encountered during the maintenance phase.

1. Correction
2. Adaptation
3. Enhancement
4. Prevention

1. Correction:

Even with the best quality assurance activities is lightly that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects. If our OLMS has any bug leaking or release then these will be fixed later.

2. Adaptation

Over time, the original environment for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate change to its external environment. As our OLMS environment may be change with time so adaptation is important for a robust system.

3. Enhancement

As software is used, the user will recognize additional functions that will provide benefit. Perceptive maintenance extends the software beyond its original function requirements. Benficial and emergence feature will have to add to our OLM system as cope with modern technology.

4. Prevention

Computer software deteriorates due to change, and because of this, preventive maintenance, often called software re-engineering, must be conducted to enable the software to serve the needs of its end users. In essence, preventive maintenance makes changes to computer programs so that they can be more easily corrected, adapted, and enhanced.

5.8 SOFTWARE METHODOLOGY

The software methodology followed in this project includes the object-oriented methodology and the application system development methodologies. Although there are a growing number of applications that should be developed using an experimental process strategy such as prototyping, a significant amount of new development work continue to involve major operational applications of broad scope. The application systems are large highly structured. User task comprehension and developer task proficiency is usually high. These factors suggest a linear or iterative assurance strategy. The most common method for this stage class of problems is a system development life cycle modal in which each stage of development is well defined and has straightforward requirements for deliverables, feedback and sign off.

The basic idea of the system development life cycle is that there is a well-defined process by which an application is conceived and developed and implemented. Our online library management system will follow by SDLC (Software Development Life Cycle).

6. CONCLUSION

Library Management System of the entire system improves the efficiency. It provides a friendly graphical user interface which proves to be better when compared to the existing system. It gives appropriate access to the authorized users depending on their permissions. It effectively overcomes the delay in communications. Updating of information becomes so easier. System security, data security and reliability are the striking features. The System has adequate scope for modification in future if it is necessary.

7. FUTURE ENHANCEMENTS:

Use barcode technology

Updating more features like e-book reading

Implement in Android version