Below are five scenario-based questions designed to test students' ability to **recommend an appropriate software architecture style**. The six possible styles are **Pipe-and-Filter, Blackboard, Peer-to-Peer, Client-Server, Broker, and MVC**.

Software Architecture Style — Scenario Questions

Scenario 1 — Online Banking System

A bank is developing an online banking application. Customers use a web or mobile application to check their balance, transfer money, pay bills, and download statements. The application sends requests to a central banking server, which authenticates users, processes transactions, and retrieves account information.

Question:

Which software architecture style would you recommend for this system?

Choose from: Pipe-and-Filter, Blackboard, Peer-to-Peer, Client-Server, Broker, MVC.

Explain why your chosen architecture is appropriate.

Recommended Architecture Style: Client-Server

The **Client-Server architecture** is appropriate because customers use client applications, such as a web browser or mobile app, to send requests to a **central banking server**. The server is responsible for authentication, processing transactions, and retrieving account information.

Why it fits:

- Clients request services from a centralized server.
- The server manages important data and business operations.
- Centralized control is suitable for sensitive banking information.

Scenario 2 — Image Processing System

A company is developing software that processes large numbers of images. Each image goes through several sequential processing stages:

1. Image loading
2. Noise removal
3. Color correction
4. Edge detection
5. Compression
6. Output generation

The output of one stage becomes the input of the next stage. Developers also want to be able to add, remove, or replace individual processing stages without significantly changing the rest of the system.

Question:

Which software architecture style would you recommend?

Choose from: Pipe-and-Filter, Blackboard, Peer-to-Peer, Client-Server, Broker, MVC.

Explain why your chosen architecture is appropriate.

Recommended Architecture Style: Pipe-and-Filter

The **Pipe-and-Filter architecture** is appropriate because image processing is divided into a sequence of independent processing stages, where the output of one stage becomes the input of the next.

Why it fits:

- Each processing stage acts as a separate **filter**.
- Data flows from one filter to another through **pipes**.
- Filters can be added, removed, or replaced without significantly affecting the other stages.
- It is suitable for sequential data processing.

Flow:

Image Loading → Noise Removal → Color Correction → Edge Detection → Compression → Output

---

Scenario 3 — Medical Diagnosis System

A hospital is developing an intelligent medical diagnosis system. Different knowledge sources provide information about a patient's symptoms, laboratory results, medical history, and diagnostic rules. Multiple independent reasoning modules examine the available information and contribute possible diagnoses. A central shared data repository stores the current patient information and the hypotheses generated by the reasoning modules.

Question:

Which software architecture style would you recommend?

Choose from: Pipe-and-Filter, Blackboard, Peer-to-Peer, Client-Server, Broker, MVC.

Explain why your chosen architecture is appropriate.

Recommended Architecture Style: MVC (Model-View-Controller)

The **MVC architecture** is appropriate because the system needs to separate the **user interface, application logic, and data**.

- **Model:** manages product, order, and shopping-cart data.
- **View:** displays products and other information to the user.
- **Controller:** handles user input and interactions and coordinates between the Model and View.

Why it fits:

- It separates the user interface from the business logic and data.
- The user interface can be modified without significantly changing the underlying logic.
- It improves maintainability and separation of responsibilities.

Scenario 4 — File-Sharing Application

A company wants to develop a file-sharing application where users' computers can directly communicate with one another. There is no single central server responsible for storing

all files. Each computer can request files from other computers and can also provide files to other participants.

The system should continue functioning even if some computers disconnect from the network.

Question:

Which software architecture style would you recommend?

Choose from: Pipe-and-Filter, Blackboard, Peer-to-Peer, Client-Server, Broker, MVC.

Explain why your chosen architecture is appropriate.

Recommended Architecture Style: Peer-to-Peer

The **Peer-to-Peer architecture** is appropriate because each computer can act as both a **client and a server**.

Why it fits:

- Computers communicate directly with one another.
- There is no single central server storing all files.
- Each peer can request and provide files.
- The system can continue functioning even when some peers disconnect.

Scenario 5 — Online Shopping Website

A development team is building an online shopping website. The system contains a user interface for displaying products, a component responsible for handling user interactions, and components responsible for managing shopping-cart data, orders, and product information. The team wants to keep the user interface separate from application logic and data so that the interface can be changed without significantly modifying the underlying business logic.

Question:

Which software architecture style would you recommend?

Choose from: Pipe-and-Filter, Blackboard, Peer-to-Peer, Client-Server, Broker, MVC.

Explain why your chosen architecture is appropriate.

Recommended Architecture Style: MVC (Model-View-Controller)

The **MVC architecture** is appropriate because the system needs to separate the **user interface, application logic, and data**.

- **Model:** manages product, order, and shopping-cart data.
- **View:** displays products and other information to the user.
- **Controller:** handles user input and interactions and coordinates between the Model and View.

Why it fits:

- It separates the user interface from the business logic and data.
- The user interface can be modified without significantly changing the underlying logic.
- It improves maintainability and separation of responsibilities.