

# HMR

Institute of Technology and Management



Laboratory File

## **OBJECT ORIENTED SOFTWARE ENGINEERING**

For 4<sup>th</sup> Year Student

Department : Computer Science & Engineering

**ETCS – 456**

**Submitted To :**

**Ms. Deepika Rawat**

**CSE Department**

**Submitted By :**

**Mohd. Aftab Alam**

**CSE – 8A – 02013302717**

# INDEX

<u>S. NO.</u>	<u>EXPERIMENT</u>	<u>DATE OF EXP.</u>	<u>SIGNATURE</u>
	<b>Case Study 1 – LIBRARY MANAGEMENT SYSTEM</b>		
1.	Identify and Create Use Cases and Actors for the problem statement	20/04/2021	
2.	Define Classes in the Logical View associated with each Use Case and define their Structure	27/04/2021	
3.	Draw Sequence Diagram for each Use Case identified in the problem statement	04/05/2021	
4.	Draw Collaboration Diagram for the problem statement	11/05/2021	
5.	Draw Class Diagram for the problem statement	18/05/2021	
	<b>Case Study 2 – UNIVERSITY MANAGEMENT SYSTEM</b>		
1.	Identify and Create Use Cases and Actors for the problem statement	25/05/2021	
2.	Define Classes in the Logical View associated with each Use Case and define their Structure	01/06/2021	
3.	Draw Sequence Diagram for each Use Case identified in the problem statement	08/06/2021	
4.	Draw Collaboration Diagram for the problem statement	15/06/2021	
5.	Draw Class Diagram for the problem statement	17/06/2021	

## **CASE STUDY 1**

### **LIBRARY INFORMATION MANAGEMENT SYSTEM**

A Library Information System is to be developed for automating the various functions of the library. Faculty, Students and Employees of the Institute can take the membership of the library. Each member can issue a book up to specific limit and specific duration. However, the limit may vary from one type of member to the other type of member. The Library Information Management System performs the Cataloguing, Managing Members, Issue and Return of the Book.

## EXPERIMENT 1

### Aim :

Identify and Create Use Cases and Actors for the problem statement.

### Theory :

#### **What is a Use Case Diagram?**

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

#### **When to apply Use Case Diagrams?**

A use case diagram doesn't go into a lot of detail. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Use cases are represented with a labelled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modelled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself. Use case diagrams are ideal for:

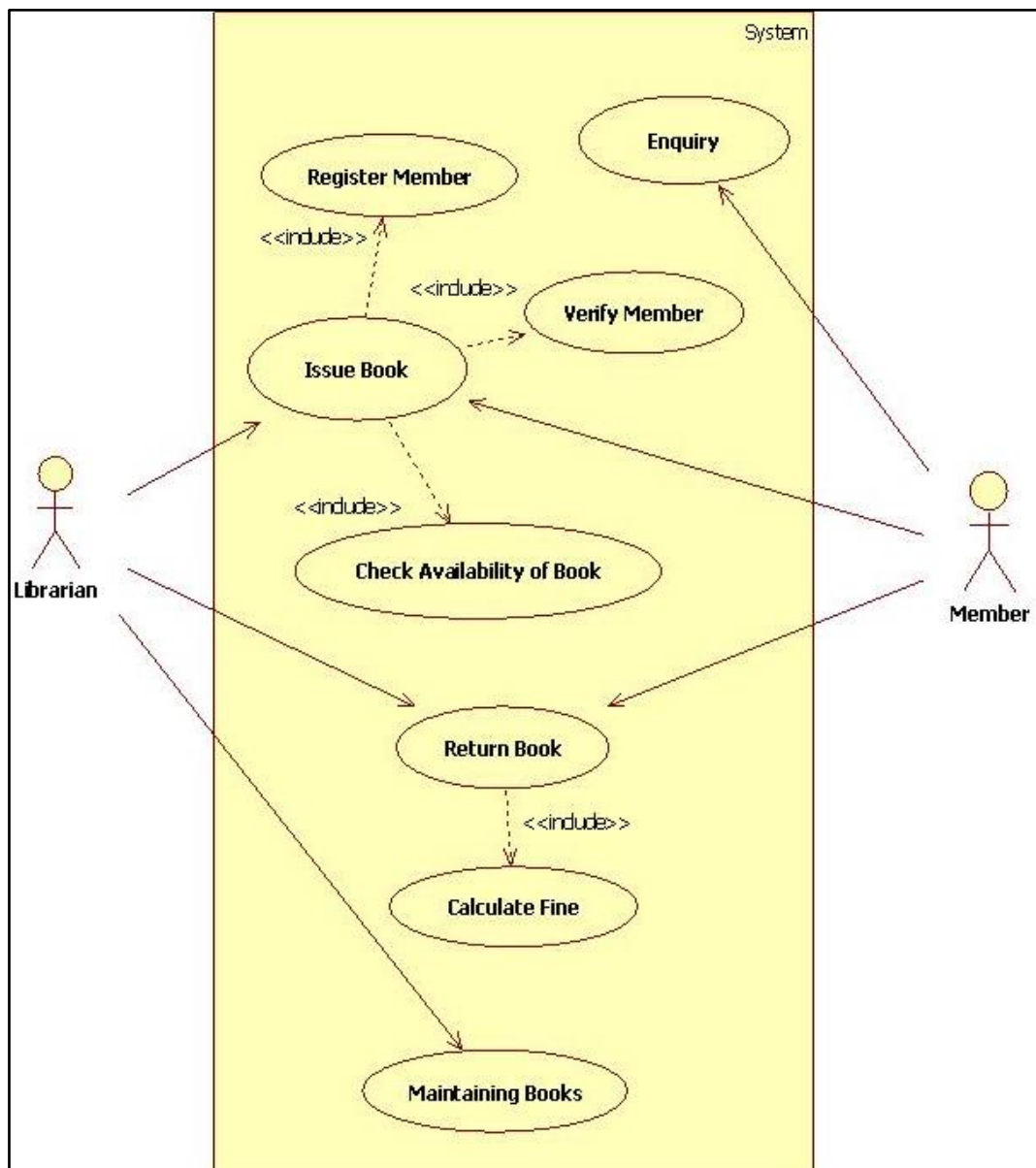
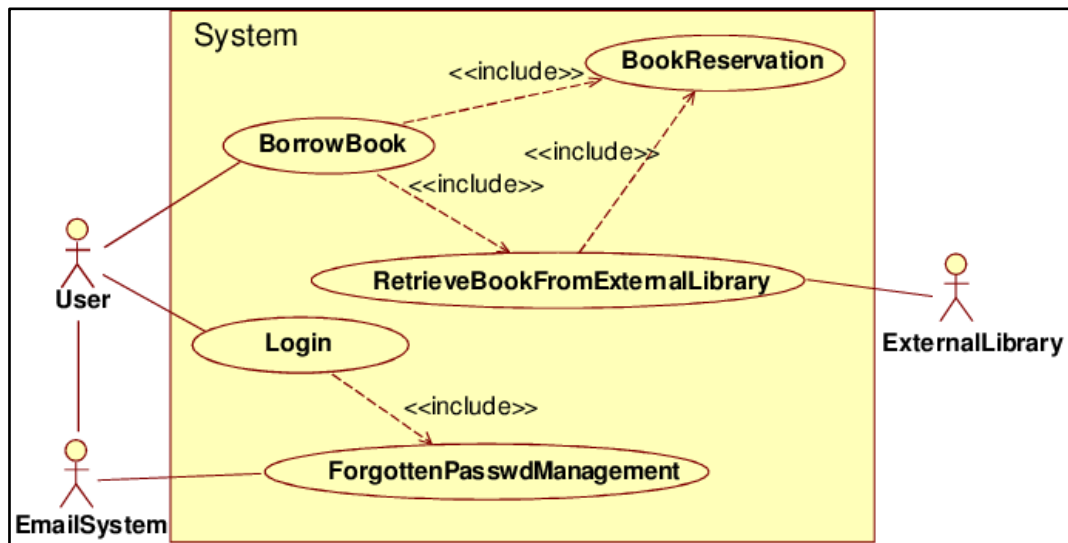
- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modelling the basic flow of events in a use case

