LIBRARY MANAGEMENT SYSTEM

PROJECT REPORT

18CSC202J- OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

Ankit Patra (RA2111003011178)

Under the guidance of

G. Malarselvi

Assistant Professor

Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**LIBRARY MANAGEMENT SYSTEM"** is the bonafide work

of

ANKIT PATRA (RA2111003011178)

who undertook the task of completing the project within the allotted time.

Signature of the Guide

G. Malarselvi

Assistant Professor

Department of CINTEL,

SRM Institute of Science and Technology

Signature of the II Year Academic Advisor

Professor and Head

Department of CINTEL

SRM Institute of Science and Technology

About the course:-

18CSC202J/ 8AIC203J - Object Oriented Design and Programming are 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives:

The student should be made to:

- Learn the basics of OOP concepts in C++
- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques

Course Learning Rationale (CLR): The purpose of learning this course is to:

1. Utilize class and build domain model for real-time programs
2. Utilize method overloading and operator overloading for real-time application development programs
3. Utilize inline, friend and virtual functions and create application development programs
4. Utilize exceptional handling and collections for real-time object-oriented programming applications
5. Construct UML component diagram and deployment diagram for design of applications
6. Create programs using object-oriented approach and design methodologies for real-time application development

Course Learning Outcomes (CLO): At the end of this course, learners will be able to:

1. Identify the class and build domain model
2. Construct programs using method overloading and operator overloading
3. Create programs using inline, friend and virtual functions, construct programs using standard templates
4. Construct programs using exceptional handling and collections
5. Create UML component diagram and deployment diagram
6. Create programs using object oriented approach and design methodologies

TABLE 1: RUBRICS FOR LABORATORY EXERCISES

(Internal Mark Splitup:- As per Curriculum)

CLAP-1	5=(2(E-lab Completion) + 2(Simple Exercises)(from CodeZinger, and any other coding platform) + 1(HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-2	7.5=(2.0(E-lab Completion)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	Elab test
CLAP-3	7.5=(2.0(E-lab Completion(80 Pgms)+ 2.0 (Simple Exercises)(from CodeZinger, and any other coding platform) + 3.5 (HackerRank/Code chef/LeetCode Weekend Challenge)	**2 Mark -** E-lab Completion **80 Program** Completion from 10 Session (Each session min 8 program) **2 Mark -** Code to UML conversion GCR Exercises **3.5 Mark - Hacker Rank** Coding challenge completion
CLAP-4	5= 3 (Model Practical) + 2(Oral Viva)	• **3 Mark** – Model Test • **2 Mark** – Oral Viva
Total	25	

COURSE ASSESSMENT PLAN FOR OODP LAB

S.No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3,	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given
13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10
15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

LIST OF EXPERIMNENTS FOR UML DESIGN AND MODELLING:

TO DEVELOP A MINI-PROJECT BY FOLLOWING THE EXERCISES LISTED BELOW.

1. To develop a problem statement.
2. Identify Use Cases and develop the Use Case model.
3. Identify the conceptual classes and develop a domain model with UML Class diagram.
4. Using the identified scenarios, find the interaction between objects and represent themusing UML Sequence diagrams.
5. Draw relevant state charts and activity diagrams.
6. Identify the User Interface, Domain objects, and Technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

SUGGESTED SOFTWARE TOOLS FOR UML:

StarUML, Rational Suite, Argo UML (or) equivalent, Eclipse IDE and Junit

ABSTRACT

A Library Management System is a software built to handle the primary housekeeping functions of a library. Libraries rely on library management systems to manage asset collections as well as relationships with their members. Library management systems help libraries keep track of the books and their checkouts, as well as members' subscriptions and profiles.

Library management systems also involve maintaining the database for entering new books and recording books that have been borrowed with their respective due dates.

We will focus on the following set of requirements while designing the Library Management System:

1. Any library member should be able to search books by their title, author, subject category as well by the publication date.

2. The system should be able to collect fines for books returned after the due date.

3. There could be more than one copy of a book, and library members should be able to check-out and reserve any copy. We will call each copy of a book, a book item.

4. The system should be able to retrieve information like who took a particular book or what are the books checked-out by a specific library member.

MODULE DESCRIPTION

The Unified Modelling Language (UML) prescribes a standard set of diagrams and notations for modelling object oriented systems, and describes the underlying semantics of what these diagrams and symbols mean. Whereas there has been to this point many notations and methods used for object-oriented design, now there is a single notation for modellers to learn. UML can be used to model different kinds of systems: software systems, hardware systems and real-world organisations.

UML offers nine diagrams in which to model systems:

- Use Case diagram for modelling the business processes
- Class diagram for modelling the static structure of classes in the system
- Sequence diagram for modelling message passing between objects
- Communication diagram for modelling object interactions
- State Chart diagram for modelling the behaviour of objects in the system
- Activity diagram for modelling the behaviour of Use Cases, objects, oroperations
- Package diagram for modelling the static structure of objects in the system
- Component diagram for modelling components
- Deployment diagram for modelling distribution of the system.

The following sections present a more detailed look at Modelling with UML. Avery simple Airline Reservation System is used to illustrate UML Modelling techniques and diagrams.

The following topics are covered:

- Organising your system with packages.
- Modelling with Use Cases.
- Modelling with Sequence and Collaboration diagrams.
- Analysing and designing with the Class diagram.
- Modelling behaviour with State and Activity diagrams.

Use case diagram with explanation

Explanation:

The objects used in the use-case diagram are: -

1. System.
2. Actors.
3. Use cases.
4. Relationships.

The two actors involved in the library management system are: -

1. Student, who acts as the primary actor interacting with the library performing various functions.
2. Librarian, who acts as the secondary actor, managing while interacting with the library taking and fulfilling requests by the primary actor.

The use cases are described below as follows: -

1. Request Book: - The primary actor can request a book from the library upon requesting, one more use case is used in checking whether the actor is a member of the library or not. There is also one use case which is used only sometimes if the user ID is invalid.

