Architecture of CineStream

VJS PRANAVASRI

Compiled October 12, 2022

This document is a MicroService Architecture of CineStream - An OTT Platform

---

1. INTRODUCTION

CineStream is an OTT startup. It aims on providing users content at the comfort of their homes. This is made available to users in their TV, Computer, Phones and other media devices.
Cinestream has 2 features that makes it stand apart from all other existing services.

- It allows for watching media from other OTT platforms (like prime video, netflix, hotstar etc). This would mean all your content at one place with one subscription
- Cine stream also has the feature to allow pay per view, i.e. pay only for what the user is watching.

Alongside these cinestream offers a flexible subscription model which greatly helps user to utilise the service to the fullest.

- A regular monthly/yearly subscription service which will give access to the user to all the available content.
- A regular monthly/yearly subscription service which will give access to the user to a subset of all available services.
- pay per view, paying only for the content user wants to watch.

This kind of planning gives a greater amount freedom and control to user, maximising the chance that he will use cinestream over other platforms.
**User also gets selective discounts based on his metrics like watch time, likes and reviews which benefit both the platform and user*

2. REQUIREMENTS

There are many different types of requirements in software systems, but two of the most common are functional, non-functional. For our Cinestream before discussing the design, we'll broadly list out our functional and non functional requirements, this will help us scope the project and design it better. Functional requirements detail what the system should do, while non-functional requirements place constraints on how the system operates.

2.1. Functional Requirements

Functional requirements are those that specify what a system should do, while non-functional requirements are those that specify how a system should do it. In order to provide an OTT service, our platform must be able to support the following functional requirements:

1. Content ingestion: The platform must be able to ingest content from a variety of sources in order to make it available to subscribers. As this is one of the major feature of our model

2. Content management and delivery: The platform must be able to manage content in order to ensure that it is available to subscribers when they want it and deliver content to subscribers in a format that is compatible with their devices.

3. User management: A user must be able to register, login, update and view their details. And authorized users should be able to view data on platform.

4. Subscriber management: The platform must be able to manage subscribers in order to ensure that they have access to the content they want.

5. Billing and payments: The platform must be able to bill subscribers for their use of the service and collect payments from them. And also maintain payments across ott services.

6. Customer management: The platform must be able to manage customers in order to ensure that they are able to use the service.

7. Network management: The platform must be able to manage devices and networks in order to ensure that they are able to connect to the service and that they are able to support the delivery of content to subscribers.

8. Feedback: A user must be able to submit reviews and ratings for content on the OTT platform. This will be our user metrics. Our system should be able to use this feedback to give user discounts.

9. Recommendation System: Being able to provide personalized suggestions and recommendations to user based on the UserMetrics.

2.2. Quality Attributes

There are a few key quality attributes that are important for our platform.

1. Reliability: The platform must be reliable and able to handle a large number of users.

2. Scalability: The platform must be able to scale to accommodate a growing number of users.

3. Performance: The platform must be fast and responsive.

4. Security: The platform must be secure and protect user data.

5. Privacy: The platform must respect user privacy and allow users to control their data.

3. DESIGN

The platform could be designed in multiple different ways with and without using micoservices. We will be looking at the way CineStream is designed. Cinestream uses a microservices based approach. We'll start by first defining the contexts, then recognising and describing the microservices in the next section.

3.1. Context

Before looking at services or microservices, we broadly classify our problem into contexts. These form the crux of our platform. From our functional requirements we can broadly divide our problem into the following contexts.

1. User Management: This is the context for maintaining user registration, login and to tell whether a user is authorized

2. Payment: This will be responsible for payments across user, cinestream and 3rd Party OTT's.

3. Feedback System: This will be responsible for recording feedback from users and storing in a respective system.

4. Machine Learning Context: This will be for all the machine learning tasks, in our case the recommendation system will be aprt of this context.

5. Content Management: This is responsible for managing content between user and 3rd party platforms.

6. OTT Services: 3rd Party OTT services.

All the other services like video streaming, subscription etc will be internal parts of the mentioned contexts.