#### **What are the Components of Use Case Diagram?**

- **Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
- **System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
- **Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

#### **Use Case Diagram Symbols and Notation:**

- **Use Cases:** Horizontally shaped ovals that represent the different uses that a user might have.
- **Actors:** Stick figures that represent the people actually employing the use cases.
- **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- **System Boundary Boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system.
- **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

Use Case Diagrams **Library Management System** :

## EXPERIMENT 2

### Aim :

Define Classes in the Logical View associated with each Use Case and define their Structure.

### Theory :

#### **What is Activity Diagram?**

The Unified Modelling Language (UML) includes several subsets of diagrams, including structure diagrams, interaction diagrams, and behaviour diagrams. Activity diagrams, along with use case and state machine diagrams, are considered behaviour diagrams because they describe what must happen in the system being modelled.

Stakeholders have many issues to manage, so it's important to communicate with clarity and brevity. Activity diagrams help people on the business and development sides of an organization come together to understand the same process and behaviour. We'll use a set of specialized symbols – including those used for starting, ending, merging, or receiving steps in the flow – to make an activity diagram.

#### **Benefits of Activity Diagram:**

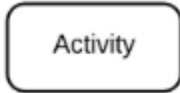
- It demonstrates the logic of an algorithm.
- It describes the steps performed in a UML use case.
- It illustrates a business process or workflow between users and the system.
- It simplifies and improves any process by clarifying complicated use cases.
- It models software architecture elements, such as method, function, and operation.

#### **What are the Components of Activity Diagram?**

- **Action:** A step in the activity wherein the users or software perform a given task.
- **Decision Node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
- **Control Flows:** Another name for the connectors that show the flow between steps in the diagram.
- **Start Node:** Symbolizes the beginning of the activity. The start node is represented by a black circle.
- **End Node:** Represents the final step in the activity. The end node is represented by an outlined black circle.

**Activity Diagram Symbols and Notation:**
**Symbol**
**Name**

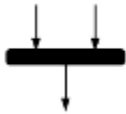

Start Symbol



Activity Symbol



Connector Symbol



Joint Symbol



Fork Symbol



Decision Symbol



Note Symbol



Send Signal Symbol



Receive Signal Symbol



Shallow History Pseudo State Symbol



Option Loop Symbol



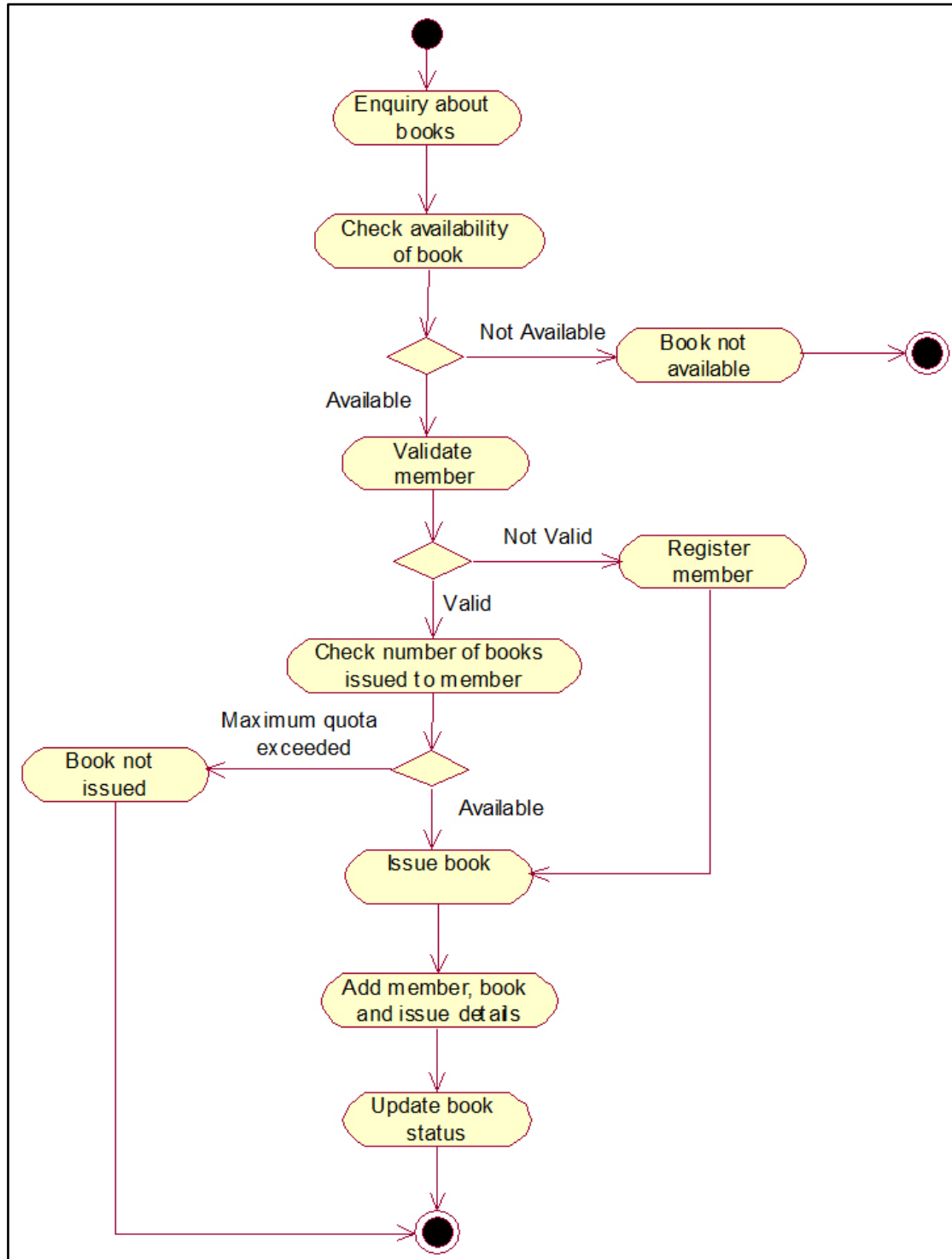
Flow Final Symbol

[Condition]

Condition Text



End Symbol

Logical View for Library Management System :



## EXPERIMENT 3

### Aim :

Draw Sequence Diagram for each Use Case identified in the problem statement.

