Indira Gandhi Institute of Technology
Computer Science &Engineering Department

Lab Manual

on ***Object Oriented Software Engineering***
(To Be Revised …)

INDRA THANAYA B
Asst . Professor
Dept of CSE.

Experiment Details: Use Case Diagram. **Experiment No.: 1**

Subject: Object Oriented Software Engineering

Experiment Name: Use Case.

Resource Used:

1) **Equipment:**
   UML tool.

2) **Consumables:**
   Printer.

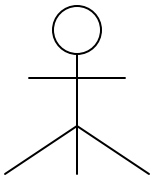
Theory:

Actors

Actors represent anyone or anything that interact with the system. An actor may

- Only input information to a system
- Only retrieve information from a system
- Both input and retrieve information to and from a system
  Typically, the actors are found in the problem statement, and also from conversation with the customers and domain experts.
  There are three types of actors:
  1. users of the system,
  2. external application systems, and
  3. external devices that can independently interact with the system.

In UML, an actor is represented stickman symbol, as shown below:

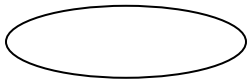


Use cases:

Use cases eventually map to the menu option. Use cases represent the functionality provided by the system. Each individual functionality provided by a system is captured as a use case.

A use case thus represents a dialog between an actor and the system. A collection of use cases for a system reflects all the defined ways in which a system can be used. A use case can be defined as a sequence of transactions performed by a system, that yields a measurable result of values for a particular actor.

In UML, a use case is represented as an oval, as shown below:

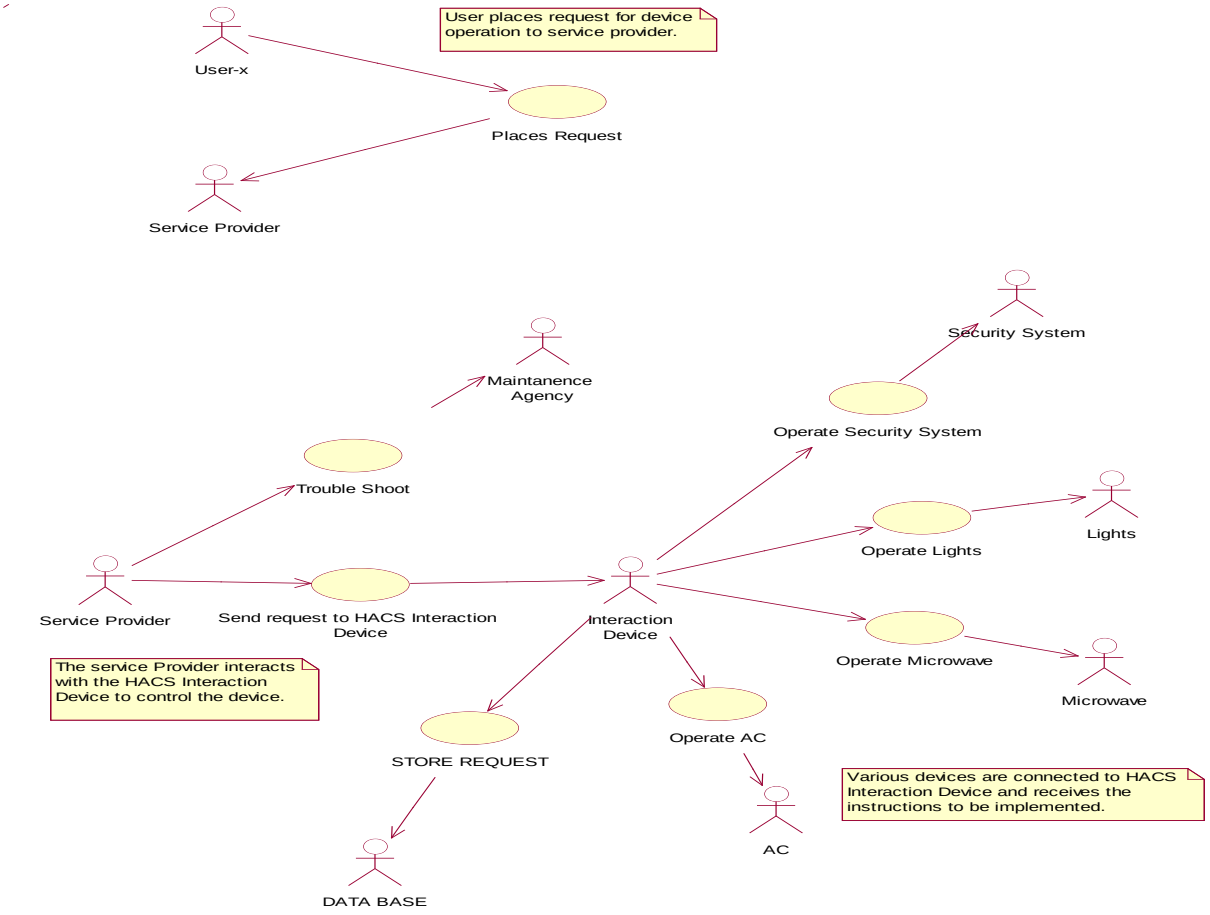


Use case Diagram:

A use case diagram is a interaction view of a some or all the actors, use cases and their interactions identified for a system. Each system typically has a main use case diagram, which is a picture of the system boundary (actors) and the major functionality provided by the system (use cases). Other use case diagrams may be created as needed. Some examples are:

- A diagram showing all the use cases for a selected actor.
- A diagram showing all the use cases being implemented in an interaction.
- A diagram showing all the use cases and all its relationship.

Example:



Actors:

1. User
2. Service Provider
3. Maintenance Agency
4. Interaction Device
5. Database
6. Lights

Use cases:

1. Places Request
2. Trouble Shoot
3. Send request to interaction devices
4. Store Request
5. Operate Light

Description of Use case Diagram:

- **User:** User places request for device operation to service provider.

- **Service Provider:** The service provider interacts with the HACS interaction device to control the device.
- **Interaction Device**: Various devices are connected to HACS interaction Device
  and receive the instructions to be implemented.
- **Places Request**: This use case enables the user to place request for a specific operation to be implemented on a particular appliance. The user sends its ID followed by its Personal Identification Number (PIN) along with the device ID & the operation to be implemented.
- **Send request to HACS Interaction Device**: This refers to the interaction between the Service Provider & the HACS Interaction Device, where the service provider sends the command required to implement the requested operation.
- **Operate Lights**: This use case sets the state of the specified lights as requested. For example if the state of light-1 is OFF & its requested state is ON , it will be switched ON. If the requested state is OFF the state will remain unchanged.

Appendix:

1. **Stereotype:** Stereotypes defines a new model element in terms of another model element. It is represented by <<stereotypes>>
2. **System boundary boxes:** We can draw a rectangle around the use cases, called the System boundary box, to indicate the scope of your system.
3. **Abstract use case:** Use case, which is inherited, by some use case is called as abstract use case.
4. **Concrete use case:** Use case, which is directly inherited by actor, is called as concrete use case.

Conclusion:

Use cases define the behavior provided by the system. They are a central theme for the entire development process. They play a role in a various stages of the development process as follows:

- During requirements determination, they specify what the system should do from the user's point of view.
- During analysis and design, the use cases are realized in a design model .Use case realizations describe the interaction between various objects in the design model.
- During implementation, the design model becomes the implementation specification.
- During testing, the use cases are the basis for identifying test cases. The system is verified by performing each use case.
- As part of project management, they provide a basis for planning the iterations.

Experiment Details: Class Diagram. **Experiment No.:2**

Subject: OOSE

Experiment Name: Class Diagram.

Recourses Used:

3) **Equipment:**
   UML tool.

4) **Consumables:**
   Printer.

Theory:

Class diagrams are perhaps the most commonly used diagrams in modeling object-oriented systems.

A class diagram shows a set of classes and interfaces, and their relationship.

In an object-oriented system, no class can stand in a complete isolation from all the other classes. Classes share various types of relationship with other classes. In fact, a system is a collection of various collaborating classes. It is through collaboration between various classes that a system can achieve its final goal.

Contents of a class Diagram

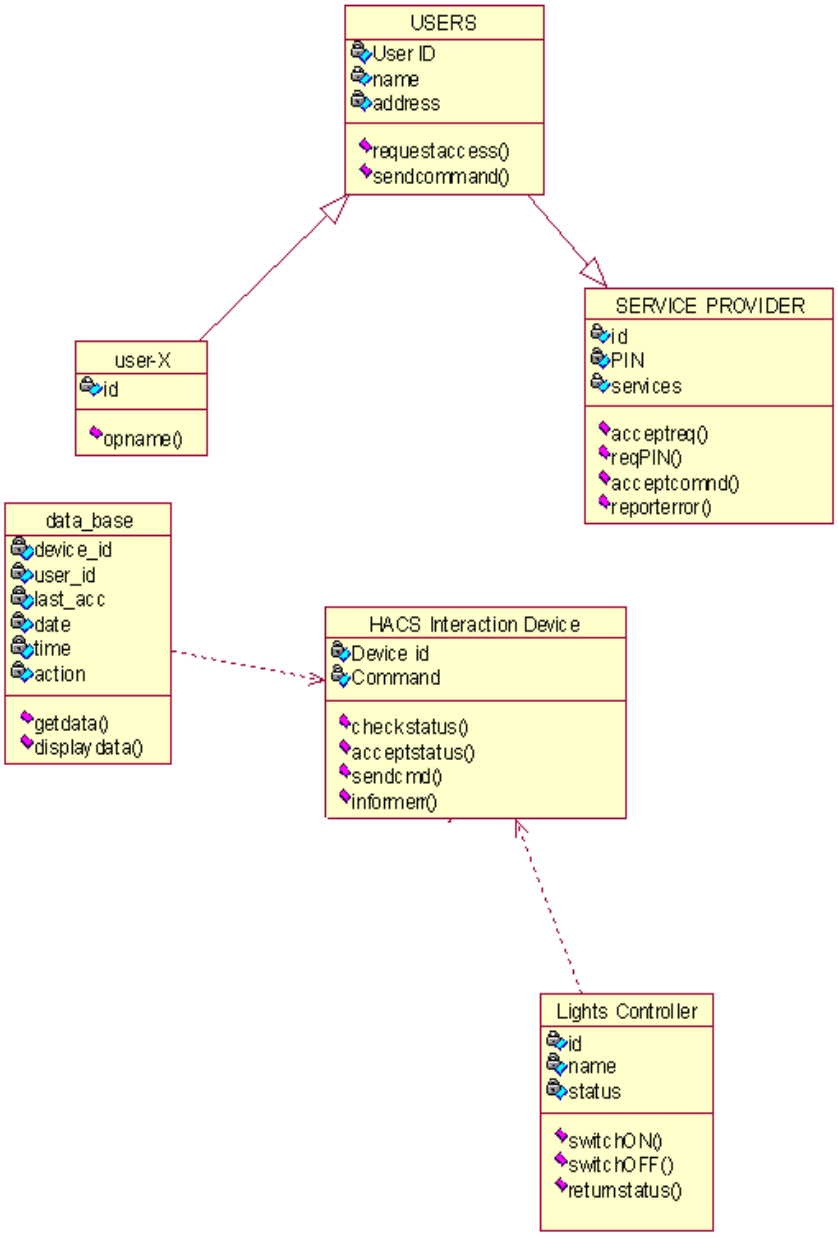
A class diagram consists of some logically related classes and interfaces from some aspects of a system, along with the relationship as well as collaborations between these classes. The number of classes and interfaces in a non-trivial system is likely to be such that a single diagram showing all classes and interfaces, and all their relationship, may not be practical. We, therefore, make a number of classes diagrams, each diagram representing some specific aspects of a structural relationships and collaborations between classes. A class may appear in multiple class diagrams.

Representing classes in a Class Diagram

The complete UML notation for a class diagram is a rectangle with three compartments. The first compartment has an optional stereotype and the class name. The second compartment can be used to show the attributes of a class, while the third compartment is used for listing the responsibilities of operations of the class. Only the first compartment is mandatory, the next compartments are optional, and often omitted.

Including all attributes and operations for a class diagram may not only be unnecessary in most contexts, but it may also clutter up the diagram and make them difficult to use. UML also allows you to render some selected attributes and operations in the respective compartments.

Example: Class Diagram for Light



Description of the Classes:

1. USERS: This class holds the information of various users. This information refers to their UserID , name & address associated to each other. Functions defined for this class includes request access which places an request with the service provider for the access to HACS. The other function is send command which is used to pass on the requested command along with the user & PIN to the service provider.

2. user-X: this is a sub class of USERS & inherits the properties of the class USERS. This class holds the information of a particular user, named 'X'. The

functions associated with this class are: opname(). It is used to send the details of operation to be performed.