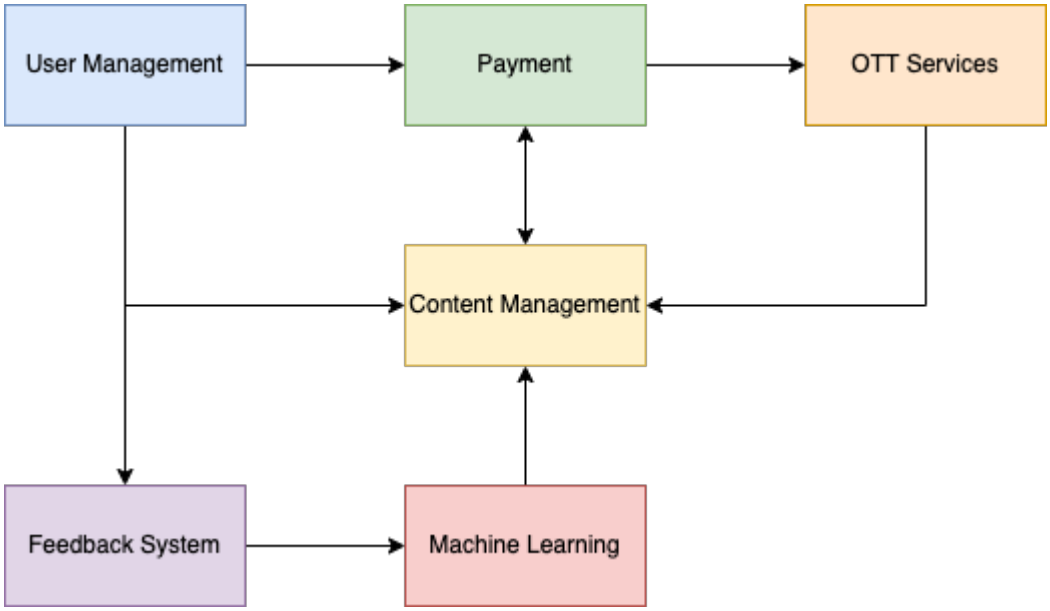

Fig. 1. Contexts of the Cinestream

3.2. Interaction Between Context

Here we'll briefly be talking about the interactions between the contexts as shown in Fig. 1. We'll look at how each context is independently dependent on one another. One thing which we can clearly notice is how loosely coupled the contexts are with minimal inter dependencies.

- User Management: User in itself stores registration and auth data. It interacts with *Content Management* for view content and initiating order[1], *Payment* to complete payment towards an OTT service, *Feedvack System* to leave feedback for either the content or the service.

- Payment: The payment system creates order for *ott services* and our *Content Management System*. These orders are forwarded to respective parties to complete the payments.

[1]We refer to a payment as an order. as per our terminology(rather razorpays), before any payment an order must be created at the payment gateway. This order os not the same as the order placed for the content

- Feedback System: Feedback system solely stores all types of feedback, and sends it to *Machine Learning*
- Machine Learning: This takes care of a couple of important tasks. This uses the feedback received to tailor user specific discounts and recommendations and sends it to content management.
- Content Management: This plays a central role in multiple ways. This Send data to *Payment* services to create order. It sends the order to *User Management* so that payment can be done.
- OTT Services: These send their respective content and payment orders to *Content management*.

4. MICROSERVICES

In this section we actually talk about all the services(micro services) we have implemented, and how we ensure the most efficient way of communication and ensure loose coupling. Now that we are clear about the contexts, We'll now further divide and list out all the services needed for our implementation.

4.1. Functional Logic

The way we want to implement the whole service is as follows:

- Any user can register and login at CineStream.
- Any OTT service can contact cinestream and get their PPV(Pay Per View) content on cinestream. They would be required to follow certain guidelines with respect to their api gateways and adhere to how cinestream consumes data.
- Each platform provides the per content cost through the said communication channels.
- If a user wishes to subscribe, For every video the user watches we will be sending a fixed amount to the respective OTT service to which the video belongs to (only if the user is in a proper[2] subscription)
- If a user wishes to Pay and watch each time, then 20% of the cost will go to cinestream and the rest will be sent to the respective OTT service.

[2]By proper we mean to say that, the platform might be providing multiple subscription plans and not all plans will allow all content.

- For payment, we use razor pay gateway. For every time a user wants to purchase, an order is created with the details and payment is handled by razor pay.
- For OTT services, the payments will be done on a monthly basis, to minimalise transaction charges. All the amount accumulated over the month can be requested as a order by an OTT platform and we will be sending them the amount. A platform can even prematurely[3] request to cash in their amount in which case cinestream will be taking 5% of total transaction charges as service fee.

Now I'll briefly go over how security for a user and while streaming for OTT is ensured. This has been shown in Fig 2[4]

- For User authentication we use JWT (JSON web tokens). These are generated for every login session with a timeout and are discared once a user disconnects from a session for a certain amount of time.
- The JWT are made up of a secret key and user data together so as to make sure no one else is able to generate one. In the backend we check for both correctness and validity of a token a user claims to hold. This is done by storing the token both at frontend and backend.
- Now if an user wants to watch any video which he has purchased, he'll directly send a request to the OTT service, Now the OTT service can directly talk to user auth system, to verify if user is valid and whether the user has actually purchased the content. Once verified the OTT server can start serving the video to the user.

