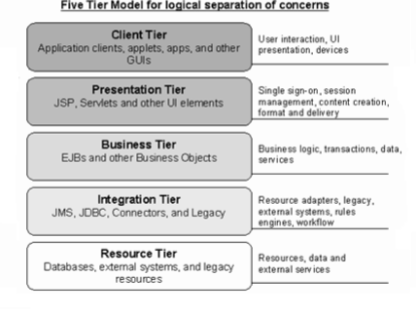
Group 3: Film Review Application

Milestone 2: Systems Architect

Nana Owusu-Ansah

**Architecture Model**

The n-tier client/server architecture in software engineering is well suited as it provides several benefits. With separation of concerns the distinct division of responsibilities among various layers is enforced by the architecture. As a result, the application is easier to comprehend, maintain, and expand and becomes more modular. Every layer serves a distinct purpose, and modifications made to one do not always impact the others. You can lessen inter-layer dependencies by making sure that every layer can only access the layer that is immediately below it. This aids in limiting the impact of modifications made to one layer on the others and isolating those changes.



The architecture makes it possible to incorporate redundancy, which can improve fault tolerance and guarantee system availability. Furthermore, it offers a strong basis for scalability, as you can improve fault tolerance and guarantee system availability. It also offers a strong basis for scalability because each layer can be scaled independently to accommodate growing loads. Future modifications are anticipated by the architecture. You can make changes to the relevant layer without impacting the remainder of the application when the requirements for presentation, data access, or business logic change.

If a single layer is significantly changed, one of the challenges with this architecture model is that the entire application must be redeployed. Applications that need continuous availability such as ours might find it undesirable to have planned downtime during releases because of this.

UML Diagram showcasing data elements


**Presentation Tier**

A few major components listed below are:

User Interface (UI): This part oversees showing users the application. It offers the user interface for interacting with the system, watching highlights, and uploading game footage. User authentication enables users to control their accounts and log in.

Highlights Interface: Offers play-by-play information, annotations, and controls for playing back videos in an easy-to-use manner for viewing game highlights.

**Application Tier**

A few major components listed below are:

The main element in charge of examining uploaded game footage, recognizing plays, and producing highlights is the game analysis engine. It contains business logic and algorithms for game analysis.

User Management: Operations pertaining to users, such as account management, authentication, and registration, are handled by user management.

Play Data Management: Oversees the storage, retrieval, and analysis of play-by-play data.

Collaboration and Sharing: Enables users to work together by sharing highlights and analysis with coaches or teammates.

**Abstraction Layers:**

A major component is the

API Layer: An abstraction layer that provides access to several APIs for the Presentation Tier and Application Tier to communicate with one another. This layer lays out unambiguous interfaces for user authentication, data retrieval, and communication with the game analysis engine.

We will take the following actions to address coupling and cohesion in our application. To reduce dependencies, use clearly defined interfaces and API contracts between layers. This makes it possible to modify a layer without affecting other areas of the program. Make sure that every module or component in a layer has a single, distinct duty. For instance, the "User Management" module should oversee user management, while the "Game Analysis Engine" should only concentrate on game analysis which should grant high cohesion. To encapsulate communication between the Presentation and Application Tiers, the API Layer serves as a clearly defined abstraction layer. High cohesion and loose coupling are encouraged by this abstraction. Our team can create a reliable, modular, and maintainable application by adhering to these guidelines.

**Architectural Elements**

I will identify important components pertaining to hardware, security, user interface, internal interface, and external interface for our application's architecture.

**Security:**

Authorization and Authentication of Users: To guarantee safe access to the application and its contents, put strong user authentication and authorization procedures in place.

Data Encryption: To safeguard user information and game footage, encrypt critical data both during transmission and while it is at rest.

**Hardware:**

Hosting Infrastructure: To host the application, choose the right hardware resources (servers, storage, networking), making sure they are scalable and reliable.

Video Processing Hardware: To improve the speed and precision of game analysis, take into consideration specialized hardware for video processing.

**User Interface:**

Web-based User Interface: Create a web-based user interface where viewers can interact with highlights, upload game footage, and access tools for analysis.

Make sure the user interface (UI) is responsive to different screen sizes and devices by using responsive design.

**Internal Interface:**

Business Logic API: To enable communication with the game analysis engine and other business logic components, create internal APIs that connect the Presentation Tier and Application Tier.

Data Access APIs: Establish interfaces between the persistence layer and data access.

**External Interface:**

Integration with Video Platforms: Create interfaces with streaming services or video platforms to upload and edit game footage.

Third-party Authentication: To ensure a smooth login process, integrate with authentication providers.

Data processing, retrieval, and storage are managed by the persistence and data tier. The main elements of this tier are as follows, along with their functions:

**System for managing databases :**

Goal: The database management system oversees keeping track of structured data, such as user accounts, play-by-play information, metadata from game footage, and annotations made by users.

**File and Storage System:**

Goal: The purpose of this component is to store user-uploaded video files. Large media files are stored effectively and securely thanks to it.

**Play Data Archive:**

Goal: Play-by-play information gleaned from game footage is stored and managed for this purpose. Time stamps, player locations, and play descriptions are among the details it contains.

**User Data Bank:**

Goal: Manages user-related data, such as preferences, account details, and annotations created by users. It guarantees the security and privacy of user data.

**Caching Mechanism:**

Goal: The implementation of a caching mechanism is intended to enhance data retrieval and analysis performance. reduces latency by caching frequently accessed data.

**Data Processing Mechanism:**

Goal: In charge of analyzing game footage to pinpoint plays, produce highlights, and give viewers insights. For effective video analysis, this part can make use of specialized hardware.

These key elements in the persistence and data tier enable our application to reliably and securely store, manage, and process data to give users insightful game highlights and analysis.

References:

<https://dev.to/zachgoll/introduction-to-software-architecture-monolithic-vs-layered-vs-microservices-452>

<https://rubygarage.org/blog/monolith-soa-microservices-serverless>

13-Architecture.pdf