OODP PROJECT

UML DIAGRAM

LIBRARY MANAGEMENT SYSTEM

Submitted By -

GROUP MEMBERS	REGISTERATION NUMBER
RAGHAV KAPOOR	RA2011026010117
VISHVESH BHARDWAJ	RA2011026010109
UTKARSH SRIVASTAVA	RA2011026010104

INDEX

Abstract :

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.

UML Diagram -

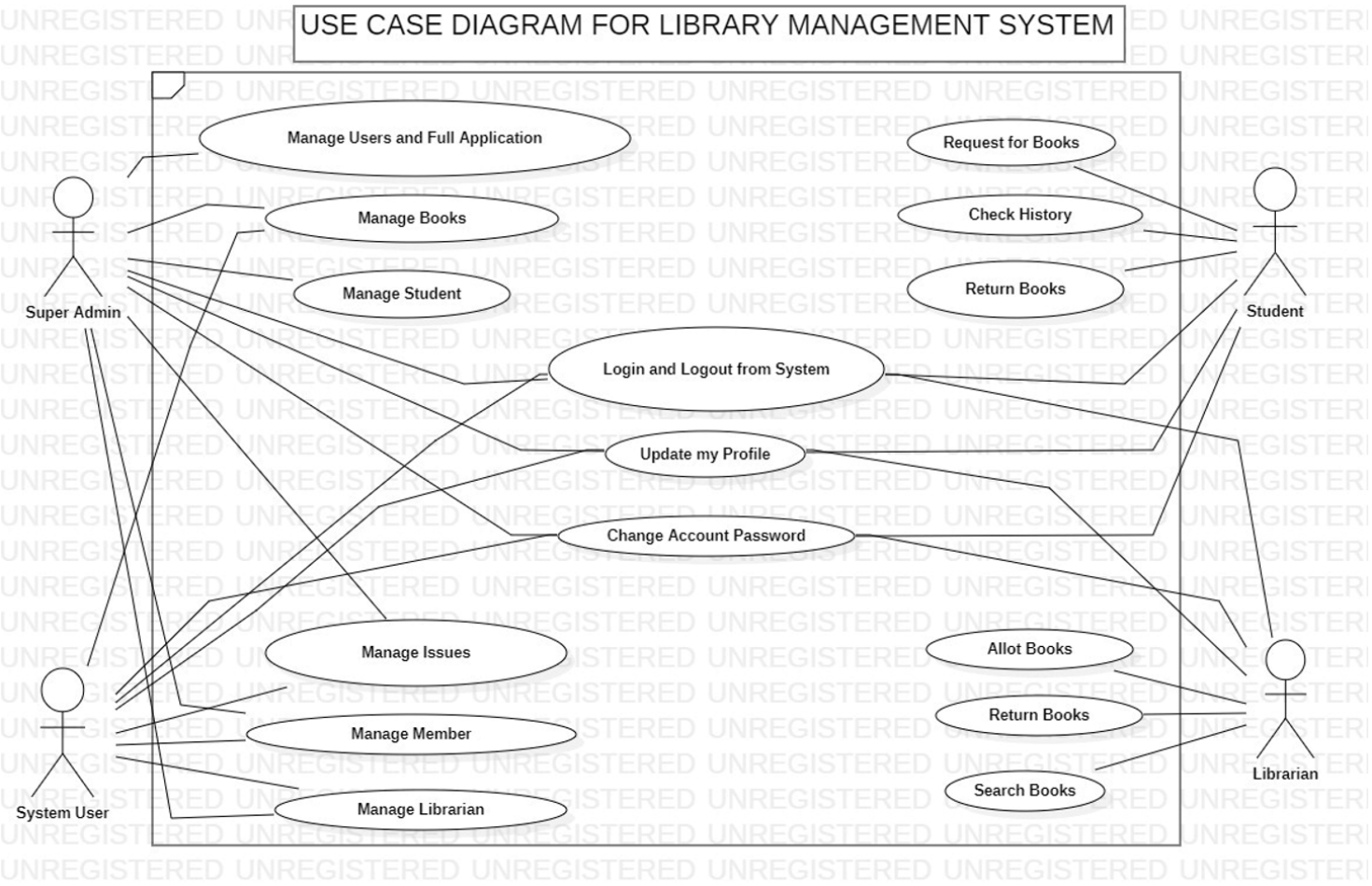
UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development.