### Theory :

#### **What is Sequence Diagram?**

A Sequence Diagram is a type of interaction diagram because it describes how – and in what order – a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as Event Diagrams or Event Scenarios. Note that there are two types of Sequence Diagrams: UML diagrams and code – based diagrams. The latter is sourced from programming code.



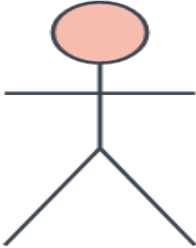

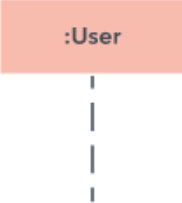


#### **Benefits of Sequence Diagram:**

- Represents the details of a UML use case.
- Models the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.







#### **Use Cases for Sequence Diagram:**

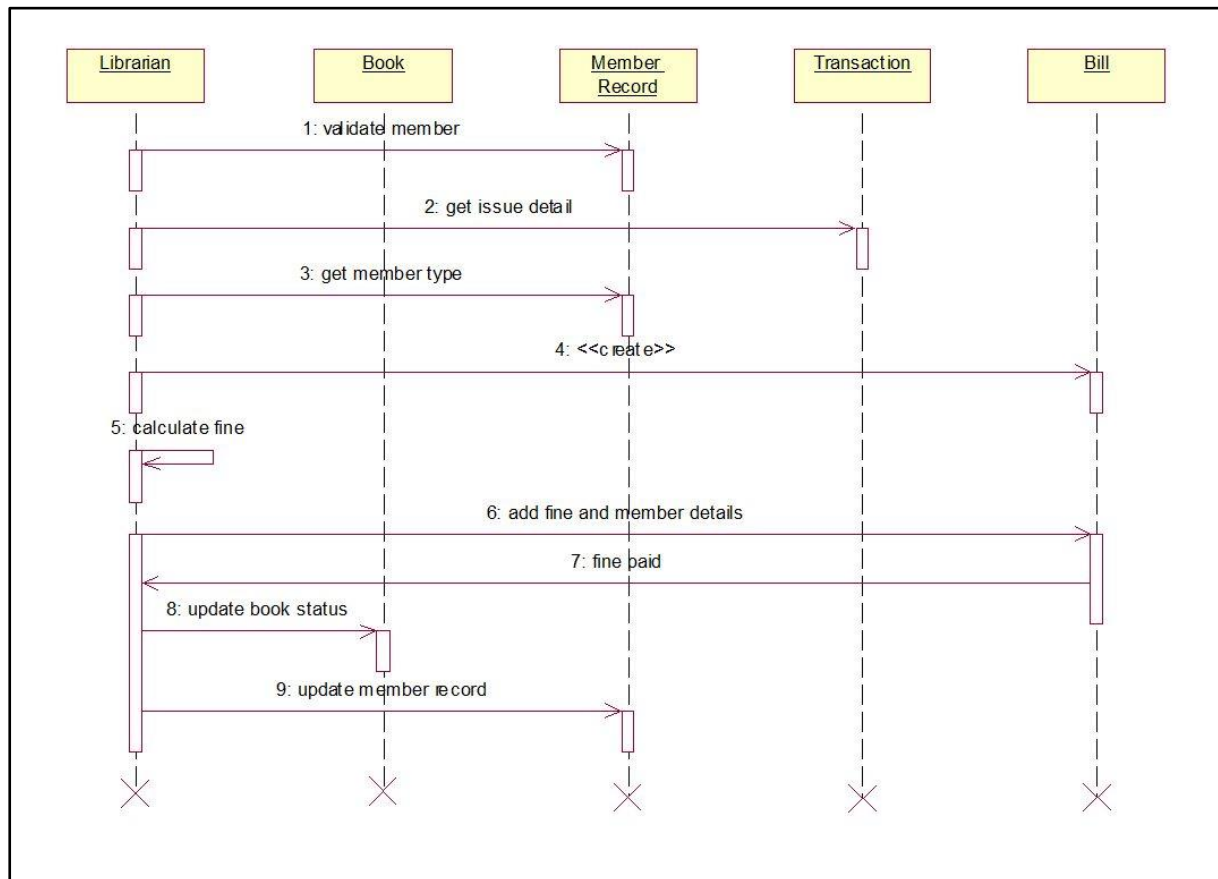
- **Usage Scenario:** A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
- **Method Logic:** Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
- **Service Logic:** If you consider a service to be a high – level method used by different clients, a sequence diagram is an ideal way to map that out.

**Basic Symbols & Components:**

<u>Symbol</u>	<u>Name</u>	<u>Description</u>
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Actor symbol	Shows entities that interact with or are external to the system.
	Package symbol	Also known as a frame, this rectangular shape has a small inner rectangle for labelling the diagram.
	Lifeline symbol	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labelled rectangle shape or an actor symbol.
	Option loop symbol	Used to model if/then scenarios, i.e., a circumstance that will only occur under certain conditions.
	Alternative symbol	Symbolizes a choice between two or more message sequences. To represent alternatives, use the labelled rectangle shape with a dashed line inside.

**Common Message Symbols:**

<u>Symbol</u>	<u>Name</u>	<u>Description</u>
	Synchronous message symbol	Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
	Asynchronous message symbol	Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram.
	Asynchronous return message symbol	Represented by a dashed line with a lined arrowhead.
	Asynchronous create message symbol	Represented by a dashed line with a lined arrowhead. This message creates a new object.
	Reply message symbol	Represented by a dashed line with a lined arrowhead, these messages are replies to calls.
	Delete message symbol	Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object.

Sequence Diagram for Library Management System :

## **EXPERIMENT 4**

### **Aim :**

Draw Collaboration Diagram for the problem statement.

### **Theory :**

#### **What is Collaboration Diagram?**

A Collaboration Diagram also known as Communication Diagram offers the same information as a Sequence Diagram, but while a Sequence Diagram emphasizes the time and order of events, a Collaboration Diagram emphasizes the messages exchanged between objects in an application. Collaboration Diagrams offer the broader perspective within a process.

