Week-1:

Aim:

Familiarization with Rational Rose / star UML / umbrella / visual paradigm / Microsoft visio Environment.

1. Rational Rose:

Rational Rose is a legacy software design tool from IBM used for visual modeling and component construction of Enterprise-level software applications. It supports various languages like UML (unified modeling language) and helps in designing and maintaining object-oriented software.

Key features:

* Supports multiple programming languages.
* Provides round-trip Engineering.
* Integrates with other IBM rational tools.

* Allows for creation of UML diagrams like use case diagrams, class diagrams, sequence diagrams, etc.

2. Star UML:

Star UML is an open-source software modeling tool that supports the creation of UML diagrams. It is designed to be lightweight and fast, providing a robust environment for creating detailed and extensive UML models.

Key features:

* Supports multiple UML 2x diagrams.
* Extensible with plugins.
* Real-time collaboration through cloud services.
* Code generation and reverse engineering capabilities.
* User-friendly interface with drag and drop features.

Umbrello:

Umbrello is a UML modelling tool that is part of the KDE software compilation. It allows users to create diagrams

for software design and other modeling needs.

key features:

* Supports all standard UML diagrams.
* provides code generation for various programming languages.
* open-source and cross-platform.
* Simple and intuitive interface.
* Supports XMI [XMI metadata Interchange] for model interchange.

4. visual paradigm:

Visual paradigm is a sophisticated modeling tool that supports UML and other standards like BPMN [Business process model and notation]. It is widely used for both software development and business process modeling.

key features:

* Comprehensive support for UML, BPMN, ERD and other diagrams.
* Agile and scrum support.

* Integration with various IDE's (Integrated development Environments)
* Collaboration tools for team-based modeling.
* Visual modeling with drag and drop and automated layout features.

5. Microsoft visio:

Microsoft visio is a versatile diagramming tool from microsoft that supports a wide range of diagram types including flowcharts, network diagrams, organisational charts, and UML diagrams.

Key features:

* Extensive template library for various types of diagrams.
* Integration with microsoft office suite.
* Real-time collaboration and sharing features.
* Easy to use with drag and drop functionality.
* supports both simple and complex diagramming needs.

Comparison and use cases:-

Rational rose:- Best for large Enterprises with complex systems needing integration with other IBM tools.

Star UML:- Suitable for developers looking for an open-source, lightweight, and fast UML modeling tool.

Umbrello:- Ideal for KDE users and those looking for an open source alternative with basic UML modeling needs.

Visual Paradigm:- Great for comprehensive modeling needs, including software development and business processes with Extensive collaboration features.

Microsoft visio:- perfect for users who need a versatile diagramming tool that integrates well with microsoft office products and can be used for a variety of diagram types beyond just UML.

Week-2

Aim:

Understanding different views that the UML aims to visualize through different modelling diagrams.

user's view: use case diagram, structural view: class diagram, object diagram, Behavioral view: sequence diagram, collaboration diagram, state chart diagram, Activity diagram, Environmental view: Deployment diagram, Implementation view: component diagram.

Use case diagram:

*usecase diagrams describe the functionality of a system and users of the system.

*usecase diagrams are considered for high level requirement analysis of a system. so when requirements of a system are analysed the functionalities are captured in usecases.

*so we can say that uses cases are nothing but the system

functionalities written in an organised manner.

* The actors can be human user, some internal application or may be some External applications.

* So in a brief when we are planning to draw an use-case diagram we should have the following items identified.

* Functionalities to be represented as an usecase.

- Actors,
- Relationship among the usecases and actors.

* Use case diagrams are drawn to capture the functional requirements of a system.

* These diagrams contain the following Elements

Actors:- which represent users of a systems. Including human users and other systems.

usecase:- which represent functionality or services provided by a system to users.

Actor

Usecase

Structural view:

class diagram:

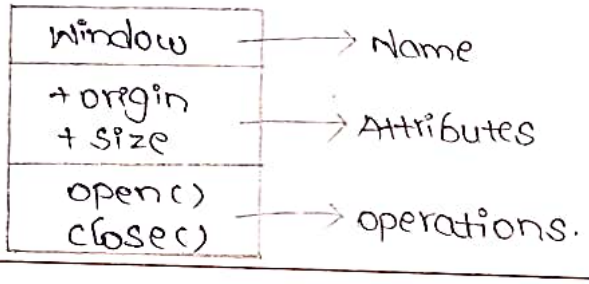
* class diagram is basically a graphical representation of static view of system and represents different aspects of application. so a collection of class diagrams represents the whole system.

* The following points should be remembered while drawing a class diagram:

- Each element and their relationship should be identified in advance.

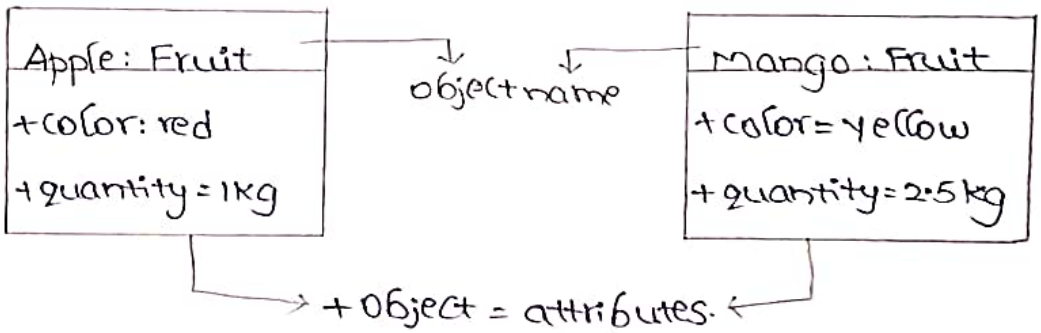
Responsibility of each class should be clearly identified.

For each class minimum no. of properties should be specified use nodes when enter required to describe some aspect of the diagram.



Object diagram:

* Object diagrams are dependent on the class diagram as they are derived from class diagram.
* It represents an instance of a class diagram.
* The objects help in protraying in a static view of an object oriented system at a specific instant.
* It helps in visualizing a particular functionality of system.



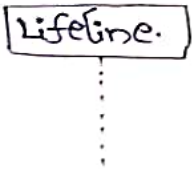
Behavioral view:

Sequence diagram:

* The sequence diagram represents the flow of messages in the system and is, also termed as an Event diagram.
* It helps in Envisioning several dynamic scenarios.
* It incorporates the iterations as well as branching.

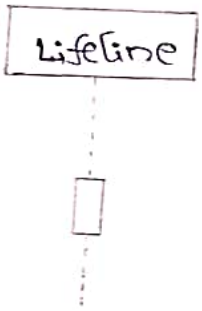
Notations of sequence diagram:

Lifeline: An individual participant in sequence diagram is represented by a lifeline.



Actor: A role played by an Entity that interacts with the subject is called as an actor

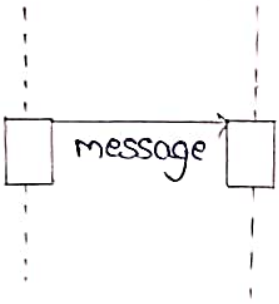
Activation: It is represented by a thin rectangle on the lifeline. It describes that timeperiod in which an operation in performed by an Element, such that top and the bottom of rectangle is associated with initiation and completion time.



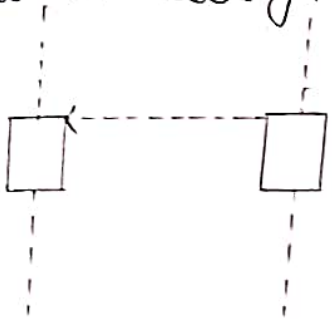
Messages:- The messages depict the interaction between the objects and are represented by arrows.

Types of messages:-

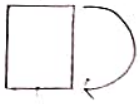
call message :-



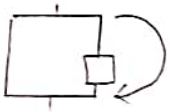
Return message:-



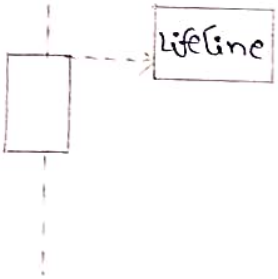
self message:-



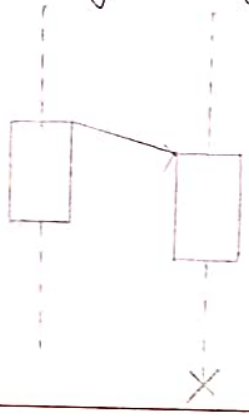
Recursive message:-



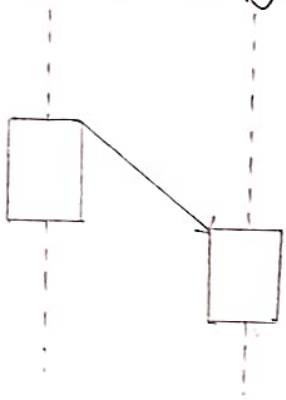
create message:-



Destroy message:-



Duration message:-

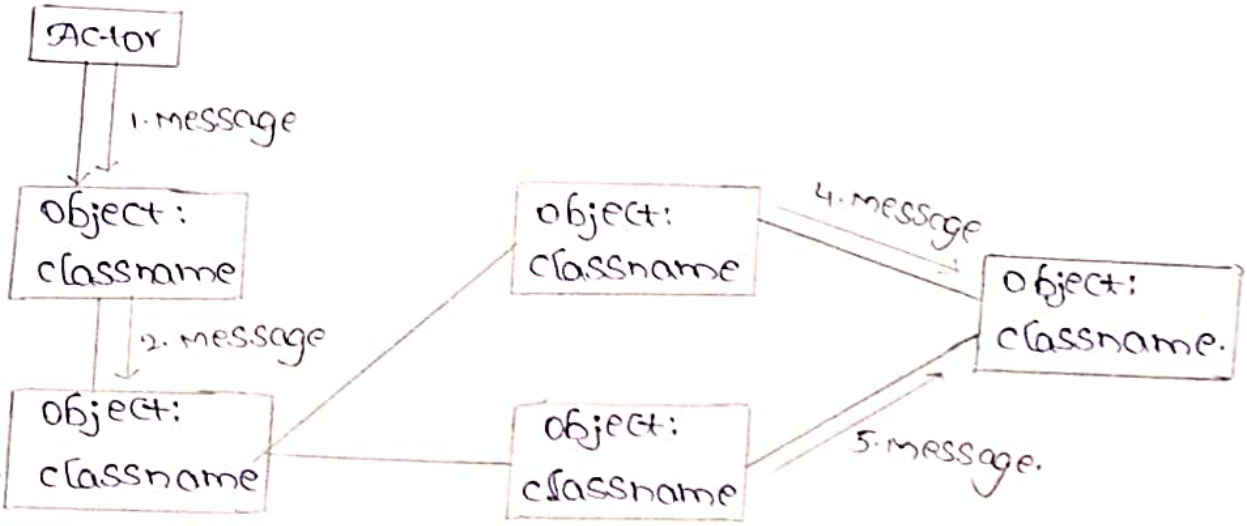


collaboration diagram:

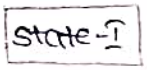
* The collaboration diagram is used to show the relationship between the objects in a system.
* The collaboration diagram, which is also known as a communication diagram.
* It is used to portray the objects architecture in system.

components of collaboration diagram:

- objects
- Actors
- Links
- message.