UML stands for Unified Modelling Language. UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices.

- Improved integration between structural models like class diagrams and behaviour models like activity diagrams.
- Added the ability to define a hierarchy and decompose a software system into components and sub-components.
- The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite structure diagram, interaction overview diagram, and timing diagram. It also renamed state chart diagrams to state machine diagrams, also known as state diagrams.

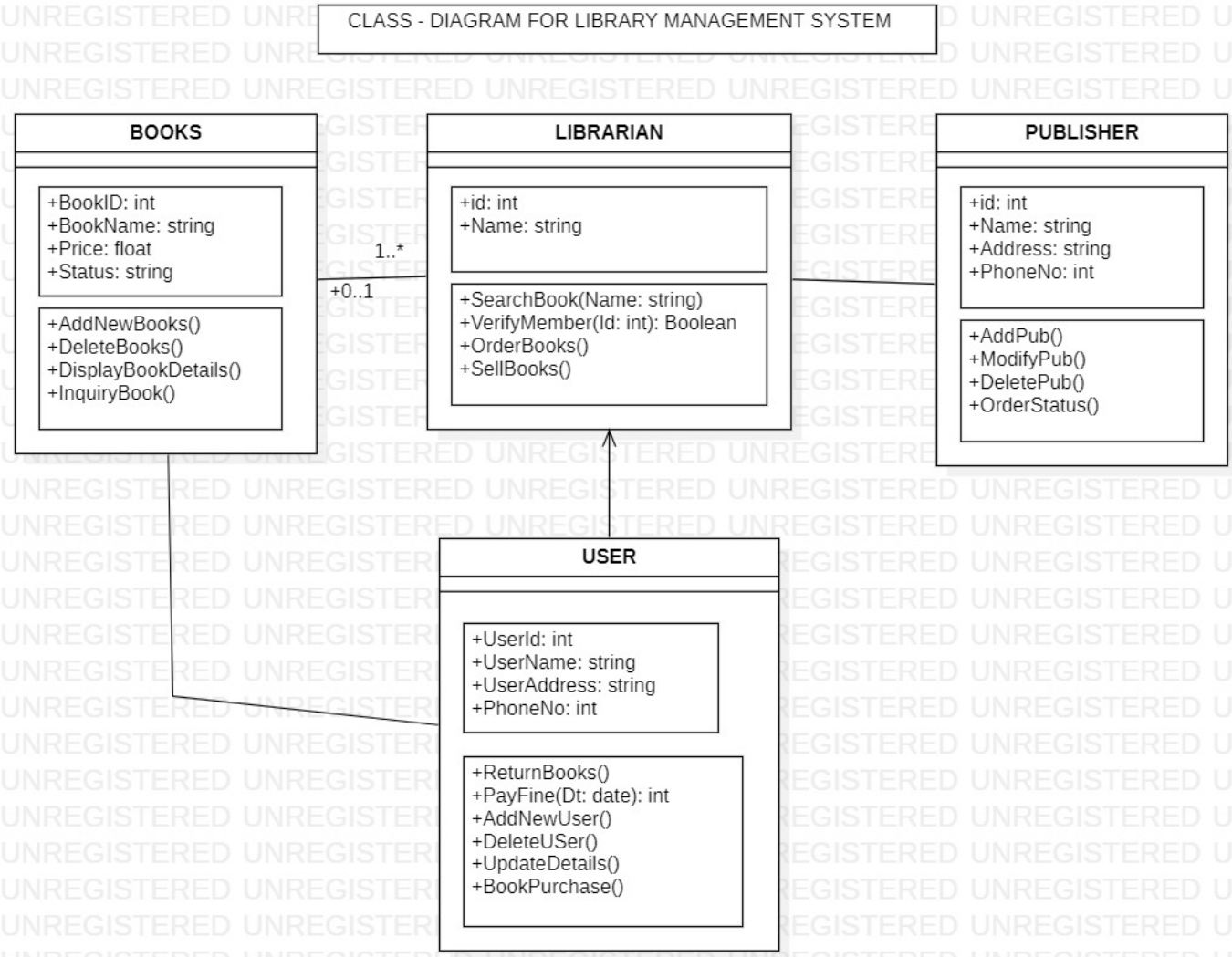
USE - CASE DIAGRAM :

Use case diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. The purpose of use case diagram is to capture the dynamic aspect of a system.