3. SERVICE PROVIDER: This class holds the details of users associated with their unique PIN along with the services they can access. As this class holds all the information of the users, we can say that this is the super class of USERS. The functions associated with this class are: acceptreq(), reqPIN, acceptcmnd, reporterror().
acceptreq() is used to process the request it receives for the access of a particular HACS.
reqPIN() sends a request to the user for the PIN.
acceptcmnd processes the request for an operation sent by the user & forwards it to the HACS Interaction Device.
reporterror() sends a message to the user that the requested task cannot be performed.

4. HACS Interaction Device: This class holds the IDs for various appliances & the command that can be processed for a particular device. The various functions performed by this class are:

checkstatus() sends a request for the status of a device.
acceptstatus() accepts the status sent by an appliance.
sendcmd() forwards a request for a command to be implemented.
informerr() sends a message to the service provider reporting an error if the status check tells it so.

5.Lights Controller: This class holds the information of various lights at the house. The functions it performs are switchON, switchOFF & returnstatus().

Analysis / Conclusion:

Class diagram fully describe the attributes and methods that exist for a class.

For each class that has significant temporal behavior, you can create an activity diagram or a state chart diagram to describe this behavior.

Subject: OOSE

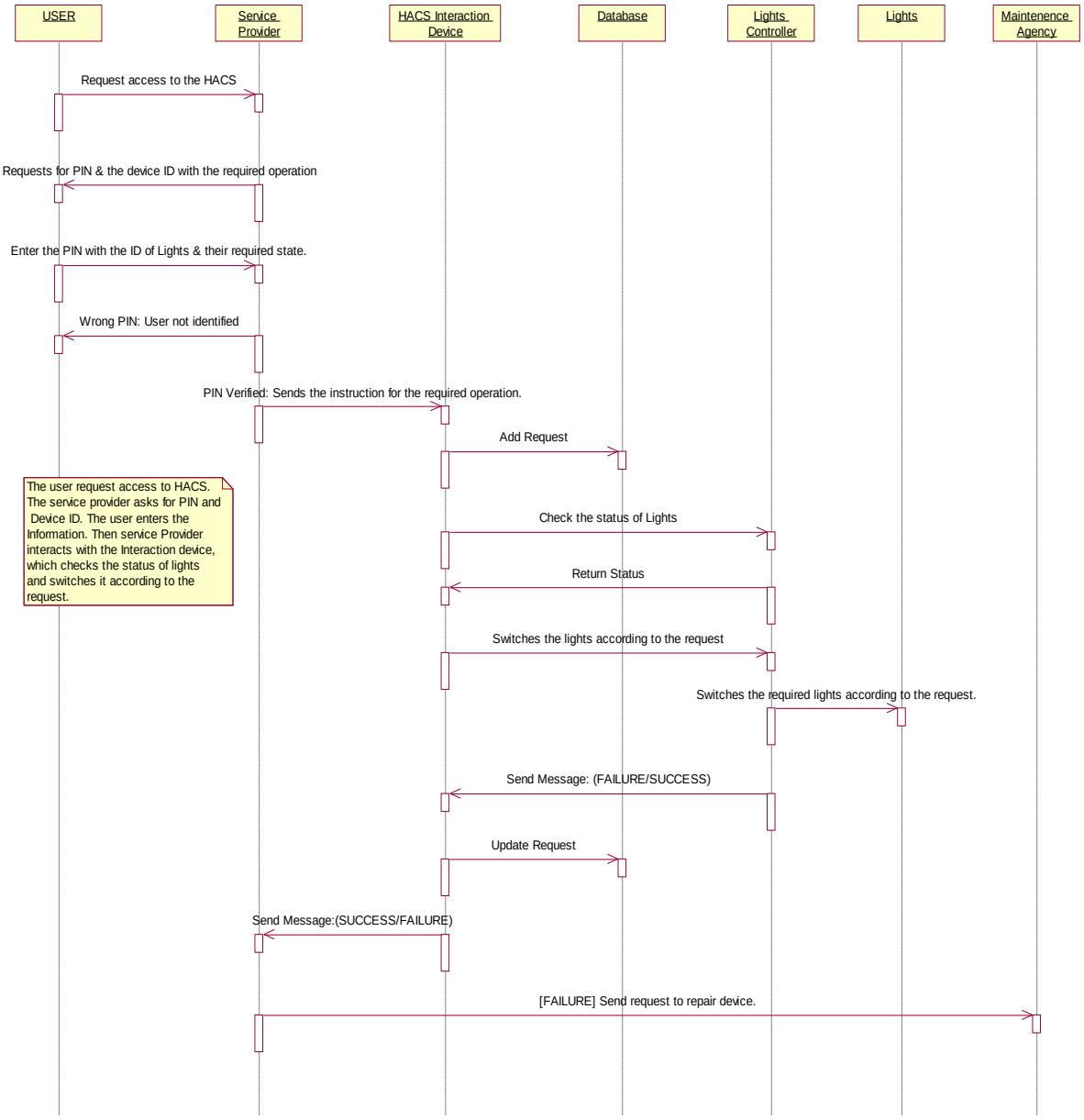
Experiment Name: Sequence Diagram.

Recourses Used:

1) Equipment:
UML tool.
2) Consumables:
Printer.

Theory:
A sequence diagram shows a pattern of interaction among objects, emphasizing the sequencing of messages. A message is the mechanism by which an object to execute a certain responsibility.
Normally, we begin drawing a sequence diagram by placing the initiating actor at the left, and subsequent classes or actors increasingly to the right. Linking the participating objects or actors by line with an arrow shows a message. The message line originates from the object or the actor who initiates the message and arrow points to the object or the actor who is responsible for that message to be executed.
The message line should be labeled with a suitable description of the message being passed. We can also specify the message.
Interaction between actors should not be shown in a sequence diagram. Since actors are outside the system, any interaction between them is also outside the system, and therefore should not be modeled
Sequence diagram demonstrate the behavior of objects in a use case by describing the objects and the message they loss. The diagrams are read left to right and descending.

Example: The user request access to HACS. The service provider asks for PIN and device ID. The user enters the information. Then service providers interact with the interaction device, which checks the status of lights, and switches it according to the request.



Appendix:

- **Asynchronous message:**

  The handler return immediately, but the actually work is done in the background. The sender can move on to other tasks while processing goes on.

- **Synchronous message:**

  The sender waits until the handler completes (blocks). This is a normal method call in a single threaded application.

- **Balking:**
  The receiving object can refuse to accept the message request could happen if an active objects message queue files.

- **Time outs:**
  The message handler typically blocks, but will return after a predetermined amount of time, even if the work of handler is not complete.

Conclusion:

1. Sequence diagram generally show the sequence of events that occur.
2. Sequence diagram are used to model flows of control by time ordering.
3. Sequence diagram can be used to demonstrate the interaction of objects in a use case.
4. Sequence diagram are used when you want to model the behavior of several objects in a use case.
5. Sequence diagram show how object interact with one another.

Experiment Details: Collaboration diagram **Experiment No.:**4

Subject: OOSE

Experiment Name: Collaboration diagram
Recourse Used:

5) **Equipment:**
   UML tool.

6) **Consumables:**
   Printer.

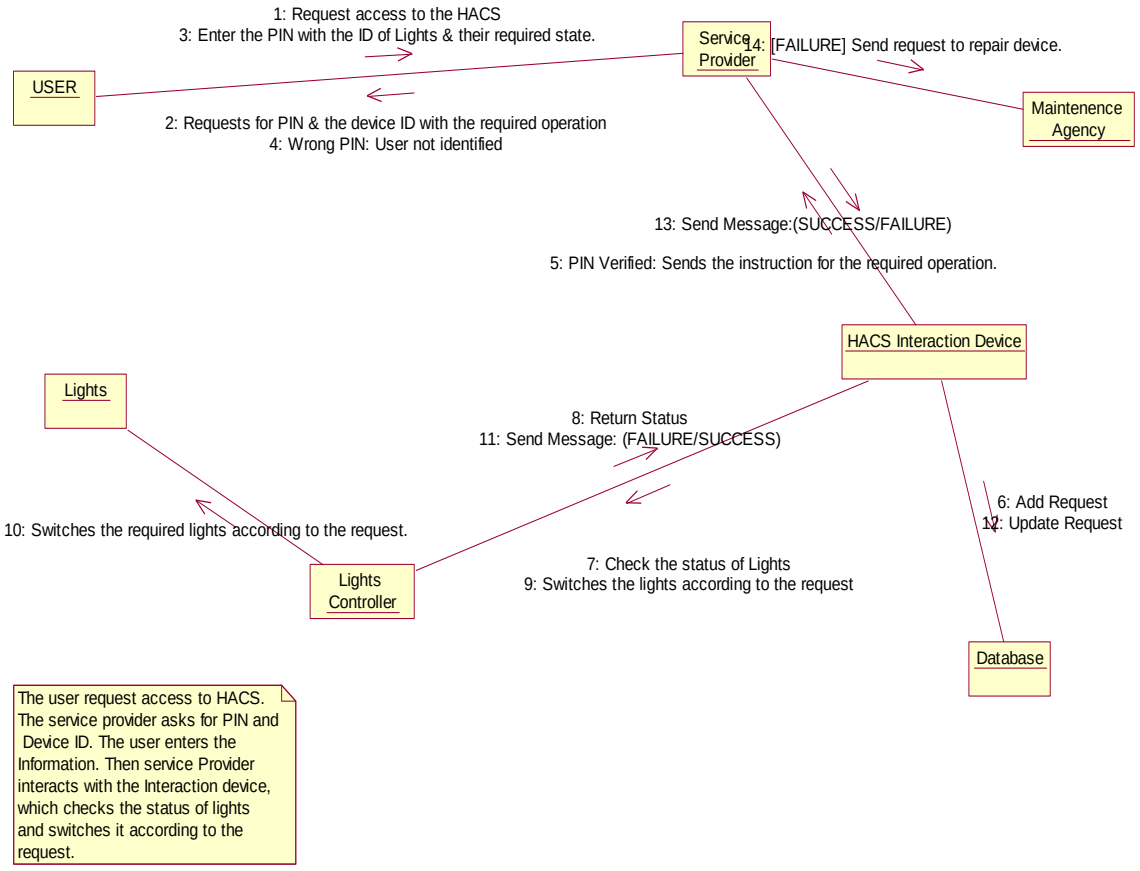
Theory:
A collaboration diagram is an alternative way of displaying the pattern of interaction between various objects and actors in a use case. While a sequence diagram emphasizes the sequencing of communication between objects, a collaboration diagram emphasizes the organization of the objects participating in iteration.
Collaboration diagram can be used to show how objects in a system interact over multiple use cases. Collaboration diagram are useful during the exploratory phases of a development process. Since there is no explicit representation of time in Collaboration Diagrams, the message are numbered to denote the sending order.
In a collaboration diagram, all interactions between any pair of objects/actors are shown at the same level.

Collaboration diagram are also relatively easy to draw. They show the relationship between objects and the order of message passed between them. The objects are listed as icons and arrow indicates the message being passed between them. The number next to the message is called sequence numbers. As the name suggests, they show the sequence of the message as they are passed between the objects. There are many acceptable sequence-numbering schemes in UML. A simple 1,2,3…format can be used or for more detailed and complex diagrams a 1,1.1,1.2,1.2.1… schemes can be used.

Example:

The user request access to HACS . The service provider asks for PIN and device ID. The user enters the information. Then service provider interact with the interaction device, which checks the status of lights, and switches it according to the request.

Collaboration Diagram for Lights



Appendix:

- **Path:** For indications of how are object is linked to another.
- **Sequence number:** For indication of message, use prefixes the message with a number, increasing monotonically for each new message in the flow of control.
- **Iteration:** It is denoted as *. Iteration shows that the given message will be repeated in accordance with the expression.
- **Semantic Equivalence:** Sequence diagrams and collaboration diagrams are semantically equivalent as they both derive from the same information in the UML's meta-model. Therefore, we can take one form of a diagram and convert it to the other without any loss of information.

Conclusion:

Collaboration diagram demonstrate how objects are statically connected.

Experiment Details: State Chart Diagram **Experiment No.:**5

Subject: OOSE

Experiment Name: State Chart Diagram.

Recourses Used:

1) Equipment:
UML tool.
2) Consumables:
Printer.

Theory:

State chart diagrams model the dynamic behavior of individual classes or any other kind of object. They show the sequences of states that an object goes through, the events that cause a transition from one state to another, and the actions that result from a state change.

State chart diagrams are closely related to activity diagrams. The main difference between the two diagrams is state chart diagrams are state centric, while activity diagrams are activity centric. A state chart diagram is typically used to model the discrete stages of an object's lifetime, whereas an activity diagram is better suited to model the sequence of activities in a process.

Each state represents a named condition during the life of an object during which it satisfies some condition or waits for some event. A state chart diagram typically contains one start state and multiple end states. Transitions connect the various states on the diagram. As with activity diagrams, decisions, synchronizations, and activities may also appear on state chart diagrams.