State chart diagram:

* The state machine diagram is also called the state chart or state Transition diagram, which shows the order of states underwent by an object within the system.
* It captures the software systems behavior.
* Types of statechart diagram:
  - Behavioral state machine.
  - Protocol state machine.
* Notations of state machine diagram:
  - Initial state 
  - State box 

  - Decision box 
  - Final box. 

Activity diagram:

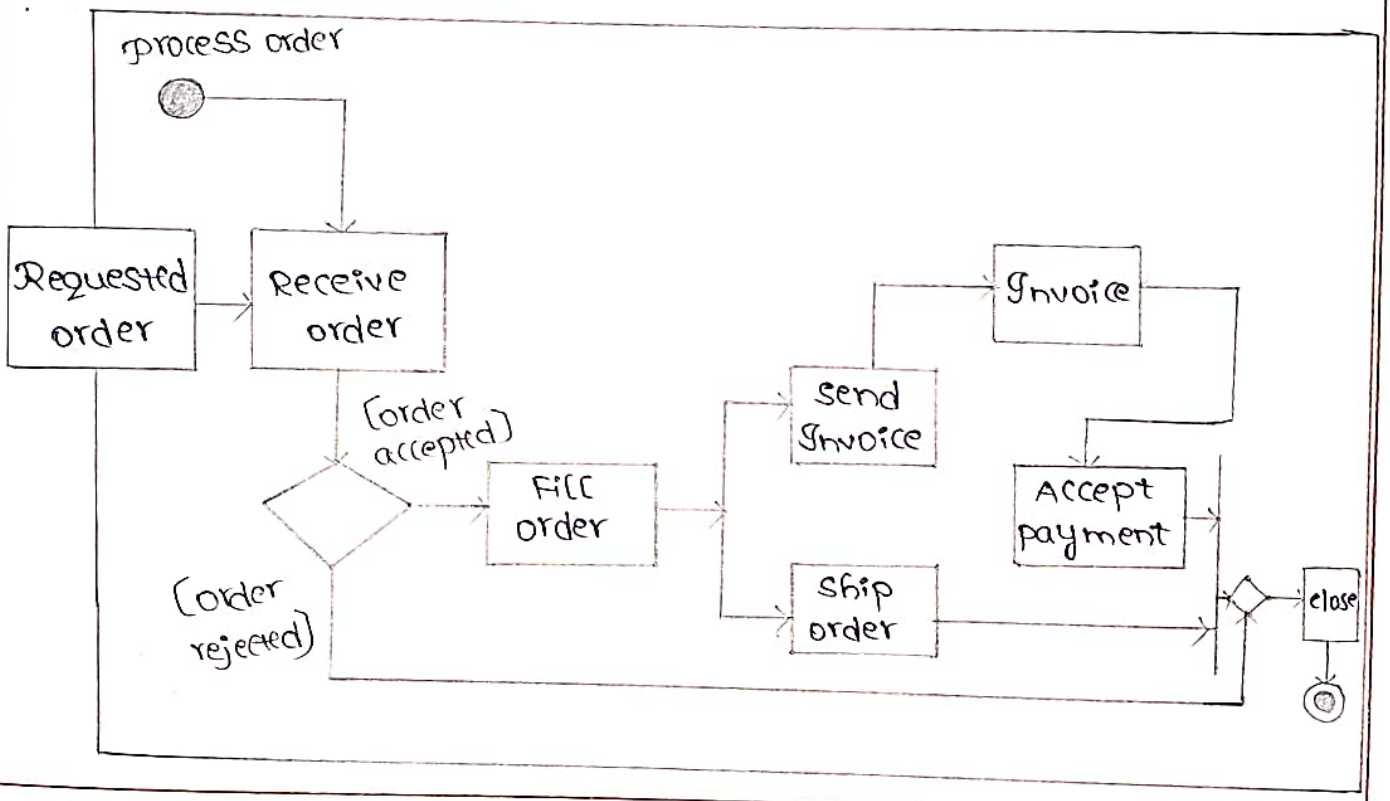
* The activity diagram is used to demonstrate the flow of control within the system rather than the implementation.
* It models the concurrent and sequential activities.

*Components of activity diagram:

- Activities
- Activity partition/swimlane.
- Forks
- Join nodes
- Pins.

*Notations of activity diagram:

- Initial state
- State box
- Decision box
- Final state.

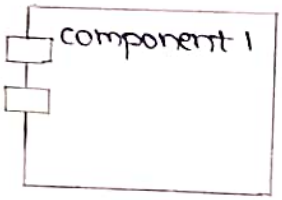


Environmental view:

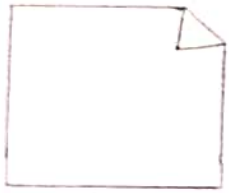
Deployment diagram:

* The deployment diagram visualizes the physical hardware on which the software will be deployed.
* It portrays the static deployment view of a system.
* It involves the nodes and their relationships.
* The deployment diagram consist of the following notations:

A component:

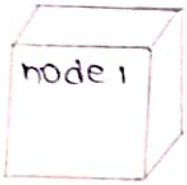


An artifact:



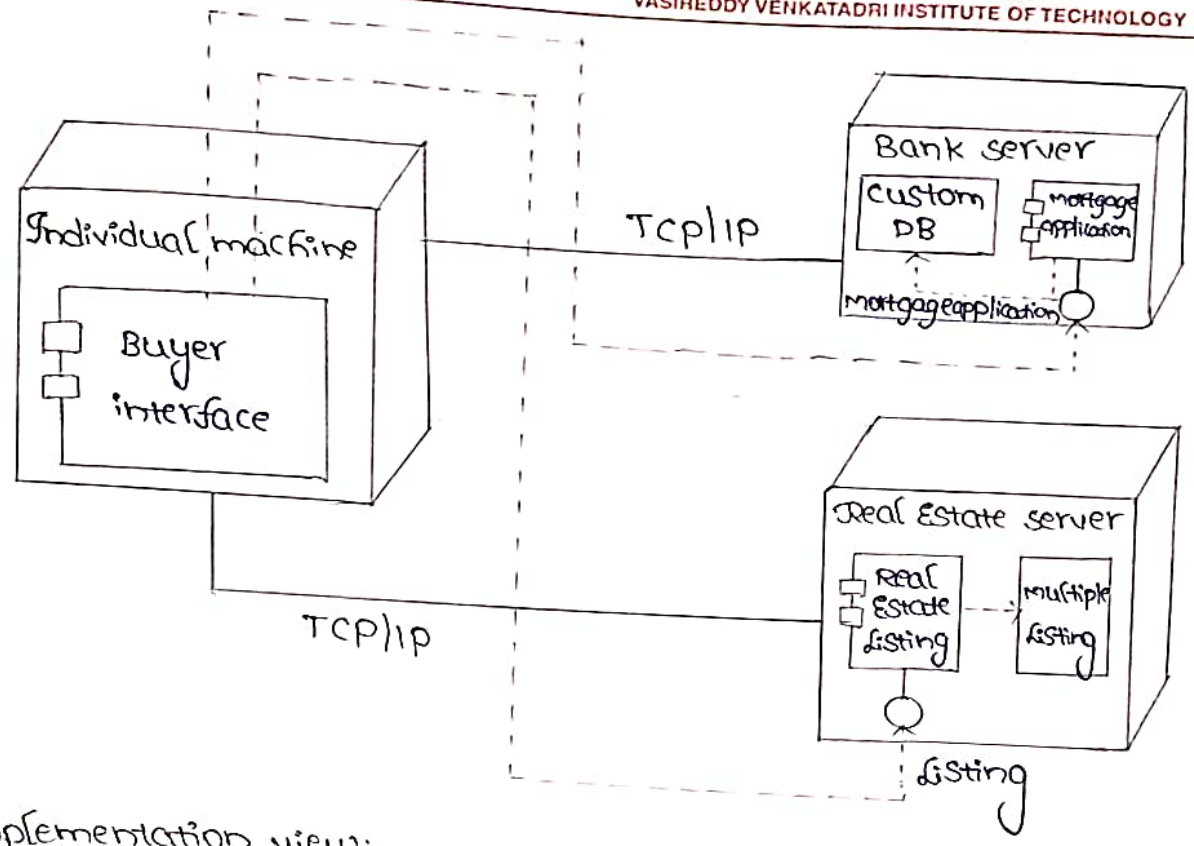
An interface:

A node:



* It plays a critical role during the administrative process, it involves the following parameters:

- High performance
- scalability
- maintainability



Implementation view:

Component diagram:

*A component diagram is used to breakdown a large object-oriented system into smaller components, so as to make them more manageable.

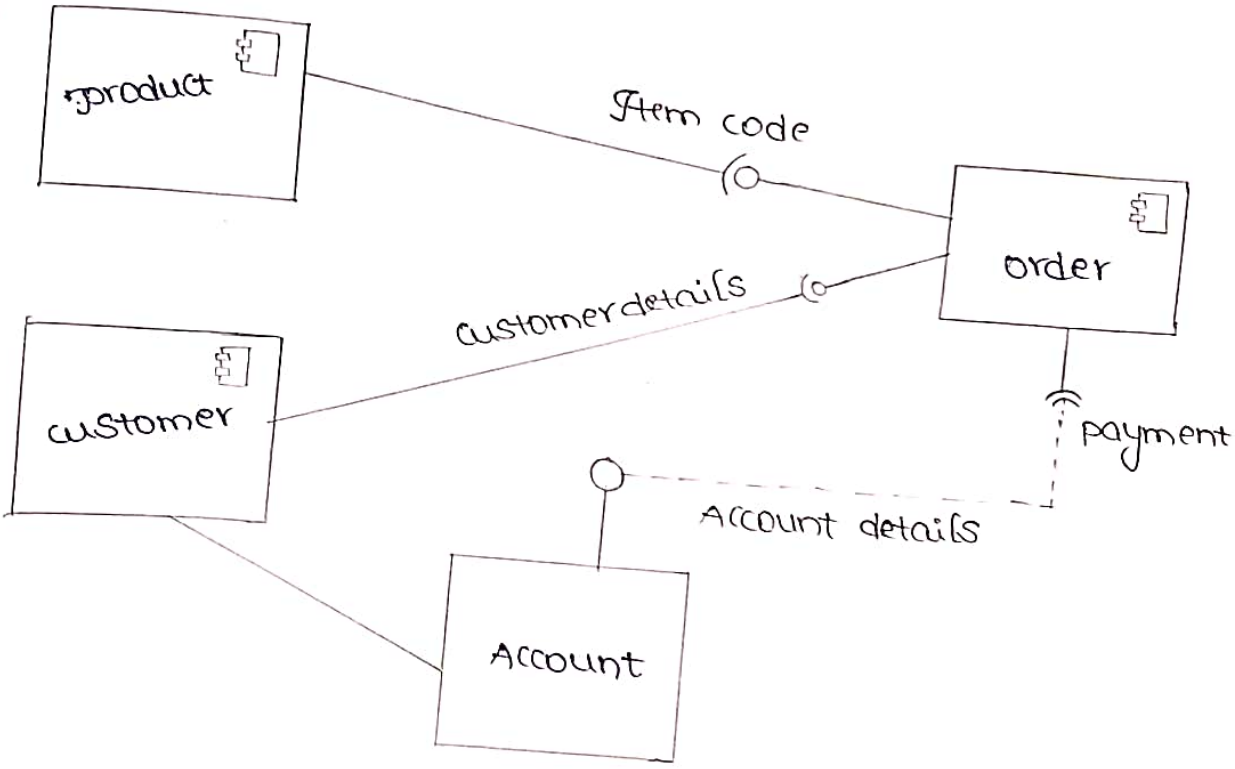
*It models the physical view of a system such as Executables, files, libraries etc. that resides with in the node.

