Software Architecture Design

Stylique – Fashion & Styling Application

•Overview of Stylique:

A fashion and styling application helping users browse products, get styling recommendations, and manage a shopping experience.

•Purpose of the presentation:

To analyze the system’s architecture and justify the selected architecture style.

Software Architecture?

Software architecture is the **high-level structure** of a system.
It defines **how components are organized**, how **they interact**, and **how data flows**.

Software Architecture Styles?

Software Architecture Styles are standard ways of designing software systems.
They act like blueprints that help developers build scalable, maintainable, and efficient systems.

TYPES OF SOFTWARE ARCHITECTURE STYLES

1. Layered Architecture
2. Client-Server Architecture
3. Microservices Architecture
4. Event-Driven Architecture
5. MVC (Model-View-Controller) Architecture
6. SOA (Service-Oriented Architecture)
7. Peer-to-Peer (P2P) Architecture
8. Pipe-and-Filter Architecture
9. Broker Architecture
10. Component-Based Architecture

1. Layered Architecture

A system design where components are organized into layers, each with a specific responsibility.

- Like a sandwich

•Top layer: what you see (screen)
•Middle layer: thinking part (logic)
•Bottom layer: storage (database)

Easy to manage, clean, and organize

#	Architecture Style	Follows? (Yes/No)	Reason (why?)
1	Layered	No	**Why it doesn't follows:** because of its **rigid structure**, which can introduce **data flow delays** between layers, affecting real-time user interactions. It also lacks the **flexibility** required to handle frequent and dynamic **UI updates**. This makes it less efficient for a **UI-driven, interactive** application like Stylique.

1. Why Stylique Doesn’t Uses Layered Architecture?

- Stylique is **component-based**, not strictly hierarchical.
- Components like **Cart, Catalog, Styling Recommendations** need **direct interaction**.
- Layered architecture forces **rigid flow** → slows updates and responses.
- Reduces **flexibility, modularity, and maintainability**.

“Layered design restricts direct component communication, which Stylique relies on for real-time interaction.”

1. What Would Happen If We Used Layered Architecture?

•Every action must pass through multiple layers, making the app slower and more complex.

•Direct communication between components is not allowed, reducing efficiency.

•Small changes affect many layers, making maintenance difficult

2. Client–Server Architecture

An architecture where clients request services and servers provide them.

- Like ordering food

•You are a (client)order
•Restaurant (server) prepares food

2	Client-Server	No	**Why it doesn't follows:** as it requires complex data processing on the client-side, which could impact performance. Additionally, handling sensitive data like user photos and recommendations on the client could raise security concerns. Centralizing processing on the server-side ensures better scalability, security, and consistency.

2. Why Stylique Doesn’t Uses Client Server Architecture?

•Stylique is component-based, not just request-response.

•Components interact internally without always going through a central server.

•Client-server forces all communication via server → slows interactions between components.

•Limits modularity and independent functionality.

“Client-server handles requests, but Stylique’s strength is direct, modular component interaction.”

2. What Would Happen If We Used Client-Server Architecture?

- All communication must go through a central server, making even simple interactions slower.

- Components lose independence and cannot interact directly with each other.

- The server becomes a bottleneck, reducing performance and scalability.

3. Microservices Architecture

A style where an application is built as a collection of small, independent services.

- like many small shops

Each shop does **one specific job**(works independently)

•One shop for users
•One shop for payments
•One shop for orders
•One shop for recommendations

If one shop has a problem, others still work

3	Microservices	No	**Why it does not follow:** Stylique is not complex enough for multiple independent services. The features (product browsing, cart, recommendations) are handled in a single application without splitting into separate deployable services.

3. Why Stylique Doesn’t Uses Microservices Architecture?

•Stylique is not fully distributed into separate services.

•Components are modular but not deployed as independent microservices.

•Microservices add unnecessary complexity for the app’s scale.

•Increases overhead in communication between services.

“Stylique uses modular components, but microservices are overkill for current functionality.”

3. What Would Happen If We Used Microservices Architecture?

- The system would become unnecessarily complex for Stylique's current scale.

- Communication between services would increase network overhead and delays.

- Managing and deploying many small services would be harder and time-consuming.

4. Event-Driven Architecture

A system design where components communicate by producing and reacting to events.

- like a doorbell

•Someone presses the doorbell
•Something happens
•System reacts immediately

Examples of events:

•Button click
•New message
•Order placed

4	Event-Driven	No	**Why it does not follow:** Stylique has minimal event-driven needs. Notifications or updates (like new products) are occasional and do not drive the core system. The app mostly works on direct user actions.

4. Why Stylique Doesn’t Uses Event-Driven Architecture?

•Stylique reacts to events, but events are **within components**, not the main system structure.

•Event-driven architecture focuses on decoupled messaging → Stylique relies on direct component interaction.

•Pure event-driven adds complexity in tracking all events globally.

“Stylique has some events, but the core architecture is component-based, not fully event-driven.”

4. What Would Happen If We Used Event Driven Architecture?

- The system would become more complex due to managing many events and listeners.

- Debugging and tracking application flow would be harder.