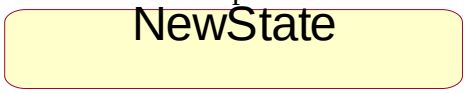
State

Definition

A state represents a condition or situation during the life of an object during which it satisfies some condition or waits for some event. Each state represents the cumulative history of its behavior.

Graphical Depiction

The state icon appears as a rectangle with rounded corners and a name (Wait). It also contains a compartment for actions:



Naming

The name of a state should be unique to its enclosing class, or if nested, within the state. All state icons with the same name in a given diagram represent the same state.

Actions

Actions on states can occur at one of four times:

· on entry

- on exit
- do
- on event.

An on event action is similar to a state transition label with the following syntax:

event(args)[condition] : the Action

You must add actions through the Action Specification. States may also appear on activity diagrams.

Start state

A start state (also called an "initial state") explicitly shows the beginning of a workflow on an activity diagram or the beginning of the execution of a state machine on a state chart diagram. You can have only one start state for each state machine because each workflow/execution of a state machine begins in the same place. If you use multiple activity and/or state chart diagrams to model a state machine, the same start state can be placed on the multiple diagrams. When you model nested states or nested activities, one new start state can be created in each context.

Normally, only one outgoing transition can be placed from the start state. However, multiple transitions may be placed on a start state if at least one of them is labeled with a condition. No incoming transitions are allowed.

You can label start states, if desired. State Specifications are associated with each start state.

Graphical Depiction

The start state icon is a small, filled circle that may contain a name (Begin Process):

End State

An end state represents a final or terminal state on an activity diagram or state chart diagram. Place an end state when you want to explicitly show the end of a workflow on an activity diagram or the end of a state chart diagram. Transitions can only occur into an end state; however, there can be any number of end states per context.

You can label end states, if desired. State Specifications are associated with each end state.

Graphical Depiction

The end state icon is a filled circle inside a slightly larger unfilled circle that may contain a name (End Process):

State Transition

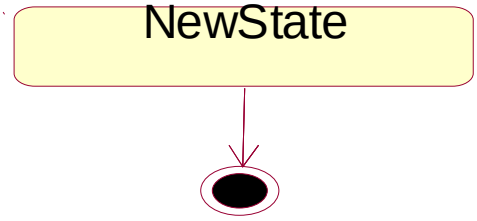
Definition

A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities, or between an activity and a state.

You can show one or more state transitions from a state as long as each transition is unique. Transitions originating from a state cannot have the same event, unless there are conditions on the event.

Graphical Depiction

The icon for a state transition is a line with an arrowhead pointing toward the destination state or activity:



Naming

You should label each state transition with the name of at least one event that causes the state transition. You do not have to use unique labels for state transitions because the same event can cause a transition to many different states or activities.

Transitions are labeled with the following syntax:

event (arguments) [condition] / action ^ target.sendEvent (arguments)

Only one event is allowed per transition, and one action per event.
Events, conditions and actions must be added by editing the label or through the State Transition Specification.

Nested States

States may be nested to any depth level. Enclosing states are referred to as super states, and everything that lies within the bounds of the super state is referred to as its contents. Nested states are called sub states.

Conclusion: In this way we have studied state chart diagram

Experiment Details: Activity Diagram **Experiment No.:** 6

Subject: OOSE

Experiment Name: Activity Diagram

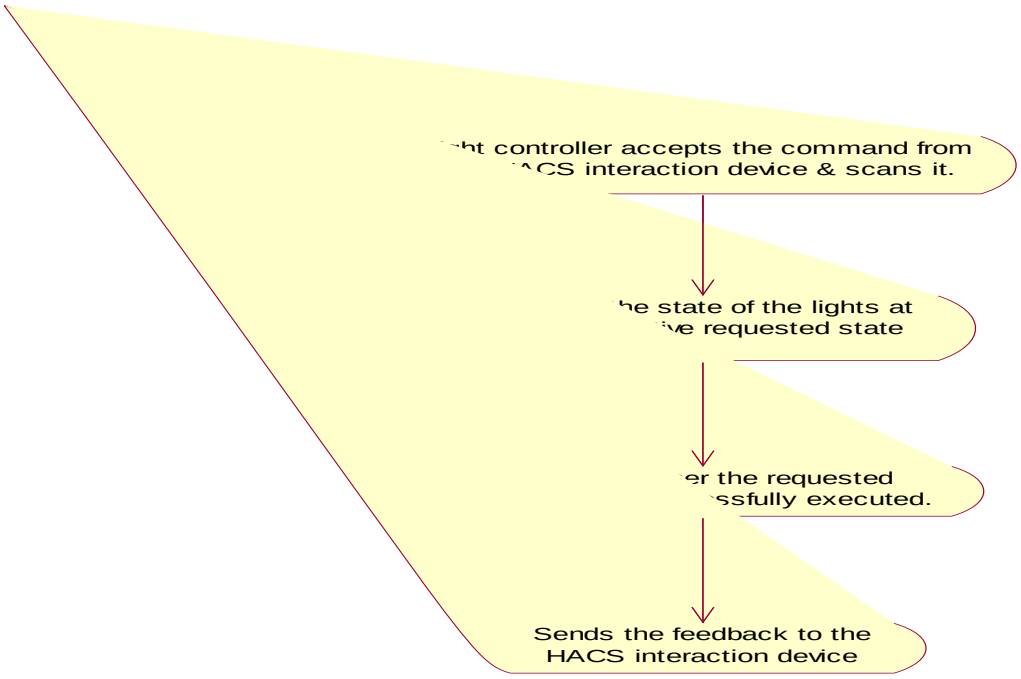
Recourses Used:

1) **Equipment:**
   UML / OOAD tool.

2) **Consumables:**
   Printer

Theory:

Activity diagrams provide a way to model the workflow of a business process. You can also use activity diagrams to model code-specific information such as a class operation. Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity. An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities. The main difference between activity diagrams and state charts is activity diagrams are activity centric, while state charts are state centric. An activity diagram is typically used for modeling the sequence of activities in a process, whereas a state chart is better suited to model the discrete stages of an object's lifetime.



Procedure:

Activity diagrams can model many different types of workflows. For example, a company could use activity diagrams to model the flow for an approval of orders or to model the paper trail of invoices. An accounting firm could use activity diagrams to model any number of financial transactions. A software company could use activity diagrams to model a software development process.

Each activity represents the performance of a group of actions in a workflow. Once the activity is complete, the flow of control moves to the next activity or state through a transition. If an outgoing transition is not clearly triggered by an event, then it is triggered by the completion of the contained actions inside the activity. A unique activity diagram feature is a swim lane that defines who or what is responsible for carrying out the activity or state. It is also possible to place objects on activity diagrams. The workflow stops when a transition reaches an end state.

You can attach activity diagrams to most model elements in the use case or logical views. Activity diagrams cannot reside within the component view

Analysis / Conclusion:

The activity diagram shows use case as it progress from start to finish.

The activity diagram shows use case as it progress from start to finish.Activity diagrams are used to show workflow in parallel and conditionally. They are useful when working out the order and concurrency of a sequential algorithm, when analyzing the steps in a business process and when working with threads.

Experiment Details: Component Diagram. **Experiment No.:7**

Subject: OOSE

Experiment Name: Component Diagram.

Recourses Used:

7) **Equipment:**
   UML tool.

8) **Consumables:**
   Printer.

Theory:

Component:
A component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. Graphically, a component is rendered as a rectangle with tabs.

Names:
Every component must have a name that distinguishes it from other components. A name is a textual string.

Components and classes:
In many ways, components are like classes. Both have names; both may realize a set of interfaces; both may participate in a dependency, generalization, and association relationship; both may be nested; both may have instances; both may be participants in interactions. However, there are some significant differences between component and classes.

Classes represent logical abstractions; components represent physical things that live in the world of bits.

- Components represent the physical packaging of otherwise logical components and are at a different level of abstraction.
- Classes may have attributes and operations directly. In general, components only have operations that are reachable only through their interfaces.

Kinds of components may be distinguished-

1. Deployment components: These are the components necessary and sufficient to form an executable system, such as dynamic libraries and executable.
2. Work Product Components: The components are essentially the residue of the development process, consisting of things such as source code files and data files from which deployment component are created.
3. Execution Components: These component are created has a consequences of an executing system, such as COM+ object, which is instantiated from a DLL.

Component Diagram:

A component diagram shows a set of components and their relationships. Graphically, a components diagram is a collection of vertices and arcs.

A components diagram is just a special kind of diagram and shares the same common properties as do all other diagrams-a name and graphical contents that are a projection into a model. What distinguishes a component diagram from all other kinds of diagrams is its particular content.

Component diagram commonly contain

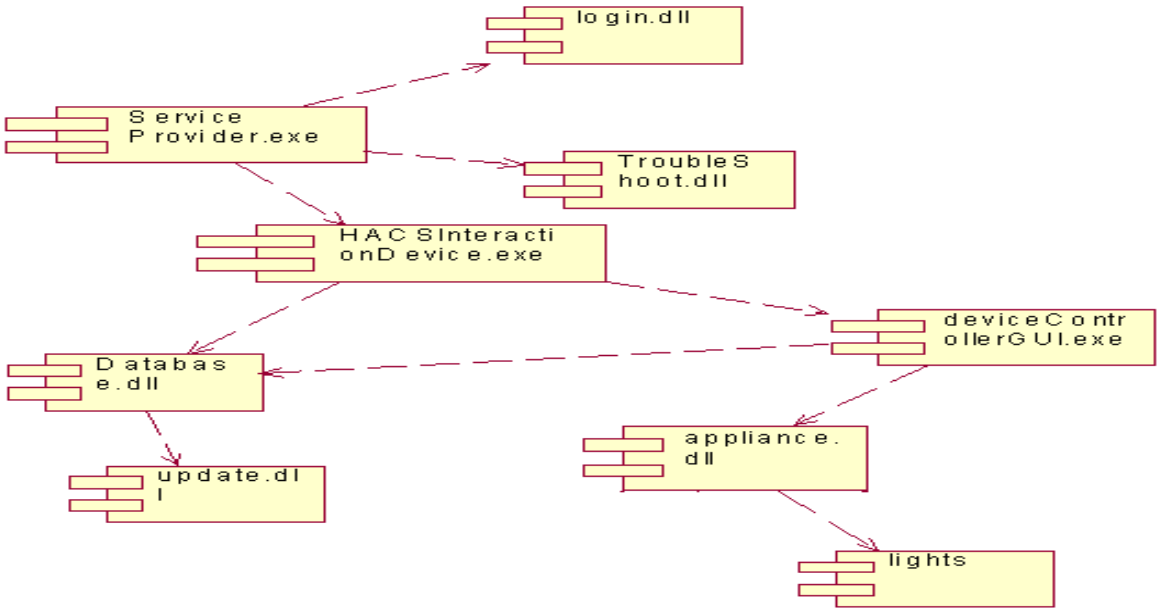
- Components
- Interfaces
- Dependency, generalization, association and realization relationships.

Component diagram may also contain packages or subsystem, both of which are used to group elements of your model into larger chunks. Sometimes you'll want to place instances in your component diagrams, as well, especially when you want to visualize one instances of a family of component- based system.

Example:

The component diagram shows the organization and dependencies among the set of components of the system. The Home Appliance Control System contains the following components:

login.dll
TroubleShoot.dll
Service Provider.exe
HACSInteractionDevice.exe
Database.dll
deviceControllerGUI.exe
update.dll
appliance.dll
Lights



Appendix:

- **Component:** A component is a physical building block of the system. It is represented as a rectangle with tabs.
- **Interfaces:** An interface describes a group of operations used or created by components.
- **Component qualification:** System requirements and architecture define the component that will be required. Reusable components are normally identified by the characteristics of their interfaces.
- **Component adaptation:** The existing reusable components must be adapted to meet the meets of the architecture or discarded and replaced by other, more suitable components.
- **Component Composition:** Architectural style plays a key role in the way in which software components are integrated to form a working system.
- **Component Update:** When systems are implemented, update is complicated by the imposition of the organization that developed the reusable component may be outside the immediate control of the software engineering corporation.

Conclusion:

1. Component diagram describe the organization of physical software components, including source code, run time (binary) code, and executables.
2. The different high level reusable parts of a system are represented in a component diagram.
3. The primary difference is that component diagram represents the implementation perspective of a system.
4. Component diagram is used to model the static implementation view of a system.

Experiment Details: Deployment Diagram **Experiment No.:**8

Subject: OOSE

Experiment Name: Deployment Diagram.

Recourses Used:

1) Equipment:

UML tool.

2) Consumables:

Printer.

Theory:

A deployment diagram shows processors, devices, and connections. Each model contains a single deployment diagram which shows the connections between its processors and devices, and the allocation of its processes to processors.
Processor Specifications, Device Specifications, and Connection Specifications enable you to display and modify the respective properties. The information in a specification is presented textually; some of this information can also be displayed inside the icons.
You can change properties or relationships by editing the specification or modifying the icon on the diagram. The deployment diagram specifications are automatically updated.