CLASS - DIAGRAM :

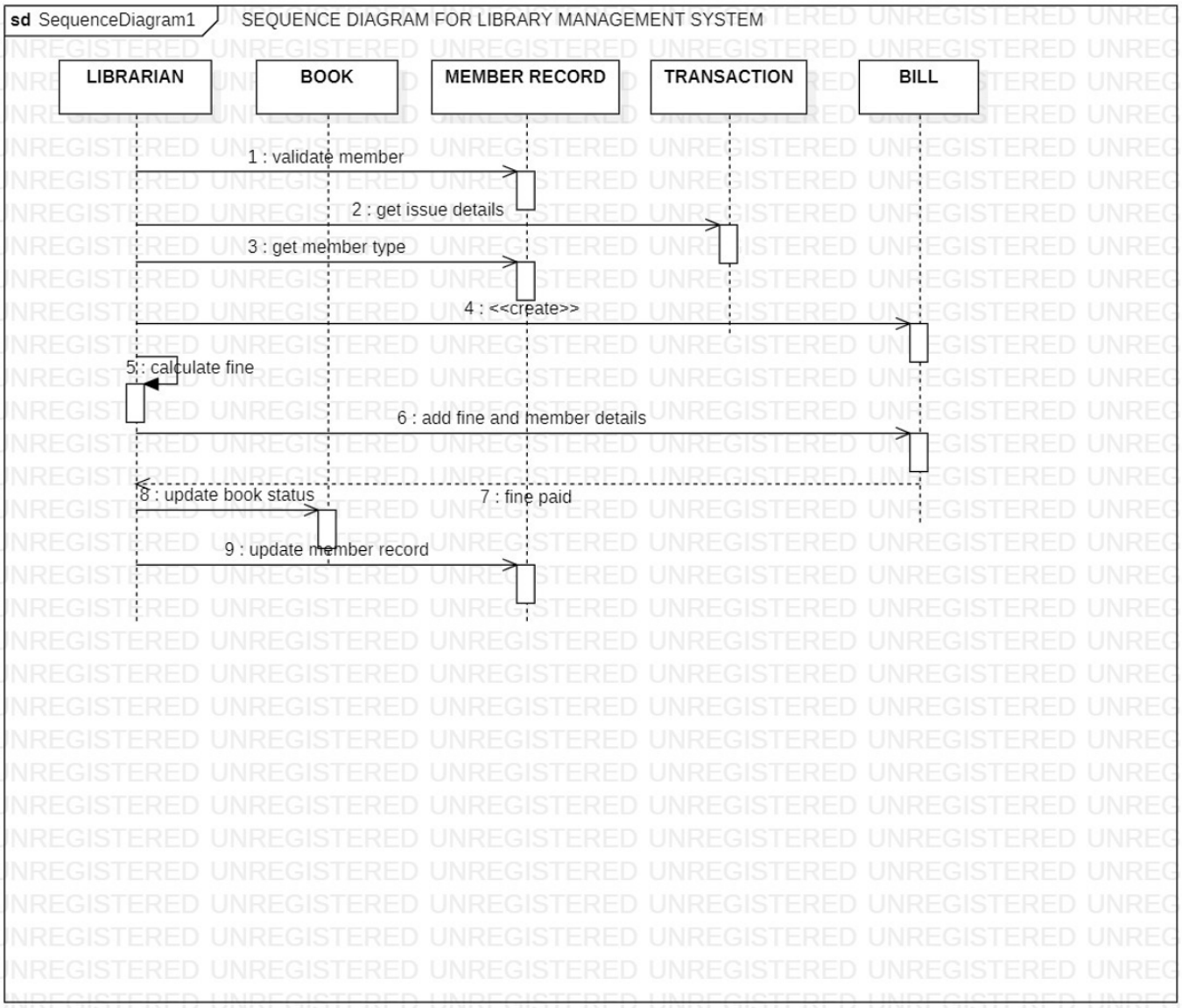
A Class Diagram describes the structure, class, attributes, operations (or methods), and the relationship among various objects. It provides a conceptual modelling of the structure of application and a detailed translation of the model into programming code. The relationship among the different attributes and operations of the system is depicted by this UML Diagram.