2. Return Book: - The primary actor can request a book which was previously requested. Upon returning the book, it is checked whether the book is being returned late or not and fine is collected accordingly.

3. Feedback: - The primary actor(s) can also provide feedback upon which they have to fill a feedback form.

4. Update Database: - The database is updated whenever a use case is activated doing any function.

5. Add new Book: - The secondary actor, librarian, can add new books to the library. Upon doing so, the database is updated.

6. Remove a Book: - The secondary actor, librarian, can remove existing books from the library. Upon doing so, the database is updated.

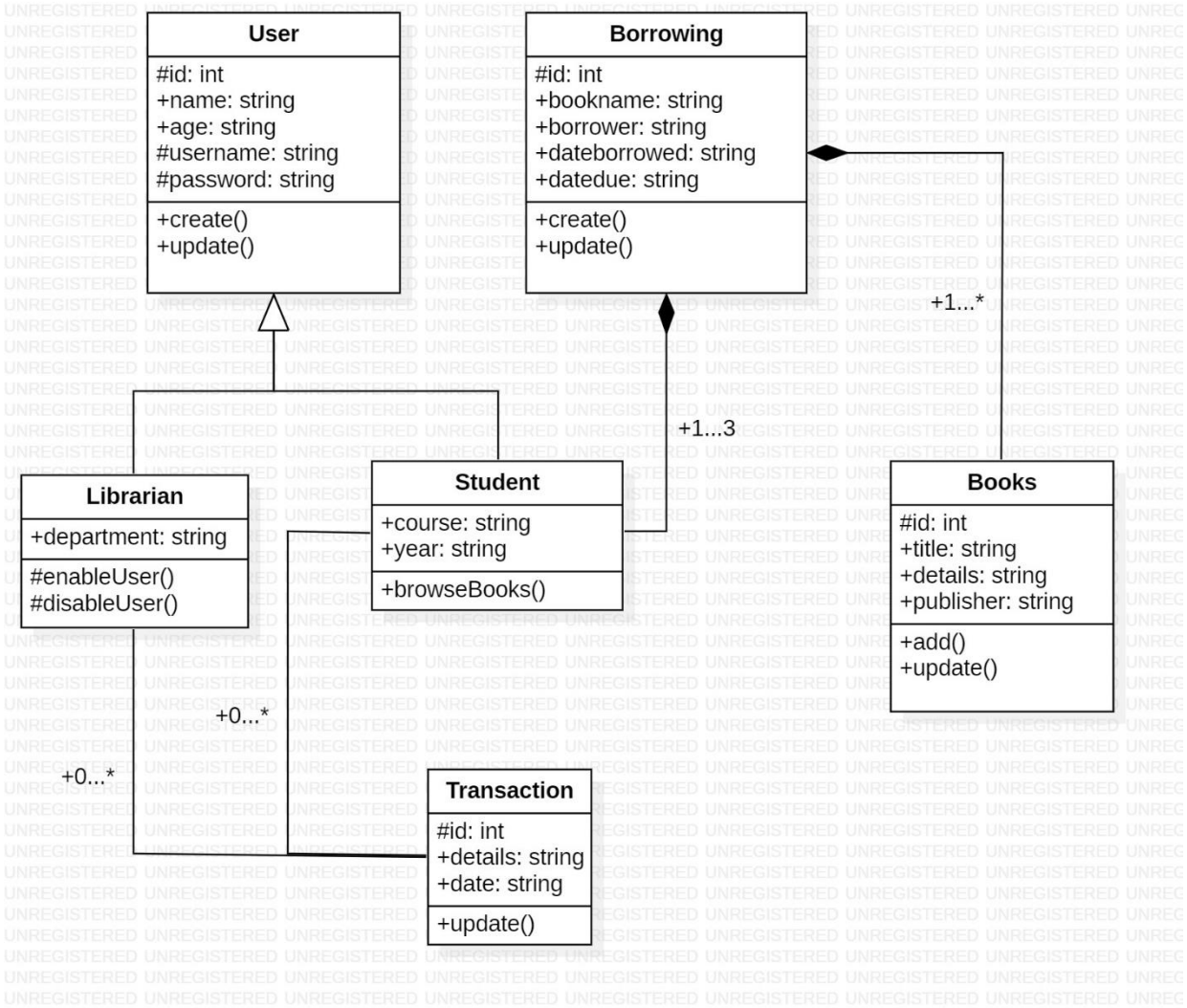
7. Update book Data: - The secondary actor, librarian, can update credentials of an existing book. Upon doing so, the database is updated.

The relationships used in the diagram are: -

1. Association, denoting a semantic condition.
2. Generalization, branching the primary actor.
3. <<include>>, denoting use cases always called upon whenever the former connecting use case is used.
4. <<exclude>>, denoting use cases which sometimes are called when the former connecting use case is used.

All the use cases and relationships are inside the system.

Class diagram with explanation



Explanation:

Class diagrams are generally used for conceptual modeling of static view of a software application, and for modeling translating models into programming code in a detailed manner. At time of developing or construction software systems, a class diagram is widely used. They are also used for data modeling. It is used to show classes, relationships among them, interface, association, etc. Class in a class diagram simply is a blueprint of an object. It simply describes and explains different type of objects in system, and different types of relationships that exist between them.

Classes of Library Management System: -

- Library class –
  It manages all operations of Library Management System. It is central part of organization for which software is being designed.

- Librarian Class – It manages all operations of Librarian.

- Book Class –
  It manages all operations of books. It is basic building block of system. It has two generalized classes connected (Journal and study_books).

- Member data Class –
  It manages member data. It has two generalized classes connected (Student and staff_member).

Attributes of Library Management System :

- Library Management System Attributes –
  Name, address, contact info.

- Librarian Attributes –
  name, contact number.

- Book Attributes –
  ID, name, price, status.

