**3. Aspect-Oriented Programming(AOP)**

**What is AOP (Aspect-Oriented Programming)?**

• **Theory:**

**1.Definition of AOP and its importance in separating cross-cutting concerns (logging, security, transaction management):-** Aspect-Oriented Programming (AOP) is a programming paradigm that helps separate cross-cutting concerns from the main business logic. It allows adding functionality (like logging, security, and transaction management) without modifying the core logic of an application.

**Importance of AOP in Separating Cross-Cutting Concerns**

Cross-cutting concerns are functionalities that are used across multiple parts of an application but are **not part of the business logic**. AOP helps in:

1️.**Logging** → Automatically logs method execution, inputs, and outputs.  
2️.**Security** → Applies authentication and authorization checks uniformly.  
3️.**Transaction Management** → Ensures database transactions are handled consistently.  
4️.**Performance Monitoring** → Tracks execution time for performance analysis.  
5️.**Error Handling** → Captures exceptions globally without duplicating code.

**2. Key components in AOP: ♣ Aspect: A module that encapsulates cross-cutting concerns. ♣ Joinpoint: A point in the program where the aspect is applied. ♣ Advice: The action taken by an aspect at a particular joinpoint (Before, After, Around). ♣ Pointcut: An expression to define where advice should be applied:-**

1️.Aspect → A module that encapsulates a cross-cutting concern (e.g., logging, security).

* Implemented as a class with @Aspect annotation in Spring.

2️.Joinpoint → A specific point in program execution where an aspect can be applied.

* Example: Method execution, object initialization, or exception handling.

3️.Advice → The action taken by an aspect at a particular joinpoint. Types of advice:

* @Before → Runs before the method execution.
* @After → Runs after the method execution.
* @AfterReturning → Runs after a method successfully returns.
* @AfterThrowing → Runs after a method throws an exception.
* @Around → Runs before and after method execution.

4️.Pointcut → A pattern to define where the advice should be applied.

* Example: Applying logging only to service package methods