SEQUENCE DIAGRAM :

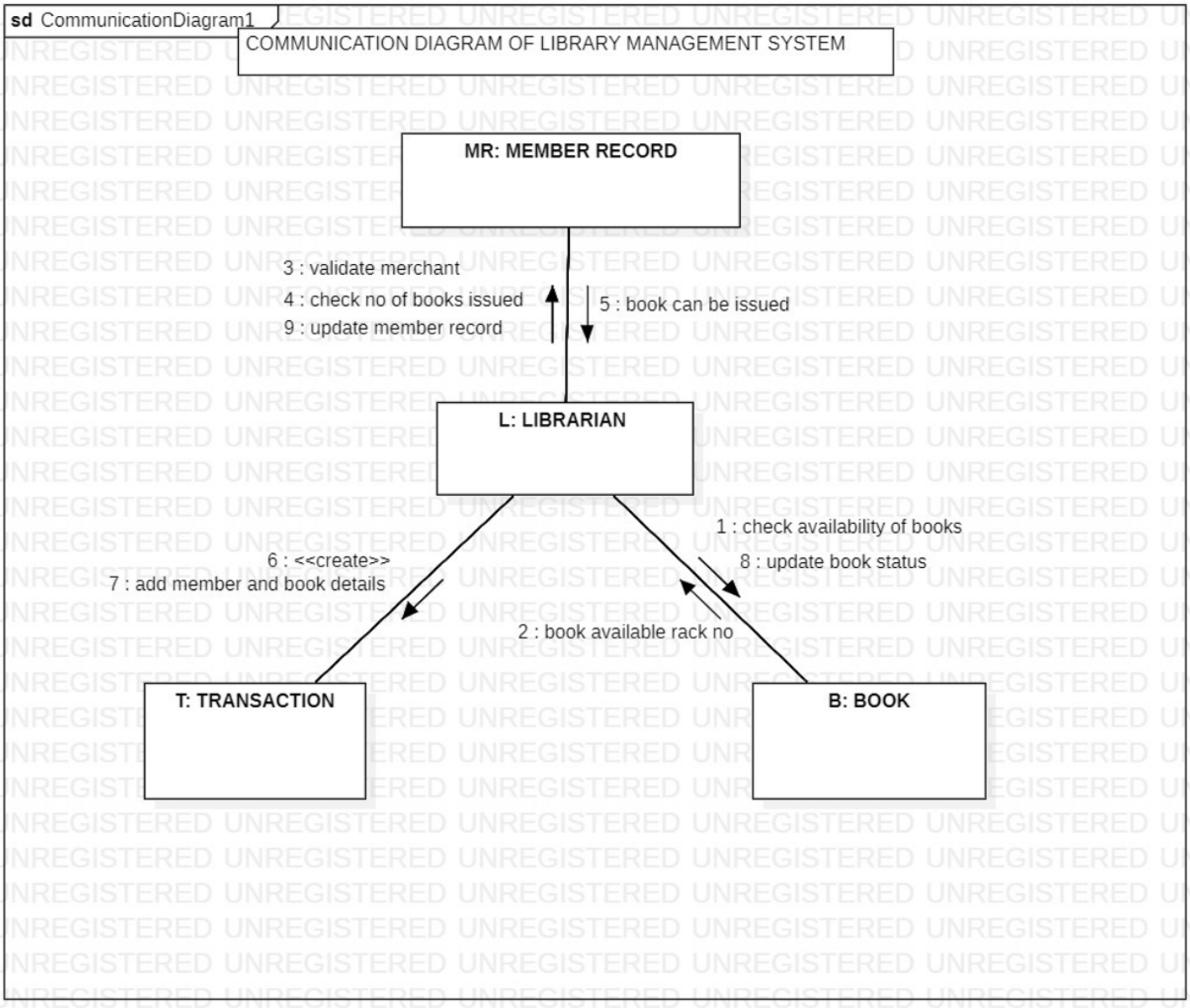
Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a number of lifelines. Sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

