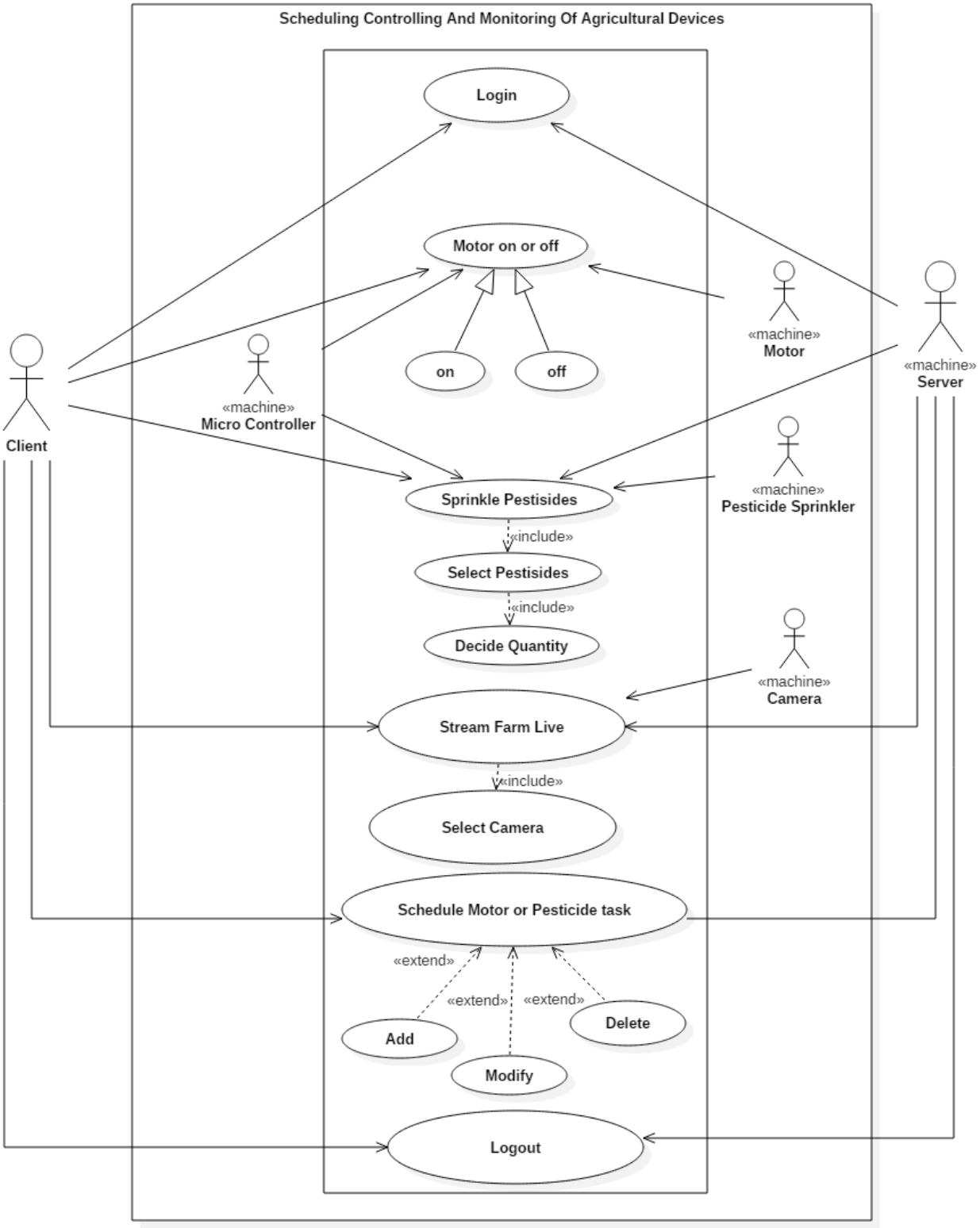
Module 2

Q1. Draw UML Use Case diagram and Class Diagram for "Smart Agriculture Monitoring System

