**Architecture Overview**

**System Design**

In this Modular Streaming Platform is I make search and designed a layered architecture centred around the **MediaSource** interface. And in this interface-driven design also enables a flexible integration of five structural design pattern as what the activity asked us to execute, and each one of them addresses specific extensibility requirements. \

**Architecture Layer**

**Core layer:** The **MediaSource** interface have the these following functions, **play(), pause(), stop(),** **getSourceInfo(),** **MediaPlayer** by implementing these interfaces and organizing the playback using some injected sources with the function of rendering strategies.

**Adaptive Layer:** I also build three adapters here, and these are the following **LocalFileAdapter**, **HLSAdapter**, **RemoteAPIAdapter** converting multiple media sources making it into a basic **MediaSource** interface, it also enables the player to function with any various sources transparently.

**Decorative Layer:** The **MediaDecorator** and its subclasses and by these following three decorators: **SubtitleDecorator**, **WatemarkDecorator**, **EqualizerDecorator** in this section it also wraps various sources to add more features dynamically. These decorators can be also chained together in order to combine multiple enhancements for the future developments.

**Composite Pattern:** The **MediaItem** that can be found in my code is an interface enables hierarchical playlist, along side with the **MediaFile** it also represents also individual items, while **Playlist** aggregates various items and it also includes the nested playlist.

**Strategy Layer:** The **RenderStrategy** interface with **HardwareRenderStrategy** and **SoftwareRenderStrategy** allows the user to execute a runtime switching between **GPU** and **CPU** and the unique factor of it, is it rendered without modifying the player logic.

**Proxy Layer:** The last of them is the CachingProxy intercepts media requests, in this last section of the pattern layer is it serves cached content when if there is an available.

PATTERN INTEGRATION

All these patterns are composed in the **MediaSource** interface. The sources are adapted, decorated with features, and also proxied for an effective caching, played by a selected strategy, and organized in composite playlists. In this design ensures modularity, extensibility, in order for an easy maintainability while eliminating the code duplication at the same.