*Notation of a component diagram:

- A component 
- A node. 

*The main purpose of component diagram are Enlisted below.

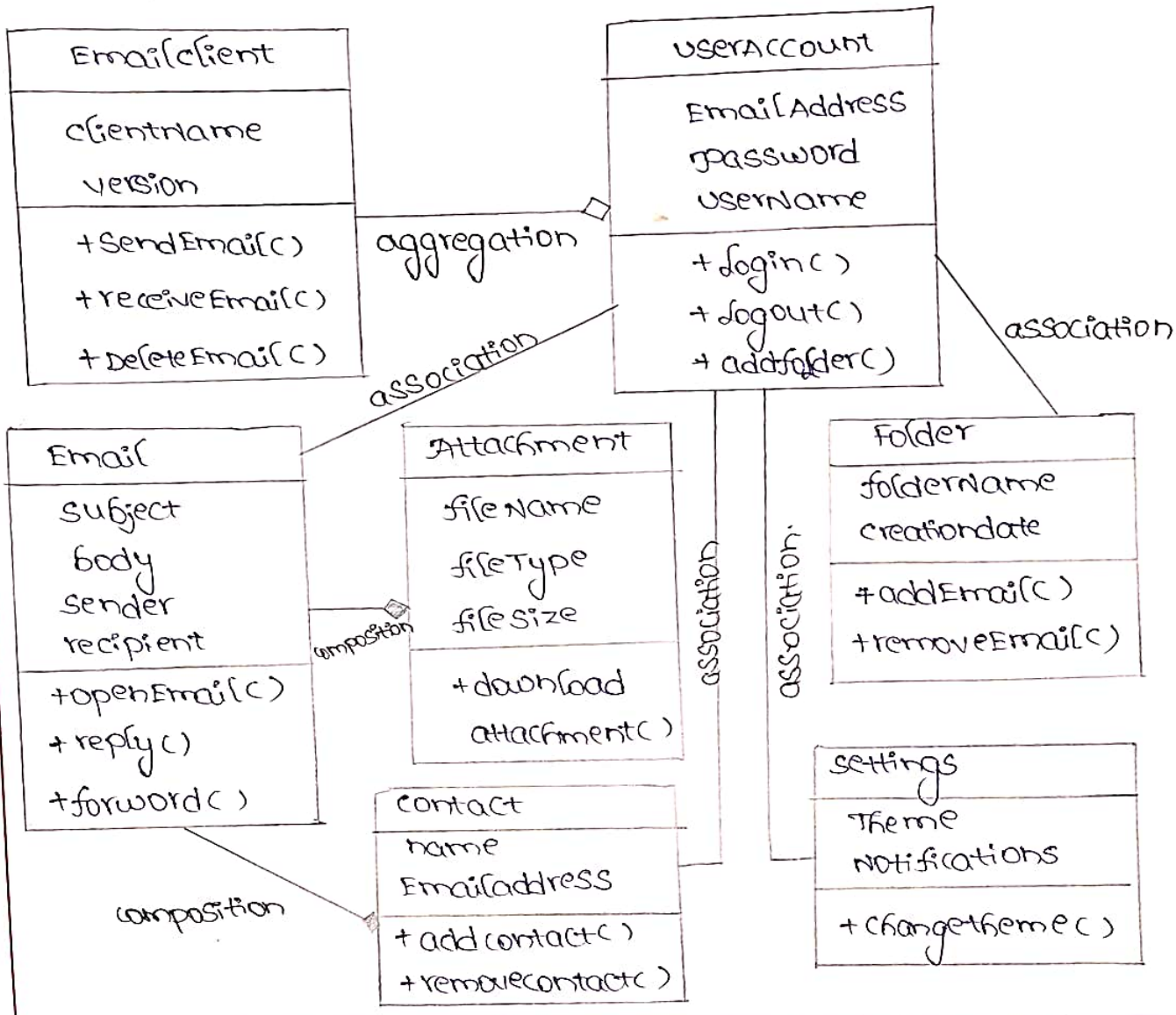
- It Envisions each component of a system
- It constructs the Executable by incorporation forward and reverse Engineering.
- It depicts the relationships and organization.



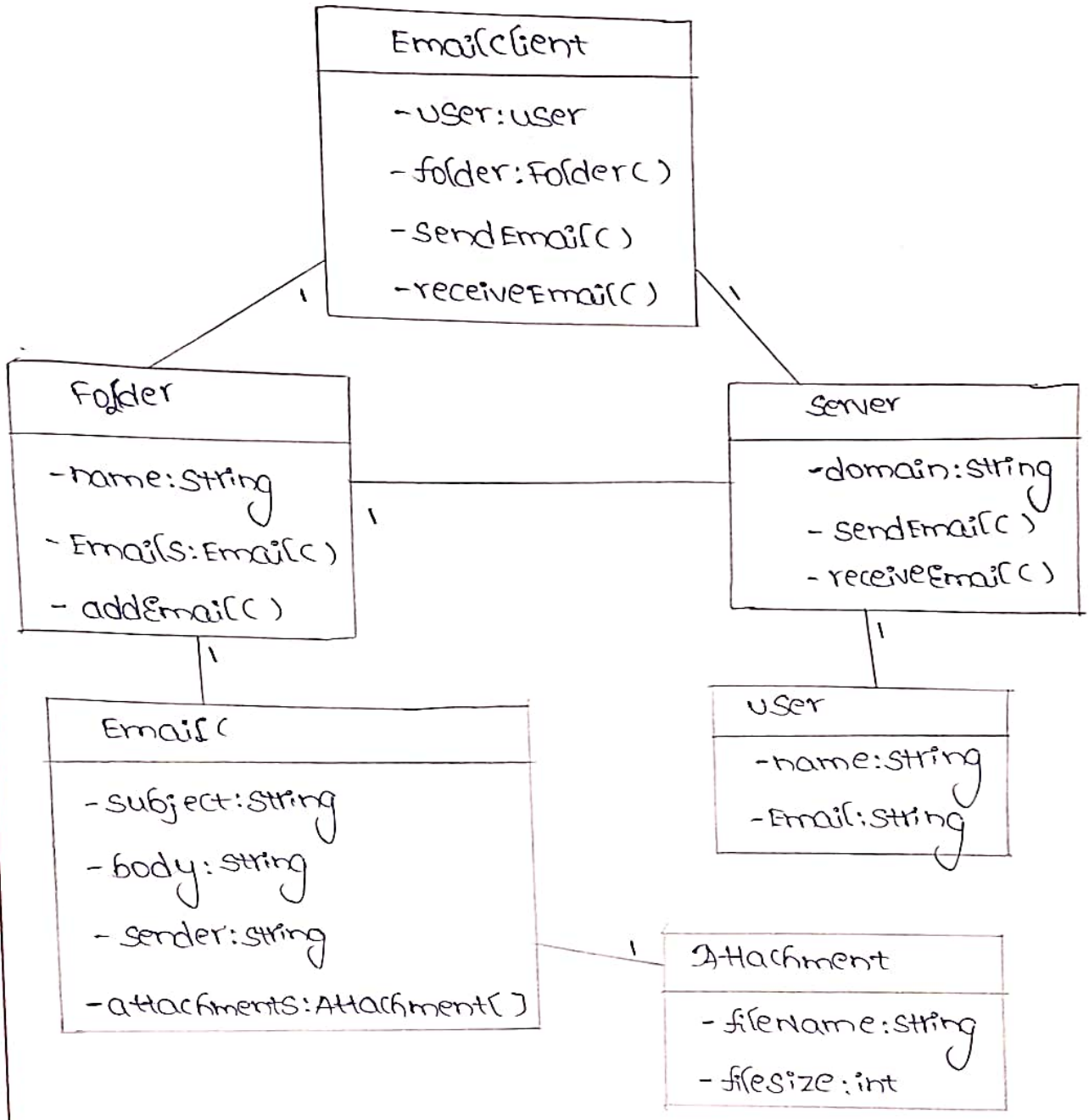
Week-3

Aim:: Create a complete UML model for E-mail client system.