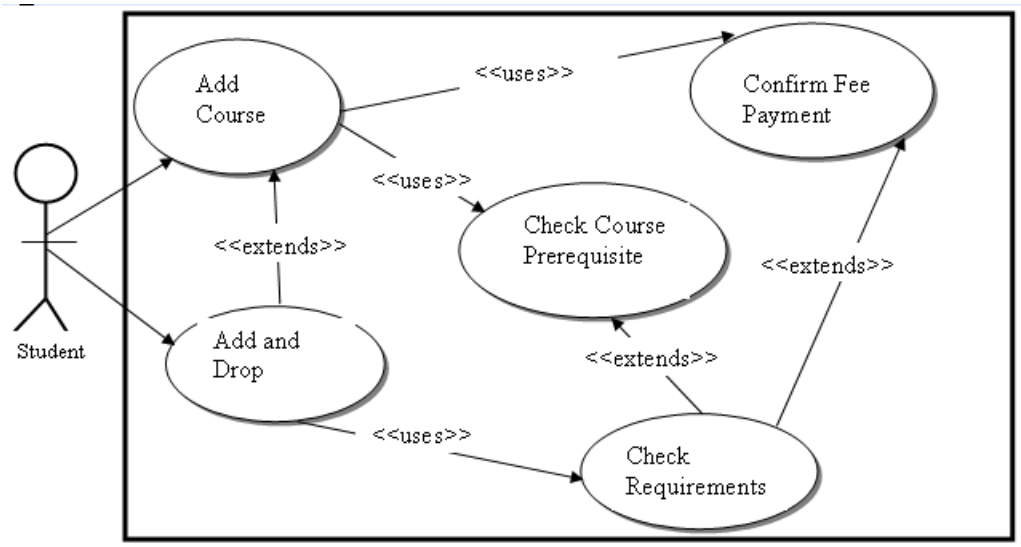
=>



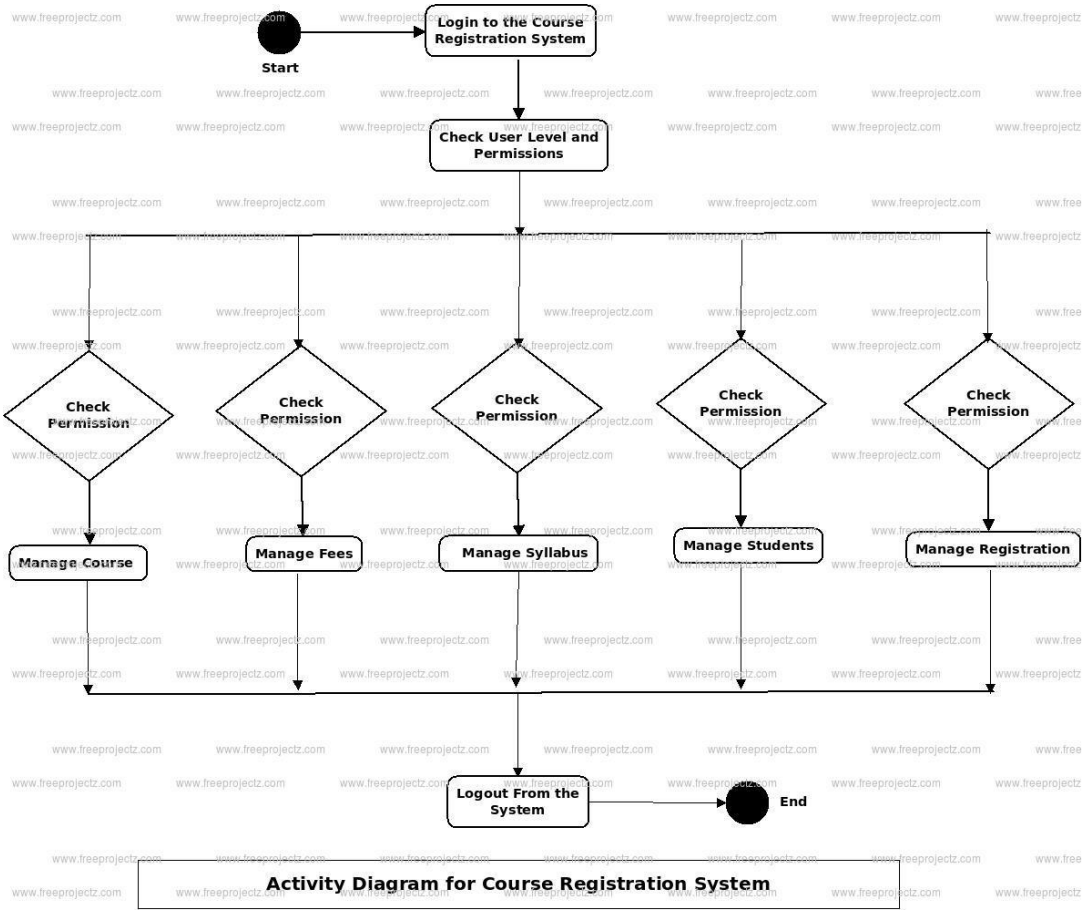
Q2. Draw the use case diagram and activity diagram for course registration system

=>

Use case diagram



Activity diagram



Q3. Explain Requirement Engineering in detail.

=>

a) Requirement Engineering involves gathering, analyzing, and documenting the software requirements.
b) This helps in creating a clear path for designing and building the software.
c) Requirement Engineering has seven main steps:

1. **Inception**
    a. Inception is the starting point where the basic ideas of the project are discussed.
    b. The development team asks questions to understand the project goals, main objectives, and who the project is for.
    c. The team identifies key stakeholders (people involved or impacted by the project) and tries to enhance communication between clients and developers.
2. **Elicitation**
    a. This step is all about gathering requirements directly from the stakeholders.
    b. It's essential to involve the right people to avoid misunderstandings.
3. **Elaboration**
    a. This phase involves expanding on the gathered requirements to create a detailed model of the software.
    b. The team may create models and prototypes (early versions of the software) to understand the system's functions and features better.
4. **Negotiation**
    a. Here, the client and developers discuss project limitations like budget, time, and resources.
    b. They prioritize requirements and resolve any conflicts, aiming to reach an agreement on the project scope.
5. **Specification**
    a. The requirements are documented in a formal way, often in a Software Requirements Specification (SRS) document.
    b. This document includes all user, system, functional, and non-functional requirements.
    c. The team presents the document in a way that the client can understand, giving a snapshot of how the final product will function.
6. **Validation**
    a. In this phase, the requirements are checked for errors and accuracy.
    b. The team ensures that all requirements are complete, clear, and meet the client's needs.

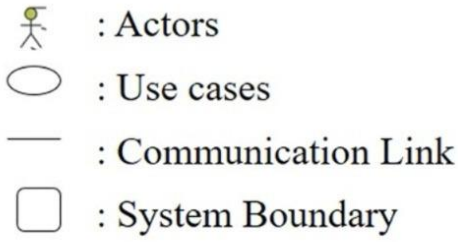
Q4. Explain Requirement Modelling in detail.