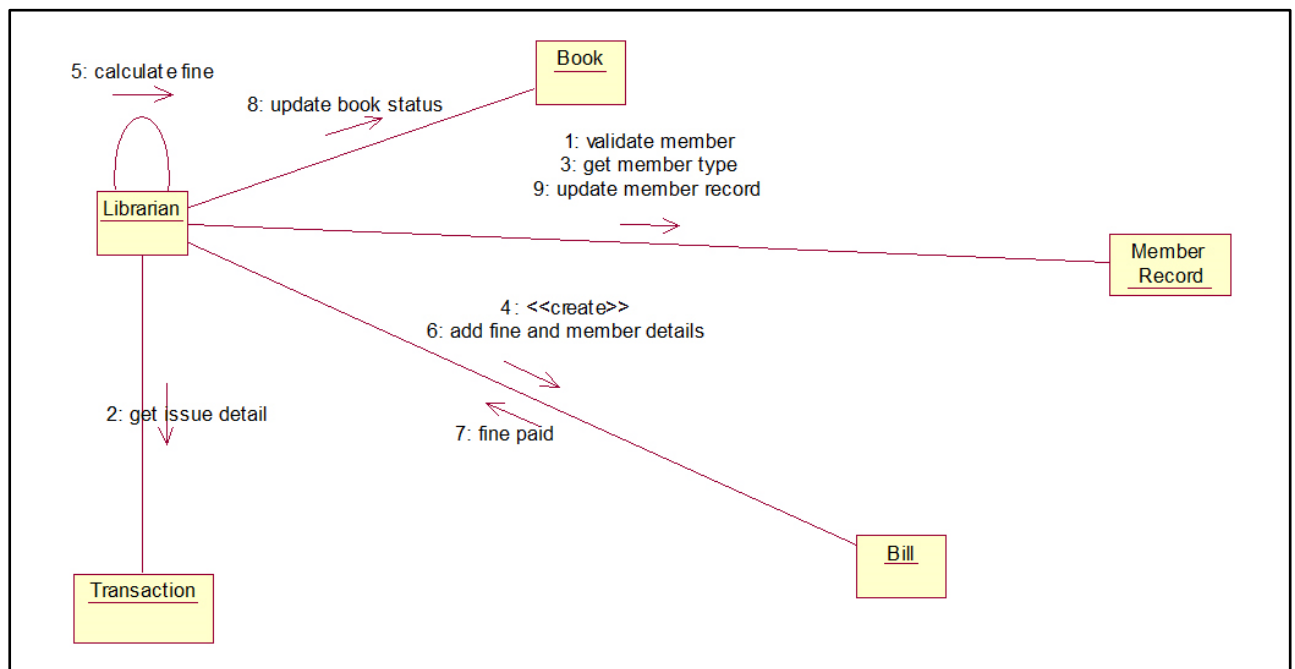
#### **Benefits of Collaboration Diagram:**

- Model the logic of a sophisticated procedure, function, or operation.
- Identify how commands are sent and received between objects or components of a process.
- Visualize the consequences of specific interactions between various components in a process.
- Plan and understand the detailed functionality of an existing or future scenario.

#### **Collaboration Diagram Symbols and Notation:**

- **Rectangles** represent objects that make up the application.
- **Lines** between class instances represent the relationships between different parts of the application.
- **Arrows** represent the messages that are sent between objects.
- **Numbering** lets you know in what order the messages are sent and how many messages are required to finish a process.

Collaboration Diagram for **Library Management System** :



## EXPERIMENT 5

### Aim :

Draw Class Diagram for the problem statement.

### Theory :

#### **What is Class Diagram?**

Class Diagrams are a type of structure diagram because they describe what must be present in the system being modelled. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects.

The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.

#### **Benefits of Class Diagram:**

- Illustrate data models for information systems, no matter how simple or complex.
- Better understand the general overview of the schematics of an application.
- Visually express any specific needs of a system and disseminate that information throughout the business.
- Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- Provide an implementation – independent description of types used in a system that are later passed between its components.

#### **What are the Components of Class Diagram?**

- **Upper Section:** Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- **Middle Section:** Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- **Bottom Section:** Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

#### **Member Access Modifiers:**

- Public ( + )
- Private ( - )
- Protected ( # )
- Package ( ~ )
- Derived ( / )
- Static ( underlined )

#### **Member Scopes:**

There are two scopes for members: Classifiers and Instances. Classifiers are static members while Instances are the specific instances of the class.

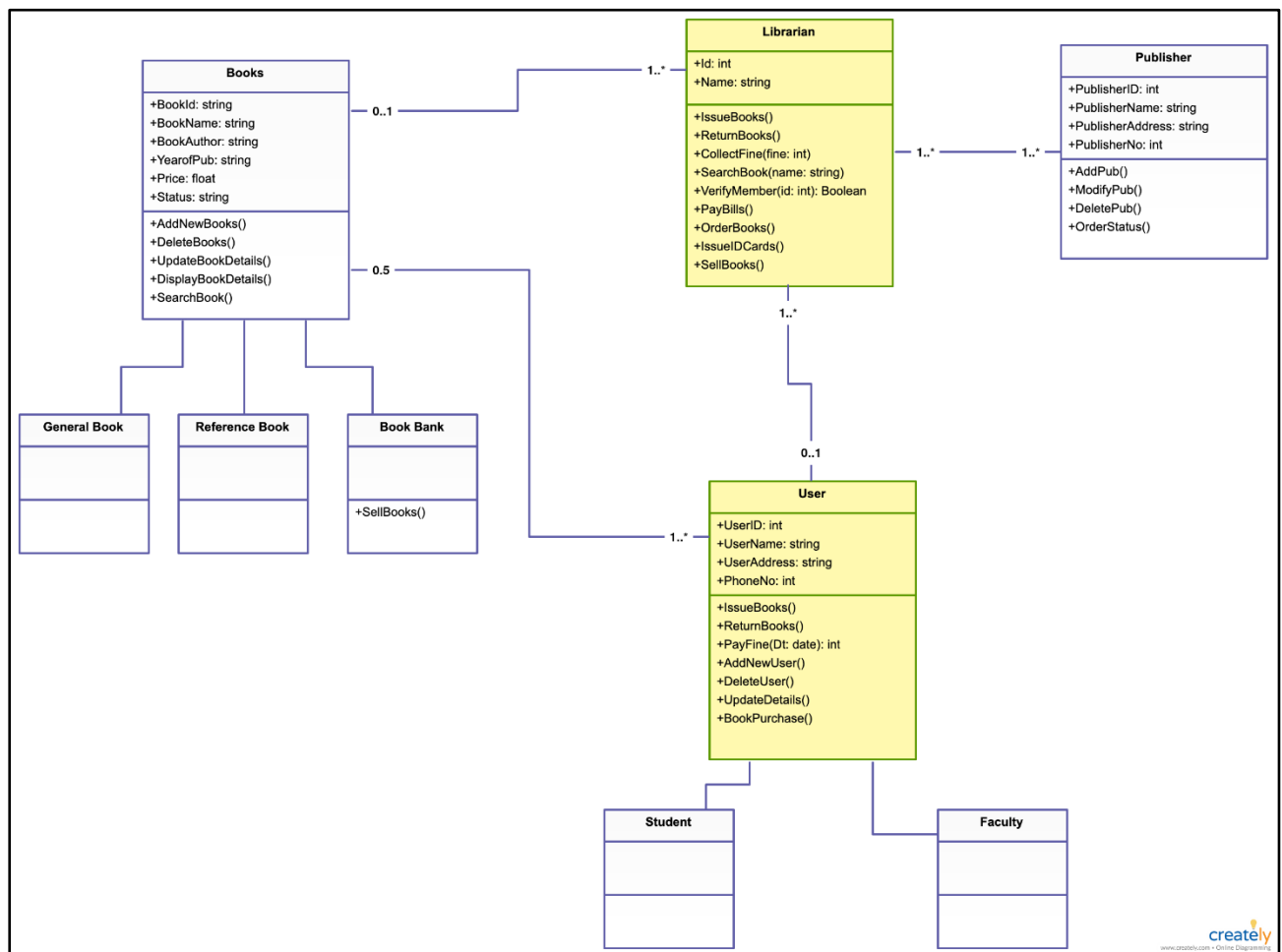
### **Additional Class Diagram Components:**