Sequence Diagrams show elements as they interact over time and they are organized according to object (horizontally) and time (vertically)



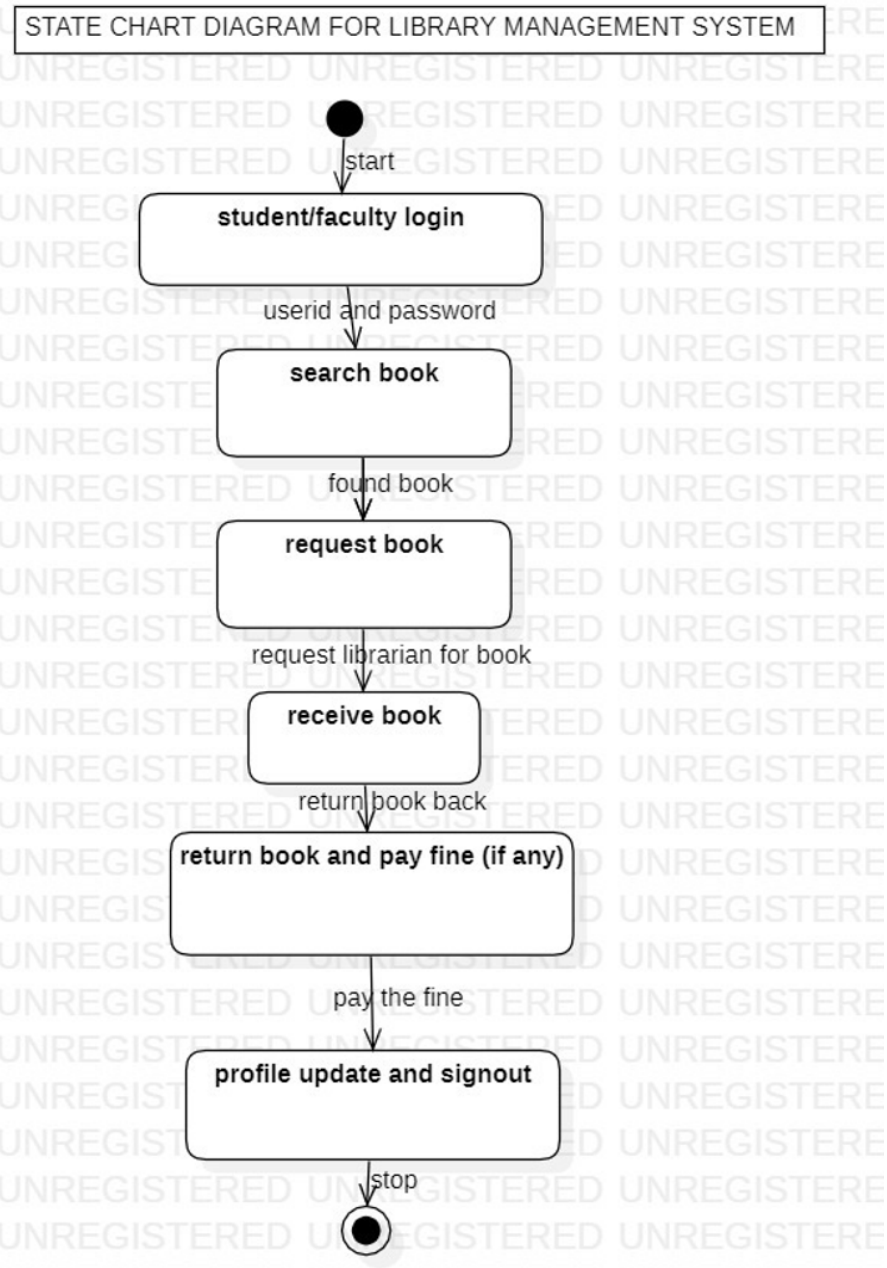
COMMUNICATION - DIAGRAM :

Communication diagrams, like the sequence diagrams - a kind of interaction diagram, shows how objects interact. A communication diagram is an extension of object diagram that shows the objects along with the messages that travel from one to another. In addition to the associations among objects, communication diagram shows the messages the objects send each other.