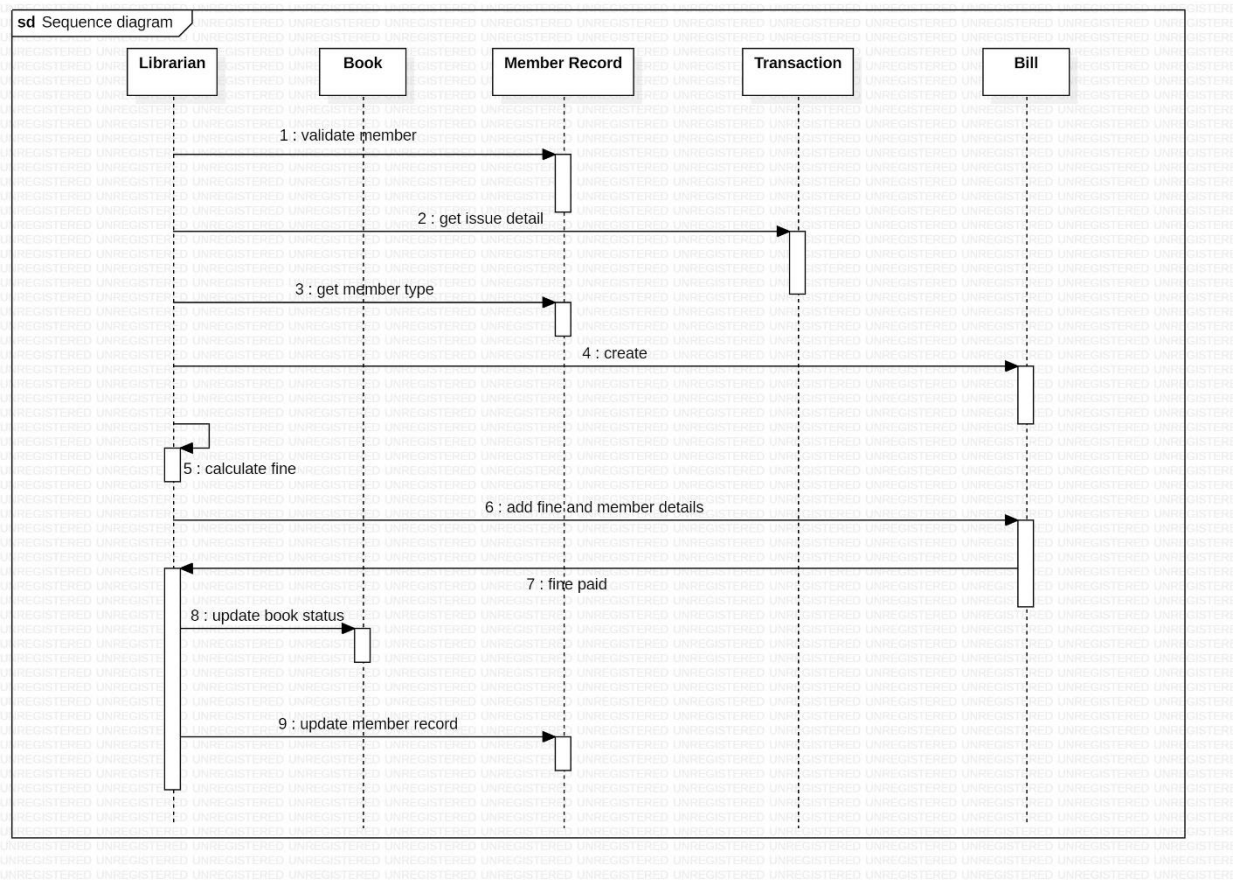
- Member_data Attributes –
  name, id, contact_info.

Methods of Library Management System :

- Librarian Methods –
  search_book(), issue_book(), collect_fine(), verify_member()

- Book Methods –
  Display_book_details(), update_status().