- Direct and simple component communication would be replaced by delayed event handling.

5. MVC (Model–View–Controller) Architecture

An architecture that separates data, user interface, and control logic into three components.

- like a classroom

•Book = Model (Data)

Stores information like clothes, users, orders

•Board = View (Display)

Shows app screens, products, and styling ideas

•Teacher = Controller (Control)

Takes actions like button clicks, filters, add to cart

Makes the app easy to understand and manage.

5	MVC (Model-View-Controller)	NO	**Why it doesn't follows:** because its components require direct interaction with each other. MVC enforces a strict separation between Model, View, and Controller, which limits flexibility.

5. Why Stylique Doesn’t Uses Model-View Controller Architecture?

- Stylique uses MVC **inside components**, not for the entire app.

- MVC alone doesn’t capture **component modularity and independent communication**.

- Forces strict separation of UI, logic, and data globally, which is unnecessary.

“MVC is a pattern within components, not the overall architecture.”

5. What Would Happen If We Used Model View Architecture?

- The app structure would become tightly bound to UI flow, reducing component independence.

- It would not support Stylique's modular, feature-based components effectively.

- Managing large features like Cart, Catalog, and Recommendations would become harder.

6. SOA – Service-Oriented Architecture

A design style where software functionality is provided as reusable services.

- like different counters in a mall

•One counter for payments
•One counter for delivery
•One counter for customer service

Each counter works **independently** and can be used by **other apps too**.

6	SOA (Service-Oriented Architecture)	No	**Why it does not follow:** Stylique doesn't expose multiple independent services to external systems. APIs are internal, so SOA is unnecessary.

6. Why Stylique Doesn’t Uses Service-Oriented Architecture?

•Stylique is not a fully service-oriented system.

•Components are modular but not deployed as independent services communicating over a network.

•SOA adds unnecessary communication layers and complexity.

“Stylique is modular but does not require service orchestration across the network.”

6. What Would Happen If We Used SOA Architecture?

•It would introduce unnecessary complexity for Stylique's size and scope.

•Communication overhead between services would slow down interactions.

•Managing and coordinating multiple services would increase development effort.

7. Peer-to-Peer (P2P) Architecture

An architecture where all nodes act as both clients and servers without central control.

- like friends sharing files directly

•Friends talk to each other **directly**
•No teacher, no boss, no main server
•Everyone can send and receive data

Examples:

•File sharing apps
•Direct device-to-device sharing

7	Peer-to-Peer (P2P)	No	**Why it does not follow:** Stylique clients do not communicate directly with each other. All interactions go through the central server, so P2P is not applicable.

7. Why Stylique Doesn’t Uses Peer-to-Peer Architecture?

•Stylique is client-server based for user-server communication.

•P2P requires users to share data directly → not needed for Stylique.

•Security, consistency, and control become difficult in P2P.

“Stylique relies on centralized data and modular components, not peer-to-peer sharing.”

7. What Would Happen If We Used Peer to Peer Architecture?

•Data security and consistency would become difficult to control.

•The app would lose its centralized data management structure.

•It would be unreliable for managing user accounts, orders, and products.

8. Pipe-and-Filter Architecture

A style where data flows through a series of processing components.

- like a water filter system

•Water goes through **pipe 1** → filter
•Then through **pipe 2** → another filter
•Each filter does **one small job**

Examples:

•Data cleaning
•Image processing
•Step-by-step data transformation

8	Pipe-and-Filter	No	**Why it does not follow:** Stylique is interactive, responding to user actions, not a data pipeline. Data is requested and displayed on-demand, not processed sequentially through filters.

8. Why Stylique Doesn’t Uses Pipe-and-Filter Architecture?

•Stylique does not process data in a sequential pipeline.

•Components interact directly and independently, not in a linear chain.

•Pipe-and-filter adds unnecessary constraints and overhead.

“Stylique processes actions inside components, not through sequential filters.”

8. What Would Happen If We Used Pipe and Filter Architecture?

- The app would be forced into a linear data flow, which does not match Stylique's interactive nature.

- Components like Cart, Catalog, and Recommendations cannot work independently in a pipeline.

- It would add unnecessary complexity for simple user actions and interactions.

9. Broker Architecture

An architecture where a broker manages communication between distributed components.

- like a middleman

•Client does **not** talk directly to service
•Client talks to a **broker**
•Broker finds the right service and sends the message

Example:

•Customer → shopkeeper → worker

9	Broker	No	**Why it does not follow:** Stylique has direct communication between client and server. There is no middleware broker needed for message routing between components.

9. Why Stylique Doesn’t Uses Broker Architecture?

•Stylique has **direct communication** between components.

•Broker architecture adds a middleware layer for message routing → unnecessary complexity.

•Increases latency and reduces simplicity of component interactions.

“Stylique’s components communicate directly, so a broker is not needed.”

9. What Would Happen If We Used Broker Architecture?

•Introducing a broker would add unnecessary middleware, increasing complexity.

•Direct communication between components would be slowed down.

•It would create extra overhead, making the system less efficient and harder to maintain.

10. Component-Based Architecture

A system built from reusable, independent components with well-defined interfaces.