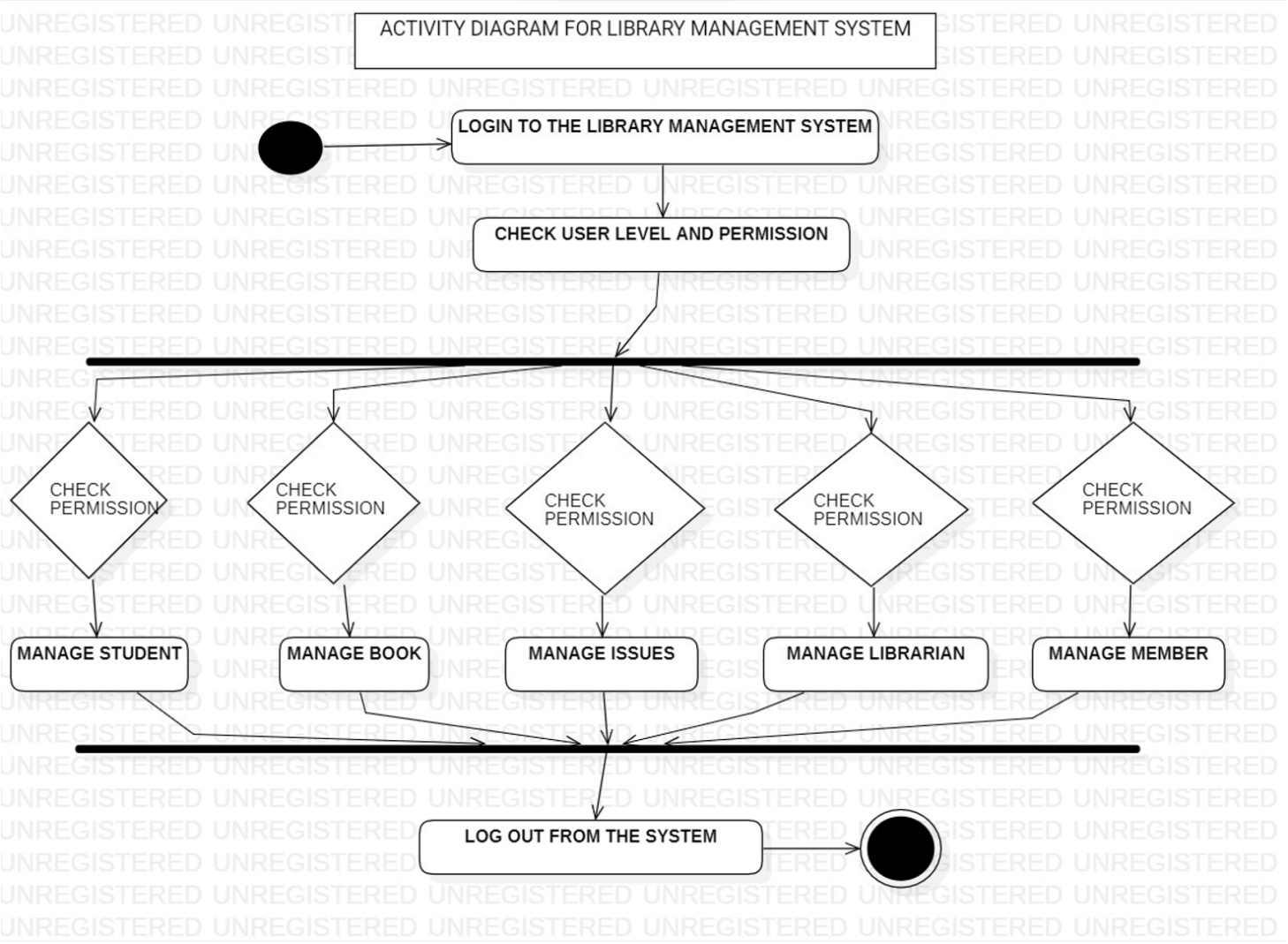
STATE - CHART DIAGRAM :

It describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.