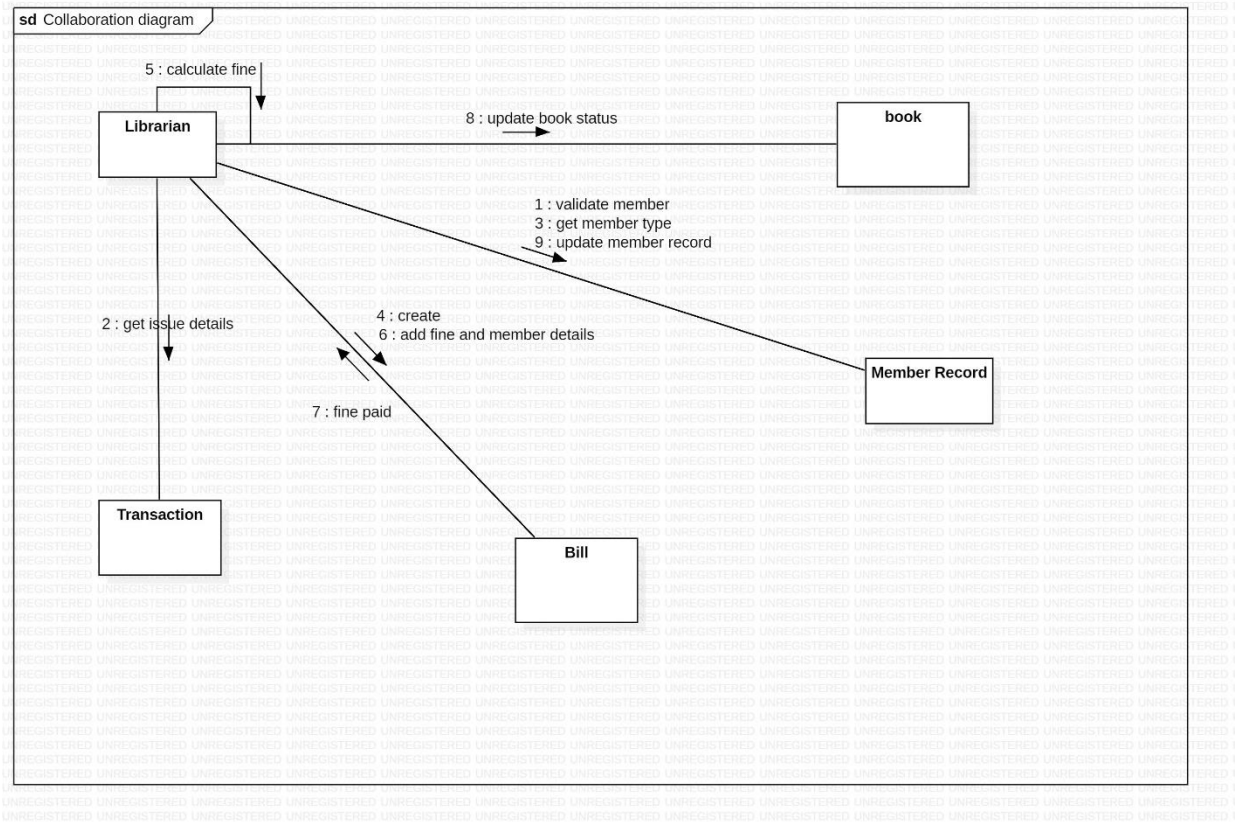
Sequence diagram with explanation



Explanation:

A sequence diagram shows an interaction arranged in time sequence; it shows object participating in interaction by their lifeline by the message they exchange arranged in time sequence. Vertical dimension represents time and horizontal dimension represent object.

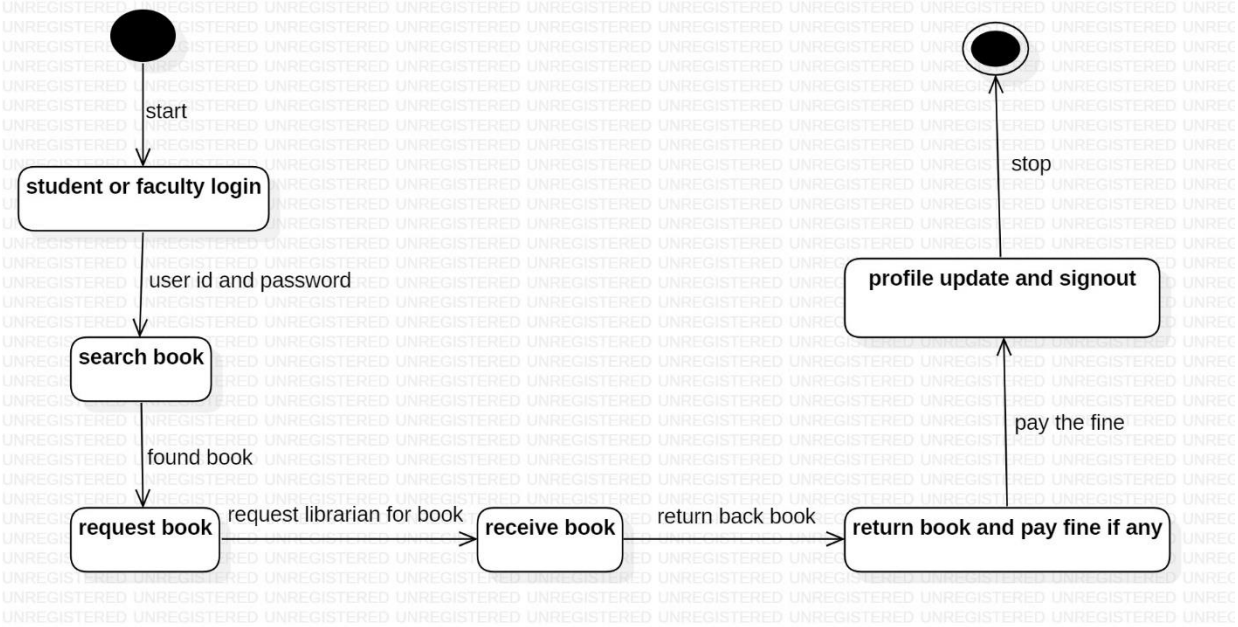
Communication diagram with explanation



Explanation:

A collaboration diagram is similar to sequence diagram but the message in number format. In a collaboration diagram sequence diagram is indicated by the numbering the message. A collaboration diagram, also called a communication diagram or interaction diagram, A sophisticated modeling tool can easily convert a collaboration diagram into a sequence diagram and the vice versa. A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time.

State chart diagram with explanation



Explanation:

The state chart diagram contains the states in the rectangle boxes and starts in indicated by the dot and finish is indicated by dot encircled. The purpose of state chart diagram is to understand the algorithm in the performing method.

ELEMENTS OF STATE DIAGRAM:

Initial State: This shows the starting point of the state chart diagram that is where the activity starts.

Transition – We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.

State – We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.

Fork – We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.

Join – We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.