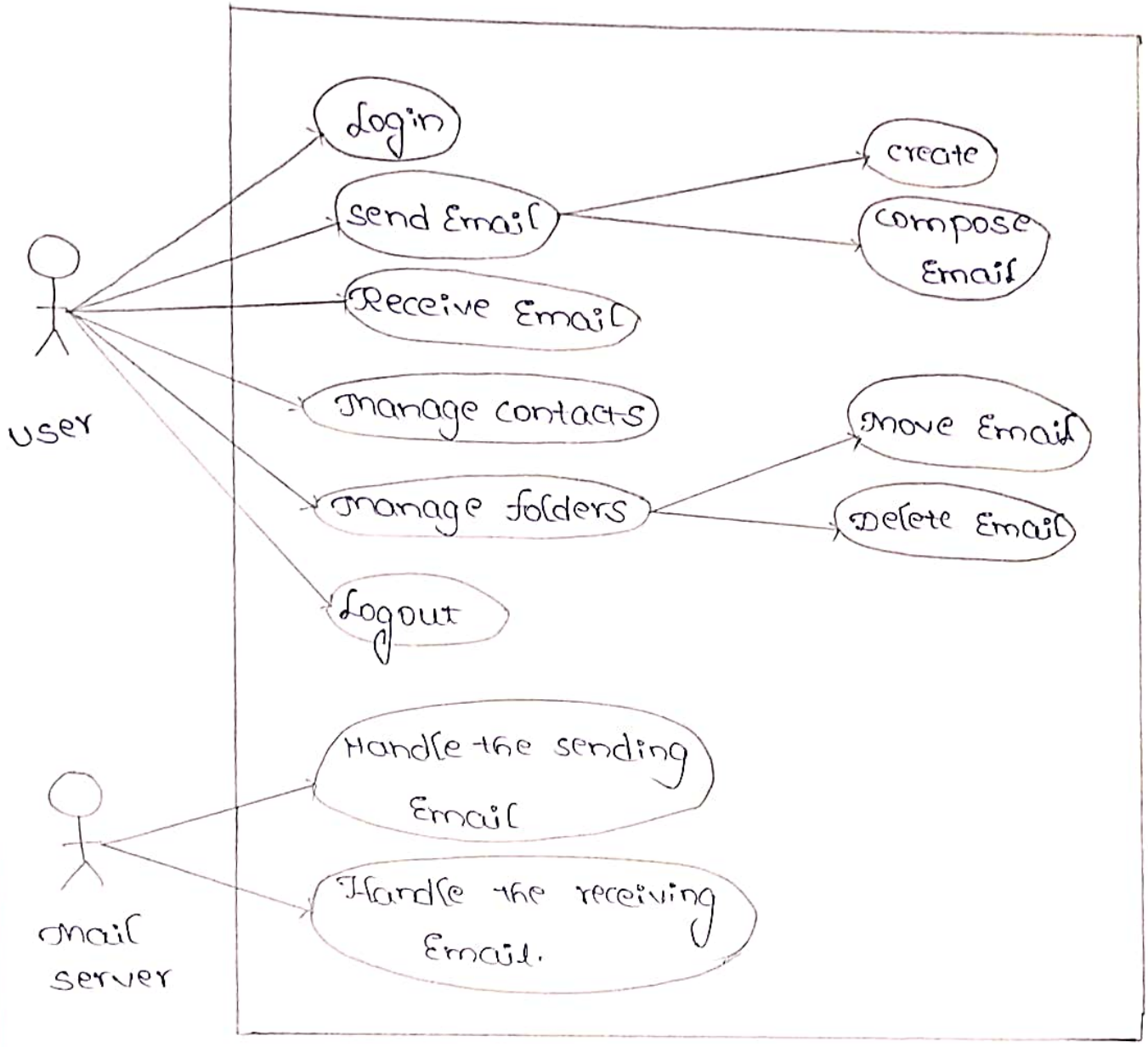
class diagram::



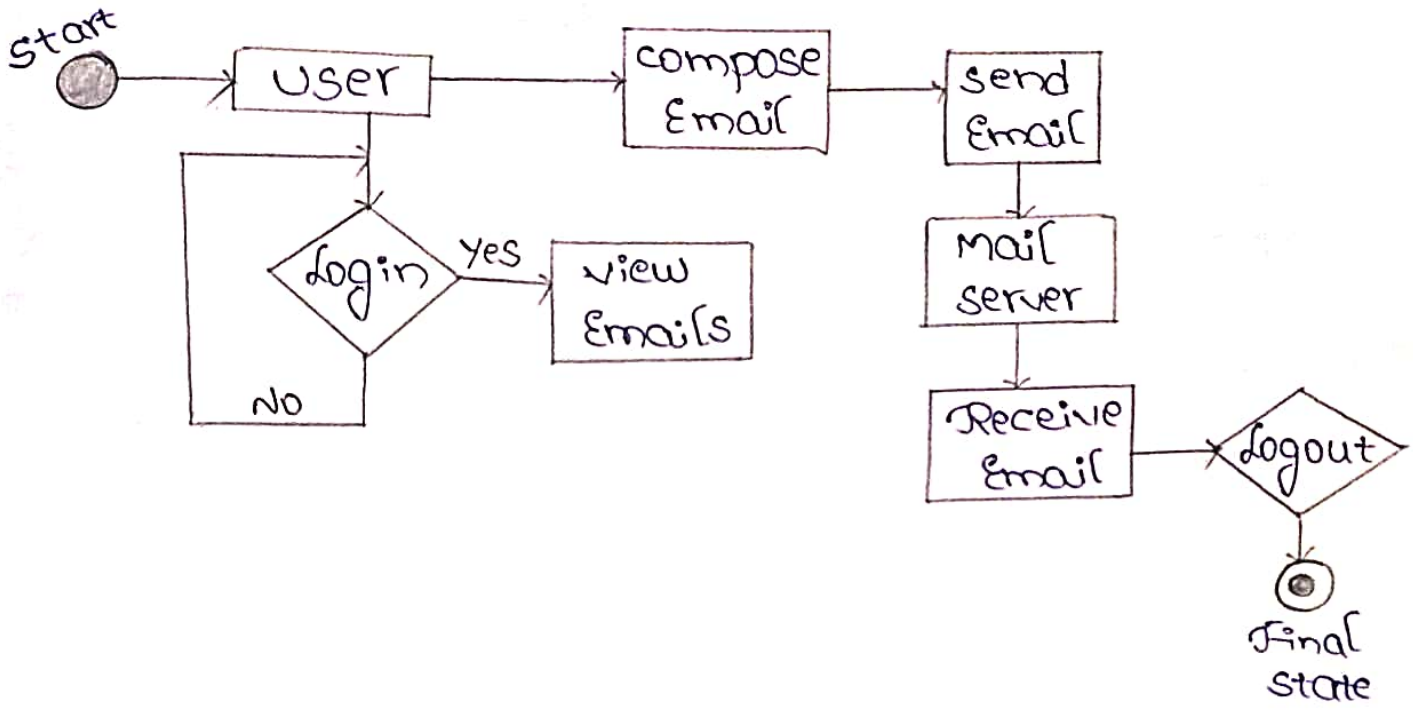
object diagram:



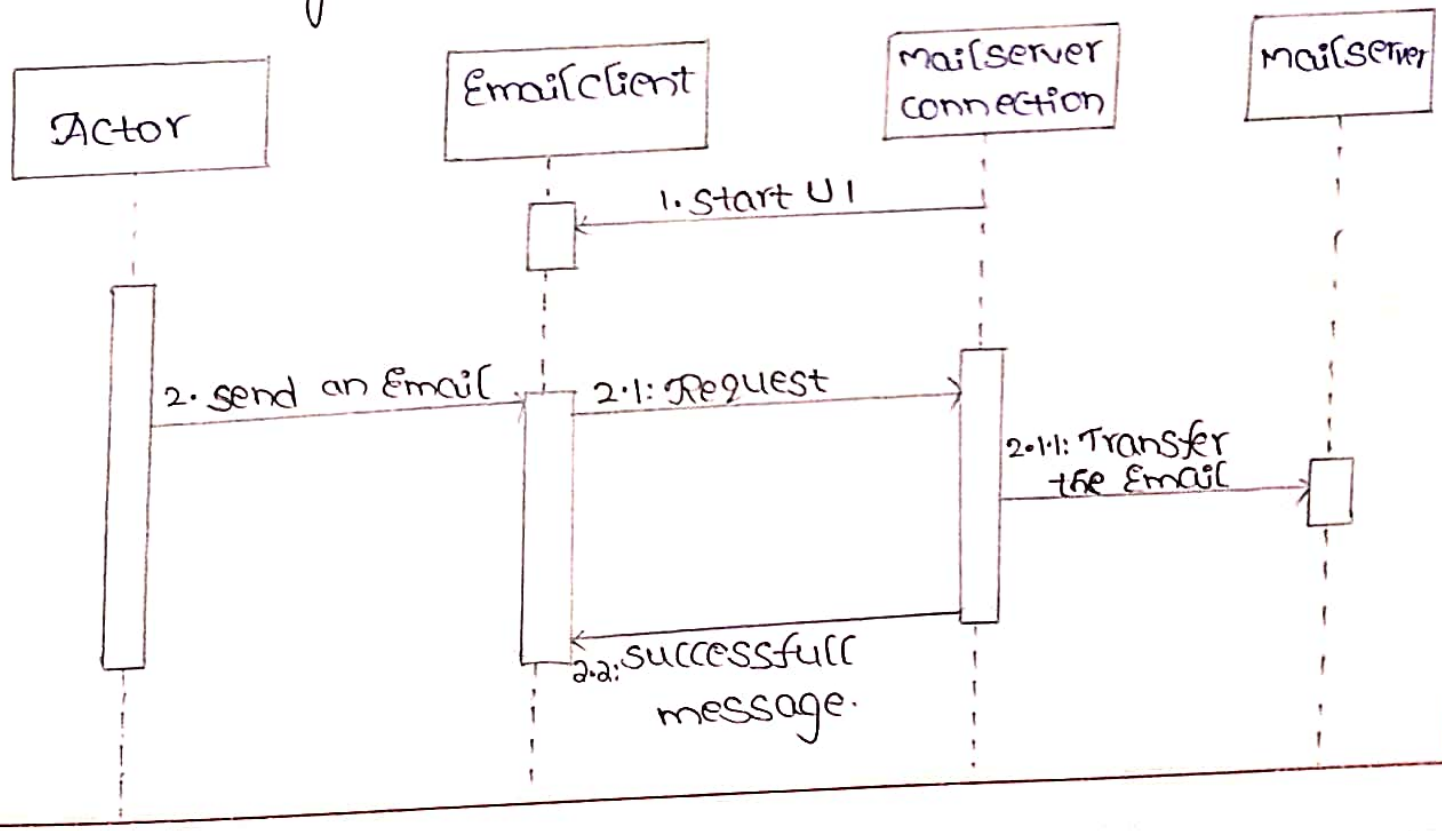
Usecase diagram:



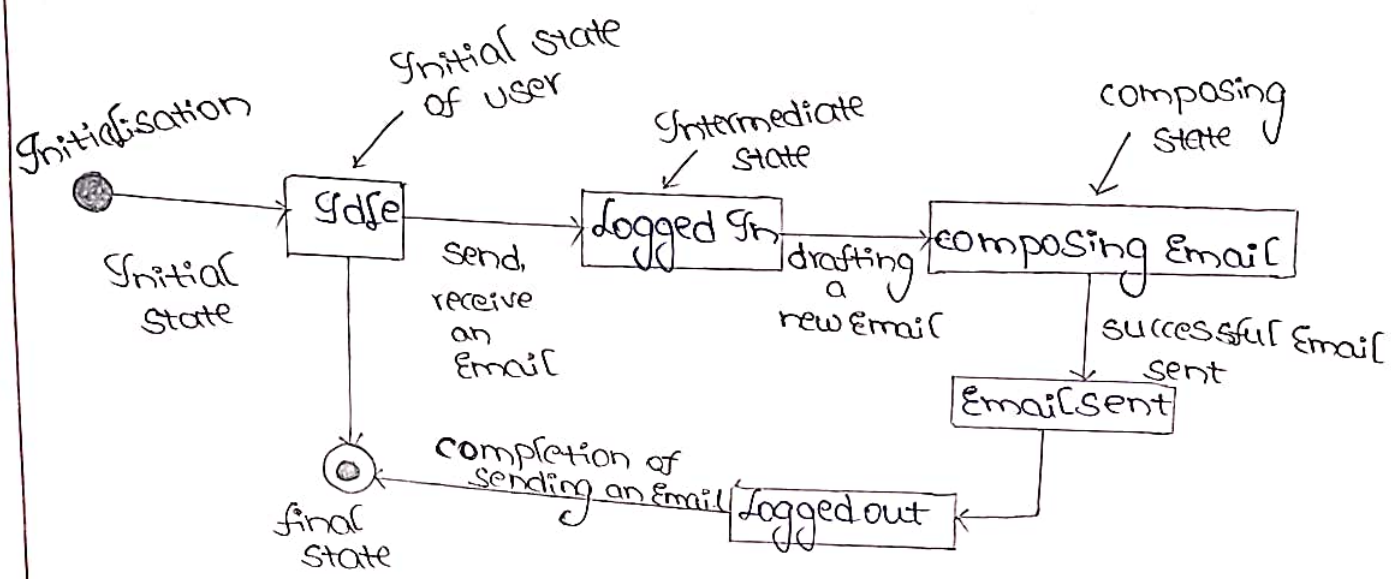
Activity diagram:



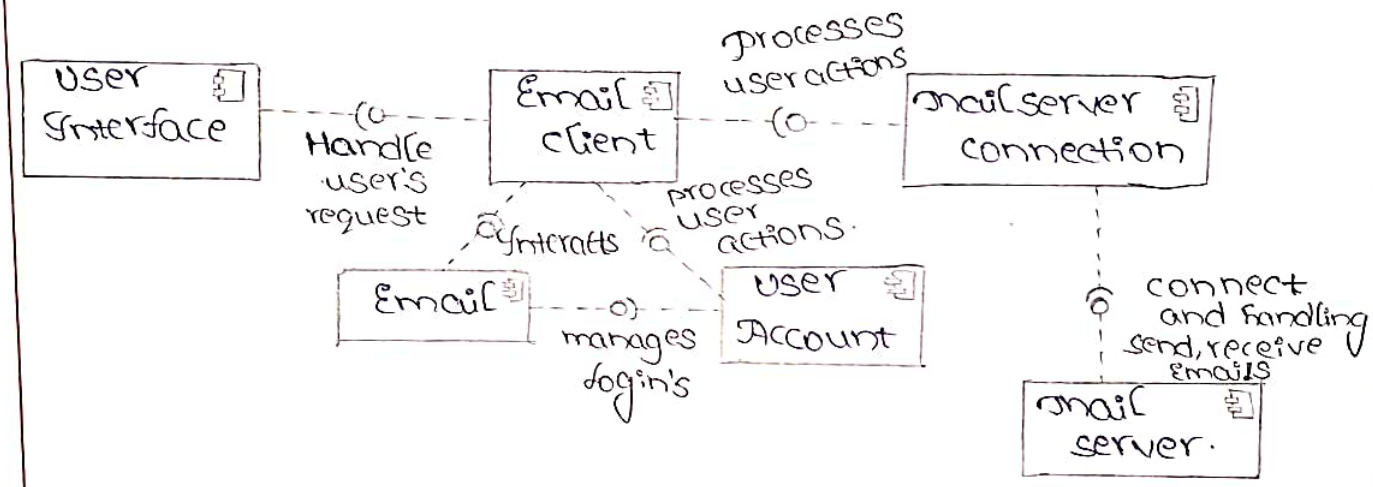
sequence diagram:



State chart diagram:



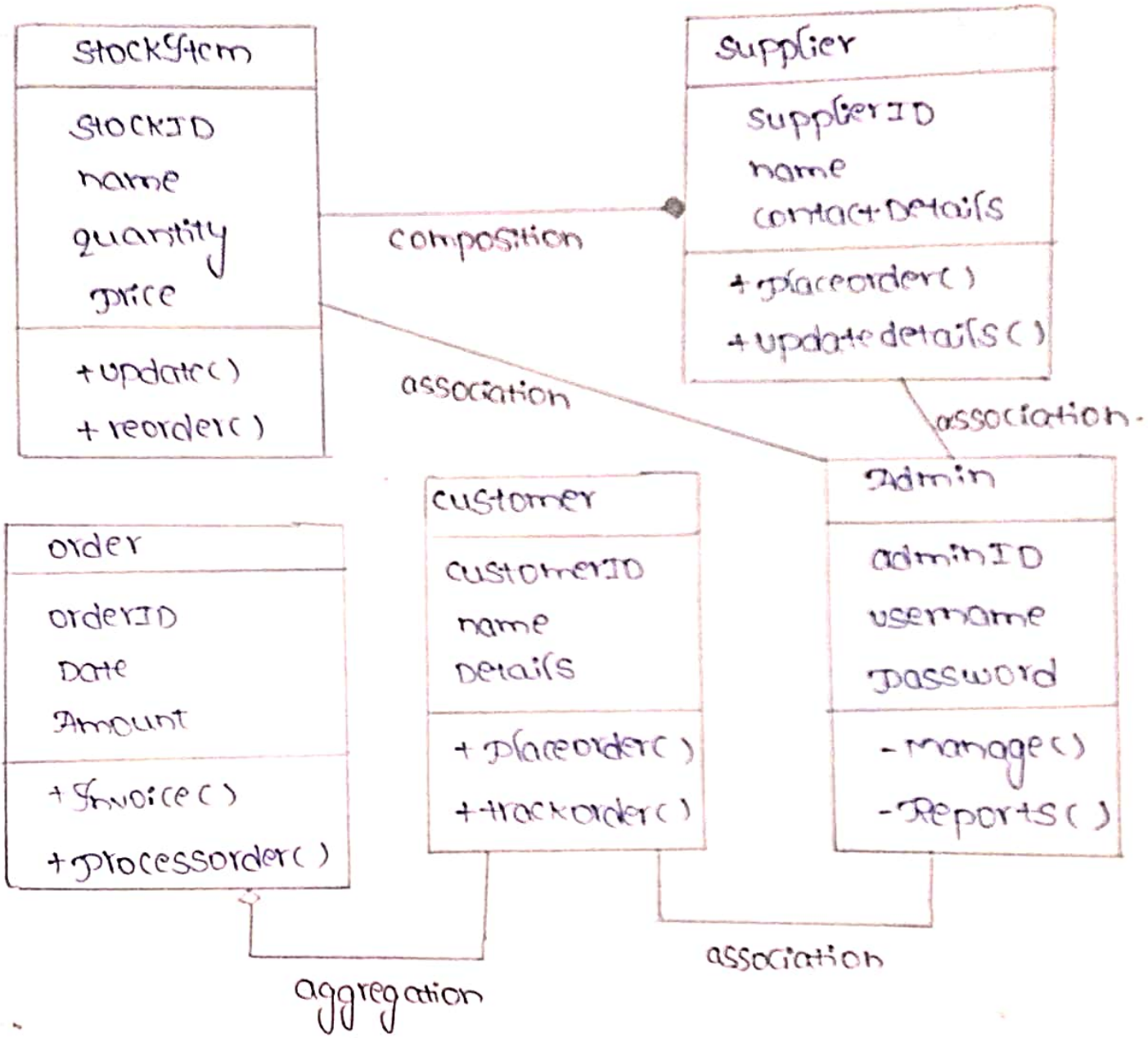
Component diagram:



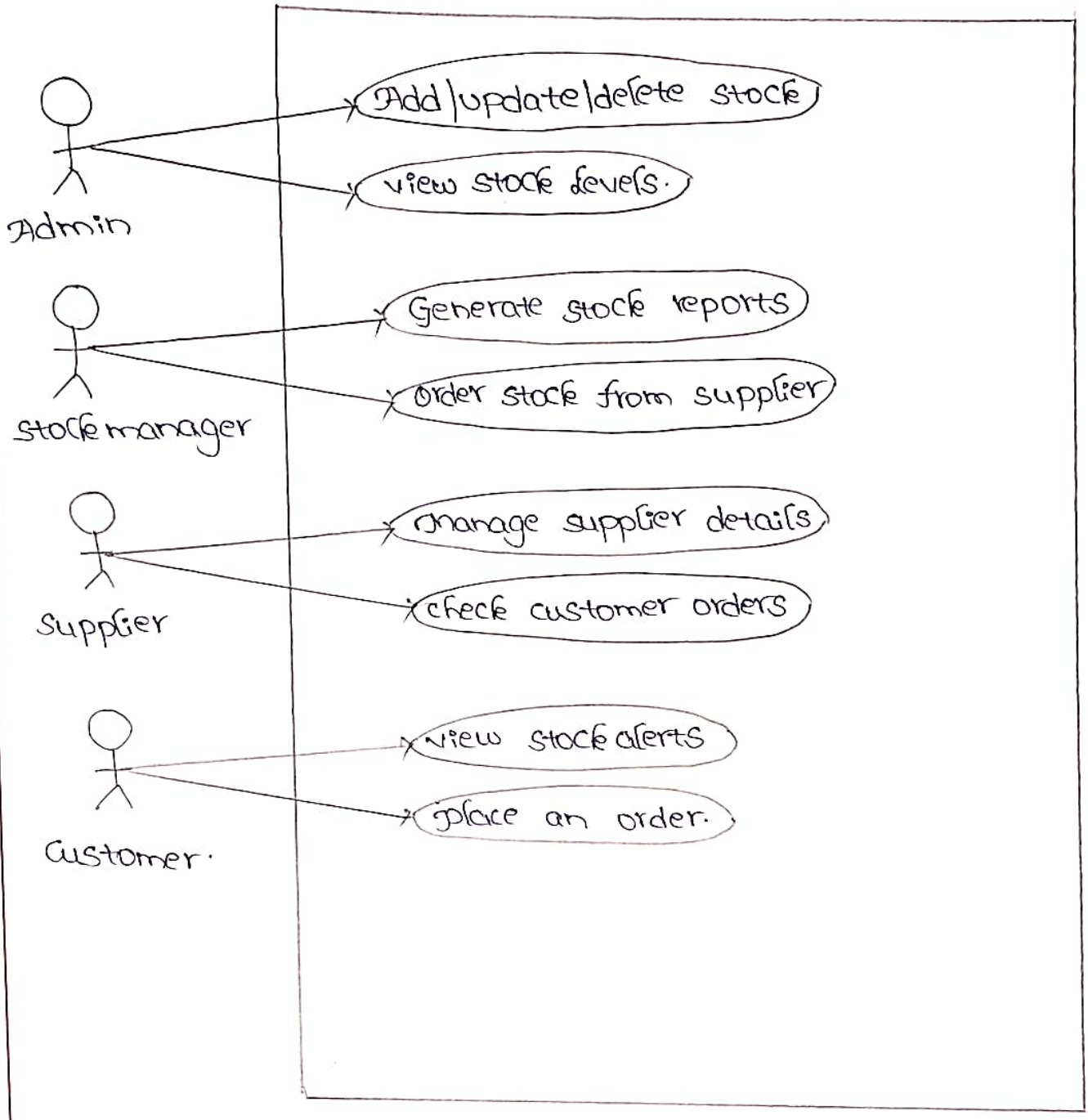
Week-4

Aim: create a complete UML model for stock maintenance system.