- **Classes:** A template for creating objects and implementing behaviour in a system. A class represents an object or a set of objects that share a common structure and behaviour. They're represented by a rectangle that includes rows of the class name, its attributes, and its operations.
  - **Name:** The first row in a class shape.
  - **Attributes:** The second row in a class shape. Each attribute of the class is displayed on a separate line.
  - **Methods:** The third row in a class shape. Also known as operations, methods are displayed in list format with each operation on its own line.
- **Signals:** Symbols that represent one-way, asynchronous communications between active objects.
- **Data Types:** Classifiers that define data values. Data types can model both primitive types and enumerations.
- **Packages:** Shapes designed to organize related classifiers in a diagram. They are symbolized with a large tabbed rectangle shape.
- **Interfaces:** A collection of operation signatures and/or attribute definitions that define a cohesive set of behaviours.
- **Enumerations:** Representations of user – defined data types. An enumeration includes groups of identifiers that represent values of the enumeration.
- **Objects:** Instances of a class or classes. Objects can be added to a class diagram to represent either concrete or prototypical instances.
- **Artefacts:** Model elements that represent the concrete entities in a software system, such as documents, databases, executable files, software components, etc.

### **Interactions:**

- **Inheritance:** The process of a child or sub – class taking on the functionality of a parent or superclass, also known as generalization. It's symbolized with a straight connected line with a closed arrowhead pointing towards the superclass.
- **Bidirectional Association:** The default relationship between two classes. Both classes are aware of each other and their relationship with the other. This association is represented by a straight line between two classes.
- **Unidirectional Association:** A slightly less common relationship between two classes. One class is aware of the other and interacts with it. Unidirectional association is modelled with a straight connecting line that points an open arrowhead from the knowing class to the known class.



Class Diagram for Library Management System :

## **CASE STUDY 2**

### **EASTERN STATE UNIVERSITY COURSE REGISTRATION**

At the beginning of each semester, students may request a course catalog containing a list of course offerings for the semester. Information about each course, such as professor, department and prerequisites will be included to help students make informed decisions.

The new system will allow students to select four courses offering for the coming semester. In addition, each student will indicate two alternative choices in case a course offering becomes filled or canceled. No course offering will have more than ten students or fewer than three students. A course offering with fewer than three students will be cancelled. Once the registration process is completed for a student, the registration system sends information to the billing system so the student can be billed for the semester.

Professor must be able to access the online system to indicate which courses they will be teaching, and to see which students signed up for their course offerings.

For each semester, there is a period of time that students can change their schedule. Students must be able to access the system during this time to add or drop courses.

## EXPERIMENT 1

### Aim :

Identify and Create Use Cases and Actors for the problem statement

### Theory :

#### **What is a Use Case Diagram?**

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

#### **When to apply Use Case Diagrams?**

A use case diagram doesn't go into a lot of detail. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Use cases are represented with a labelled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modelled with a line between the actor and use case. To depict the system boundary, draw a box around the use case itself. Use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modelling the basic flow of events in a use case

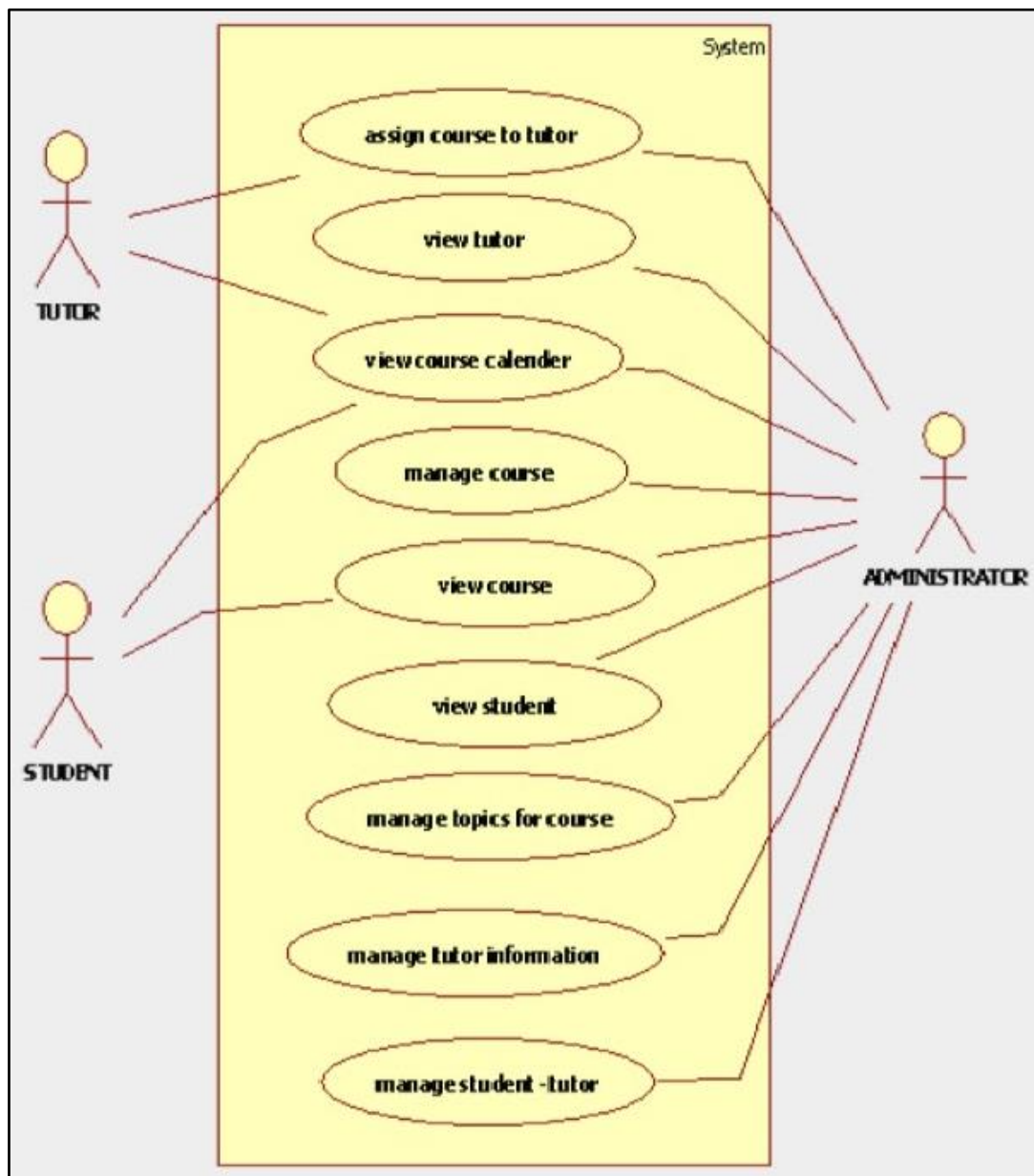
#### **What are the Components of Use Case Diagram?**