Self transition – We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the

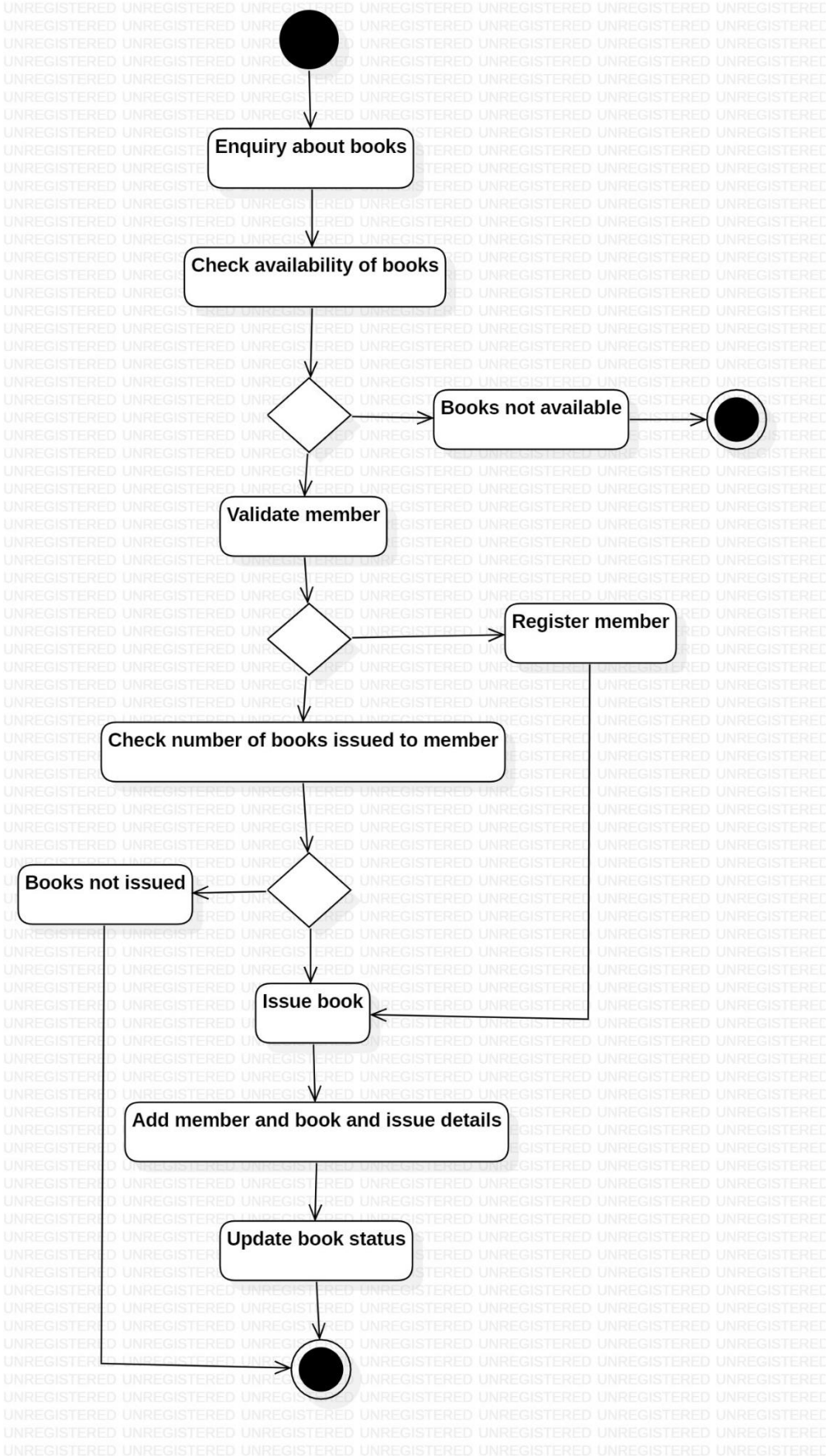
object does not change upon the occurrence of an event. We use self transitions to represent such cases.

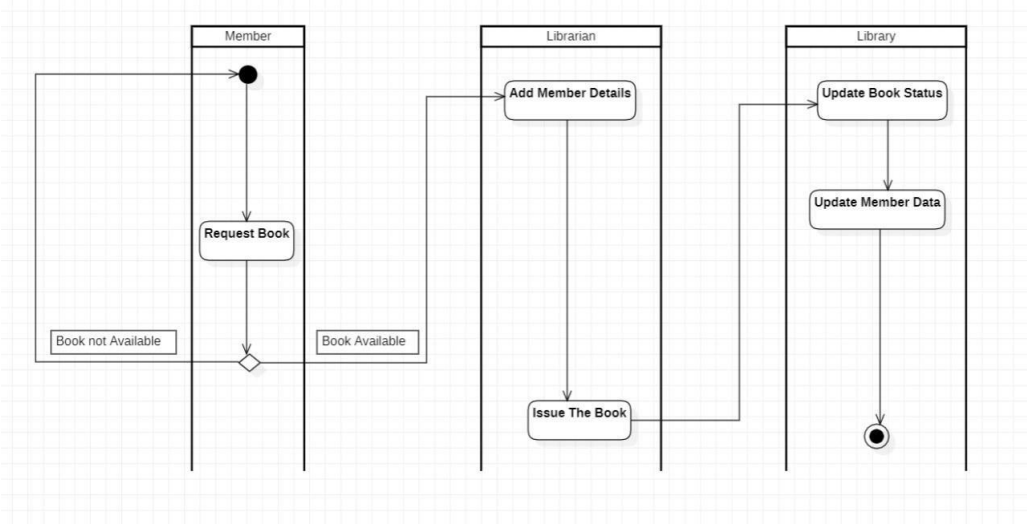
Composite state – We use a rounded rectangle to represent a composite state also.We represent a state with internal activities using a composite state.

Final state – We use a filled circle within a circle notation to represent the final state in a state machine diagram.

In the above state diagram of the library management system the student request the book from the the library and the librarian check if book is available or not and if book is not available than the student met the final state. If book is available with the librarian then the student issue the book and after that librarian update the the detailes of the member and the book in the system.In the last librarian update the member details and met with the final state.

Activity diagram with explanation





Explanation:
An activity diagram is a variation or special case of a state machine in which the states or activity representing the performance of operation and transitions are triggered by the completion of operation. The purpose is to provide view of close and what is going on inside a use case or among several classes. An activity is shown as rounded box containing the name of operation.
ELEMENTS OF STATE DIAGRAM:

Activity;

Is used to illustrate a set of actions.

It shows the non-interruptible action of objects.

Action Flow

It is also called edges and paths

It shows switching from one action state to another. It is represented as an

arrowed line.

Object Flow

Object flow denotes the making and modification of objects by activities.

An object flow arrow from an action to an object means that the action

creates or influences the object.