And the final part which I will describe would be the feedback system logic. We try to make this as fair and beneficial as possible for every party in the loop. I'll try to go over them in a partially case wise scenario.

- Every user can give 3 types of feedback: Liking content, Commenting on Content, Reviewing OTT Service.
- The Machine Learning model will accordingly incentivise or penalise a user:

    1. Incentives 'like' action but also make sure that a user is not spamming likes, in which case he'll be penalised.

[3] Before the end of a month

[4] This is not the exhaustive list of interactions. This just shows what happens when a user wants to stream a video along with a minimal set of basic dependent actions

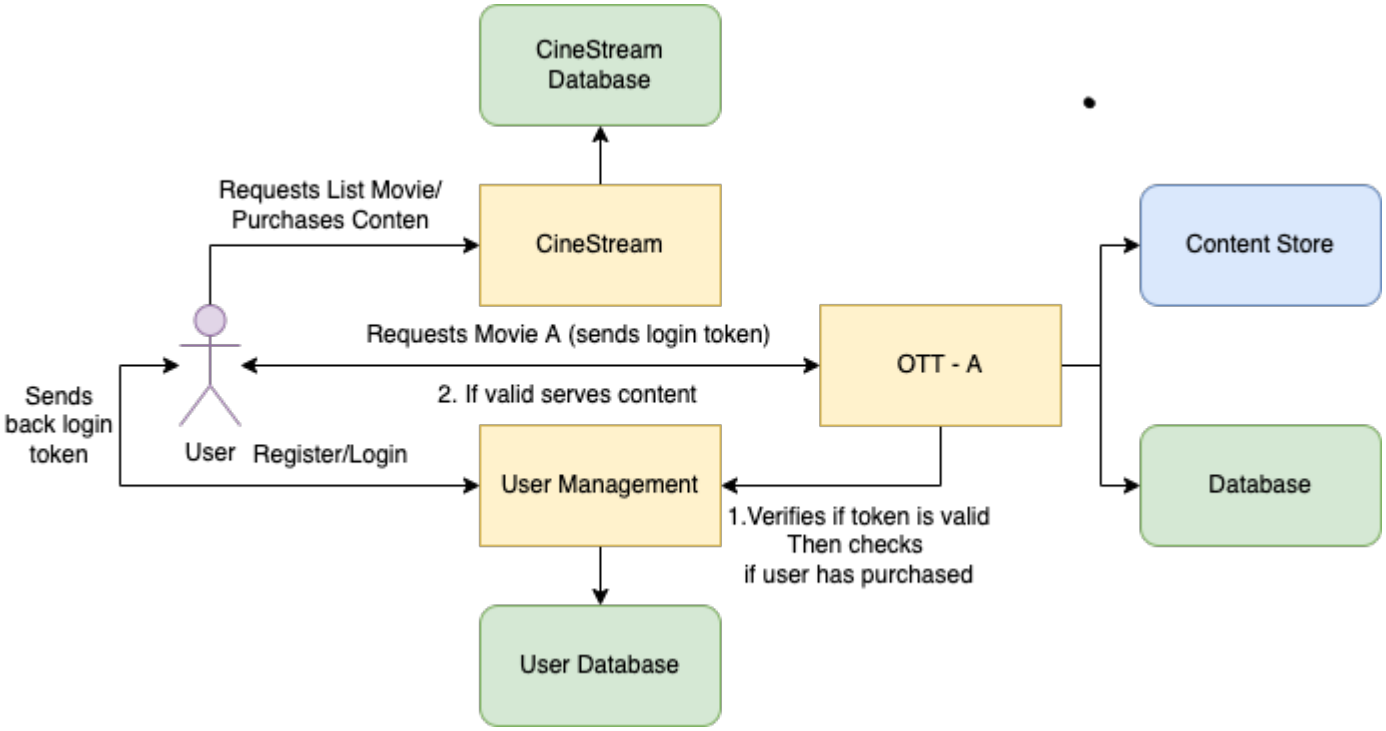

Fig. 2. Security Workflow of Cinestream

2. Checks the comment for spam or explicit content. If a user has a lot of this he'll be unable to comment anymore along with not receiving any rewards.

3. The reviews against OTT will be used both to incetivise/penalise the user and the OTT service respectively.

- These Incentives and Penalties are counted for a user and are used to decide how much a user can get a discount if he has enough incentives.
- Similarly we'll also use the reviews against content of a particular OTT service and the against the OTT service itself to gauge popularity and hence accordingly send some extra percentage as reward giving the OTT service an incentive to to push out better content.
- The more the users, the more the content, the more the revenue we gain.