Processor

A processor is a hardware component capable of executing programs.

Naming

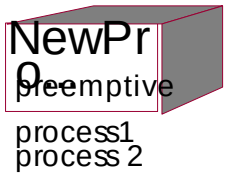
Each processor must have a name. There are no constraints on the processor name because processors denote hardware rather than software entities.

Graphical Depiction

The icon for a processor is a shaded box:

Adornments

You can further define a processor by identifying its processes and specifying the type of process scheduling it uses. You can set the following adornments in the Processor Specification. You can display the information in the deployment diagram by selecting an item from the processor shortcut menu.



Scheduling

You can specify the type of process scheduling used by this processor by setting a scheduling type:

Type	Description
Preemptive (default)	**Higher-priority processes that are ready to execute can preempt lower-priority processes that are currently executing.**
Nonpreemptive	**The current process continues to execute until it relinquishes control.**
Cyclic	**Control passes from one process to another.**
Executive	**An algorithm controls process scheduling.**
Manual	**Processes are scheduled by the user outside of the system.**

Processes

Processes represent single threads of control. Examples include the main program from a component diagram or the name of an active object from a collaboration diagram. To add a process to the processor, double-click on <New> in the Processes field to displays the Process Specification.

Device

A device is a hardware component with no computing power. Each device must have a name. Device names can be generic, such as "modem" or "terminal."

Graphical Depiction

The icon for a device is a box:

Connection

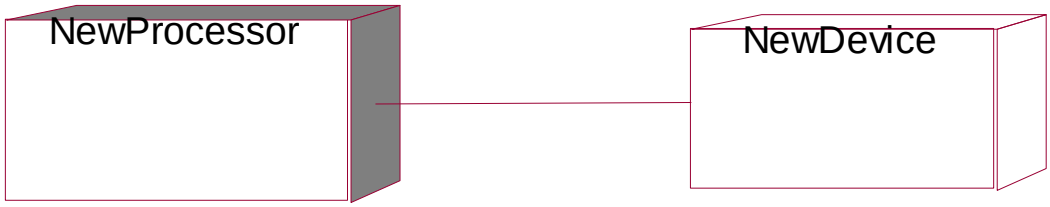
A connection represents some type of hardware coupling between two entities. An entity is either a processor or a device. The hardware coupling can be direct, such as an RS232 cable, or indirect, such as satellite-to-ground communication. Connections are usually bi-directional.

Naming

You can optionally label the connection with its name.

Graphical Depiction

The icon for a connection is a straight line:



Conclusion: In This way we have studied Deployment Diagram.

Library management system

Any organization who wishes to manage the library should assign 3 basic posts in library for better management.
1.Librarian. 2.Clerk. 3.Helper.
Library management system provides facilities for book, magazines issues and return. librarian should be able to search for a book & magazine.

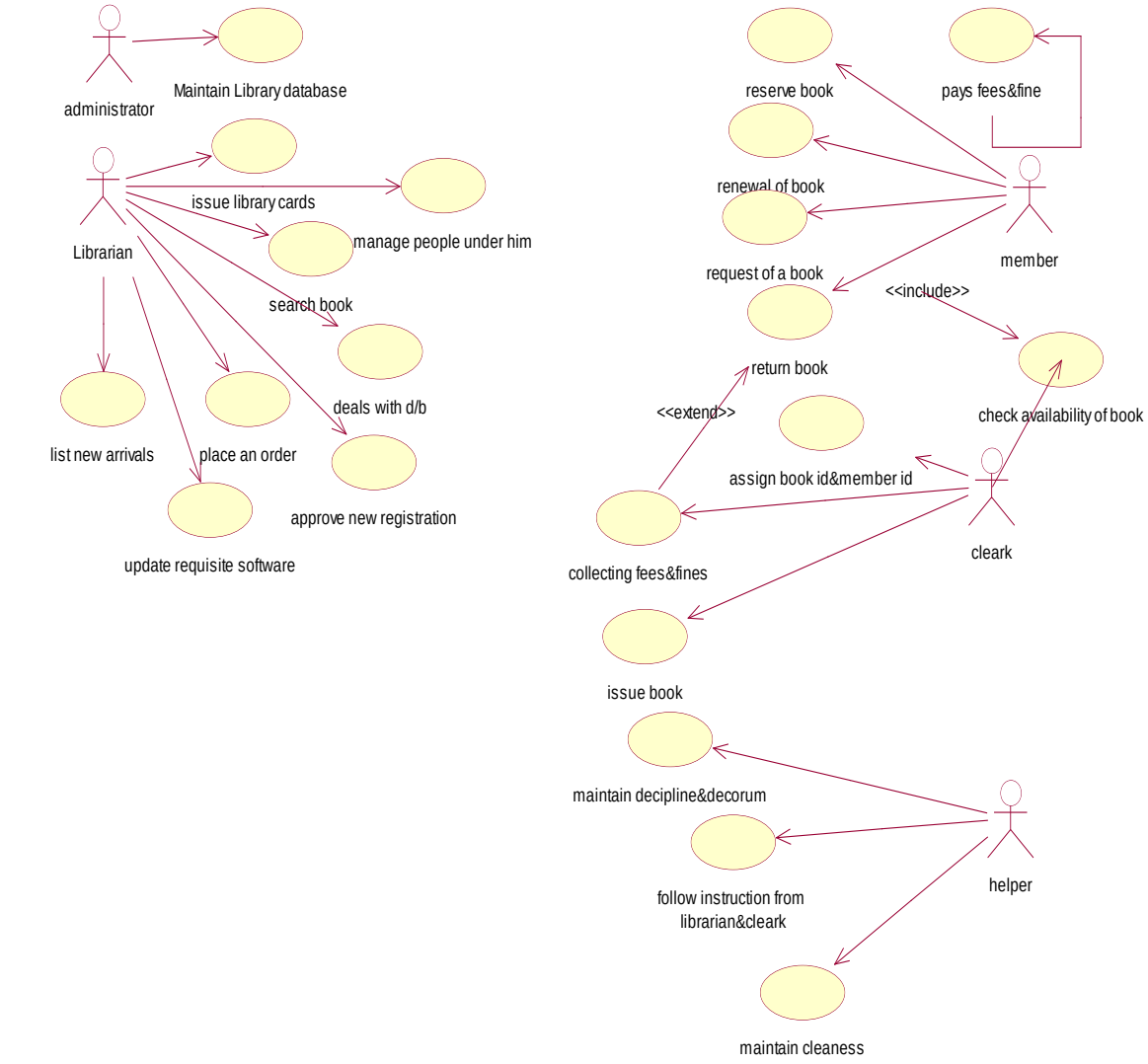
The Library has 10000 books; each book is assigned a unique identification number. There are four categories of members of the library .under graduates student, post graduates student and faculty members. Each library member is assigned a unique library membership code number, number of books issued and period for which book is issued depend upon the type of member. LMS registers each book issued to a member, when a member accounts and makes the book available for future issue.

A member who are registered visits the library to return the existing books& obtain another one. A member hands the book over to the clerk& asks for a particular book based on the book title/author ,name/publisher,name/price.&issue the same to the member alternatively the librarian may need to search for the book in it's library database .The book may be present in the library or it may be issued to another member or it may not be available at all .accordingly the member is informed and if appropriate the book is issued to member.& If the members who are registered wants a book which is not in the library, the librarian should be allowed to place an order for the book or magazine. Also it handles the purchase of new titles for the library. Popular titles are brought in multiple copies, old books and magazines are removed when they are out of date or in a poor condition.

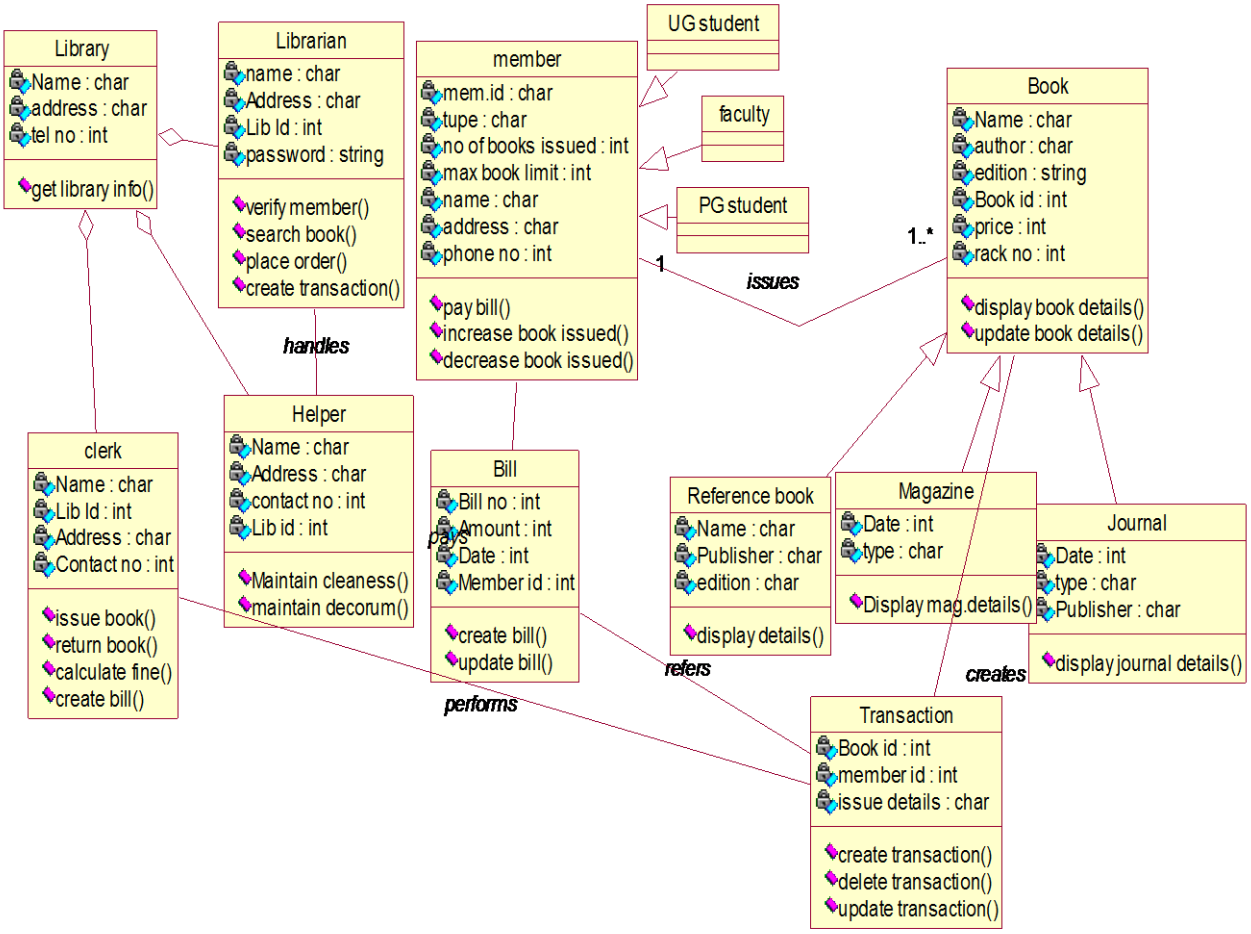
When a member returns the book the LMS prints a bill for the penalty charge for overdue books. If a book is available, the LMS displays the rack number in which book is located. A command should be supported to allow the librarian to issue a command to print reminder messages to members against whom books are overdue. LMS can easily create update and delete information about
title/author,name/publisher,name/price,members.
The librarian is the main person in the library.He takes care of the functioning of library. His functions include:
Taking care of books.Placing an order, searching for a book & magazine
Update the requisite software.Maintain database..Issue library cards..Manage all people under him.