Decisions and Branching

A diamond represents a decision with alternate paths.

When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities.

Guards

In UML, guards are a statement written next to a decision diamond that must be true before moving next to the next activity.

Synchronization

A fork node is used to split a single incoming flow into multiple concurrent flows. It is represented as a straight, slightly thicker line in an activity diagram.

Time Event

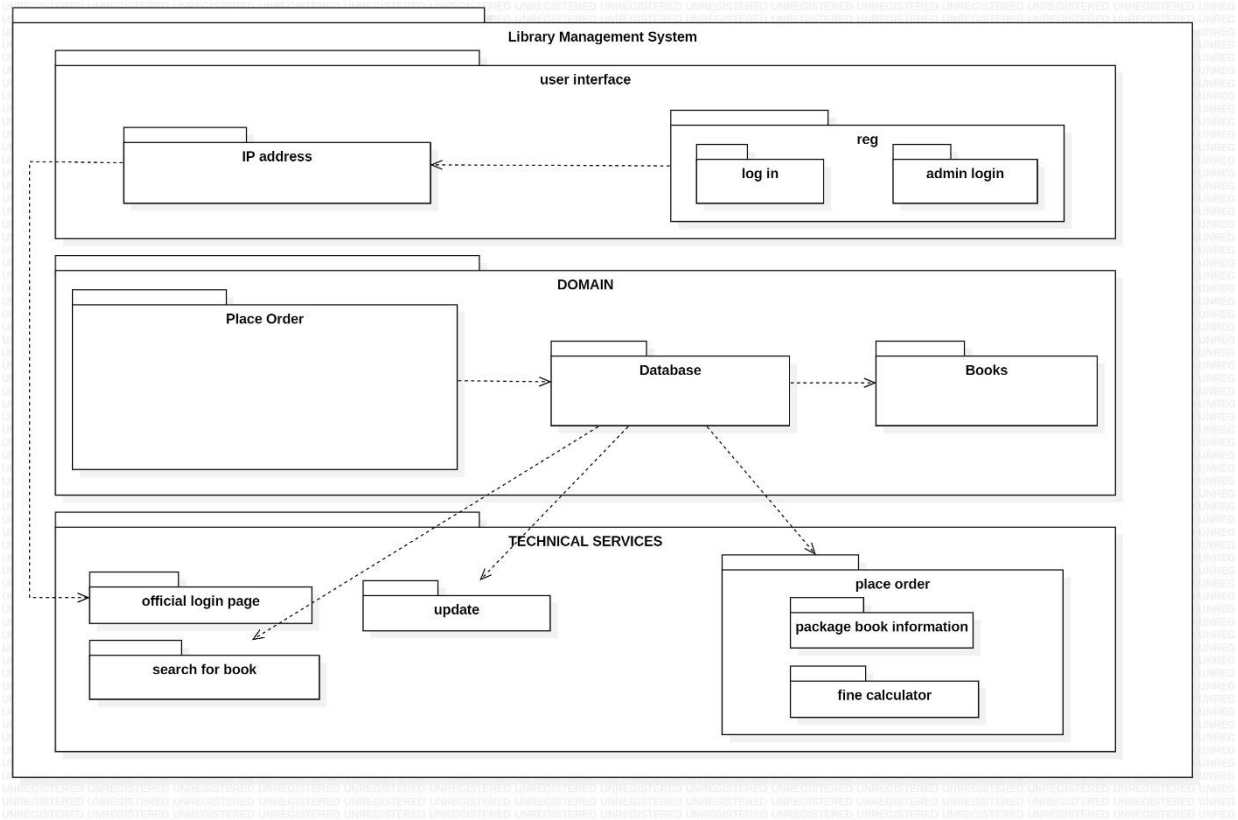
This refers to an event that stops the flow for a time; an hourglass depicts it.

Swimlane and Partition

A way to group activities performed by the same actor on an activity diagram or to group activities in a single thread

In the above activity diagram of the library management system the student request the book from the the library. If book is available we then add member details. After this we proceed to issue the book. Then we depict updating the book status. Finally we update member daitls. Then we enter the final state.