With this we can show how our feedback system is a win win situation for every party involved.

4.2. Services

- OTT-X Service(s): all the third party services each being a distinct service. Each will be having their own database where they will be storing the total amount that cinestream owes them.

- RazorPay Service: This service talks to all the services concerned with payment to take the payment forward.

- CineStream Backend: This service mainly takes care of intercation with other OTT services, collecting the content and and redirecting user to respective services when needed. This maintains it's own database which stores all orders and payments with user and across other OTT services.

- User Management: This has the main logic for user authentication including user registeration, login, session management.

- Feedback Service: This is a service responsible for taking all types of feedback. Detailed description has been provided in the functional description above.

- ML Service: For dynamically giving discounts and filtering content.

5. METHODS

For achieving this multiple ways exist. But we choose the following.

5.1. RESTFUL API

We use this for allowing communication to and from a service. REST api's are widely used and over https are really secure too. using REST api's makes it very easy for the end user to consume our application, and also for other services to communicate with us. In the scenario where there are different communication protocols like grpc or SOAP we can use an API Gateway[5], Which will simplify communications by acting as a central channel.

5.2. Service Registry

This section we effectively talk about how our application scales and handles load. This acts as an intermediary for load balancer (or in our case our user) to tell which service he can access through which route. We can see Fig 3 for a reference of how in our case the service registry can be implemented.[6]. Instead of fearing dead servers or invalid requests, we can be assured that We will always be pointed to the best server possible. Service Registry itself doesn't ensure this, but it allows us to query the available servers to make the choice for ourselves. For scaling a load balancer can be attached just before service discovery to allow a perfect scaling experience.

[5]Our current model doesn't need or include an API gateway, we are just proposing it as a method for future needs

[6]We have only shown it for OTT platform but we'll be following similar suite for other services in our control

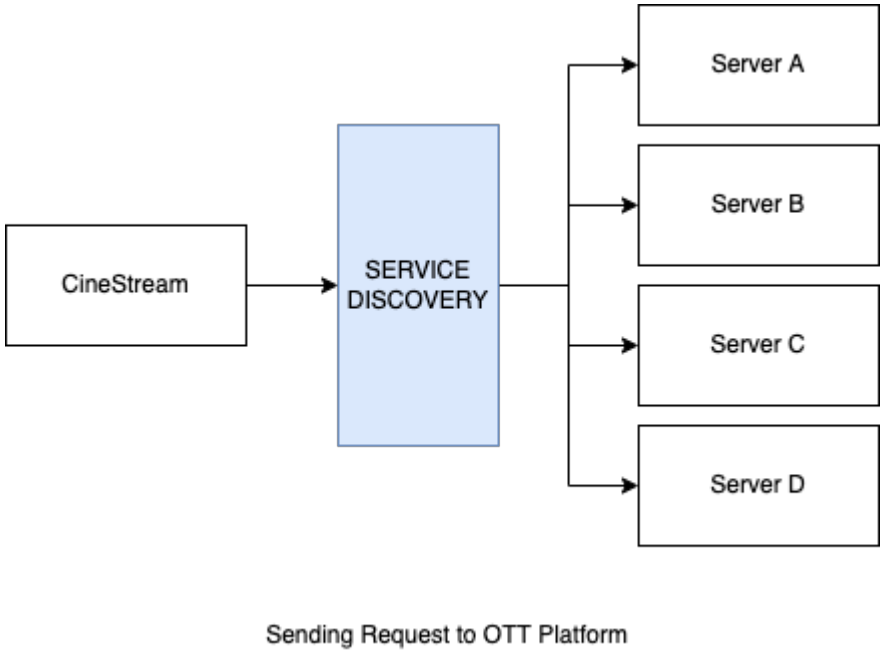

Fig. 3. Service Registry of Cinestream

5.3. Database

We use a NOSql database(Mongo) as that allows us to easily modify our collections, and work with our data without having to break dependencies or tables. This comes with a disadvantage of not having a defined structure, but it always ensures that insertions do not fail as long as they satisfy certain criteria which is very useful when we want to actively develop and improve an application.

6. ASSUMPTIONS/ENFORCEMENTS

We make the following assumptions for the entirety of the design

1. We expect that the said 3rd party ott platforms have apis exposed which provide details on per content availablity and pricing.
2. We assume that the said third party applications will be interacting with payment gateway to create orders based on all user purchases from platform.