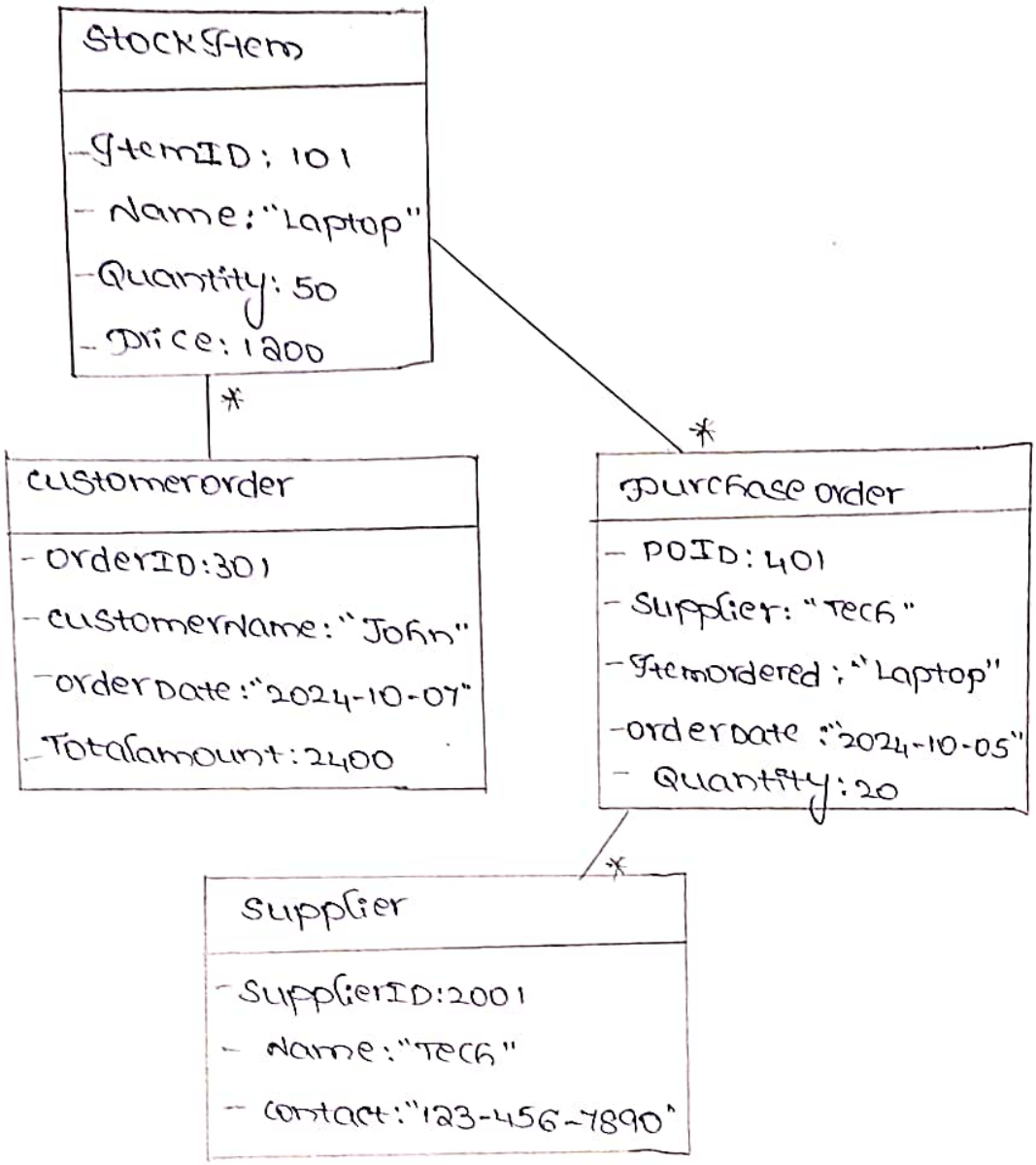
class diagram:



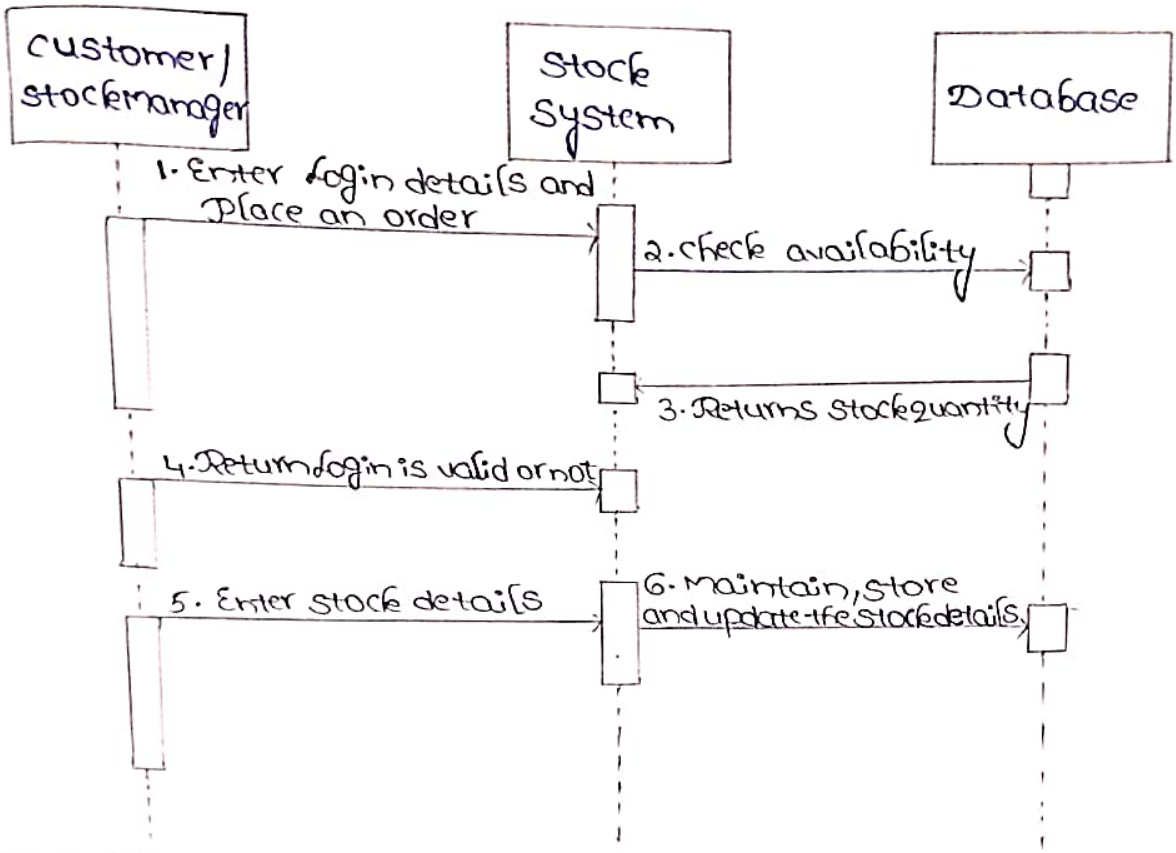
usecase diagram:.



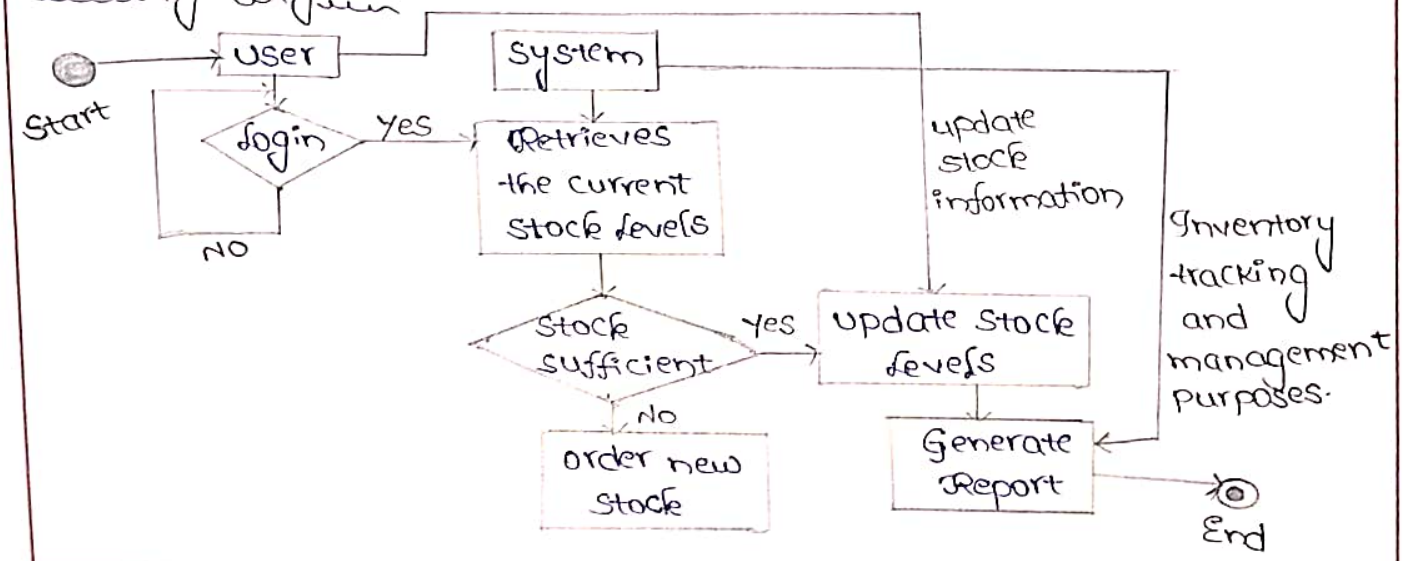
object diagram:



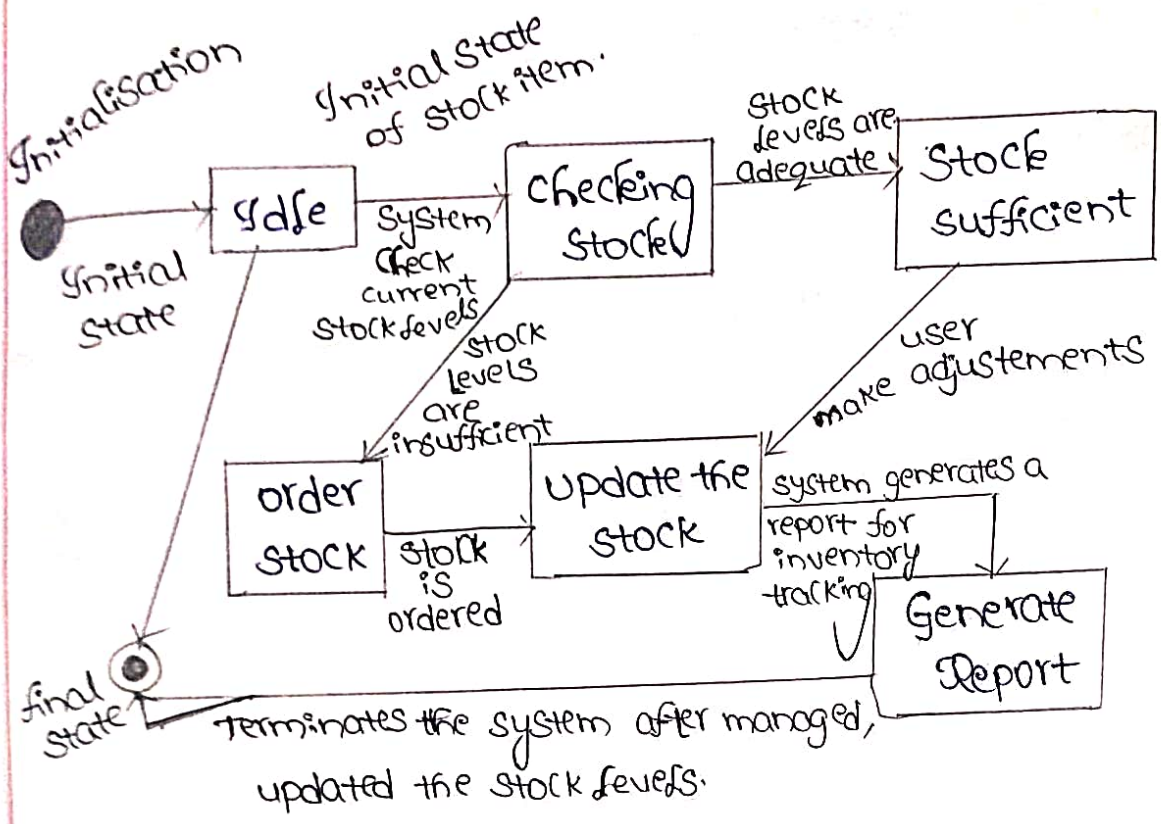
sequence diagram:



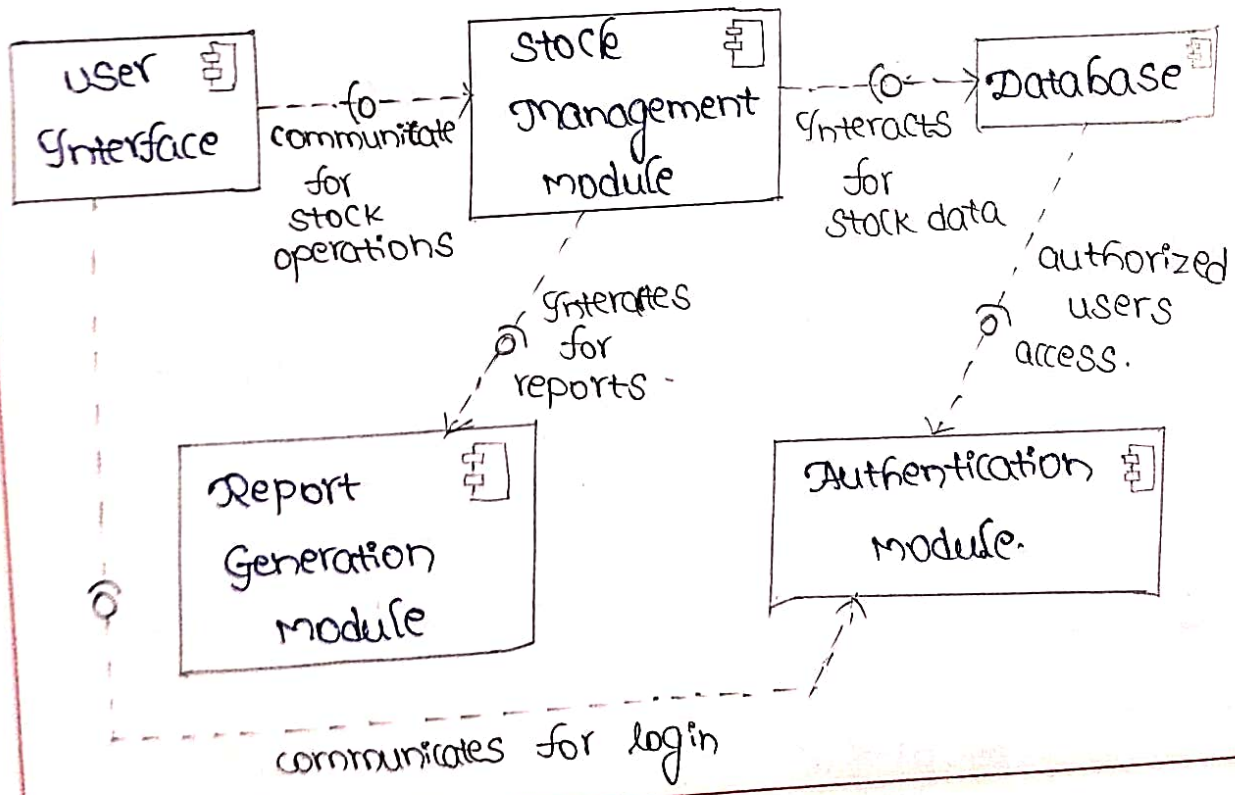
Activity diagram:



State chart diagram:



Component diagram:



Week-5

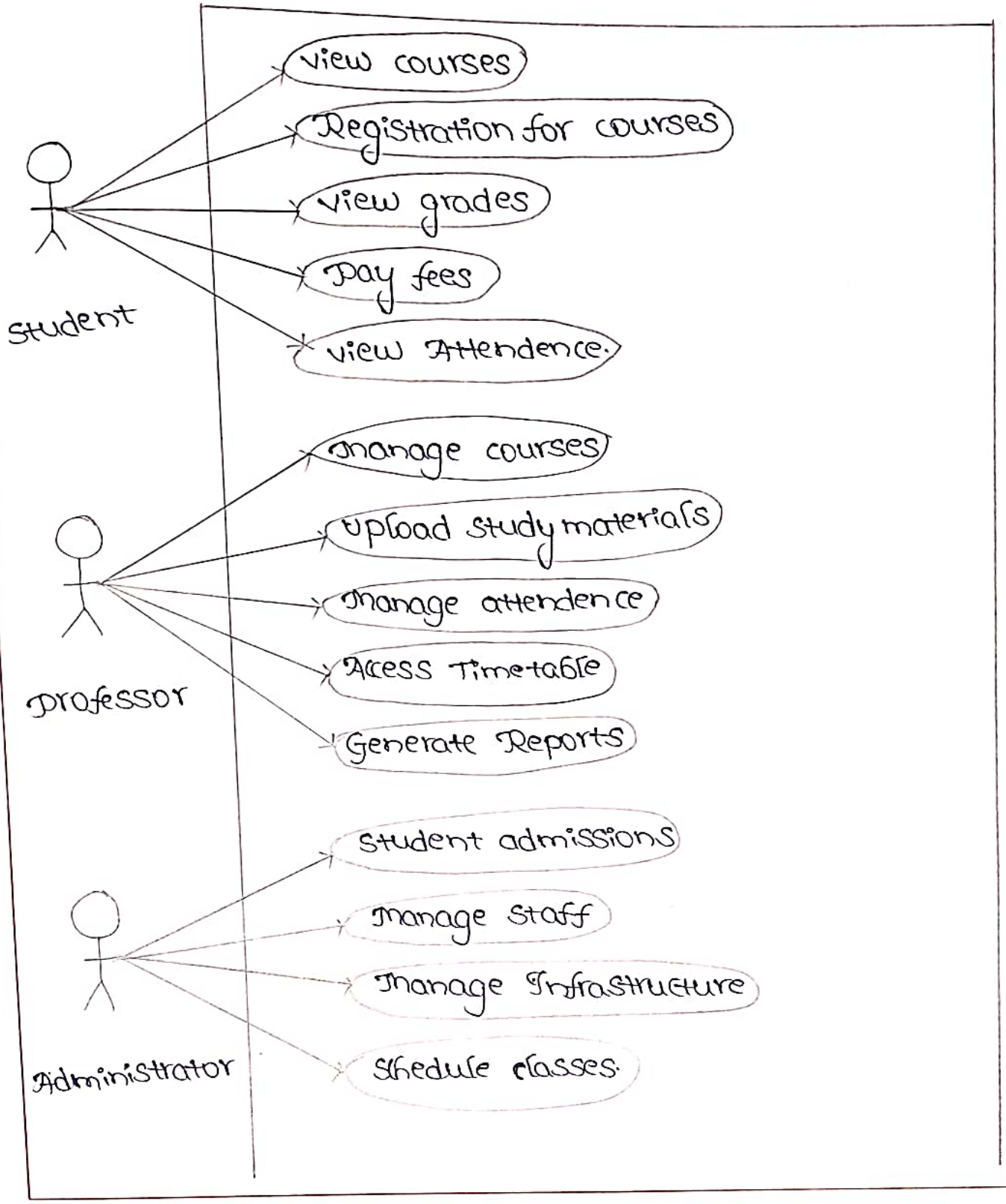
Aim:: consider the user's view of respective system: Identify the usecases, actors involved in a system and developed the usecase and sub usecase diagrams.

