**Architecture Styles & Patterns Practical Task**

**Task**

1. Try to identify architecture styles being used in legacy and proposed solutions. List benefits and limitations for each of the identified style

**Legacy**:

Based on the legacy schema, it's clear that a monolithic architecture is being utilized. The application is packaged as a single executable WAR file that contains all necessary components. A load balancer is deployed on top of the application, enabling horizontal scaling to enhance performance and availability, as well as to serve static content for the frontend.

The advantages of this approach include its ease of deployment and testing, due to its simplicity. This straightforward structure is also easy for new developers to understand, making it simpler to add new features. However, as the application grows, it becomes increasingly challenging to support and maintain.

New style:

This architecture exemplifies a **microservices architecture** that is event-driven, cloud-native, and highly modular. It emphasizes scalability, flexibility, and the ability to evolve iteratively. The use of an API Gateway, coupled with policies for rate limiting, fault tolerance, and caching, demonstrates a focus on security, performance, and reliability.

The monolithic application has been decomposed into separate, loosely-coupled components or services like grading service, user preference service, users & role management service, payment service and so on. Each representing a distinct business entity. This enables each service to be developed, deployed, and scaled independently.

Each service maintains its own database, promoting data autonomy and allowing each service to choose the most appropriate storage technology for its needs

The system leverages an event-driven architecture where services communicate asynchronously by sending events to a central Message Broker. This ensures loose coupling between services and enhances the system's scalability and fault tolerance.

The architecture utilizes an API Gateway to manage and expose APIs securely. Instead of exposing services directly, all traffic is routed through the API Gateway, which passes requests to Backend for Frontend components, ensuring a clear separation of concerns and optimized API responses tailored to different client needs.

Static content and cached materials are stored in a Content Delivery Network (CDN), improving content delivery speed and reducing the load on backend services.

Authentication is delegated to a Cloud Identity Provider that integrates with the legacy Active Directory (AD) and social networks. This provides a flexible authentication experience tailored to different consumer types, enhancing security and user experience.

1. Try to figure out what architecture patterns has been applied within solution. List what are the problem given pattern is intended to solve.

Based on the schema I can say that some patterns can be applied there. The first one can be Valet Key architecture pattern. It can be useful to provide direct access to datasources like courses, media storage etc.

The second one I can emphasize is Static content hosting pattern. Its implementation is CDN. It can be useful in terms of cost effective and efficient for delivering slow-changing content and media.

The next one I can emphasize is Strangler pattern. This pattern is used to migrate legacy app to new technologies.

Based on the schema I can see Cache Aside Pattern. This pattern is useful here because the data like course content etc requests a lot but changes rarely.

The next one is Federated Identity pattern. This pattern helps to separate auth and authZ. And thanks to this pattern It will be easy to manipulate user-account. In addition to it this pattern provides possibility to authorize via social networks like Facebook, google ect.

.