- **Actors:** The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
- **System:** A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
- **Goals:** The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

#### **Use Case Diagram Symbols and Notation:**

- **Use Cases:** Horizontally shaped ovals that represent the different uses that a user might have.
- **Actors:** Stick figures that represent the people actually employing the use cases.
- **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- **System Boundary Boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system.
- **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

Use Case Diagram for University Management System :



## EXPERIMENT 2

### Aim :

Define Classes in the Logical View associated with each Use Case and define their Structure.

### Theory :

#### **What is Activity Diagram?**

The Unified Modelling Language (UML) includes several subsets of diagrams, including structure diagrams, interaction diagrams, and behaviour diagrams. Activity diagrams, along with use case and state machine diagrams, are considered behaviour diagrams because they describe what must happen in the system being modelled.

Stakeholders have many issues to manage, so it's important to communicate with clarity and brevity. Activity diagrams help people on the business and development sides of an organization come together to understand the same process and behaviour. We'll use a set of specialized symbols – including those used for starting, ending, merging, or receiving steps in the flow – to make an activity diagram.

#### **Benefits of Activity Diagram:**

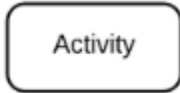
- It demonstrates the logic of an algorithm.
- It describes the steps performed in a UML use case.
- It illustrates a business process or workflow between users and the system.
- It simplifies and improves any process by clarifying complicated use cases.
- It models software architecture elements, such as method, function, and operation.

#### **What are the Components of Activity Diagram?**

- **Action:** A step in the activity wherein the users or software perform a given task.
- **Decision Node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
- **Control Flows:** Another name for the connectors that show the flow between steps in the diagram.
- **Start Node:** Symbolizes the beginning of the activity. The start node is represented by a black circle.
- **End Node:** Represents the final step in the activity. The end node is represented by an outlined black circle.

**Activity Diagram Symbols and Notation:**
**Symbol**
**Name**

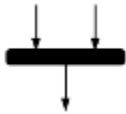

Start Symbol



Activity Symbol



Connector Symbol



Joint Symbol



Fork Symbol



Decision Symbol



Note Symbol



Send Signal Symbol



Receive Signal Symbol



Shallow History Pseudo State Symbol



Option Loop Symbol



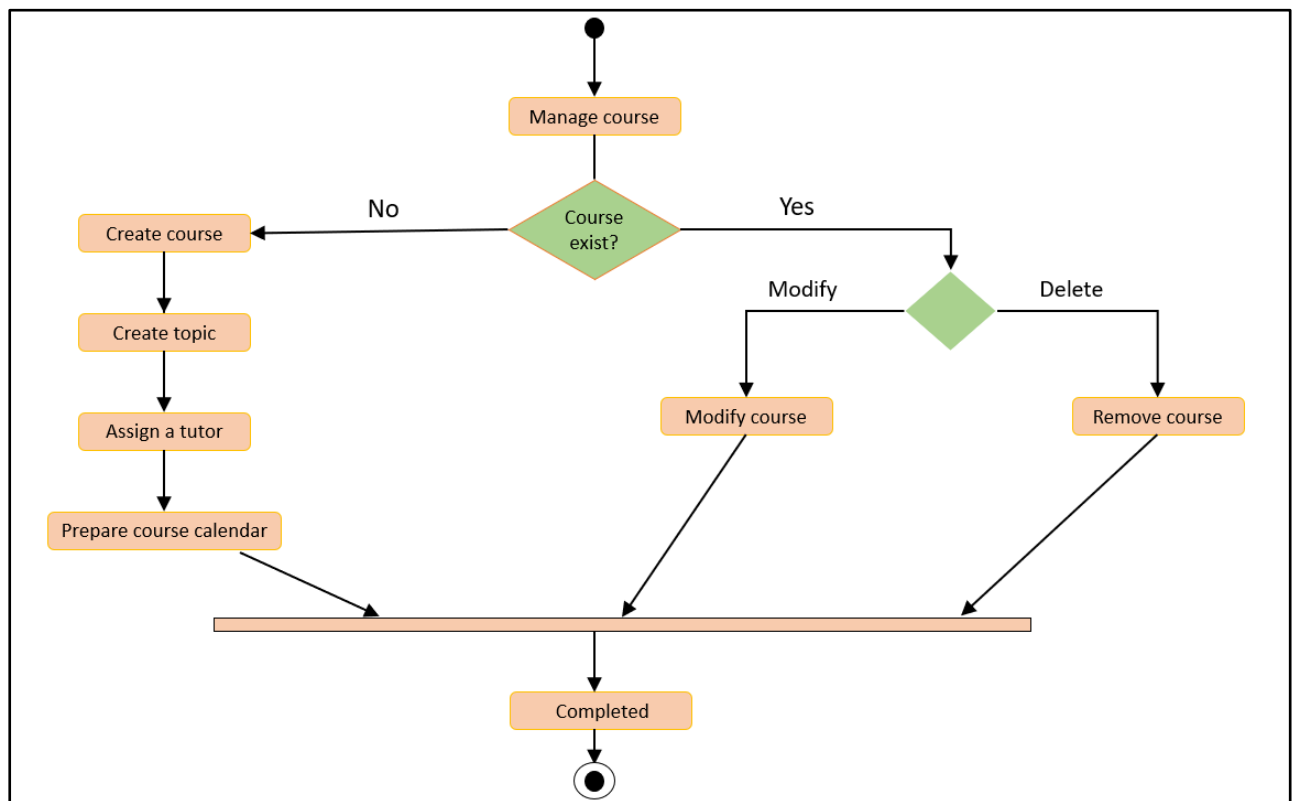
Flow Final Symbol

[Condition]

Condition Text



End Symbol

Logical View for University Management System :

## EXPERIMENT 3

### Aim :

Draw Sequence Diagram for each Use Case identified in the problem statement.