Package diagram with explanation

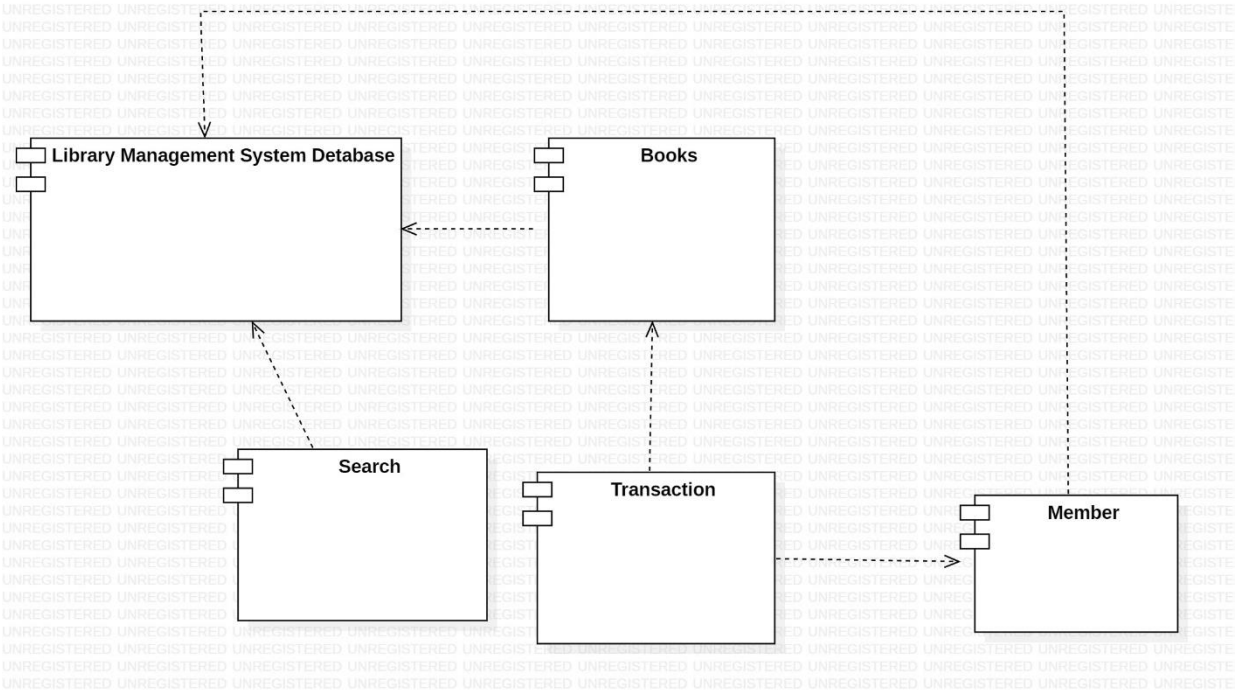


Explanation:

A package diagram is represented as a folder shown as a large rectangle with a top attached to its upper left corner. A package may contain both sub ordinate package and ordinary model elements. All UML models and diagrams are organized into package. A package diagram in unified modeling language that depicts the dependencies between the packages that make up a model. A Package Diagram (PD) shows a grouping of elements in the OO model, and is a Cradle extension to UML. PDs can be used to show groups of classes in Class Diagrams (CDs), groups of components or processes in Component Diagrams (CPDs), or groups of processors in Deployment Diagrams (DPDs).

There are three types of layers. They are: User interface layer, Domain layer and Technical services layer

Component diagram with explanation



Explanation:

The component diagram is represented by figure dependency and it is a graph of design of figure dependency.

The component diagrams have remarkable importance. It is used to depict the functionality and behavior of all the components present in the system, unlike other diagrams that are used to represent the architecture of the system, working of a system, or simply the system itself.

In UML, the component diagram portrays the behavior and organization of components at any instant of time. The system cannot be visualized by any individual component, but it can be by the collection of components.

Following are some reasons for the requirement of the component diagram:

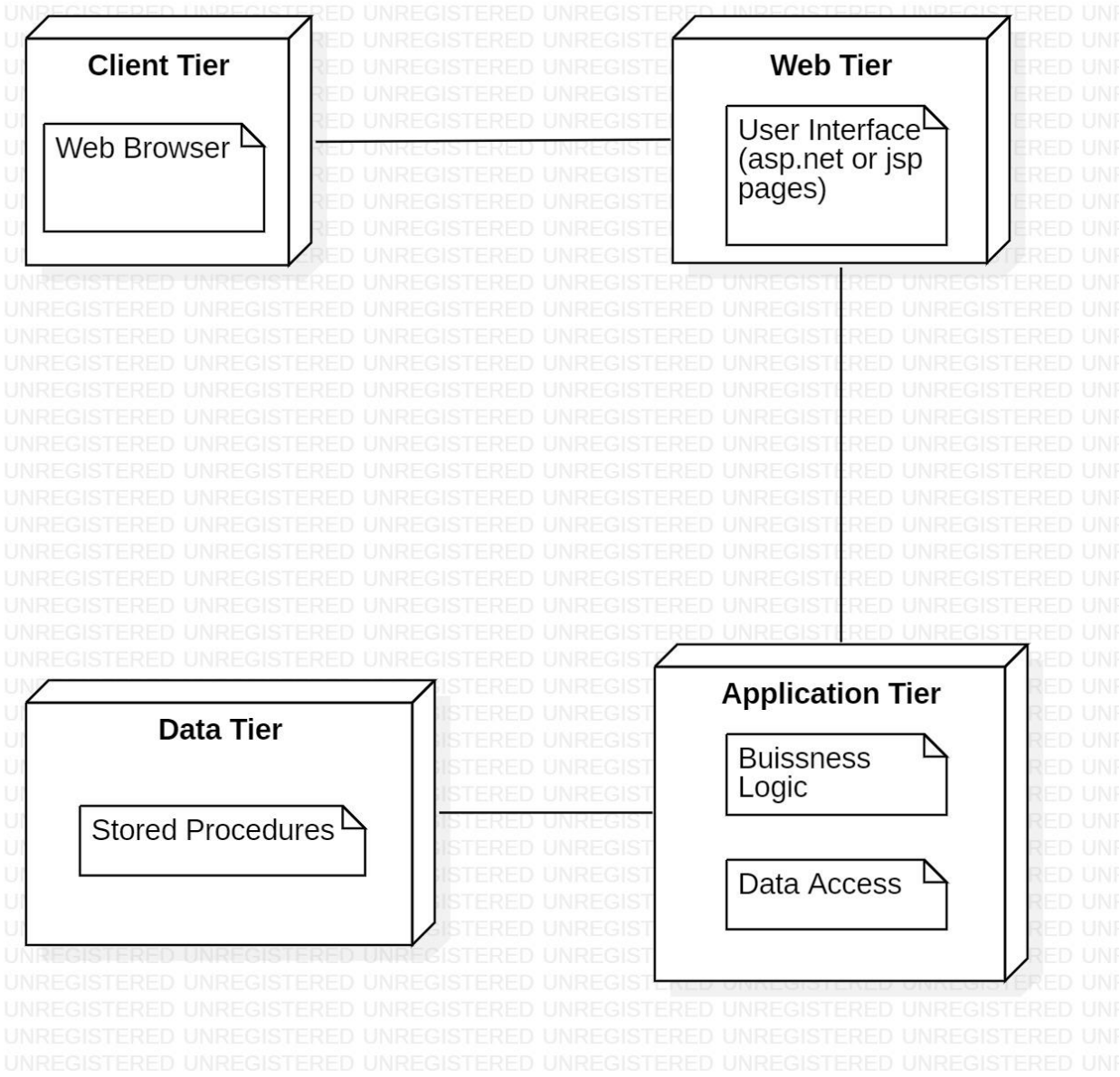
1. It portrays the components of a system at the runtime.
2. It is helpful in testing a system.
3. It envisions the links between several connections.