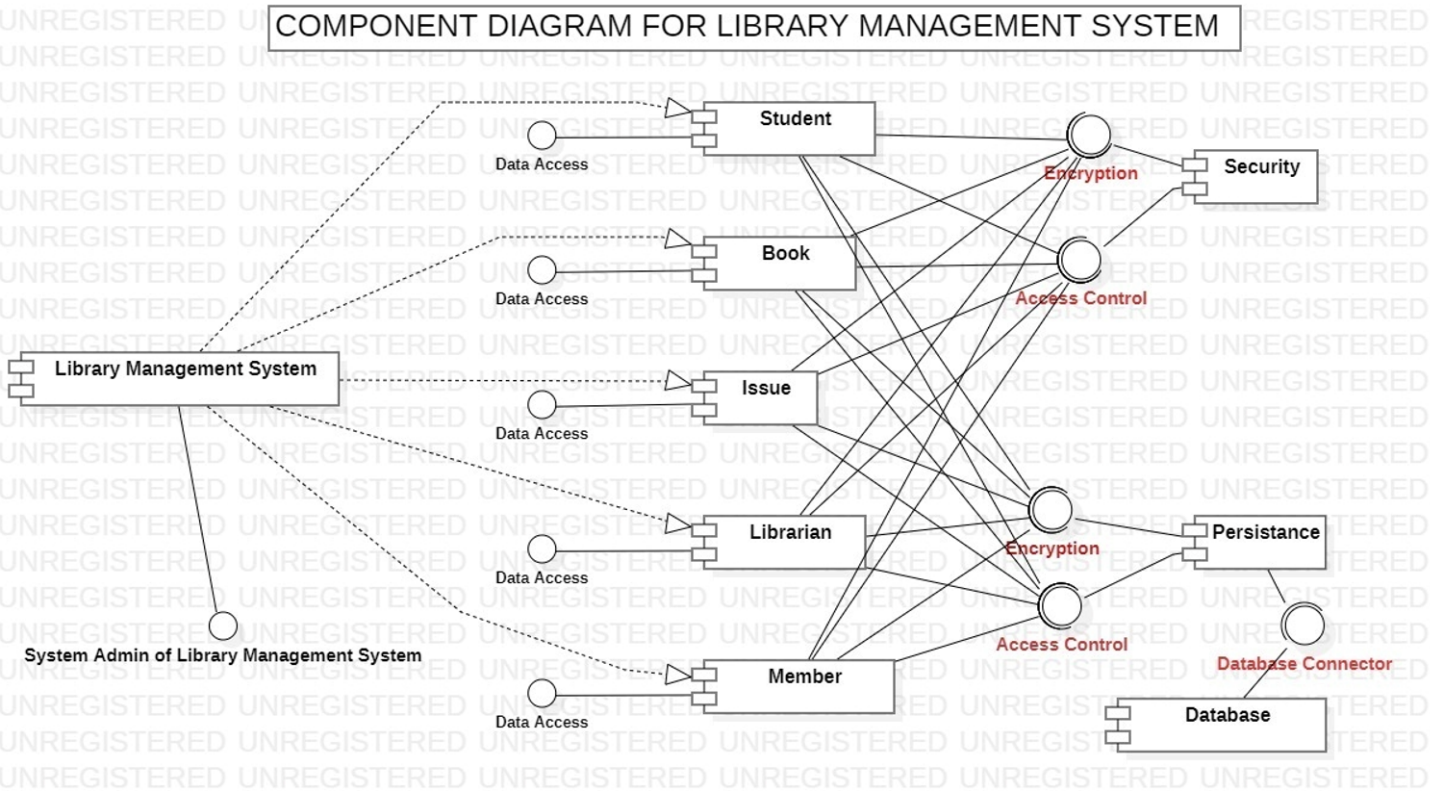
ACTIVITY - DIAGRAM :

An activity diagram is a behavioural diagram that depicts the behaviour of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.