The clerk is the issuing authority in the library.it handles the function :
Issuing the books.,Making entry in register and software..Assign returning date.
Collecting fees & fines.,Ensuring whether book is returned in proper condition.
Return deposits.
The helper is playing a supportive role in the library. it handles the function:
Maintain cleanliness.,Maintain discipline and decorum.Follow instructions from librarian and clerk.
Members can issue atmost 2 books at a time.
Members should ensure that they return books issued to them before due date in proper condition

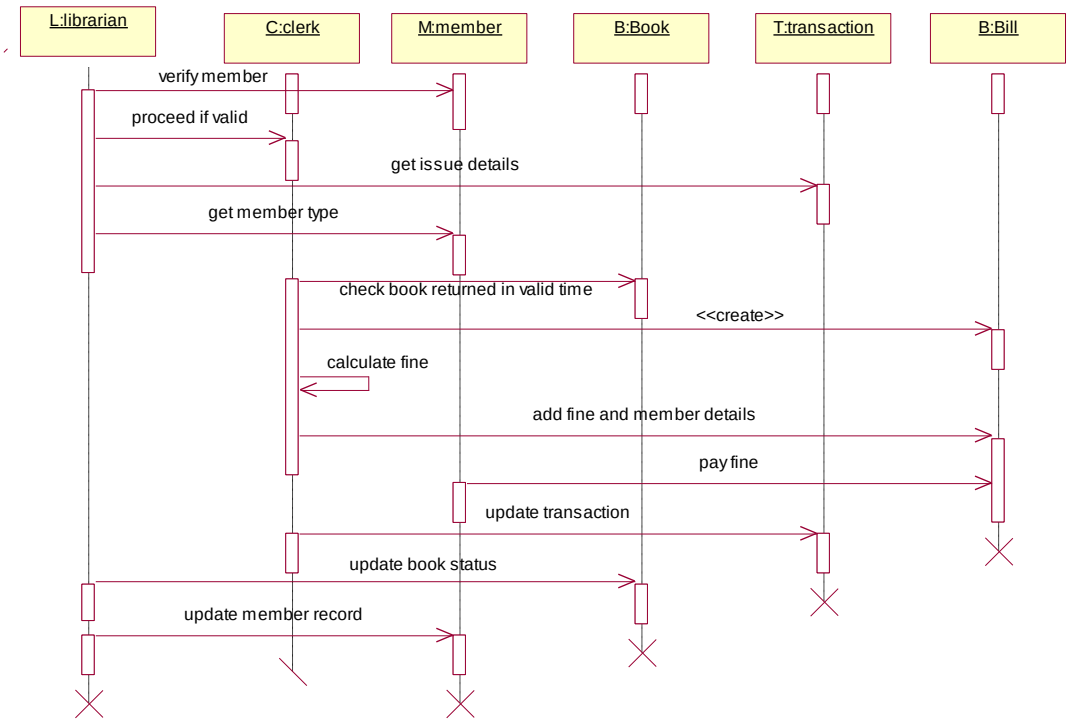
Use Case diagram for library management system



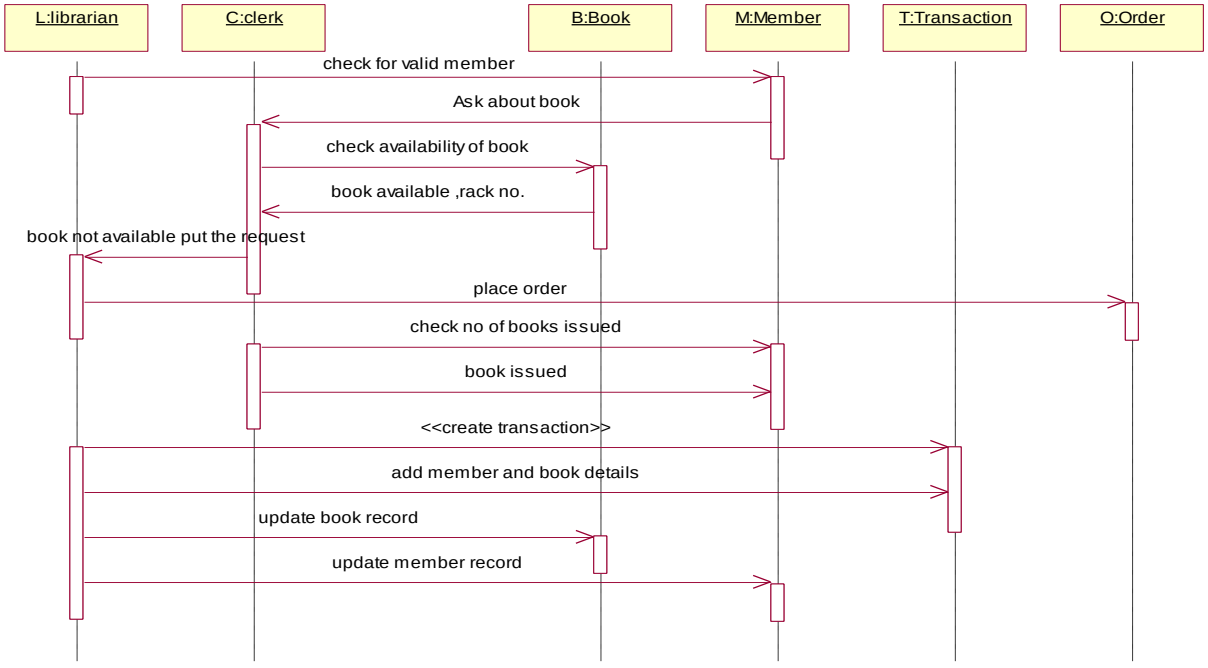
Class Diagram for Library management system