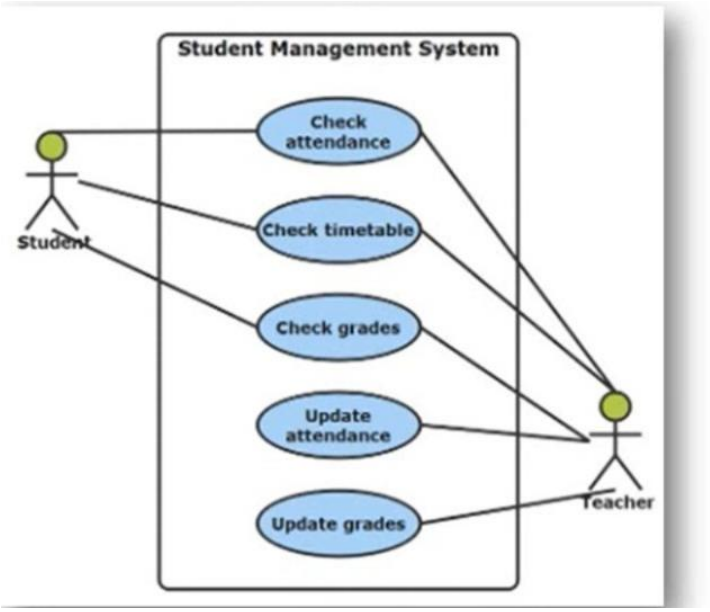
=>

1. Scenario-Based Modeling

Scenario-based modeling focuses on how users and other systems will interact with the software. This helps engineers visualize real-world usage scenarios.

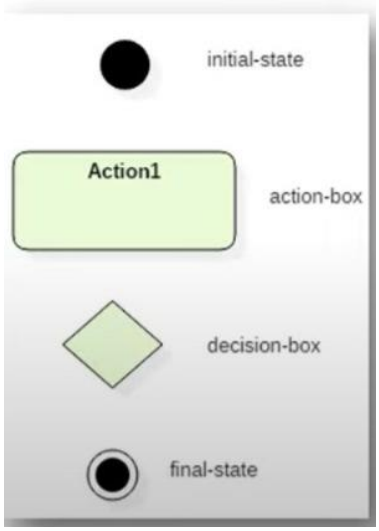
1) **Use Case Diagram**: This diagram represents all the ways users interact with the system.
   a. **Actors**: The users or external systems that will interact with the software.
   b. **Use Cases**: Specific actions or functions the system must perform for the user.
   c. **Communication Link**: A line connecting an actor to a use case, showing the interaction.
   d. **System Boundary**: Defines the scope of the system, showing what is included in or outside the software's functionalities.

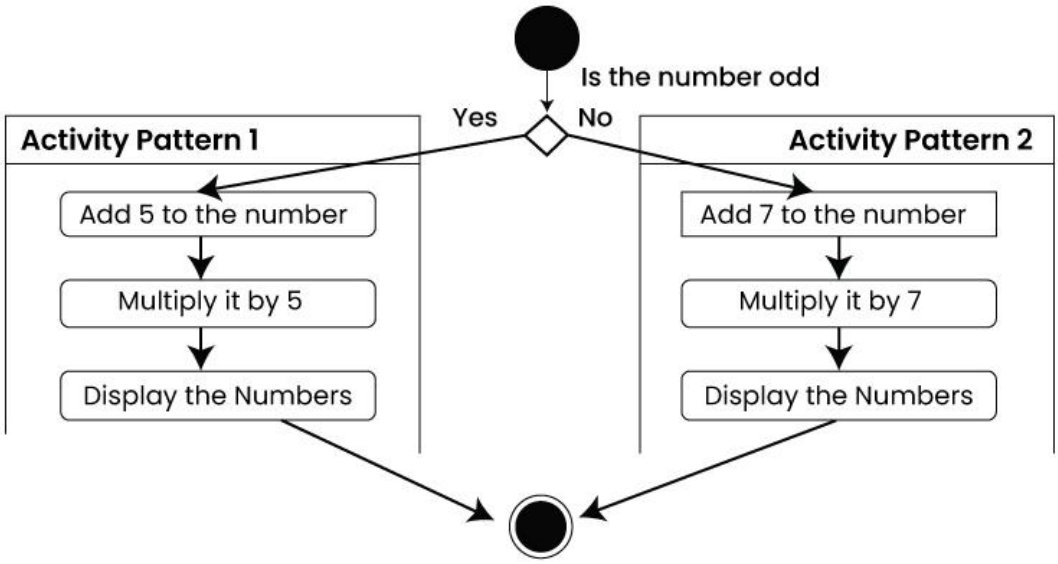




2) **Activity Diagram**: Shows the workflow from one activity to the next, capturing the sequence of operations and conditions that guide them.

a. **Flow Conditions**: Conditions under which each activity flows to the next.
b. **Order of Activities**: The logical sequence in which tasks or processes occur.



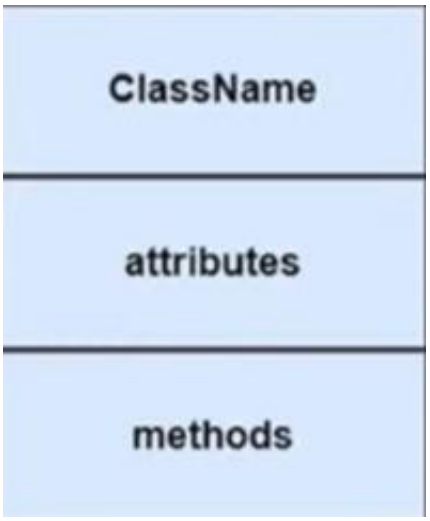


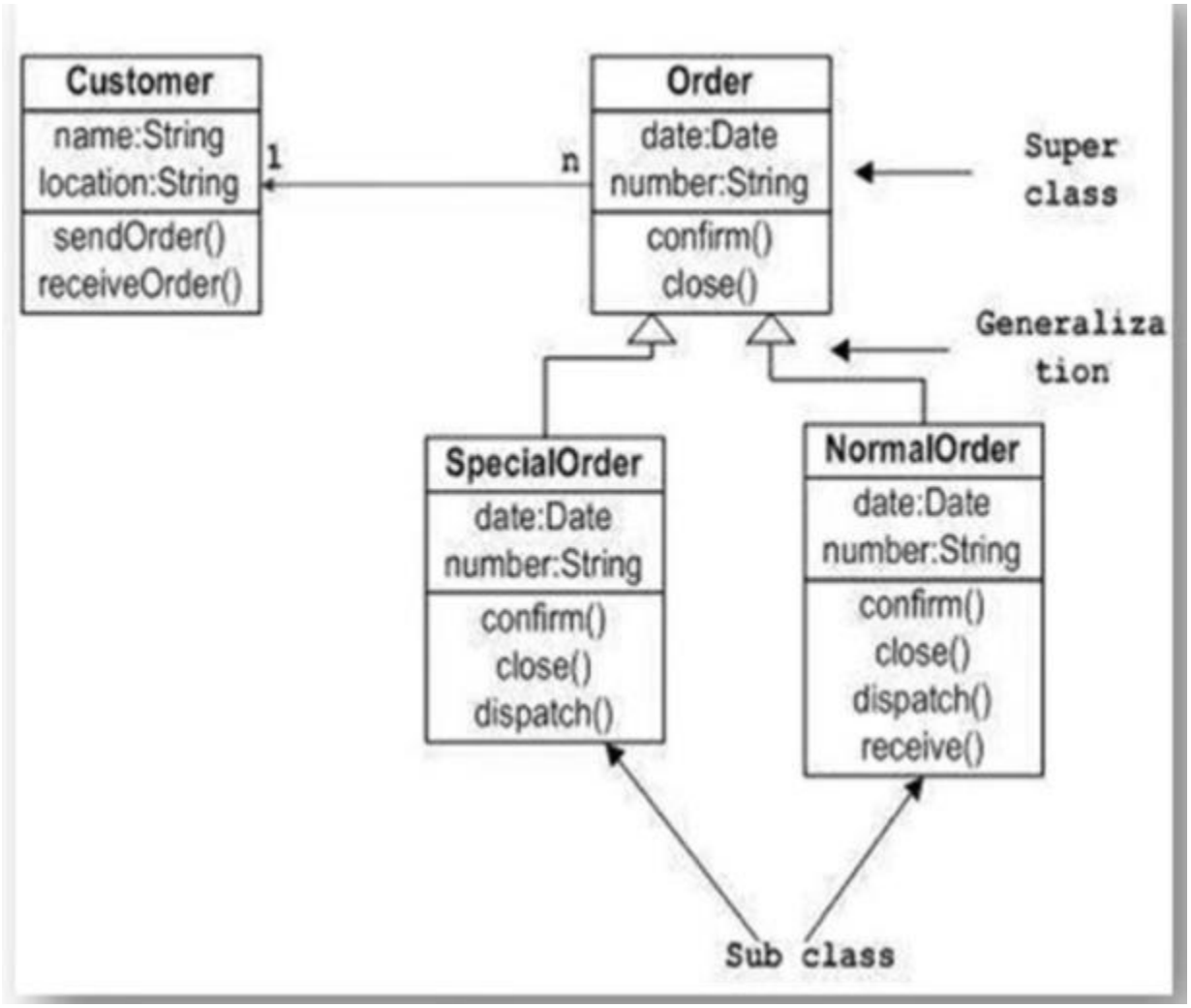
2. Class-Based Modeling

Class-based modeling represents the structure of the system by defining the main objects (or "classes") and their relationships.

- **Class Diagram**:

1. This static diagram shows different types of objects and their relationships within the system.
2. Format of a class diagram



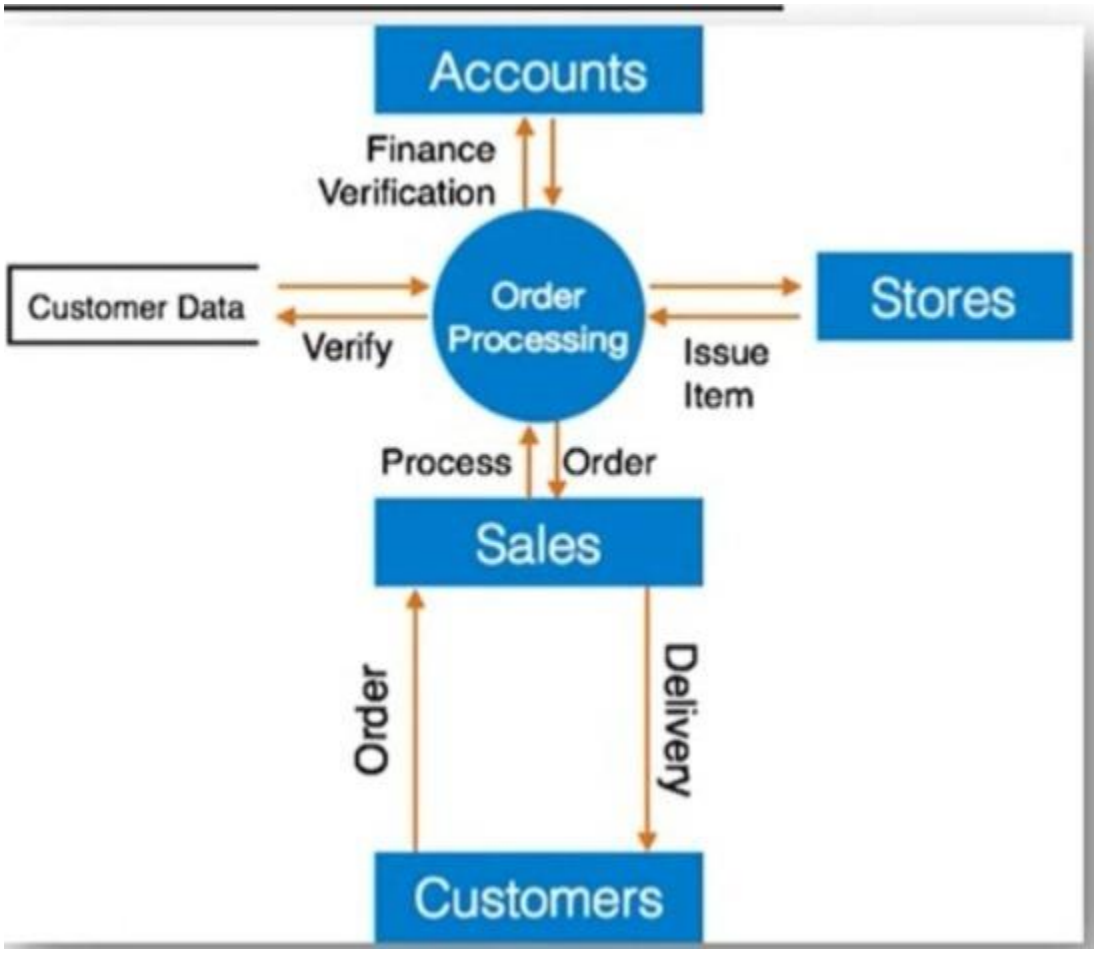


3. Flow-Oriented Modeling

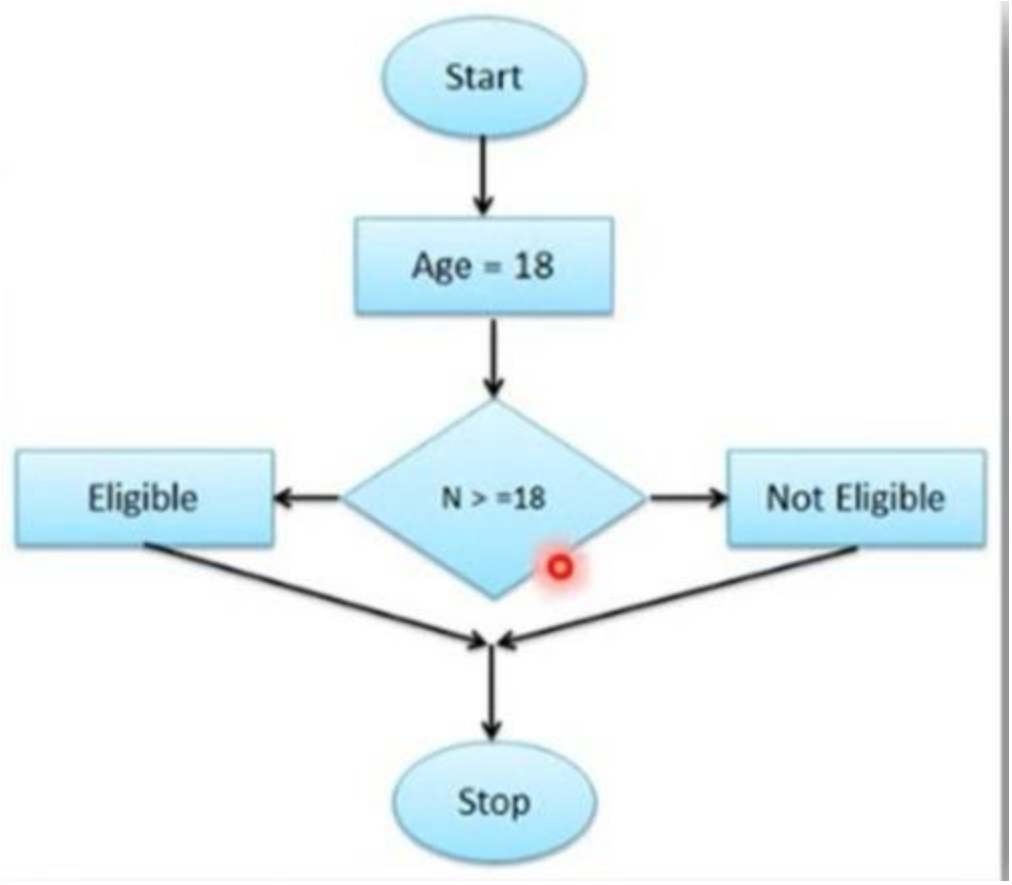
Flow-oriented modeling illustrates how data flows through the system, showing where data comes from, where it goes, and how it's transformed.

1) **Data Flow Diagram (DFD)**: A graphical representation of data moving through the system.

a) **Entities**: Sources or destinations of data, like users or external systems.
b) **Processes**: Transformations that data undergoes within the system.
c) **Data Flow**: Paths that show the movement of data between entities and processes.
d) **Data Store**: Places where data is stored within the system.



2) **Control Flow Diagram**: Shows the logical sequence of control in a program, especially useful for understanding decision-making points.
   a. **Entry Block**: Marks where control begins within a flow.
   b. **Exit Block**: The endpoint, where control flow leaves the system.
   c. **Conditions**: Decision points that dictate the control path based on true/false outcomes.



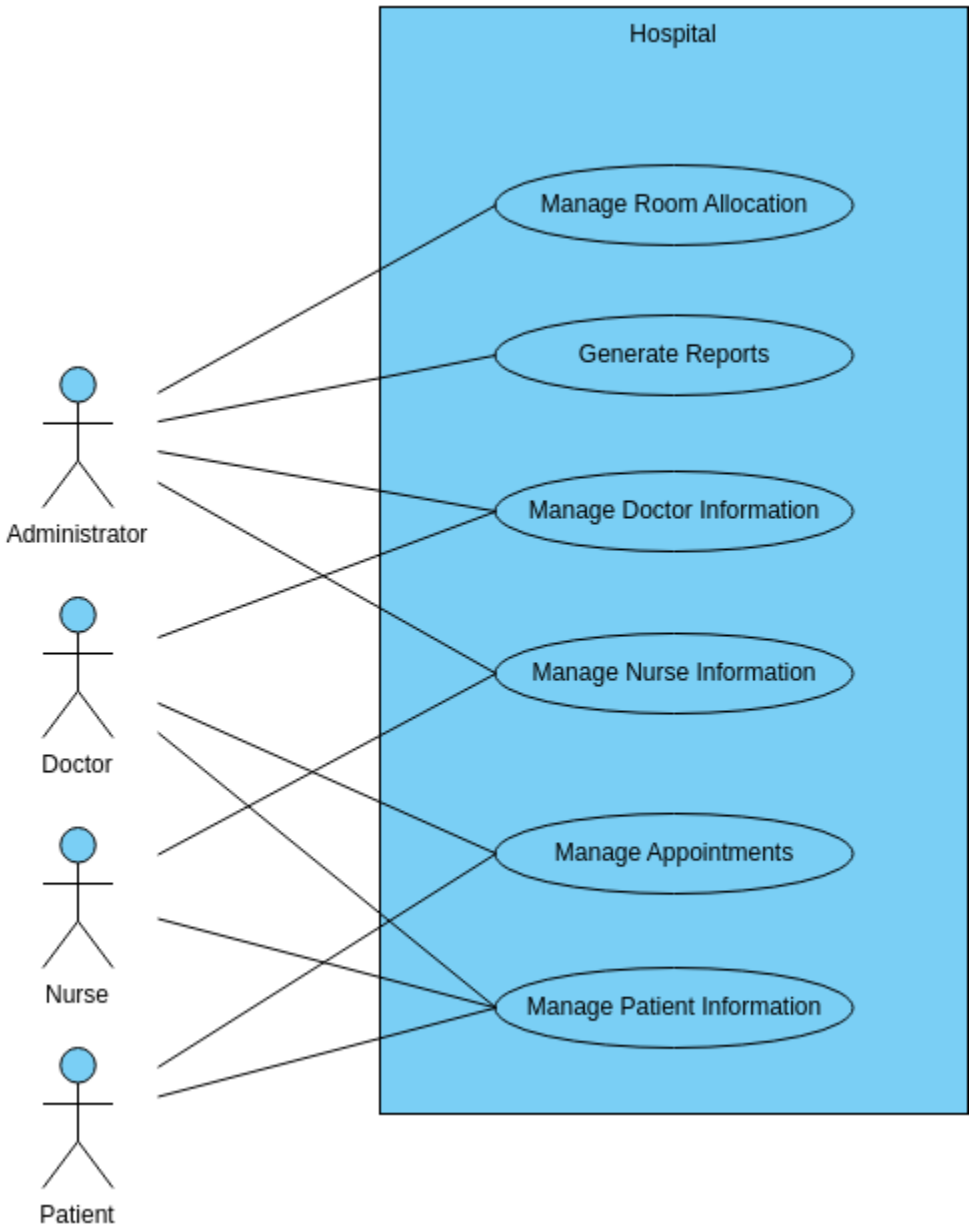
4. Behavioral Modeling

Behavioral modeling focuses on the system's reactions to various inputs and conditions, helping to capture its dynamic nature.

1) **State Transition Diagram**: Describes the different states a system can be in and the events that cause transitions between states.
   a. **States**: Different statuses or conditions the system can occupy at any time.
   b. **Events**: Triggers that move the system from one state to another.

Q5. Use Case Diagram for Hospital Management System

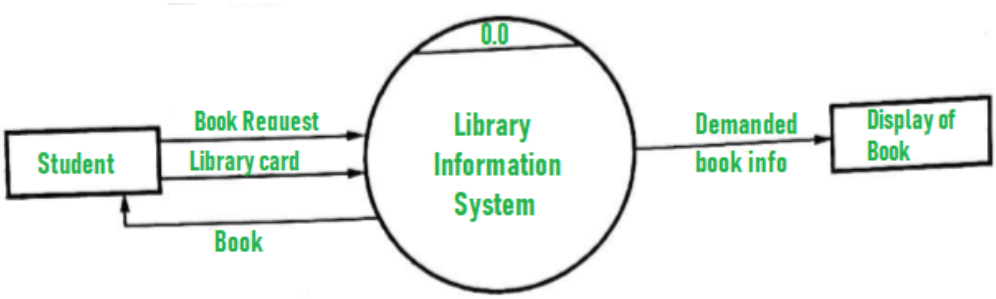
=>



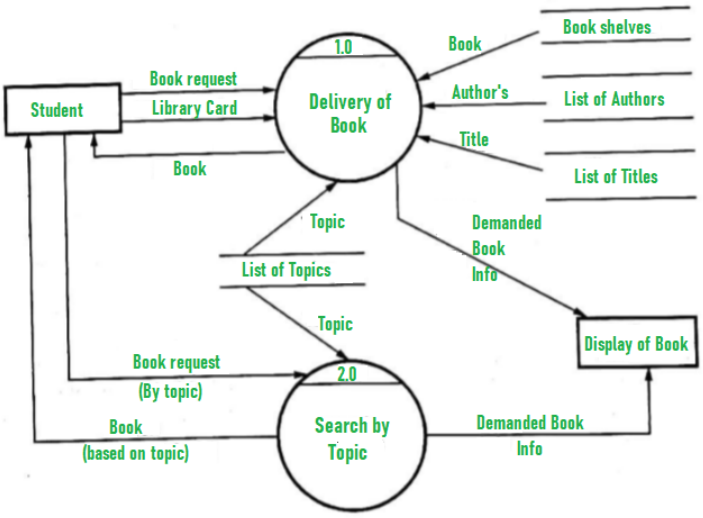
Q6. Design the DFD for Library Management System

=>

Level 0

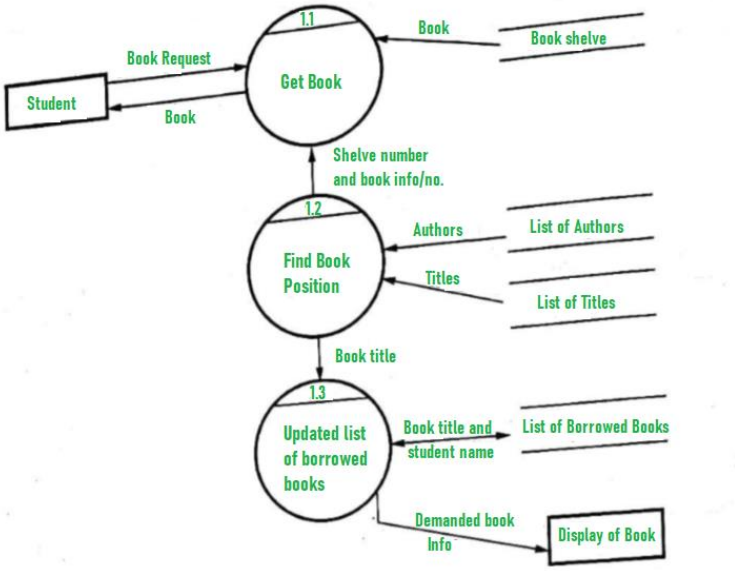


Level 1



Level 1 DFD

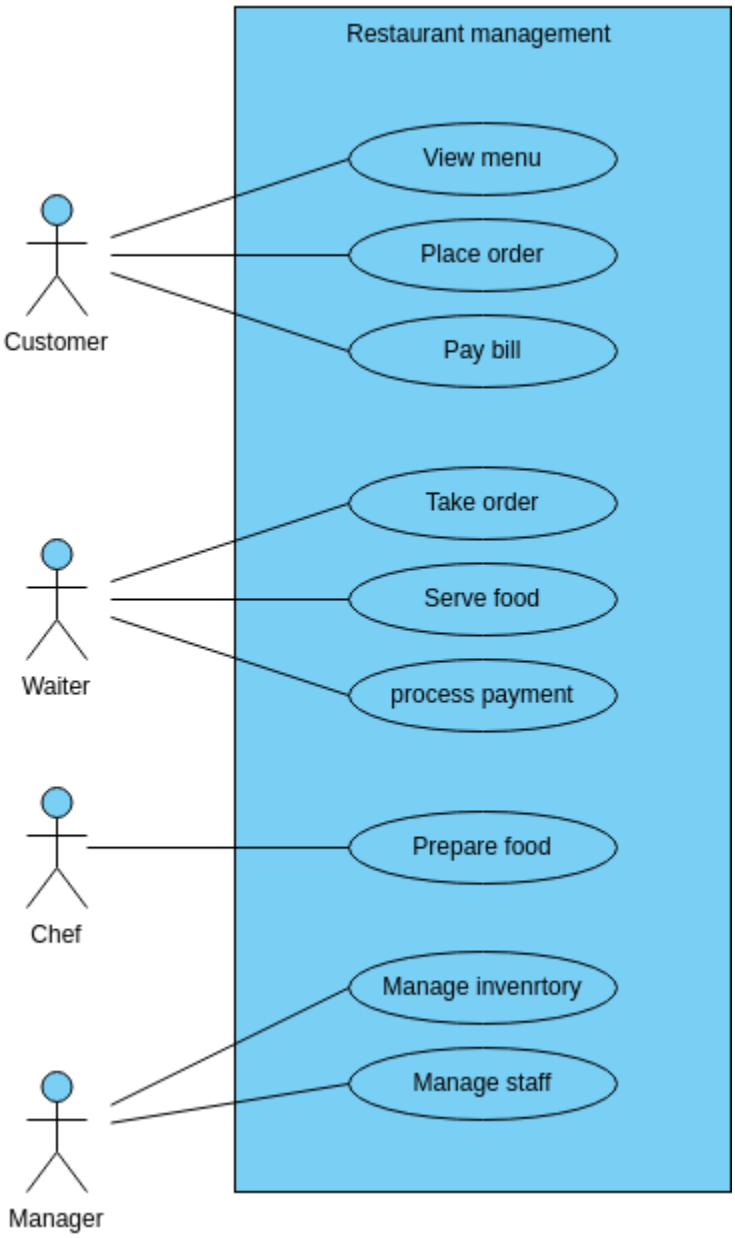
Level 2



Level 2 DFD

Q7. Draw use case diagram for restaurant management system

=>



Q8. Write an SRS for University Management Website

=>

1. Introduction

1.1 Purpose

The **University Management System (UMS)** is a software designed to manage university operations, including student information, courses, faculty details, and administrative tasks.

1.2 Scope

The initial scope includes:

- Student registration and course enrollment
- Grade management
- Fee payments
- Administrative functions

1.3 Audience

- Development Team
- Testing Team
- University Administrators
- Faculty Members
- Students

1.4 Definitions, Acronyms, and Abbreviations

- **UMS**: University Management System – the software for university management.
- **Admin**: Administrator – a user with higher system privileges.
- **Fac.**: Faculty – a teaching or research staff member.
- **Stud.**: Student – an individual enrolled at the university.
- **DB**: Database – a structured collection of stored data.
- **UX**: User Experience – overall experience while using the system.

1.5 References

- IEEE 830-1998 standard for SRS documentation.

2. Overview

This SRS document has three main sections:

1. **Overview of the UMS** – purpose and key features.
2. **Product Perspective and Functionality** – user characteristics, constraints, and dependencies.
3. **Functional and Non-Functional Requirements** – detailed requirements for the system.