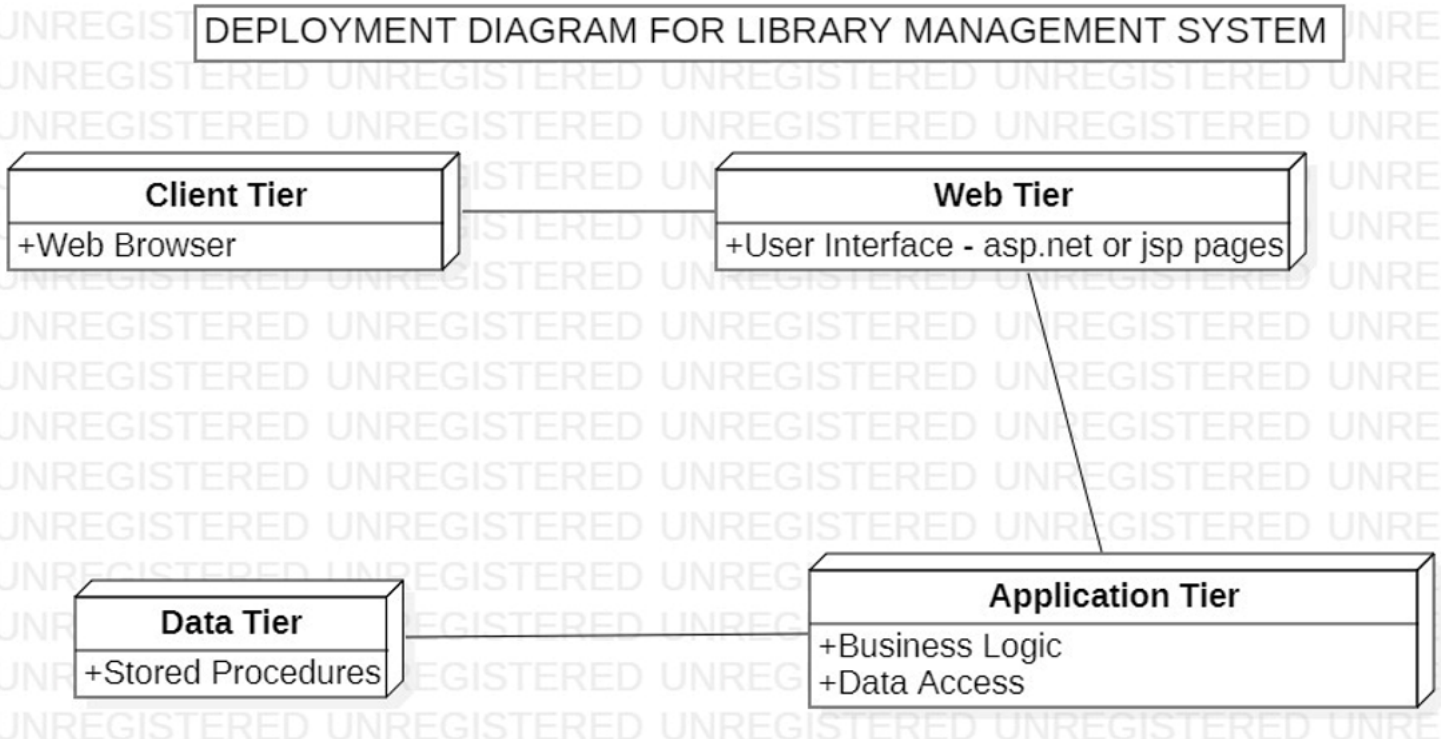
COMPONENT - DIAGRAM :

Component diagrams are used in modelling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.



DEPLOYMENT - DIAGRAM :

Deployment diagrams are used to model the configuration of run-time processing elements and the software components, processes, and objects that live on them. In the deployment diagram, you start by modelling the physical nodes and the communication associations that exist between them. For each node, you can indicate what component instances live or tun on the node. You can also model the objects that are contained within the component.



CONCLUSION

In the modern world, the use of computers is becoming rampant. The recent developments in technology have revolutionized and consequently brought a paradigm shift in the way activities are accomplished. UML helps to organize, plan and visualize a program. In addition, being a standard, it is widely used and accepted as the language for outlining programs. UML is used in a variety of purposes and its readability and re-usability make it an ideal choice for programmers. This report has presented a simple, convenient, cost-effective, but efficient system with a user-friendly, sensitive and intelligible web interface. Whereby it can be accessed at any time provided there is internet. The diagram is meant to be understood by any type of programmer and helps to explain relationships in a program in a straightforward manner. Traditionally, to understand a program, a programmer would read the code directly. This could be thousands or millions of lines of code in very large programs. Having a UML diagram helps to quickly illustrate those relationships.

Software Used: STAR UML.

Submission -

RAGHAV KAPOOR - RA2011026010117

VISHVESH BHARDWAJ - RA2011026010109

UTKARSH SRIVASTAVA - RA2011026010104