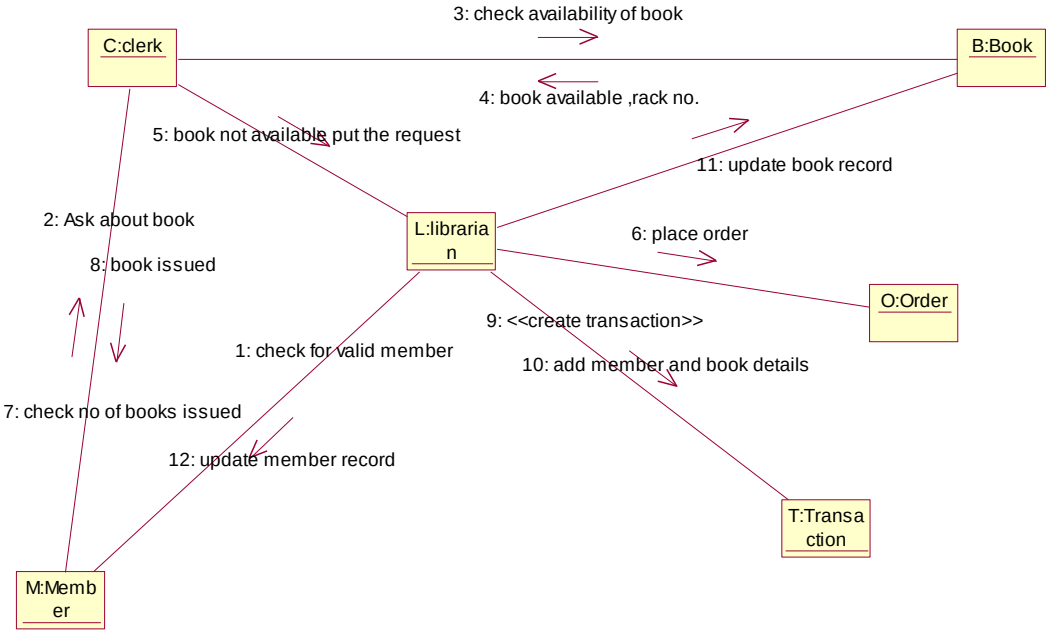
Sequence diagram for Return book



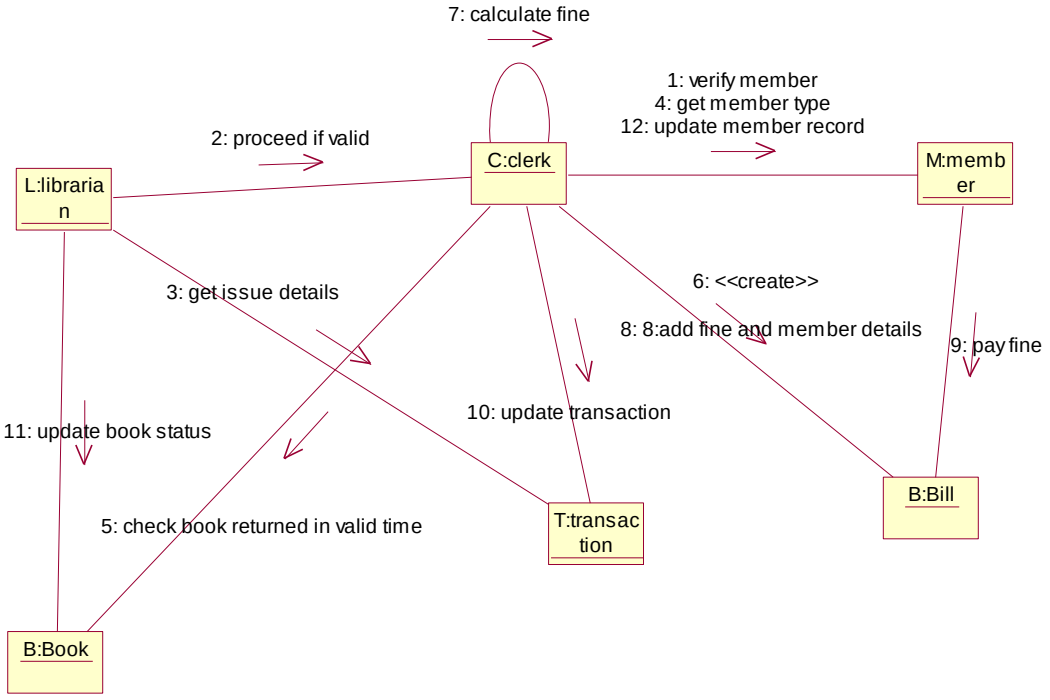
Sequence Diagram for Issue Book



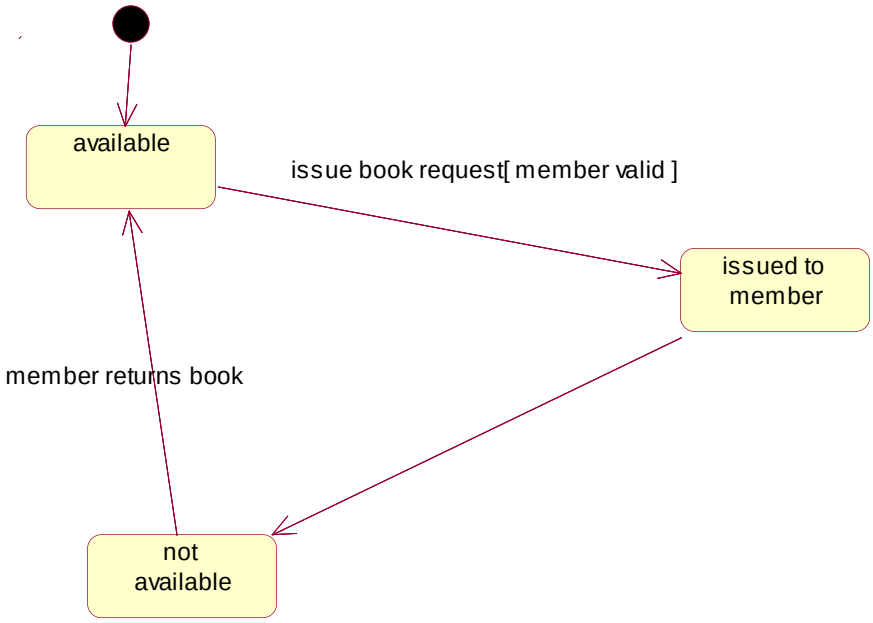
Collaboration Diagram For Issue Book



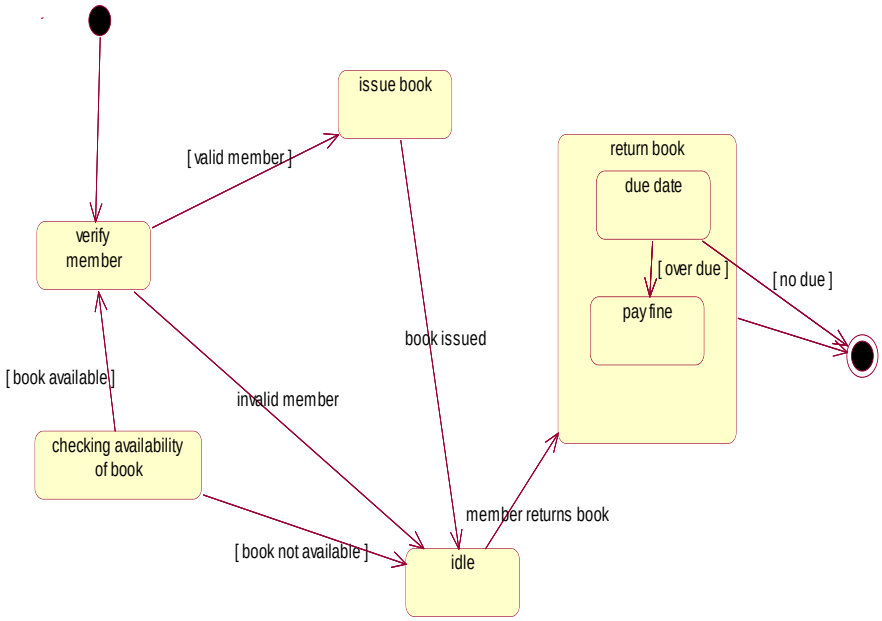
Collaboration diagram for Return Book

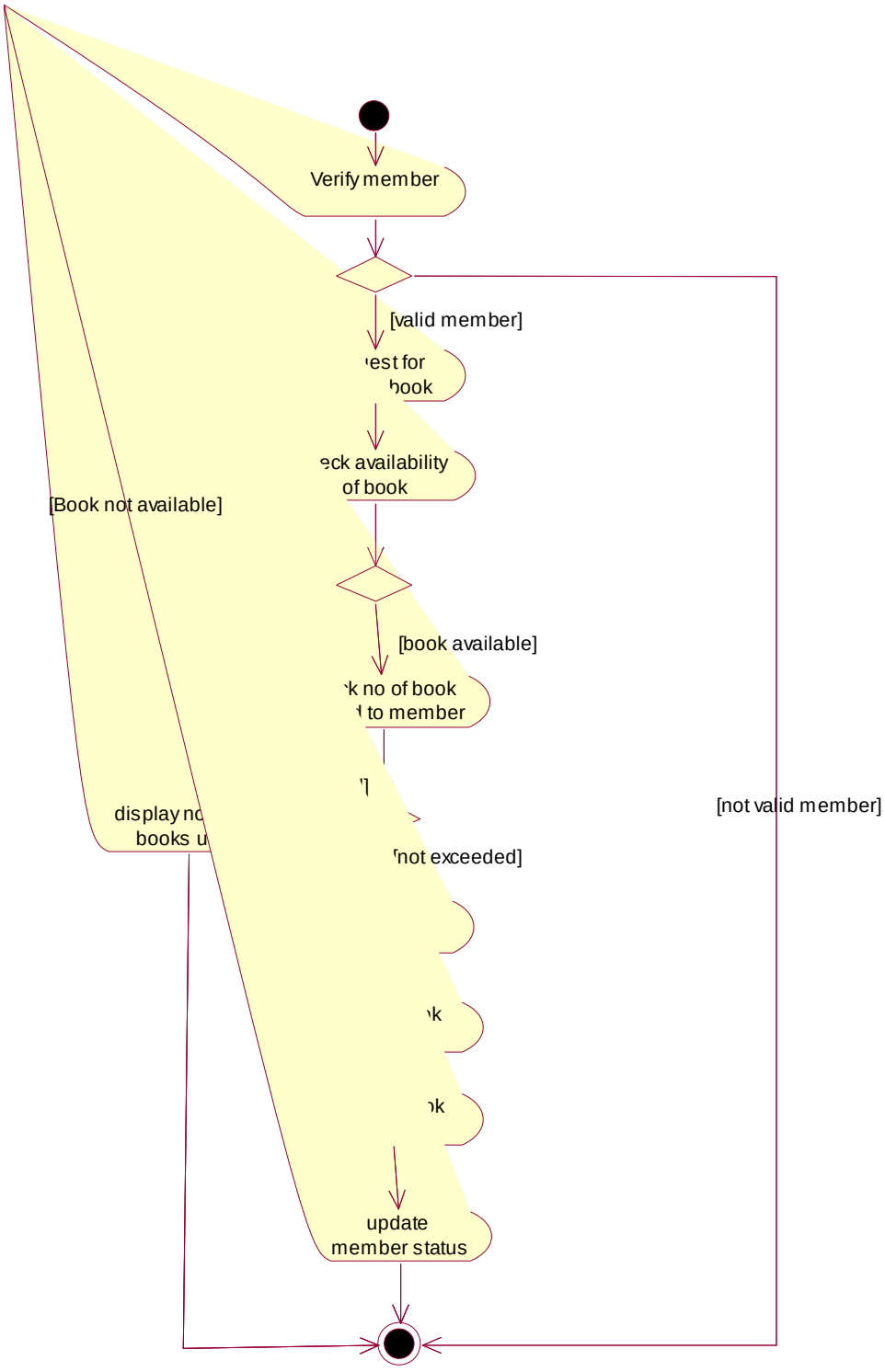


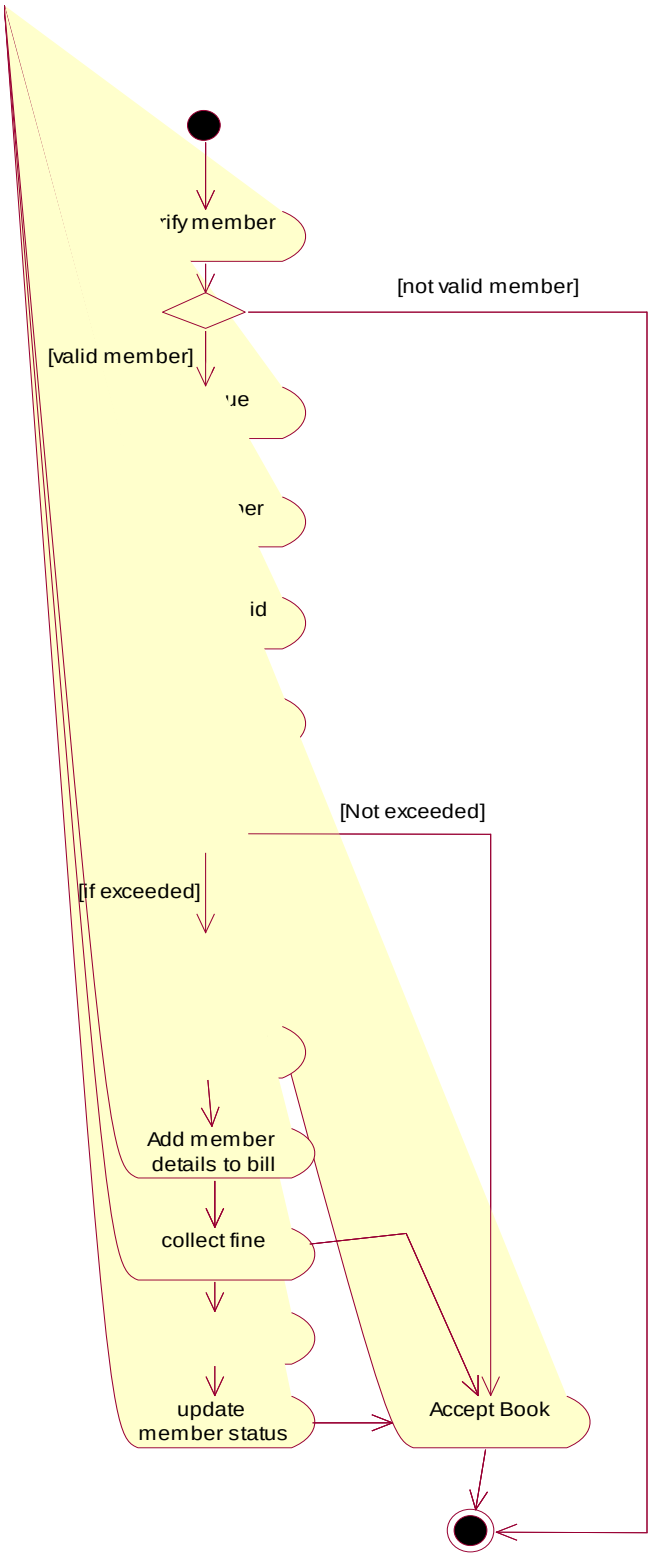
State chart diagram for book



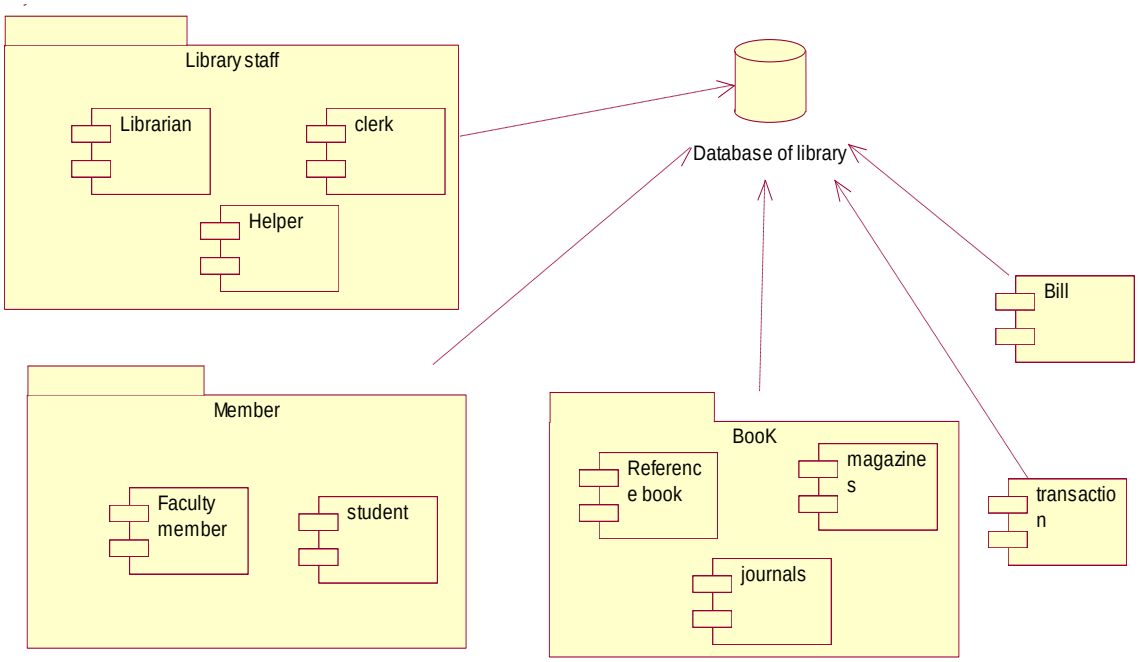
State chart diagram for Librarian




Verify member
[valid member]
est for
book
eck availability
of book
[Book not available]
[book available]
k no of book
to member
display nc
books u
not exceeded]
[not valid member]
update
member status


[not valid member]
[valid member]
[Not exceeded]
[if exceeded]
Add member details to bill
collect fine
update member status
Accept Book

Component diagram for library management system



Online Book Store

Problem Definition:

A popular bookstore wants to expand its business by going online . The bookstore has several outlets which are very popular and crowded . However customers face the problem of going to the store and locating books manually . The management feels that hosting a website which cater to the needs of these customers , will reduce crowding at the outlets , while actually increasing business. Since many more customers may like to access the online store . This will also lead to higher customer satisfaction because customers can order books from the comfort of their home or office , Further more , the store can manage its transactions efficiently.

The store maintains books according to categories viz. Medical Science , Fiction , Philosophy etc. Books are also identified by the Title , Author and Pubpblisher , Customers should be able to browse books by any of the above classification, Realizing the importance of a comprehensive customer database the management is keen to provide convenient registration facilities . This should permit the store to capture important information including contact details, professional background , preferred categories publishers authors etc. The online store should provide each registered surfer with a login id and a password . Only registered users shall be given the facility to buy books.

The store allows customers to purchase unlimited copies of a book , subject to availability . Customers should be able to browse the site when convenient and select the books for purchase . They should also be able to hand in the selected books for billing ar any point . For purchase of 5 to 10 copies of a book the store gives a discount of 5 % and for purchase of more than 10 copies , a discount of 10% is given . The store charges 2 % of the bill as delivery charge.