### Theory :

#### **What is Sequence Diagram?**

A Sequence Diagram is a type of interaction diagram because it describes how – and in what order – a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as Event Diagrams or Event Scenarios. Note that there are two types of Sequence Diagrams: UML diagrams and code – based diagrams. The latter is sourced from programming code.

#### **Benefits of Sequence Diagram:**



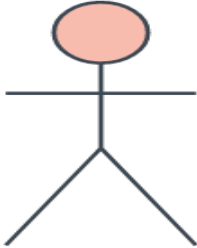

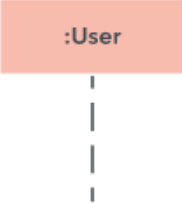


- Represents the details of a UML use case.
- Models the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.

#### **Use Cases for Sequence Diagram:**







- **Usage Scenario:** A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
- **Method Logic:** Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
- **Service Logic:** If you consider a service to be a high – level method used by different clients, a sequence diagram is an ideal way to map that out.

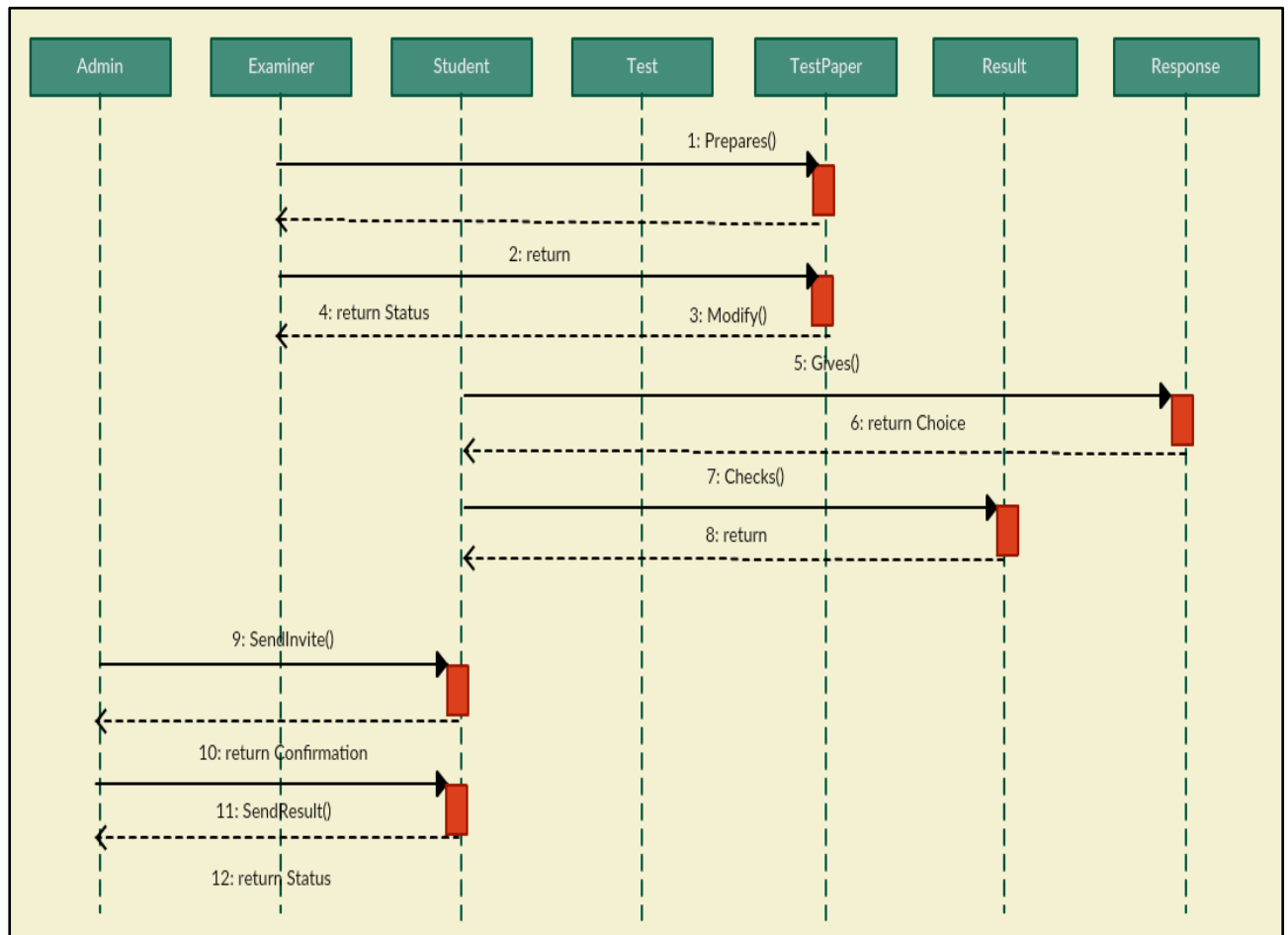


**Basic Symbols & Components:**

<u>Symbol</u>	<u>Name</u>	<u>Description</u>
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Actor symbol	Shows entities that interact with or are external to the system.
	Package symbol	Also known as a frame, this rectangular shape has a small inner rectangle for labelling the diagram.
	Lifeline symbol	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labelled rectangle shape or an actor symbol.
	Option loop symbol	Used to model if/then scenarios, i.e., a circumstance that will only occur under certain conditions.
	Alternative symbol	Symbolizes a choice between two or more message sequences. To represent alternatives, use the labelled rectangle shape with a dashed line inside.

**Common Message Symbols:**

<u>Symbol</u>	<u>Name</u>	<u>Description</u>
	Synchronous message symbol	Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
	Asynchronous message symbol	Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram.
	Asynchronous return message symbol	Represented by a dashed line with a lined arrowhead.
	Asynchronous create message symbol	Represented by a dashed line with a lined arrowhead. This message creates a new object.
	Reply message symbol	Represented by a dashed line with a lined arrowhead, these messages are replies to calls.
	Delete message symbol	Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object.

Sequence Diagram for University Management System :

## **EXPERIMENT 4**

### **Aim :**

Draw Collaboration Diagram for the problem statement.