- like LEGO blocks

•Each block does **one specific job**
•Blocks can be **used again**
•Blocks can be **changed without breaking others**

10	Component-Based	Yes	Why it follows: Each component handles a specific responsibility and can interact directly with others, making the system more organized and easier to manage. It also makes the application more flexible, scalable, and easier to update.

10. Why Stylique Used Component Based Architecture?

•Stylique is **modular**, with independent components like **Cart, Catalog, User Profile, and Styling Recommendations**.

•**Components communicate directly**, enabling fast and flexible interactions.

•**Reusable and maintainable** – each component can be updated without affecting others.

•Supports **scalability** – new components/features can be added easily.

•**Easy to understand and develop** – perfect for app growth and team collaboration.

“Component-based architecture provides flexibility, modularity, and direct communication, making it perfect for Stylique.”

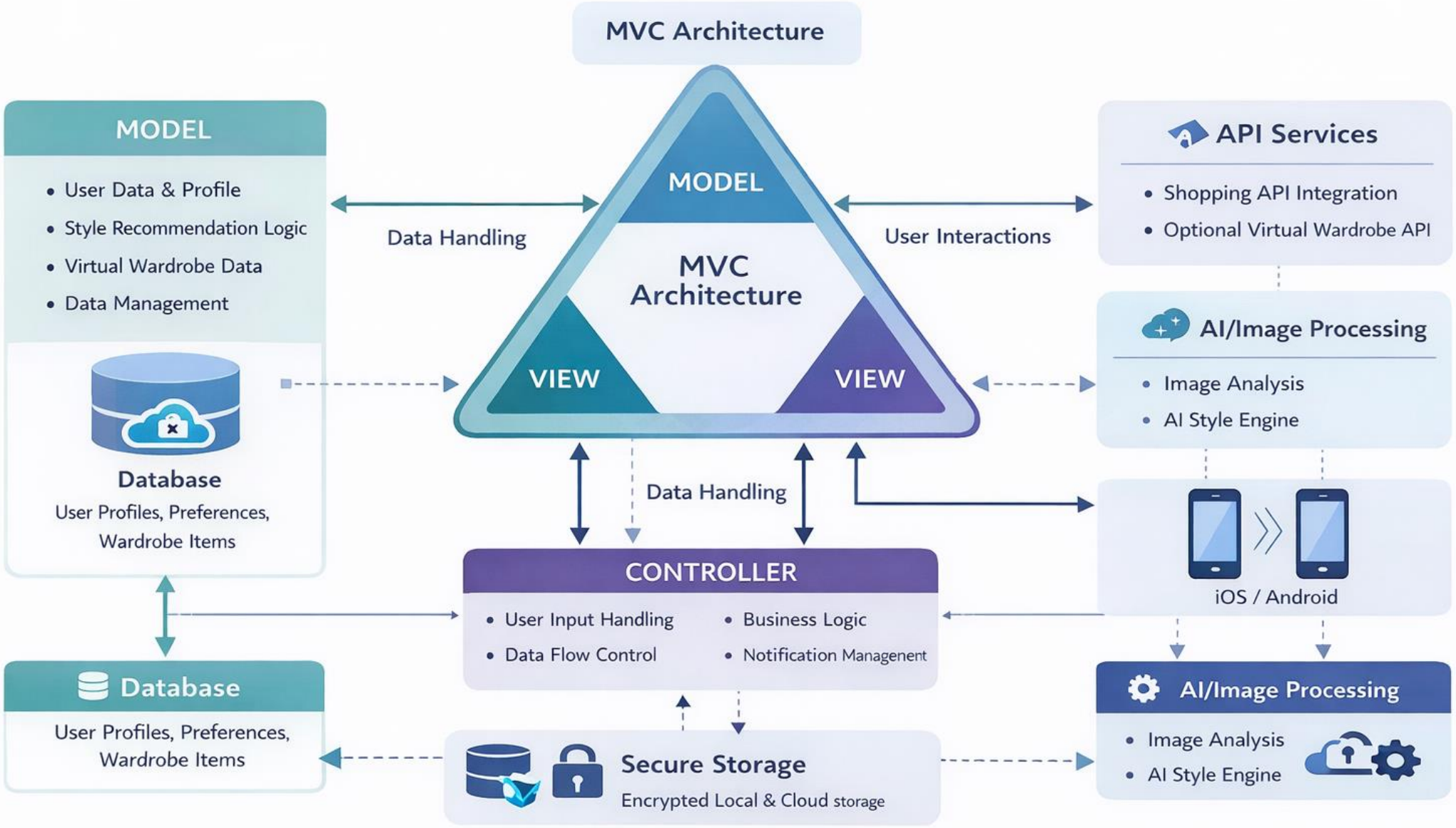
10. What Would Happen If We Used Component Based Architecture?

- Stylique would work efficiently, with **independent components communicating directly**.

- Components like Cart, Catalog, and Recommendations would be **reusable and easy to maintain**.

- Adding new features or updating existing ones would be **simple and scalable**.

Stylique Mobile Fashion Assistant App Architecture



Thank you