If they so desire , customers should be able to continue browsing after billing and buy more books as described above. On billing, the customers should be given the full details of the invoice, including invoice no, Items bought, amount of purchase etc. Books purchase by a customer should be dispatched within a week from receipt of payment along with the appropriate

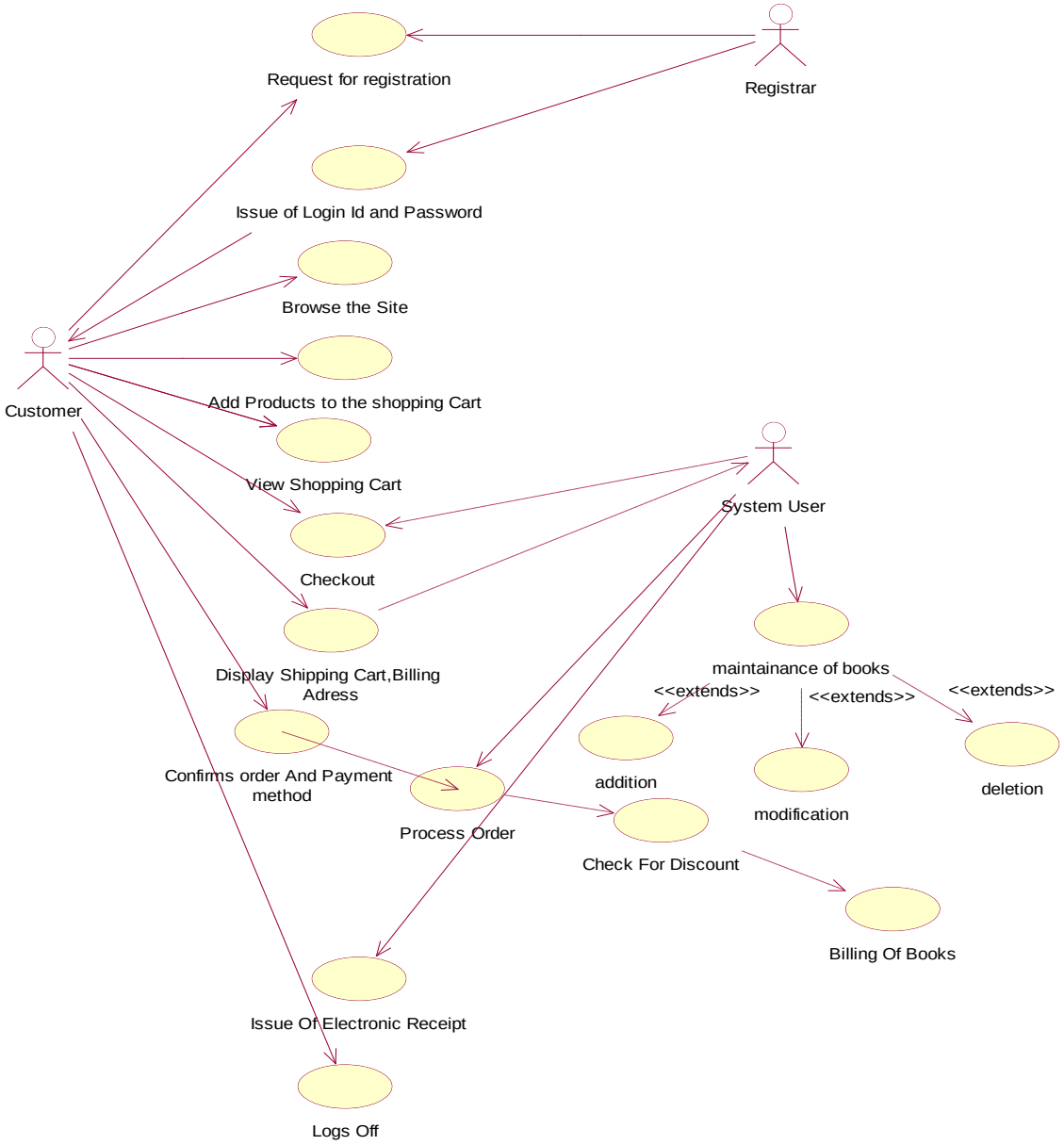
bill. The store does not bear the responsibility for the books damaged during dispatch.

At present , Inventory is being managed by a legacy system . Upgrade to include inventory within the web based system is not immediately visualized. Hence the Online system should generate invoices for all ordered items and need not check for inventory.

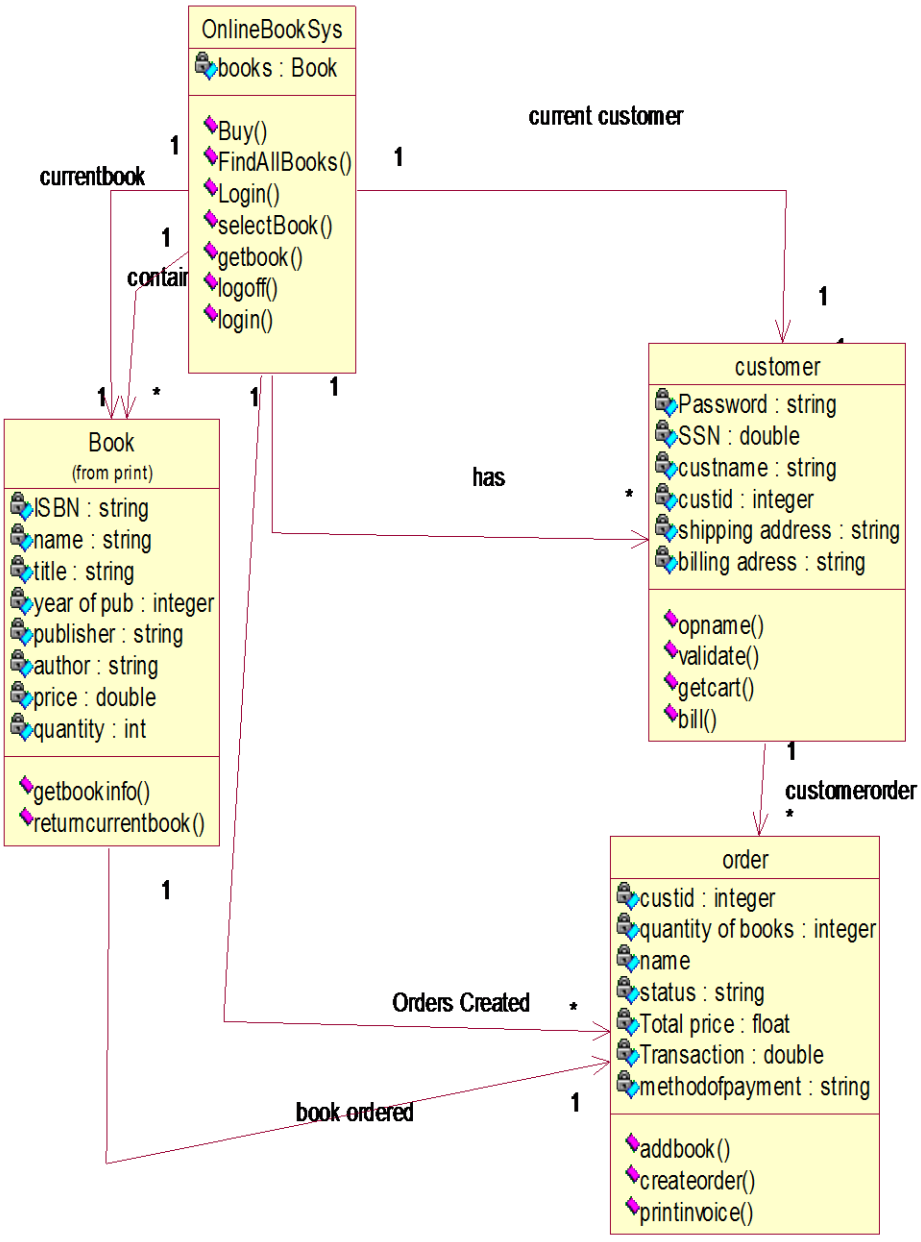
Further, the online Bookstore should provide for maintenance of details of books, book categories, publishers, and authors . Maintenance should include any addition, deletion and modification carried out.

The store management maintains complete information about the books such as Book ID, Price , Title , Edition , Year of Publication , Category , author , publisher etc. The store management also encourages the development team to add more features to the project , which shall merge well with requirements of the bookstore . However , since several competitors ere expected to go online very shortly and the fact that the management wants to be the first of the block, it becomes vital that the project is completed within the stipulated time.