### **Theory :**

#### **What is Collaboration Diagram?**

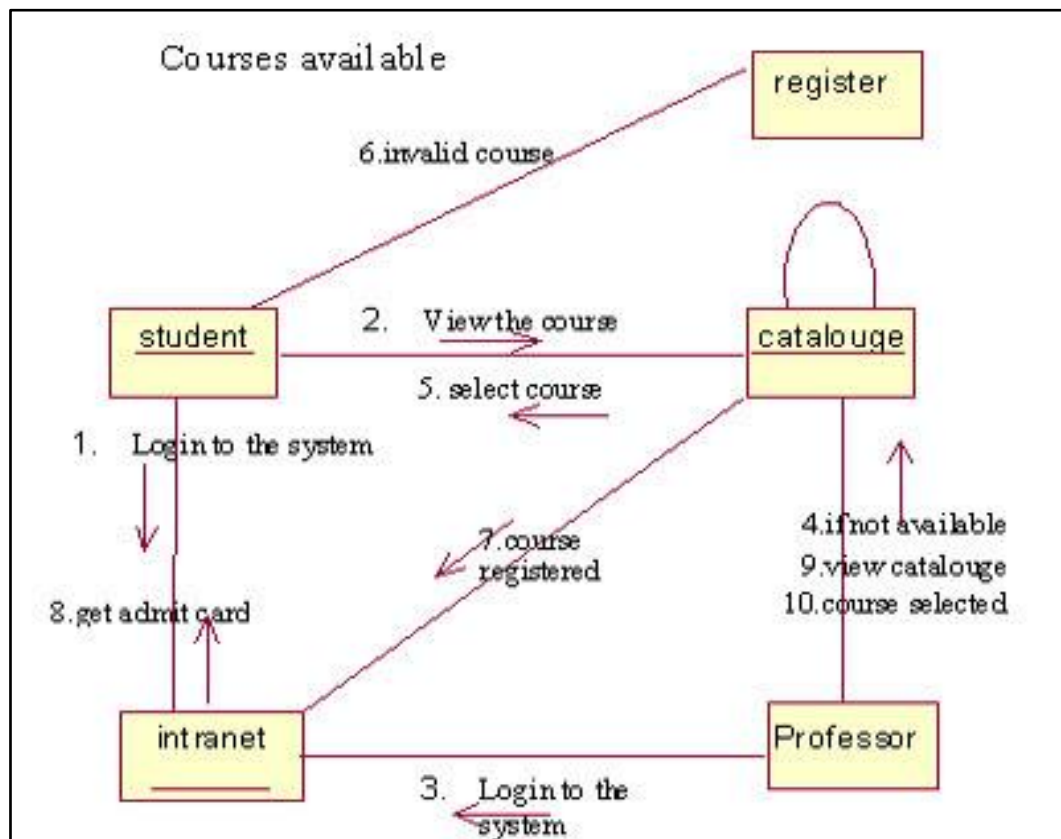
A Collaboration Diagram also known as Communication Diagram offers the same information as a Sequence Diagram, but while a Sequence Diagram emphasizes the time and order of events, a Collaboration Diagram emphasizes the messages exchanged between objects in an application. Collaboration Diagrams offer the broader perspective within a process.

#### **Benefits of Collaboration Diagram:**

- Model the logic of a sophisticated procedure, function, or operation.
- Identify how commands are sent and received between objects or components of a process.
- Visualize the consequences of specific interactions between various components in a process.
- Plan and understand the detailed functionality of an existing or future scenario.

#### **Collaboration Diagram Symbols and Notation:**

- **Rectangles** represent objects that make up the application.
- **Lines** between class instances represent the relationships between different parts of the application.
- **Arrows** represent the messages that are sent between objects.
- **Numbering** lets you know in what order the messages are sent and how many messages are required to finish a process.

Collaboration Diagram for University Management System :

## EXPERIMENT 5

### Aim :

Draw Class Diagram for the problem statement.

### Theory :

#### **What is Class Diagram?**

Class Diagrams are a type of structure diagram because they describe what must be present in the system being modelled. The various components in a class diagram can represent the classes that will actually be programmed, the main objects, or the interactions between classes and objects.

The class shape itself consists of a rectangle with three rows. The top row contains the name of the class, the middle row contains the attributes of the class, and the bottom section expresses the methods or operations that the class may use. Classes and subclasses are grouped together to show the static relationship between each object.

#### **Benefits of Class Diagram:**

- Illustrate data models for information systems, no matter how simple or complex.
- Better understand the general overview of the schematics of an application.
- Visually express any specific needs of a system and disseminate that information throughout the business.
- Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- Provide an implementation – independent description of types used in a system that are later passed between its components.

#### **What are the Components of Class Diagram?**

- **Upper Section:** Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- **Middle Section:** Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- **Bottom Section:** Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

#### **Member Access Modifiers:**

- Public ( + )
- Private ( - )
- Protected ( # )
- Package ( ~ )
- Derived ( / )
- Static ( underlined )

#### **Member Scopes:**

There are two scopes for members: Classifiers and Instances. Classifiers are static members while Instances are the specific instances of the class.

### **Additional Class Diagram Components:**

- **Classes:** A template for creating objects and implementing behaviour in a system. A class represents an object or a set of objects that share a common structure and behaviour. They're represented by a rectangle that includes rows of the class name, its attributes, and its operations.
  - **Name:** The first row in a class shape.
  - **Attributes:** The second row in a class shape. Each attribute of the class is displayed on a separate line.
  - **Methods:** The third row in a class shape. Also known as operations, methods are displayed in list format with each operation on its own line.
- **Signals:** Symbols that represent one-way, asynchronous communications between active objects.
- **Data Types:** Classifiers that define data values. Data types can model both primitive types and enumerations.
- **Packages:** Shapes designed to organize related classifiers in a diagram. They are symbolized with a large tabbed rectangle shape.
- **Interfaces:** A collection of operation signatures and/or attribute definitions that define a cohesive set of behaviours.
- **Enumerations:** Representations of user – defined data types. An enumeration includes groups of identifiers that represent values of the enumeration.
- **Objects:** Instances of a class or classes. Objects can be added to a class diagram to represent either concrete or prototypical instances.
- **Artefacts:** Model elements that represent the concrete entities in a software system, such as documents, databases, executable files, software components, etc.

### **Interactions:**

- **Inheritance:** The process of a child or sub – class taking on the functionality of a parent or superclass, also known as generalization. It's symbolized with a straight connected line with a closed arrowhead pointing towards the superclass.
- **Bidirectional Association:** The default relationship between two classes. Both classes are aware of each other and their relationship with the other. This association is represented by a straight line between two classes.
- **Unidirectional Association:** A slightly less common relationship between two classes. One class is aware of the other and interacts with it. Unidirectional association is modelled with a straight connecting line that points an open arrowhead from the knowing class to the known class.

Class Diagram for University Management System :