The main purpose of the component diagram are enlisted below:

1. It envisions each component of a system.
2. It constructs the executable by incorporating forward and reverse engineering.
3. It depicts the relationships and organization of components.

In the above component diagram for the library management system we have made the components for the database server ,books, for search operation, and for the transaction process and for the update on the member.In the library database the member asks for the book and the availiability for the book will be check in the library database and if the book is available then the transaction process will be done and update the member details and book details in the library database.

Deployment diagram with explanation



Explanation:

It is a graph of nodes connected by communication association. It is represented by a three-dimensional box. A deployment diagram in the unified modeling language serves to model the physical deployment of artifacts on deployment targets. Deployment diagrams show "the allocation of artifacts to nodes according to the Deployments defined between them. It is represented by 3-dimentional box. Dependencies are represented by communication association. The basic element of a deployment diagram is a node of two types

DEVICE NODE–

A physical computing resource with processing and memory service to execute software, such as a typical computer or a mobile phone.

EXECUTION ENVIRONMENT NODE

This is a software computing resource that runs within an outer node and which itself provides a service to host an execute other executable software element.

The device node is passport automation system and execution environment node are applicant passport administrator, regional administrator, and police.

Output

```
/**
 * Project Untitled
 */


#include "BOOK.h"

/**
 * BOOK implementation
 */


void BOOK::Display_Book_Details() {

}

void BOOK::Update_Status() {

}
/**
 * Project Untitled
 */


#ifndef _BOOK_H
#define _BOOK_H

class BOOK {
public:
    Integer bookID;
    String Name;
    Float Price;
    Boolean Status;

void Display_Book_Details();

void Update_Status();
};

#endif //_BOOK_H
/**
 * Project Untitled
 */


#include "Class1.h"
```

```
/**
 * Class1 implementation
 */

/**
 * Project Untitled
 */


#ifndef _CLASS1_H
#define _CLASS1_H

class Class1 {
};

#endif //_CLASS1_H
/**
 * Project Untitled
 */


#include "JOURNAL.h"

/**
 * JOURNAL implementation
 */

/**
 * Project Untitled
 */


#ifndef _JOURNAL_H
#define _JOURNAL_H

#include "BOOK.h"
#include "BOOK.h"


class JOURNAL: public BOOK, public BOOK {
};

#endif //_JOURNAL_H
/**
 * Project Untitled
 */


#include "LIBRARIAN.h"
```

```
/**
 * LIBRARIAN implementation
 */


void LIBRARIAN::Search_Book() {

}

void LIBRARIAN::Issue_Book() {

}

void LIBRARIAN::Collect_Fine() {

}

void LIBRARIAN::Verify_Member() {

}
/**
 * Project Untitled
 */


#ifndef _LIBRARIAN_H
#define _LIBRARIAN_H

class LIBRARIAN {
public:
    String Name;

void Search_Book();

void Issue_Book();

void Collect_Fine();

void Verify_Member();
private:
    Long Contract_Number;
};

#endif //_LIBRARIAN_H
/**
 * Project Untitled
 */


#include "LIBRARY.h"
```

```
/**
 * LIBRARY implementation
 */

/**
 * Project Untitled
 */


#ifndef _LIBRARY_H
#define _LIBRARY_H

class LIBRARY {
public:
    String Library_Name;
    String Address;
    Long Contact_Info;
};

#endif //_LIBRARY_H
/**
 * Project Untitled
 */


#include "MEMBER_DATA.h"

/**
 * MEMBER_DATA implementation
 */

/**
 * Project Untitled
 */


#ifndef _MEMBER_DATA_H
#define _MEMBER_DATA_H

class MEMBER_DATA {
public:
    String Name;
    Integer ID;
    Long Contact_Number;
};

#endif //_MEMBER_DATA_H
/**
 * Project Untitled
 */
```

```
#include "Staff_Member.h"

/**
 * Staff_Member implementation
 */

/**
 * Project Untitled
 */


#ifndef _STAFF_MEMBER_H
#define _STAFF_MEMBER_H

#include "MEMBER_DATA.h"
#include "MEMBER_DATA.h"
#include "MEMBER_DATA.h"


class Staff_Member: public MEMBER_DATA, public MEMBER_DATA, public MEMBER_DATA
{
};

#endif //_STAFF_MEMBER_H
/**
 * Project Untitled
 */


#include "STUDENT.h"

/**
 * STUDENT implementation
 */


void STUDENT::Issue_Book() {

}

void STUDENT::Return_Book() {

}
/**
 * Project Untitled
 */


#ifndef _STUDENT_H
#define _STUDENT_H
```

```
#include "MEMBER_DATA.h"


class STUDENT: public MEMBER_DATA {
public:

void Issue_Book();

void Return_Book();
};

#endif //_STUDENT_H
/**
 * Project Untitled
 */


#include "STUDY_BOOKS.h"

/**
 * STUDY_BOOKS implementation
 */

/**
 * Project Untitled
 */


#ifndef _STUDY_BOOKS_H
#define _STUDY_BOOKS_H

#include "BOOK.h"
#include "BOOK.h"


class STUDY_BOOKS: public BOOK, public BOOK {
};

#endif //_STUDY_BOOKS_H
```

Conclusion

In conclusion, from a proper analysis and assessment of the designed system, it can be safely concluded that the system is an efficient, usable and reliable records management system. It is working properly and adequately management meets the minimum expectations that were set for it initially. The new system is expected to give benefits increased overall productivity, performance and efficient records management.

References

https://www.freeprojectz.com/uml-diagrams

https://www.academia.edu/38136485/Umldiagramforlibrarymanagementsystem_1404250709

https://www.softwareideas.net/uml-diagram-library-management-system