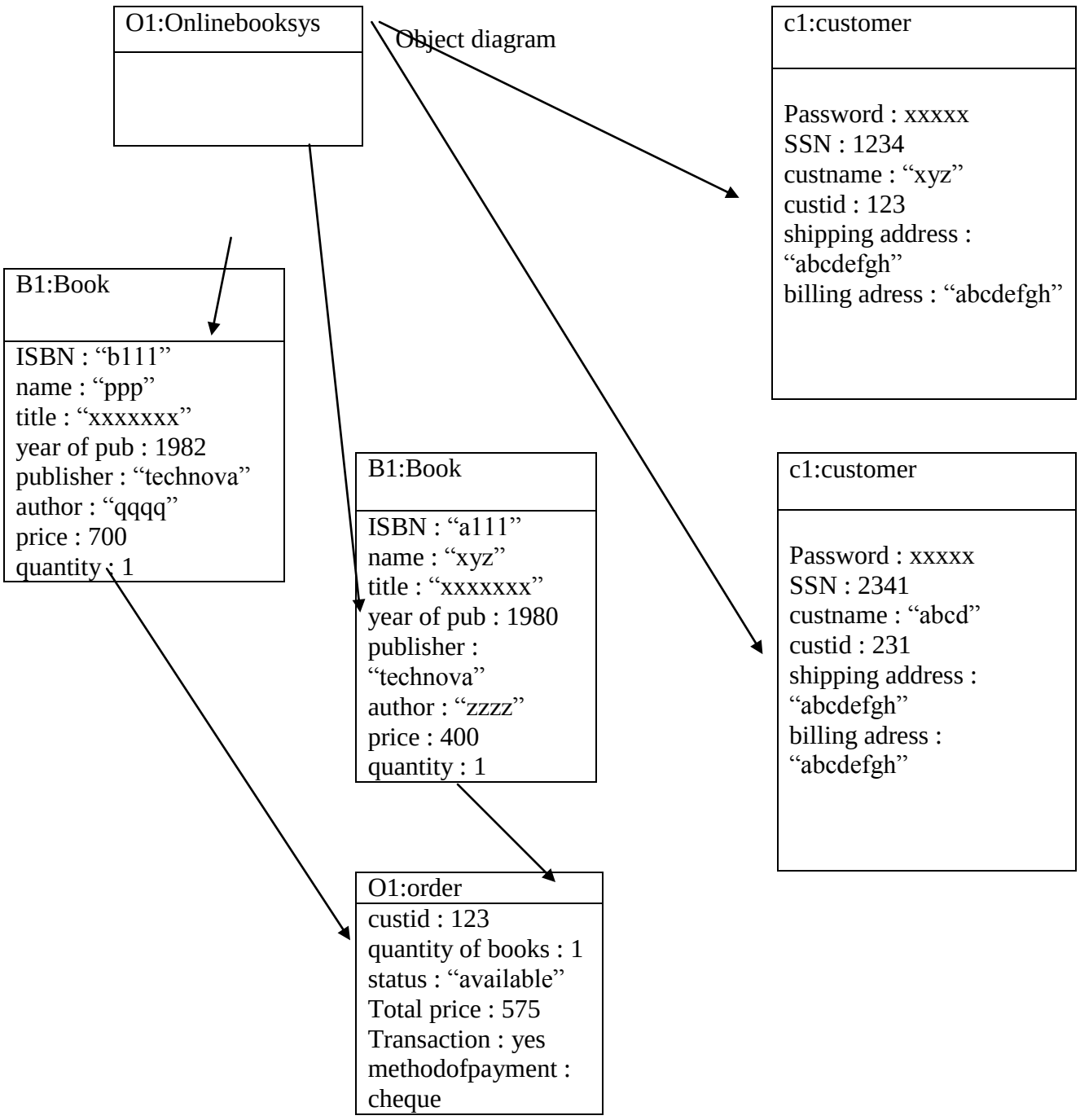
1)Usecase diagram



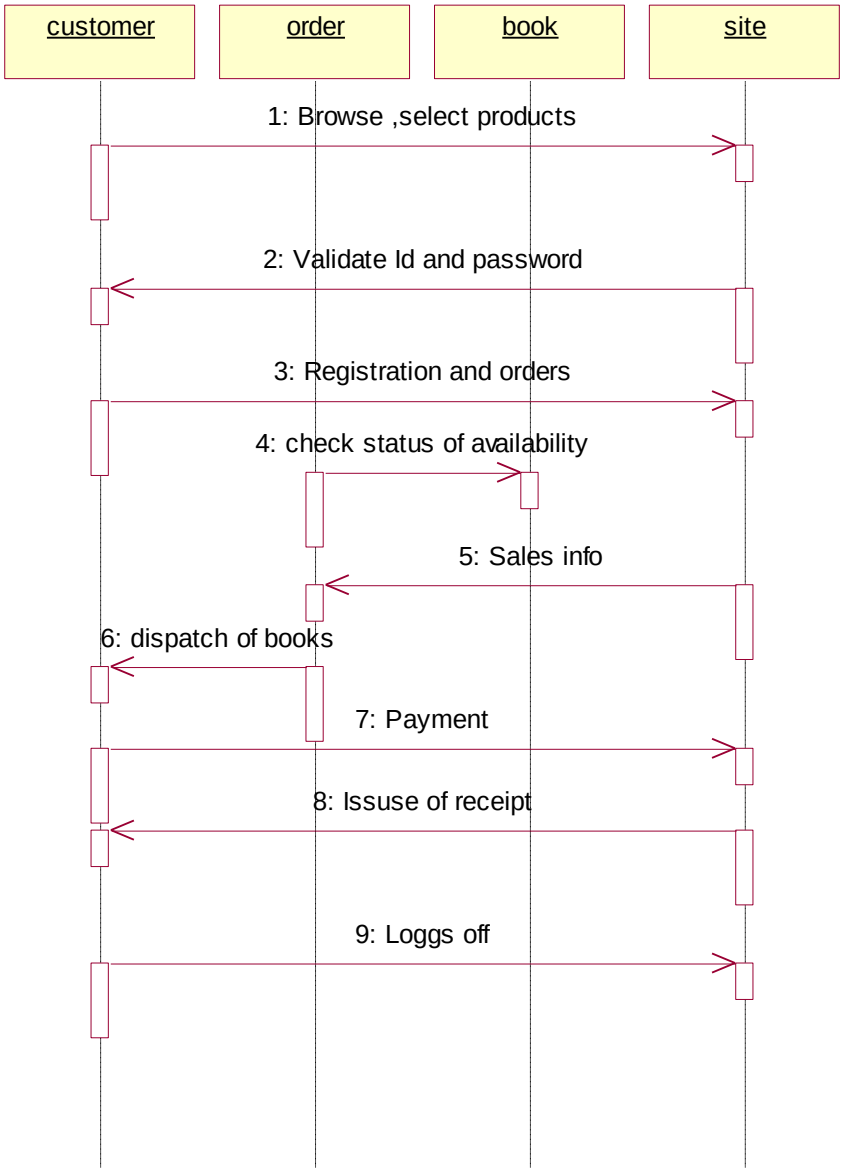
Class Diagram



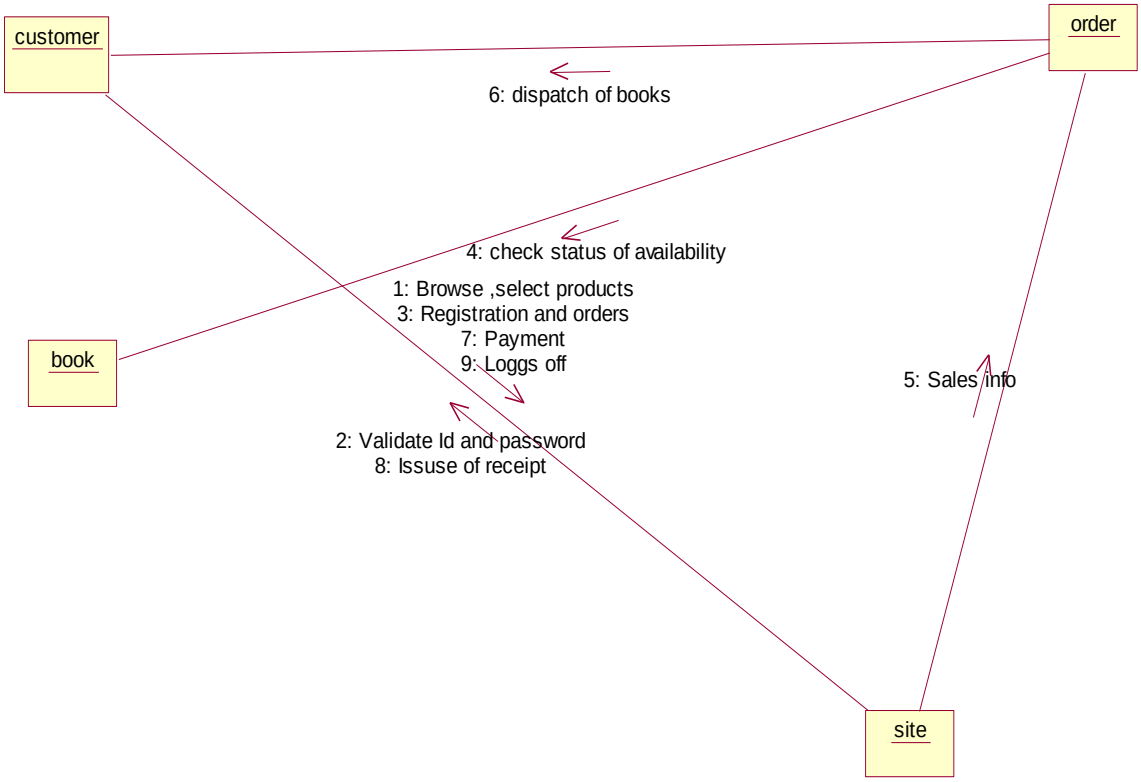
Object Diagram



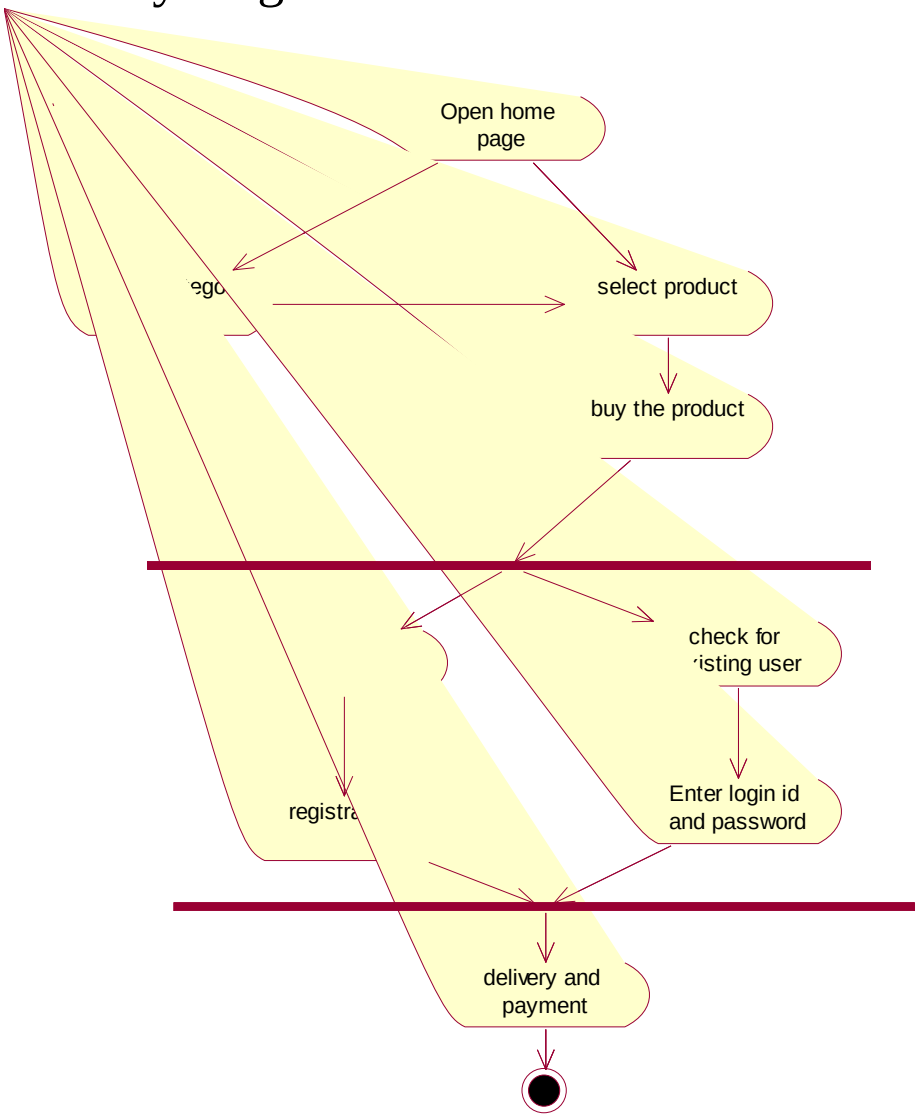
Sequence diagram



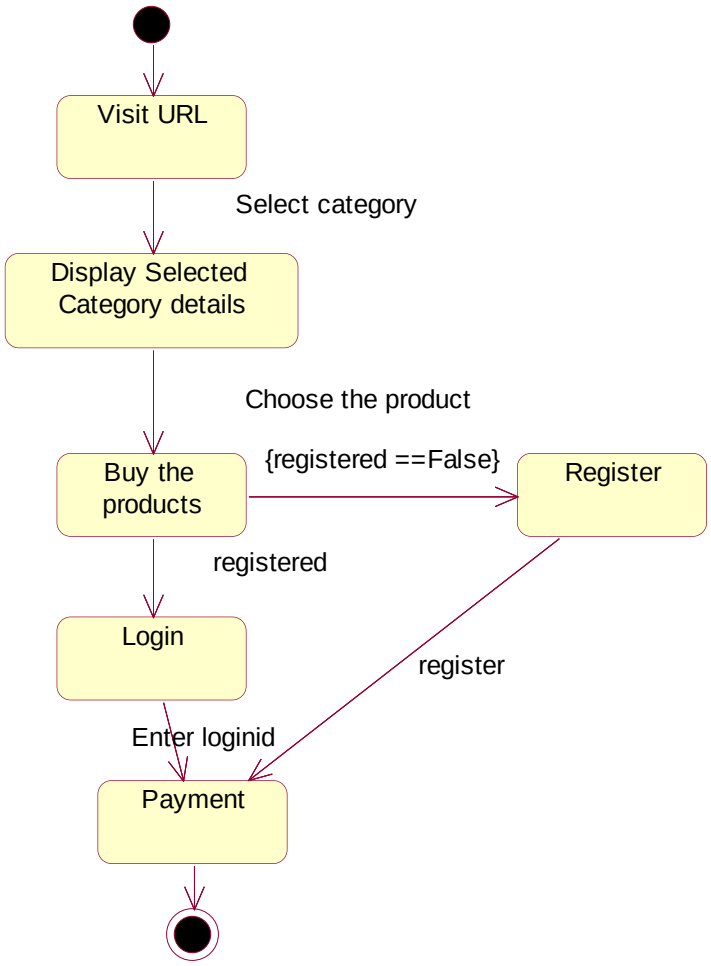
Collaboration diagram



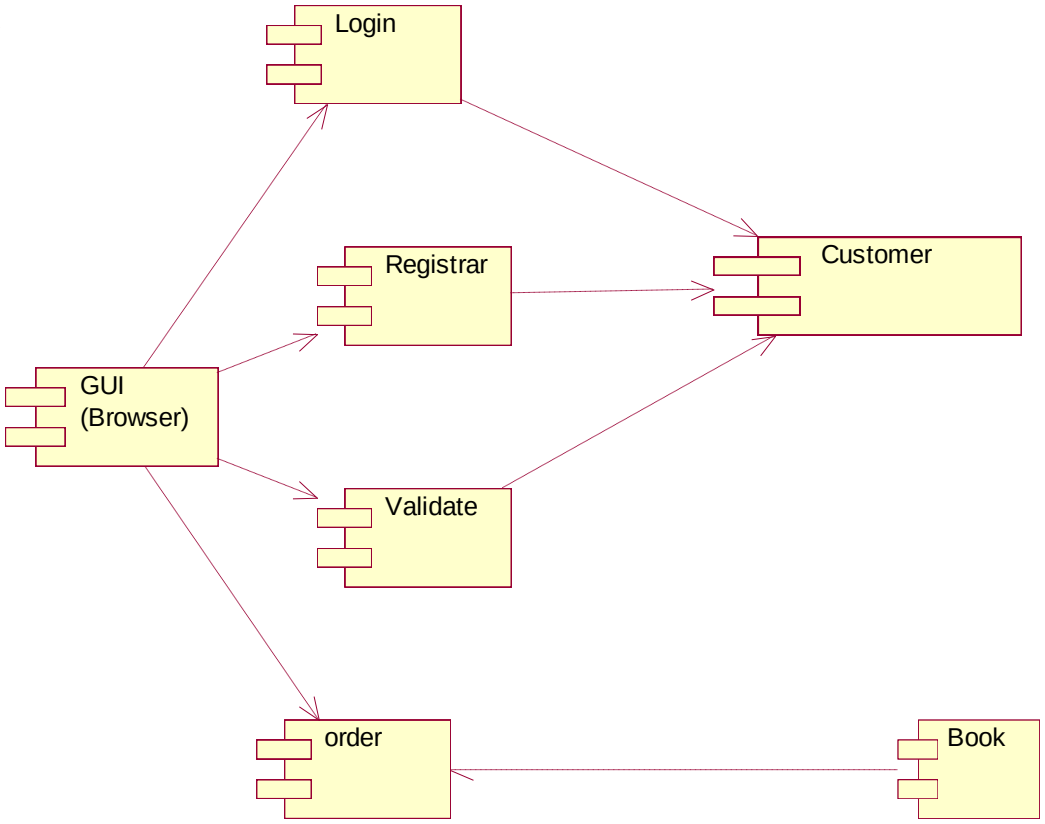
Activity diagram



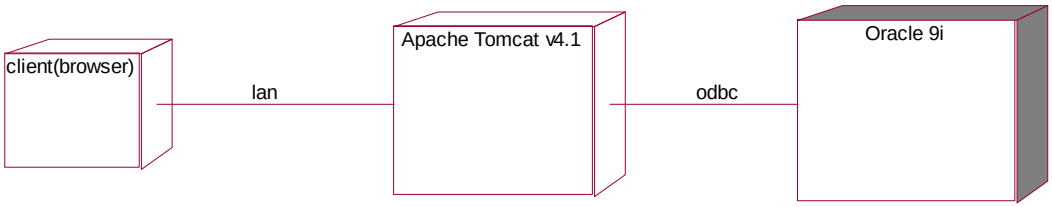
State diagram



Component diagram



deployment diagram



Student Information System

The University conducts many courses and students can register for those courses. The register university will introduce any new course. Also the register maintains the curriculum and student details. The examination will be conducted for students and the degree is issued for students who have completed the course successfully.