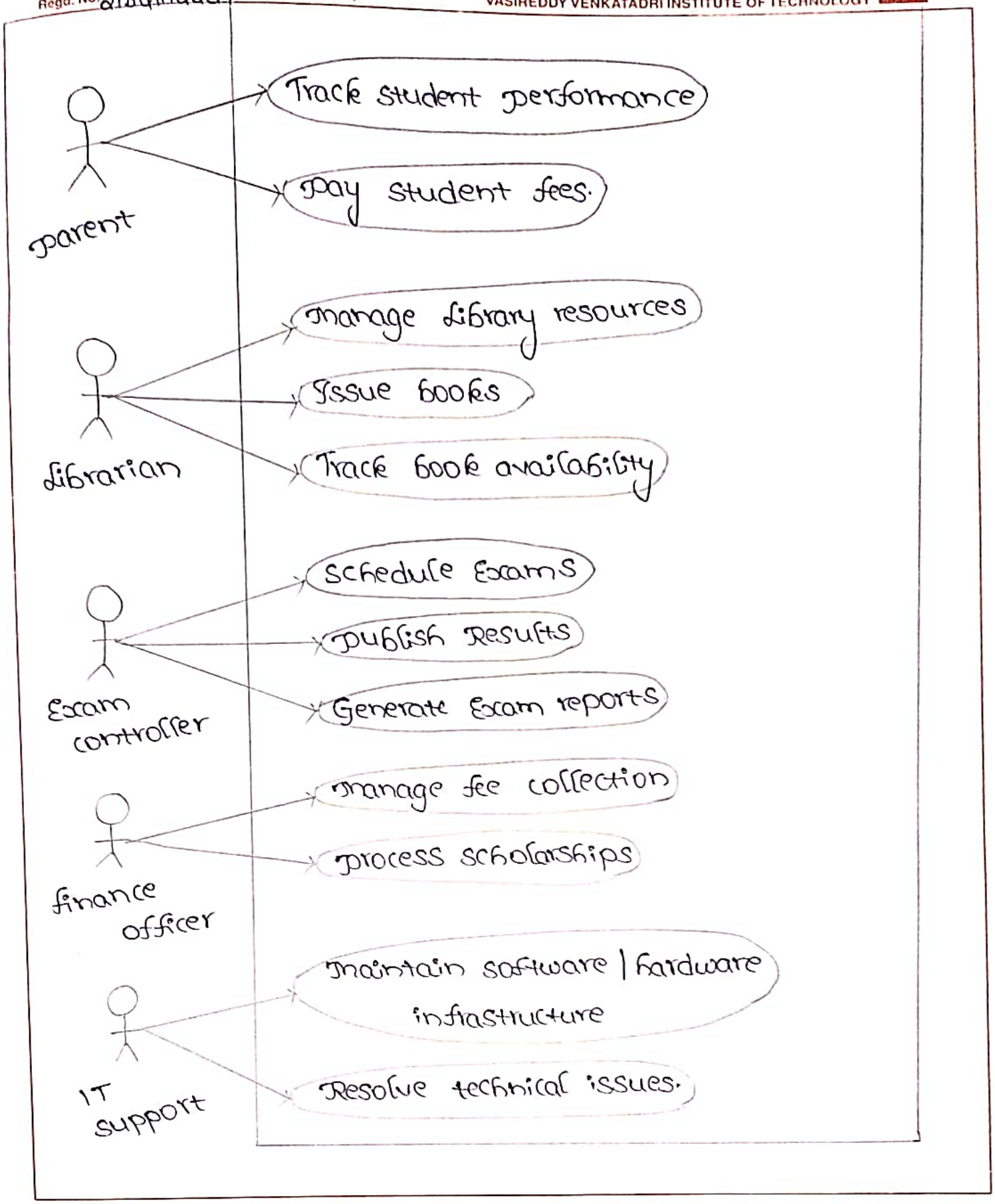
Respective system:: college management system.

Actors involved in college management system:

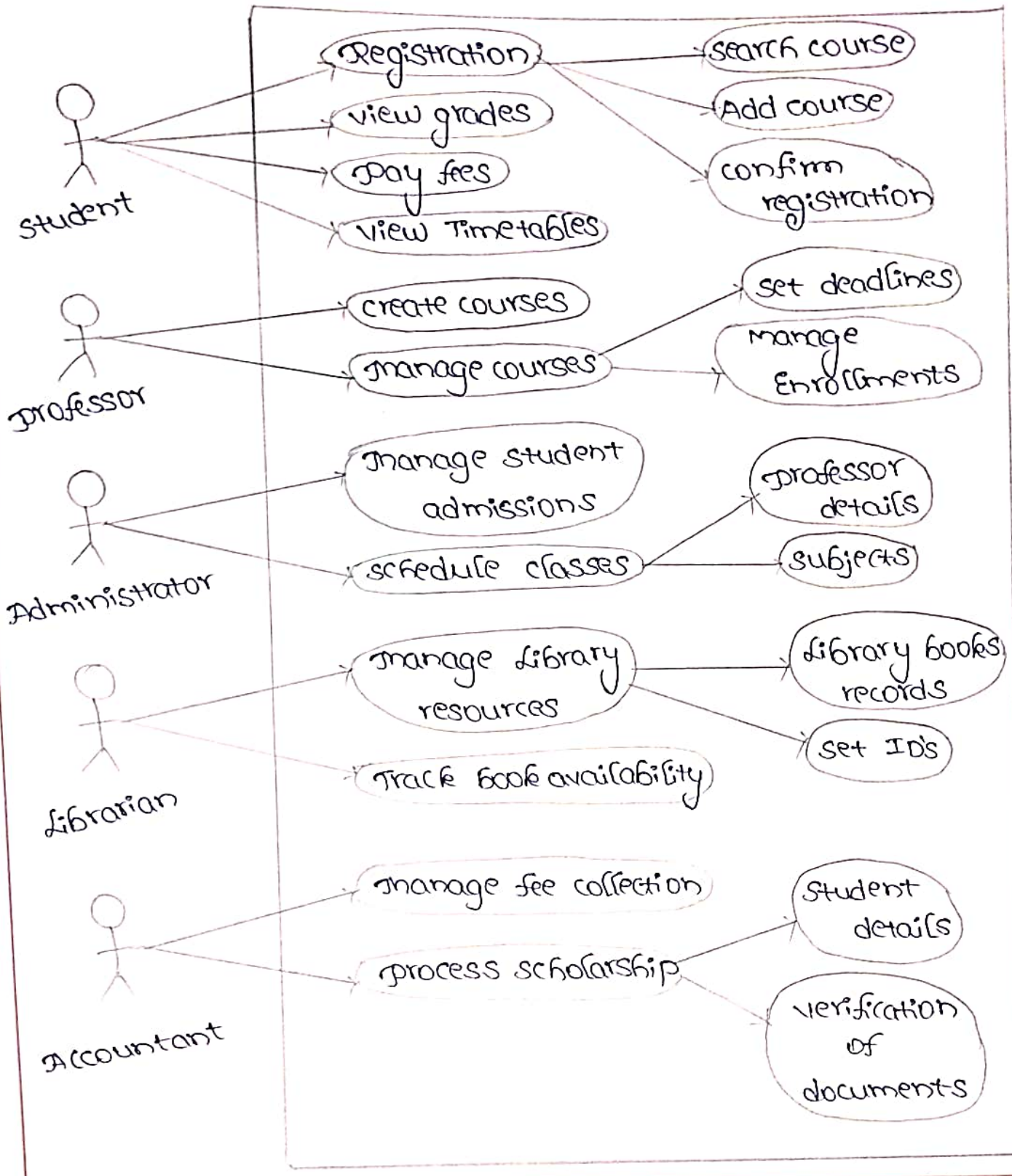
- Student
- professor
- Administrator
- parent / Guardian
- Librarian
- Exam controller
- Finance officer
- Course coordinator
- IT support

use case diagram::




Parent
Track student performance
Pay student fees.
Librarian
Manage library resources
Issue books
Track book availability
Exam controller
Schedule Exams
Publish Results
Generate Exam reports
finance officer
Manage fee collection
Process scholarships
IT support
Maintain software | hardware infrastructure
Resolve technical issues.

sub usecase diagram:



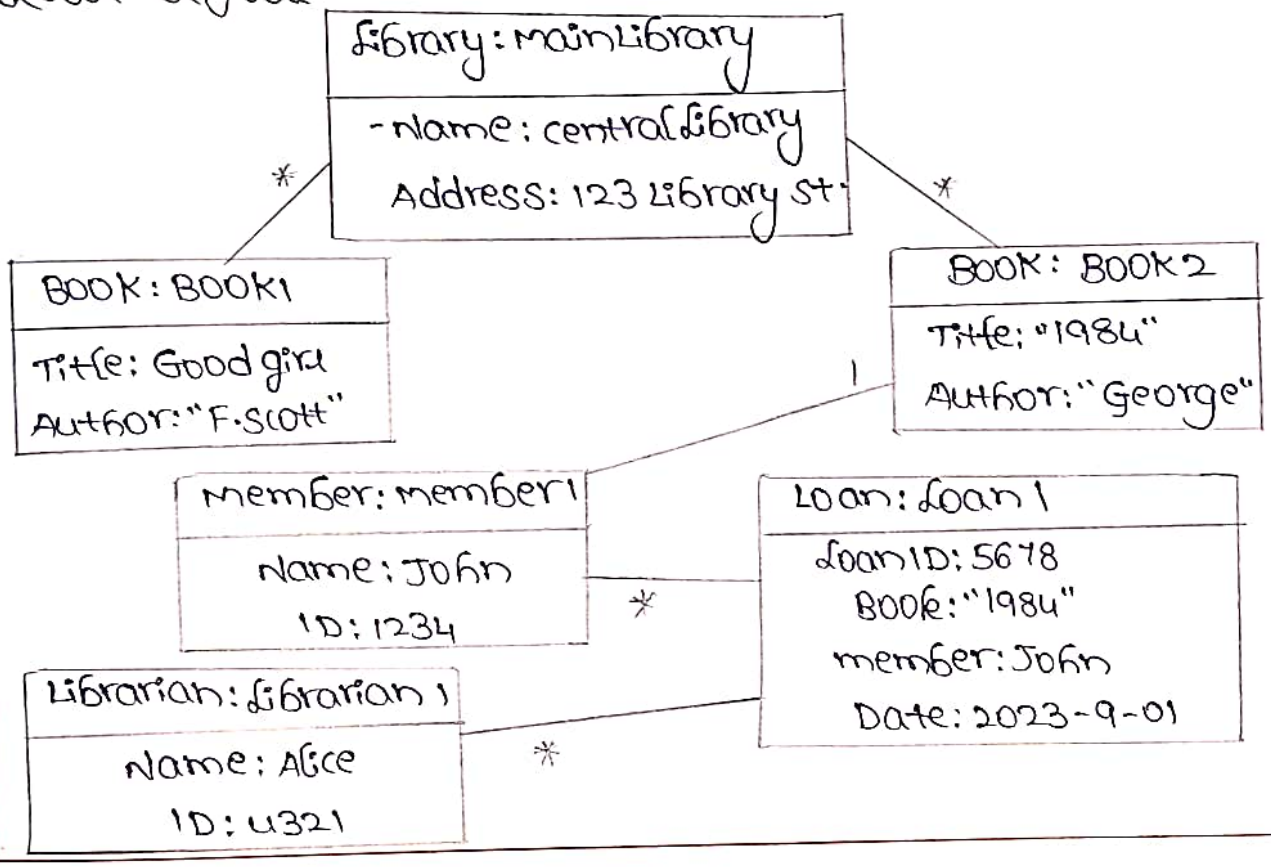
Week-6.

Aim:. consider the structural view of respective system.

- Identify the classes, their attributes, methods, relationships and develop the class diagram.
- Identify the objects and their links between and develop the object diagram.

Respective system:. Library management system.

Object diagram:.



class diagram: