MASTER OF COMPUTER APPLICATIONS

(MCA) II SEMESTER

MCA 204A – ARTIFICIAL INTELLIGENCE (WRITTEN PART)

1. Program to implement Breadth First Search (BFS) for graph traversal.

Aim:

To implement Breadth First Search (BFS) for graph traversal.

Algorithm:

1. Start from the initial node.
2. Add the node to the queue.
3. Remove node from queue and visit it.
4. Add all unvisited neighbours to the queue.
5. Repeat until the queue becomes empty.

Result:

Thus, Breadth First Search is implemented successfully

2. Program to implement Depth First Search (DFS) for graph traversal.

Aim:

To implement Depth First Search (DFS) for graph traversal.

Algorithm:

1. Start from the initial node.
2. Visit the node and mark it as visited.
3. Recursively visit all unvisited neighbours.

Result:

Thus, Depth First Search is implemented successfully.

3. Program to implement A* Search Algorithm for finding optimal path

Aim:

To implement A* algorithm to find optimal path.

Algorithm:

1. Define graph and heuristic values.
2. Calculate f(n) = g(n) + h(n).
3. Select node with minimum f(n).
4. Repeat until goal node is reached.

Result:

Thus, A* algorithm is implemented successfully.

4. Program to implement Minimax Algorithm for game decision making.

Aim:

To implement Minimax algorithm for game decision making.

Algorithm:

1. Generate game tree.
2. Apply MAX and MIN operations alternately.
3. Select optimal value.

Result:

Thus, Minimax algorithm is implemented successfully

5. Program to implement Bayes Theorem for probability calculation.

Aim:

To implement Bayes theorem for probability calculation.

Algorithm:

1.Define prior probabilities.

2. Apply Bayes formula.

3. Calculate posterior probability.

Result:

Thus, Bayes theorem is implemented successfully.

6. Program to solve Map Coloring Problem using Constraint Satisfaction Problem (CSP).

Aim:

To solve map coloring problem using Constraint Satisfaction Problem.

Algorithm:

1. Assign colors to each node.

2. Check constraints for adjacent nodes.

3. Use backtracking if constraints are violated.

Result:

Thus, map coloring problem is solved successfully

7. Program to implement Value Iteration using Markov Decision Process (MDP).

Aim:

To implement Value Iteration using Markov Decision Process.

Algorithm:

1. Initialize value function.
2. Update values using Bellman equation.
3. Repeat for given iterations.

Result:

Thus, Value Iteration is implemented successfully.

8. Program to implement Q-Learning in Reinforcement Learning.

Aim:

To implement Q-Learning in reinforcement learning.

Algorithm:

1. Initialize Q-table.
2. Select random state and action.
3. Update Q values using formula.
4. Repeat for iterations.

Result:

Thus, Q-Learning is implemented successfully.

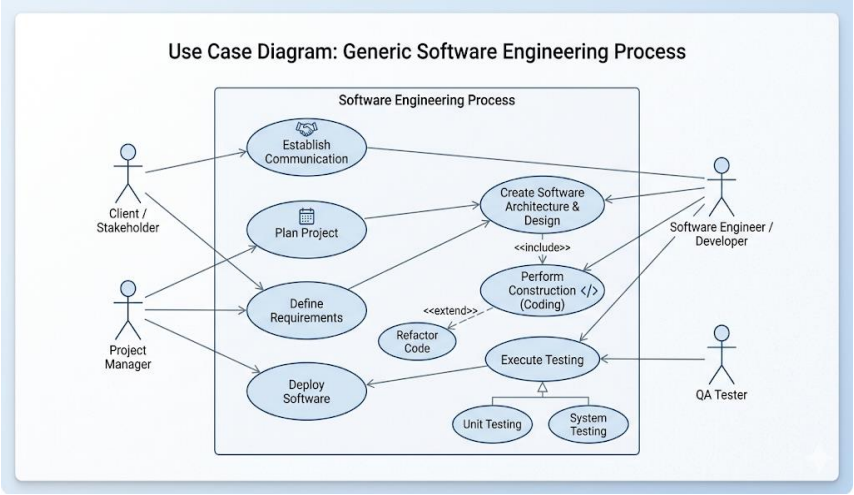
1. Explain the concept of **Software Engineering**. Describe its importance and discuss the **generic view of the software process** in detail.

Aim: To study and understand the concept of **Software Engineering**, its importance, and the generic view of the software process, and to demonstrate a simple implementation using Python.

Description: Software Engineering is the systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software. It applies engineering principles to build reliable, efficient, and scalable software systems.

Objectives

- To understand the principles of Software Engineering.
- To learn the importance of structured software development.
- To study the phases of the software process.
- To implement a simple example representing software process steps using Python.



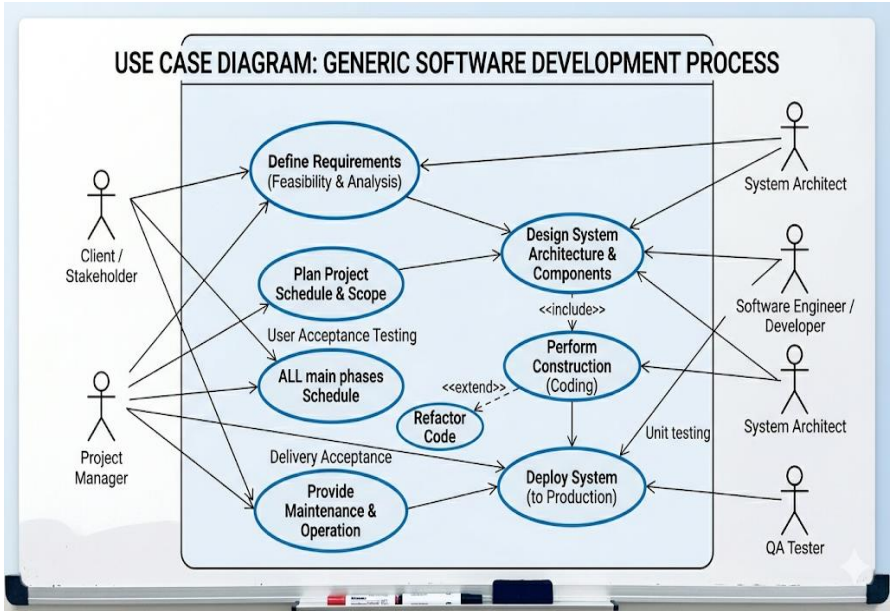
2. Discuss various **Software Process Models** in detail. Compare traditional models with **Agile process models** and explain their advantages and limitations.

Aim: **To study and understand different** Software Process Models**, compare** traditional models **with** Agile models**, and demonstrate a simple Python program that reflects process selection logic.**

Description: Software Process Models define structured approaches used to develop software systems. These models guide planning, execution, testing, and delivery.

Objectives

- Understand key software development models
- Identify differences between traditional and Agile approaches
- Analyze advantages and limitations of each model
- Apply decision logic for selecting an appropriate model



3.Explain the **Agile view of software development**. Describe Agile principles, benefits, and how it differs from conventional software development approaches.

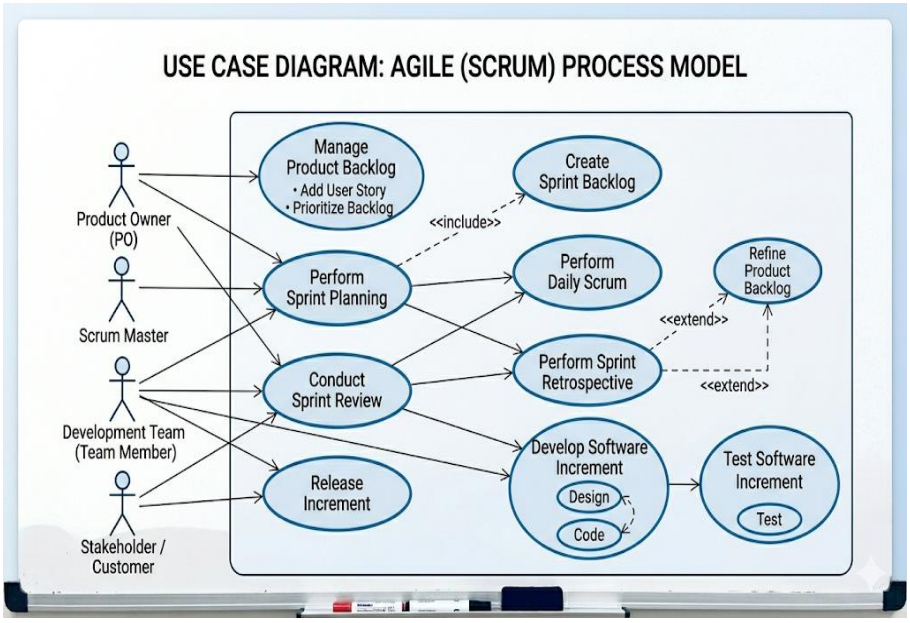
Aim: To understand the **Agile view of software development**, its principles, benefits, and to compare it with conventional development approaches using a simple Python demonstration.

Description : The **Agile approach** is a modern software development methodology that focuses on **iterative development, continuous feedback, and adaptability to change**. Instead of completing all phases sequentially like traditional models, Agile breaks the project into small units called **iterations or sprints**, where each cycle produces a working version of the software.

Agile emphasizes collaboration between developers and customers, frequent delivery of software, and continuous improvement through feedback.

Objectives

- To understand Agile software development concepts
- To study Agile principles and practices
- To compare Agile with conventional (Waterfall-based) development
- To simulate iterative vs sequential development using Python



4. Describe the **Software Engineering Practices Framework**. Explain in detail the following activities:

Aim: To study the Software Engineering Practices Framework **and understand its key activities:** Communication, Planning, Modeling, Construction, and Deployment**, along with a simple Python simulation of these phases.**

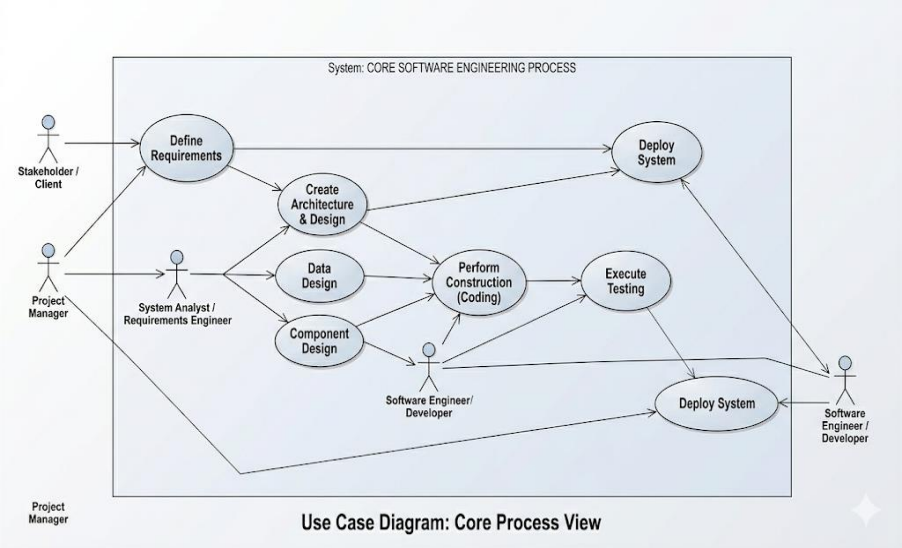
Description: The Software Engineering Practices Framework **is a structured approach that defines the essential activities required to develop high-quality software systems. It provides a** foundation for all software process models **(Agile, Waterfall, Spiral, etc.) and ensures that development is systematic, efficient, and reliable.**
The framework is divided into five main activities:

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

Each activity plays a critical role in transforming user requirements into a working software product.

Objectives

- To understand the phases of software engineering practices
- To learn how software is developed step-by-step
- To analyze the importance of each activity in software development
- To simulate software development phases using Python



Use Case Diagram: Core Process View

5. Discuss the **Communication and Planning activities** in Software Engineering. Explain their importance and challenges with suitable examples.

Aim: To study the **Communication and Planning activities** in Software Engineering, understand their importance, challenges, and demonstrate their execution flow using a Python program.

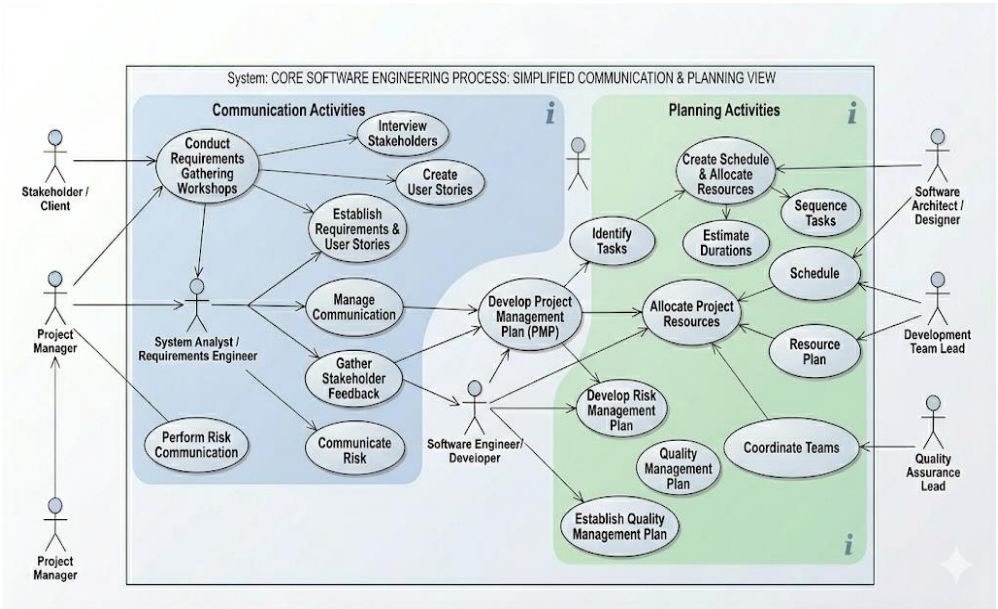
Description: In Software Engineering, Communication and Planning **are the first two major activities in the software development lifecycle.**

- **Communication** focuses on understanding **what the client wants** by gathering and analyzing requirements.
- **Planning** focuses on deciding **how the project will be executed**, including cost, schedule, resources, and risk management.

These activities ensure that the software project starts with a clear understanding and a structured roadmap.

Objectives

- To understand the role of communication in requirement gathering
- To study the importance of planning in software development
- To identify challenges in both activities
- To simulate communication and planning phases using Python



6. Define **System Engineering**. Explain the concept of **computer-based systems** and describe the **system engineering hierarchy** in detail.

Aim: To understand the concept of **System Engineering**, study **computer-based systems**, and describe the **system engineering hierarchy**, along with a simple Python program to represent system structure.

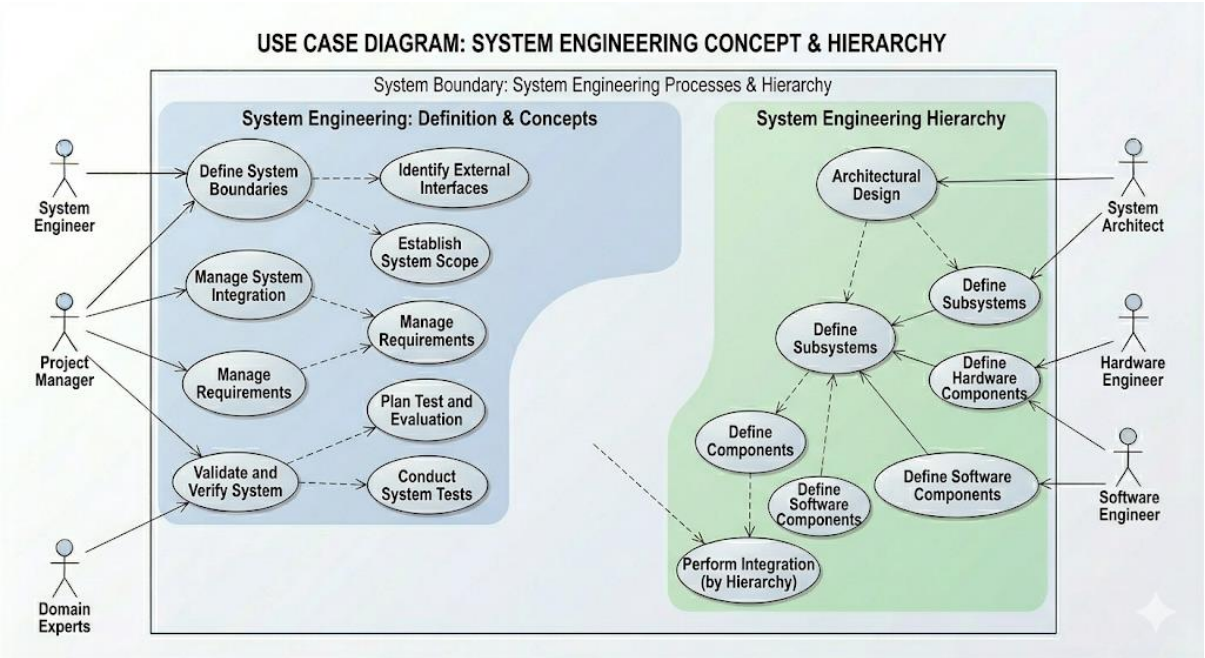
Description: System Engineering **is a discipline of software engineering that focuses on designing, analyzing, and managing complex systems over their entire life cycle. It ensures that all components of a system work together efficiently to achieve overall system goals.**

A **computer-based system** is a combination of hardware, software, data, people, and procedures that interact to perform specific tasks.

System Engineering provides a **top-down approach**, breaking a large system into smaller manageable subsystems.

Objectives

- To understand the concept of system engineering
- To study computer-based systems and their components
- To learn the system engineering hierarchy
- To represent system structure using a Python program



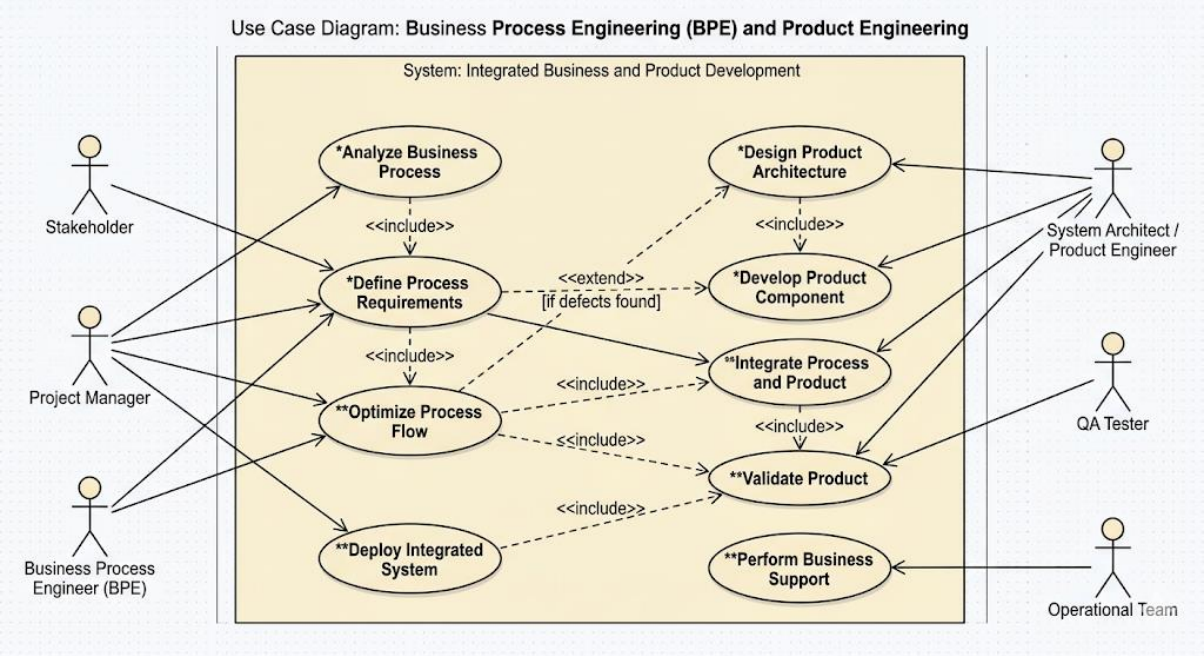
7. Explain **Business Process Engineering (BPE)** and **Product Engineering**. Discuss their roles in system development with examples.

Aim: To analyze, redesign, and improve business processes to achieve higher efficiency, reduced cost, and better quality of service.

Description: Business Process Engineering (BPE) is a systematic approach to improving an organization's workflows. It focuses on **restructuring existing business processes** or designing completely new ones to meet business goals using technology and management techniques. It is often used in organizations undergoing digital transformation.

Objectives:

- Improve process efficiency
- Reduce operational cost
- Eliminate redundant steps
- Improve customer satisfaction
- Automate manual processes
- Increase productivity



8. Discuss the **Web Engineering Process**. Explain different **analysis models for web applications** and their significance.

Aim: To understand the systematic process of developing web applications using Web Engineering principles and to study different analysis models used for designing efficient, scalable, and user-friendly web systems.

Description: Web Engineering is a disciplined approach to developing web applications using systematic methods, tools, and techniques similar to software engineering.

It focuses on:

- Requirements gathering
- Web application design
- Content structuring
- Implementation
- Testing
- Deployment and maintenance

Unlike traditional software, web applications require attention to:

- User experience (UI/UX)
- Navigation structure
- Content management
- Performance and scalability

Objectives

- Develop structured and maintainable web applications
- Improve usability and user experience
- Ensure scalability and performance
- Manage dynamic web content efficiently
- Reduce development time and cost
- Improve reliability and security of web systems