3. General Description

3.1 Product Perspective

The UMS is an independent system designed to interact with the university's databases and systems, handling information for students, faculty, and university processes.

3.2 Product Functions

Key functions of UMS:

- **Student Registration and Enrollment**
- **Course Management**
- **Faculty Management**
- **Examination and Grading**
- **Attendance Tracking**
- **Academic Records Management**

3.3 User Characteristics

The system will be used by:

- **Administrators**: manage and configure the system
- **Faculty Members**: handle courses and grading
- **Students**: register and manage their academic information

3.4 General Constraints

- Compatibility with university infrastructure
- Compliance with data protection and privacy regulations
- Performance requirements during peak times

3.5 Assumptions & Dependencies

- Availability of necessary hardware and software
- University staff cooperation for integration

4. Specific Requirements

4.1 Functional Requirements

- **Student Registration**: Allows online course registration.
- **Course Management**: Add, modify, and delete courses.
- **Faculty Assignment**: Assign faculty to specific courses.
- **Examination System**: Schedule and manage exams.
- **Attendance Tracking**: Monitor student attendance.

4.2 Non-Functional Requirements

- **Performance**: Respond to user input within 2 seconds.
- **Security**: Secure user authentication and data encryption.
- **Usability**: User-friendly and intuitive interfaces.

4.3 External Interface Requirements

- **User Interfaces**: Separate interfaces for administrators, faculty, and students.
- **Hardware Interfaces**: Compatibility with standard university hardware.
- **Software Interfaces**: Integration with university databases and systems.

4.4 Data Requirements

- **Student Records**: Personal and academic information.
- **Course Information**: Details on course schedules and prerequisites.
- **Examination Data**: Manage exam schedules, grades, and results.
- **Attendance Records**: Store attendance data for students.

5. Appendices

References

- Websites: Wikipedia, Google Scholar, ChatGPT, Google Bard
- Books: Clean Architecture: A Craftsman's Guide to Software Structure and Design by Robert C. Martin

Index

Q9. Write an SRS for Hospital Management System

=>

1. Introduction

1.1 Purpose

The **Hospital Management System (HMS)** is software designed to manage hospital operations, including patient records, doctor and staff information, appointments, billing, and administrative tasks.

1.2 Scope

The initial scope includes:

- Patient registration and management
- Appointment scheduling
- Billing and invoicing
- Inventory and pharmacy management
- Staff and administrative functions

1.3 Audience

- Development Team
- Testing Team
- Hospital Administrators
- Medical Staff (Doctors and Nurses)
- Patients

1.4 Definitions, Acronyms, and Abbreviations

- **HMS**: Hospital Management System – the software to automate hospital operations.
- **Admin**: Administrator – a user with higher system privileges for management.
- **Med. Staff**: Medical Staff – doctors, nurses, and allied health professionals.
- **Pat.**: Patient – an individual receiving medical care.
- **DB**: Database – a structured data storage system.
- **UX**: User Experience – the experience users have while interacting with the system.

1.5 References

- IEEE 830-1998 standard for SRS documentation.

2. Overview

This SRS document has three main sections:

1. **Overview of the HMS** – purpose and key features.
2. **Product Perspective and Functionality** – user characteristics, constraints, and dependencies.
3. **Functional and Non-Functional Requirements** – detailed system requirements.

3. General Description

3.1 Product Perspective

The HMS is an independent system designed to interact with existing hospital databases and systems, managing patient information, medical staff, and hospital operations.

3.2 Product Functions

Key functions of HMS:

- **Patient Registration and Management**
- **Appointment Scheduling**
- **Doctor and Staff Management**
- **Billing and Invoicing**
- **Inventory and Pharmacy Management**
- **Medical Records Management**

3.3 User Characteristics

The system will be used by:

- **Administrators**: manage hospital operations and data access.
- **Medical Staff**: manage patient care, records, and schedules.
- **Patients**: access personal information, view bills, and schedule appointments.

3.4 General Constraints

- Compatibility with hospital infrastructure.
- Compliance with healthcare data protection and privacy regulations.
- Performance constraints during high-traffic times, like peak clinic hours.

3.5 Assumptions & Dependencies

- Availability of required hardware and software.
- Cooperation from hospital staff for system integration.

4. Specific Requirements

4.1 Functional Requirements

- **Patient Registration**: System must allow registering new patients and updating their information.
- **Appointment Scheduling**: Schedule appointments with doctors and manage time slots.
- **Doctor Assignment**: Assign doctors to specific patients or departments.
- **Billing System**: Generate bills for patients and track payment status.
- **Inventory Management**: Track medical supplies and manage the pharmacy stock.
- **Medical Records**: Maintain detailed patient records including medical history, diagnosis, and treatment.

4.2 Non-Functional Requirements

- **Performance**: The system should respond within 2 seconds to user inputs.
- **Security**: Secure user authentication and data encryption to protect sensitive information.
- **Usability**: User-friendly and intuitive interfaces for staff and patients.

4.3 External Interface Requirements

- **User Interfaces**: Design tailored interfaces for administrators, medical staff, and patients.
- **Hardware Interfaces**: Ensure compatibility with hospital equipment and workstations.
- **Software Interfaces**: Integrate with existing hospital databases, laboratory systems, and electronic medical records (EMR) systems.

4.4 Data Requirements

- **Patient Records**: Store personal, medical, and treatment history.
- **Doctor Information**: Maintain records of doctors' specialties, schedules, and availability.
- **Appointment Data**: Track and manage all appointment details.
- **Billing Information**: Store billing and payment details for all patient visits.
- **Inventory Data**: Keep track of medical supplies and pharmacy stock.

5. Appendices

References

- Websites: Wikipedia, Google Scholar, ChatGPT, Google Bard
- Books: Healthcare Information Systems: A Practical Approach for Health Care Management by Karen A. Wager

Index