Software Architecture

Software Architecture is like the **blueprint** of a house, but instead of a house, it's for a software system. It describes how the different parts of the software fit together, how they work with each other, and how they communicate.

Why is Software Architecture Important?

- **Helps Organize the System**: It provides a clear plan, so developers know how to build and connect different parts of the software.
- **Makes it Easier to Maintain**: Good architecture allows for easy updates or fixing bugs without breaking other parts of the system.
- **Enables Growth**: If the software needs to handle more users or data in the future, a good architecture makes it easier to grow (scale).

Main Parts of Software Architecture:

1. **Components**:
    - **Explanation**: These are the building blocks of the software, like **modules** or **subsystems**. Each part does a specific job.
    - **Example**: In an online shopping app, one component might handle user accounts, and another component manages payments.
2. **Connections/Interactions**:
    - **Explanation**: This shows how the different parts of the software communicate with each other.
    - **Example**: The payment system sends information to the account system when a user buys something.
3. **Interfaces**:
    - **Explanation**: These define the way the parts of the software talk to each other. It's like a set of rules for how they interact.
    - **Example**: An API allows your phone app to communicate with a server to fetch data.

Common Software Architecture Patterns:

1. **Layered Architecture**:
    - **What it is**: The system is divided into layers, and each layer has a specific task.
    - **Example**: In a web application:

SHYAM-SUDHEER1602

- The top layer (User Interface) shows what the user interacts with.
- The middle layer handles business logic.
- The bottom layer manages data storage.

2. **Client-Server Architecture**:
    - **What it is**: The system is divided into two main parts: **client** (requests services) and **server** (provides services).
    - **Example**: When you use a browser (client) to access a website (server), you are using a client-server model.
3. **Microservices Architecture**:
    - **What it is**: The system is made up of small, independent services that work together.
    - **Example**: A service like Netflix uses many different small services—one for movies, another for user recommendations, etc.

SHYAM-SUDHEER1602