DECLARATION

Name: Karan Dua

Date: 25-04-2023

Final Assignment

Designing a Globally Accessible Distributed Service
Distributed Systems - CS7NS6

Submitted by:

Karan Dua (21331391)

Introduction

Content streaming services like Netflix, Amazon Prime have become a fundamental part of our daily live in this digital age. Furthermore, these services provide us with access to a vast library of not only TV shows and movies, but also lot of educational and informational content. Therefore, these content streaming services can be used for entertainment as well as to gain more knowledge from various documentaries and educational content available on these services. Another major benefit of these services is that it can be access from anywhere around the world, thus making it easier for the user to access their favourite program at their convenience.

One of the key challenging tasks for such a service is to manage user accounts and their subscriptions. This becomes an even bigger challenge if the service is accessed around the globe and has user base in multiple countries. To resolve these issues, a customer and account management system can be used in this situation to manage user subscriptions, viewing histories, and enforce content and device restrictions from a single, centralized platform.

The primary objective of this report is to design a globally-accessible customer and account management system for a content streaming service. The key goal of this system design is to offer users seamless account management, allowing them to buy subscriptions and access content on the platform while imposing device and content restrictions at the same time. The system must also satisfy requirements for performance, scalability, availability, and reliability because it may need to support millions of users globally. The system design will be cantered on attaining these objectives while reducing the cost of running and utilizing the system.

Main Functions

The primary users of our content streaming service platform are the customers who either want to purchase a new subscription i.e., new users on the platform or modify the existing one or users who have a valid subscription and want to access the content on the platform. The other set of users for this platform could be the users with administrator access. These users can manage customer accounts and subscription plans on an as-needed basis. Thus, on the basis of these specifications the main functions provided by our content streaming platform are as follows:

1. Account Creation and Management

The account management function is a critical aspect for our content streaming service platform as it is the gateway for customers to join the platform as well manage their personal information on the platform.

The primary aim of this function is that customers should be able to create a new account on the platform, update their existing account details like updating their personal information to ensure that their details are up-to-date and accurate, etc. They should also be able to view their subscription plans. Furthermore, the customers should be able to delete an existing account if they want to leave the platform.

One of the key goals for this function is that this function should be very easy to use and navigate. This will enable customer to modify their accounts with minimal hassle.

2. Subscription Management

Subscription management function is one of the most critical functions for our content streaming platform. This will enable customers to maintain their subscription on the platform.

The primary objective of this function is that customers should be able to purchase a new subscription plan. Customers should also be able to upgrade their existing subscription plan or downgrade it. Furthermore, the customer should be able to cancel their subscription too if they want to at any given time.

The primary goal for this function is that it should be very reliable and highly available at all times as it involves financial transactions. Furthermore, it should scalable to handle large volumes of transactions efficiently. It should also be easy traceable and accurate.

3. Device Management

Device management function is another key aspect of our content streaming platform. It will ensure that the terms and conditions regarding the number of devices that customer can use simultaneously on our platform as per the customer’s subscription plan should be aways be enforced.

The key object of this function is that it should be able to enforce the policy of the maximum number of devices as per subscription plan of the customer. Furthermore, it should also provide an easy way to the customer to manage the list of devices that user has authorised. User should be able to remove or add new authorised device.

The primary goal for this function is that it should be very reliable as it enforces the policy of the platform. Furthermore, it should also allow customer to modify their device with minimal hassle.

4. Content Restrictions Management

Content restriction function is also a primary aspect of our content streaming platform. It will ensure that the terms and conditions regarding the location where a customer can stream content on our platform as per the customer’s subscription plan should be aways be enforced.

Furthermore, this function is also critical as it ensure that rights related to content distribution are respected.

The key objective of this function is that it should be able to enforce the policy of restricting the customer from streaming the content on the basis of his location. The system should not allow users to access the content that is not available in the country in which the customer has purchased the subscription.

The primary goal for this function is that it should be very reliable as it ensures platform's policies are enforced at all times. It should be highly available and scalable to handle large volumes of traffic all the times

5. Viewing History Management:

It is also one of the important features of our content streaming service platform as it is key component to enhance the user experience on our platform. It enables customers to continue watching from where they left off and also helps them discover new content based on their previous viewing history.

The primary objective of this function is to track user’s watch history and store it appropriately. It will also recommend new content on the basis of customer watch history.

The primary goal for this function is that it should be very scalable to handle large volumes of data, ensuring that customers can access their entire watch history without any performance issues. Furthermore, it should also allow customer to view their watch history with minimal hassle.

Requirements

The key requirements for our content streaming platform are that it should be able to provide our customers with a seamless and reliable experience while accessing any content on our platform. Furthermore, the customer should be able to access content from any device across different locations. This requires the platform to have robust account and subscription management, effective content restriction policies, and efficient watch history tracking. Thus, following are the fundamental Functional Requirement and Non-Functional requirements for our platform:

Functional Requirements

1. **User account creation and management**
   The users should be for users to create new account on the platform or manage their existing accounts on the platform.

2. **Subscription purchase, upgrade, and cancellation**
   The users should be able to purchase, upgrade, and cancel their subscriptions as needed.

3. **Device management device limit enforcement based on subscription plans**
   The platform should be able to limit the number of devices that can access the platform based on the user's subscription plan.

4. **Content restriction based on location**
   The platform should be able to restrict access to content based on the user's location and subscription plan.

5. **Viewing history tracking and storage:**
   The user should be able to view and resume content from their watch history. Platform should be able to recommend new content on the basis of customer watch history

Non-Functional Requirements

1. **Performance**
   The platform should be able to handle high volumes of traffic generated by the concurrent users of the platform. The users should experience minimal latency while using the platform.

2. **Scalability**
   The platform should be able not only be to handle large number of users but also the growing content on the platform. It should also be able to accommodate new features and functionalities.

3. **Availability**
   The platform should be highly available at all times, and user should not experience any or minimal downtime while using system. The platform should also have a plan for recovering from any disaster.

4. **Reliability**
   The platform should provide reliable and accurate service, with a minimal error rate.

5. **Security**
   The platform should ensure the security of user data as it contains personal information of user like credit card numbers, phone numbers and emails, etc.

Key Techniques

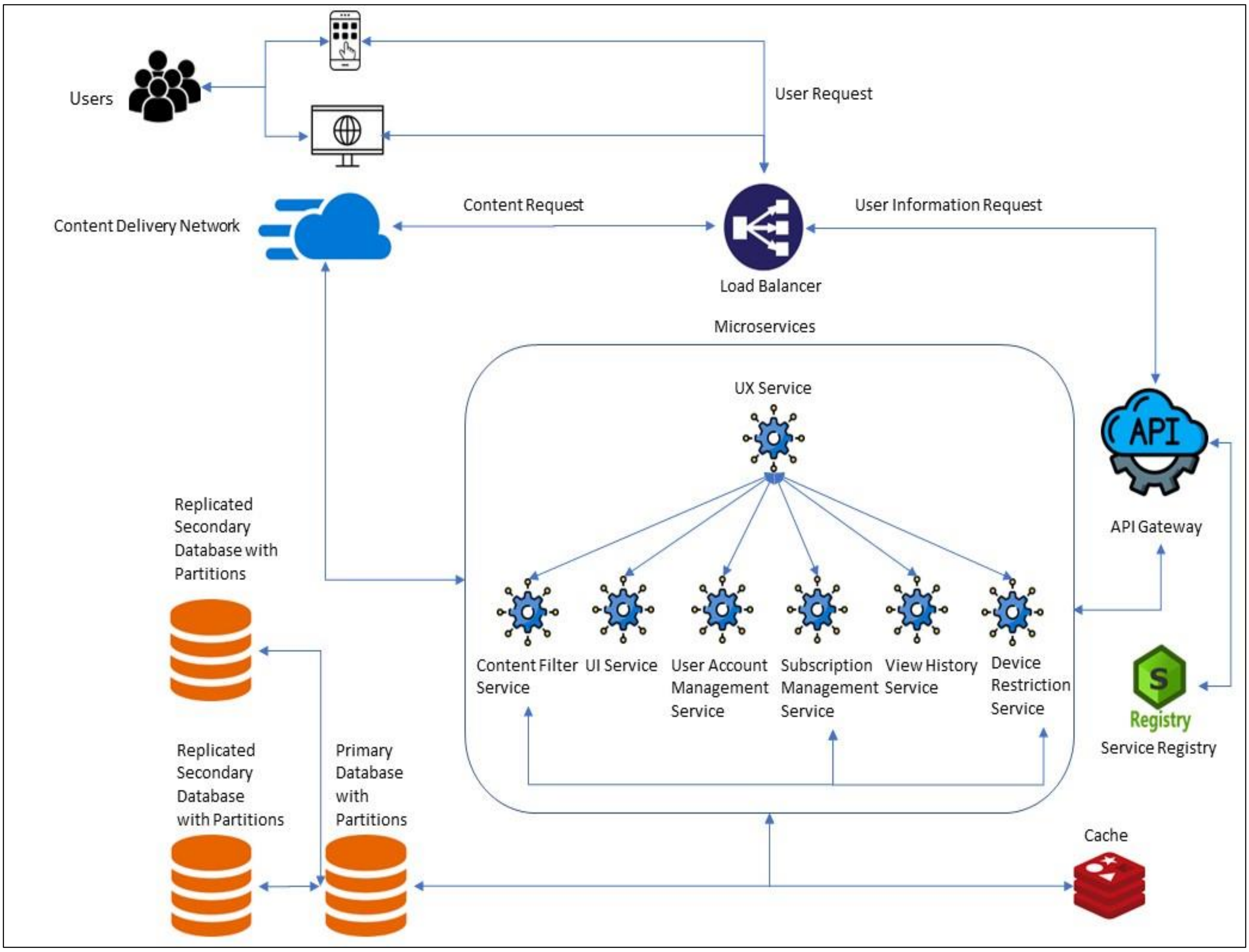
We expect users to be globally distributed and they can access the application from browser or mobile application. We also expect a lot of read and write operations on database as customers can buy a new plan or modify the subscription or they can view their watch history to get new recommendations. Based on this usage pattern, following techniques should be considered to achieve performance, scalability, availability, and reliability in the design:

1. **Caching**
   In order to improve the overall performance of the platform, we should implement caching techniques to store frequently accessed data, such as subscription details and viewing history, in the cache rather than accessing them directly from the database for each and every user request. This can help to reduce database load and improve system performance and improve response time.
2. **Load balancing**
   In order to make system highly available and improve its overall performance, we should implement load balancing mechanism on our platform. This will help us to distribute incoming user traffic across multiple servers and data centres which will improve system scalability. This will also help to ensure that the system remains available even if one of the components fails.
3. **Replication**
   In order to make system highly available and improve its overall performance, we should replicate the data across multiple servers. We should also create read replicas of the database to improve the read performance of the system and reduce latency for frequently accessed items by user like watch history, subscription plans, etc.
4. **Data Partitioning**
   In order to make system highly scalable and improve its overall performance, we should partition the database on the basis of the location and subscription plans of the users.
5. **Disaster recovery**
   In order to make system highly reliable, we should take frequent backups of the system including backup of the databases as well. We should also implement log checkpoint mechanism for our system to identify failed transaction. This will help us to rollback transactions in the event of any unforeseen circumstances like system failures. This will also help to make system more consistent in case of recovery after failures.
6. **Distributed Logs**
   In order to make system highly reliable and available, we will implement distributed logs in our platform. This will help us to track and monitor system activities across multiple servers and data centres. This will enable to get insights into any potential issues in the platform which will enable us to quickly respond to the problem ensuring a fast and smooth resolution of the problem. We can analyse system activities in real-time like all

user request, server responses and service performance. This will enable us to quickly troubleshoot and perform any optimisation if required.

7. **Service Orchestrators**
   In order to make system highly reliable, scalable and available, we will use orchestrators like Kubernetes that can help us to increase the overall performance of the system. Service orchestrators can help in automating the deployment, scaling and management of the containerized services in our platform. This will ensure that platform can handle an increasing amount of user traffic and workload demands. Service orchestrator can also provide the benefit of fault tolerance and disaster recovery in the platform as they automate the process of moving service container to healthy nodes on the even of a node failure.

Design and Architecture



Architecture Diagram for platform

* Icon Source: Google

Breakdown of the components in the architecture and justification:

1. **Load Balancer**
   It is used to distribute incoming user traffic across different servers and data centres.

2. **API Gateway**
   This service provides a single-entry point for clients to access multiple services on the platform. This service routes incoming requests to our other services responsible for different functionalities of the application. Furthermore, service failover handle mechanism is also placed in this service. This means that if a service is not available, API-gateway will redirect the request to failover handling service, so that user does not face any downtime. Before redirecting any request to a service, it internally interacts with service registry service to get information about the service.

3. **Service Registry**
   This service allows other services in the platform to register themselves and discover other services registered with the service registry. All the services self-register themselves to service registry during boot up. API Gateway service interacts with service registry to identify if a service is available in the platform or not before sending request to that service

4. **UX Service**
   This service will be used to handle user request. To fetch a particular component of UI, it will communicate with UI Service and to fetch user details it will communicate with backend services.

5. **UI Service**
   This service will be used to provide an interface to users to communicate with the platform by providing different UI components of the platform like interface for creating and managing their accounts, purchasing and managing their subscriptions, and viewing their watch history, etc.

6. **User Account Management service**
   This service handles all requests related to user account creation, modification and deletion. This service also handles user authentication and login to provide access to content to the user**. It will implement Account Creation and Management function**.

7. **Subscription Management Service**
   This service handles all request related to subscription management like purchasing new subscription or modifying an existing subscription. It also provides services related to

payment processing for subscriptions. **It will implement Subscription Management function.**

8. **Device Restriction Service**
   This service is used to enforce the policy of limiting the maximum number of devices per user that can access the platform as per subscription plan of the user. It uses Subscription Management Service to get subscription details**. It will implement Device Management function.**

9. **Content Filter service**
   This service is used to enforce the policy of restricting the user from accessing the content outside the country in which they purchased the subscription. It uses Subscription Management Service to get subscription details. **It will implement Content Restrictions Management function**

10. **View History service**
    This service is used to get watch history of the customer. It also provides recommendations to customer on the basis of their watch history. **It will implement Viewing History Management function.**

11. **Content Distribution Network**
    This service is used to provide the actual streaming of content to the user. To enforce content restriction based on customer location, it will use Content Filter service.

12. **Caches**
    This component is used to store frequently accessed information such as user account details, subscription details and viewing history in the cache to reduce the load on the database.

13. **Database**
    The database will store user account details, subscription details, viewing history, and other relevant data. All the personal information stored in the database will be encrypted for security reasons. It is also replicated to two secondary databases. The primary reason for replication is that it will ensure high availability and reliability of the platform if primary database goes down. Furthermore, it will also enhance performance as mot of read requests can be handled by secondary databases also. We will also partition both primary and replicated databases which will ensure that performance of system is at the optimal level.

Algorithms and Procedures

We can use the following algorithms to implement each of the functions that we have identified:

1. **Account Creation and Management Function**
   The user can create accounts using their email and password. The user can login into the system using these credentials to view the content and update any information through the user interface. All the credentials and personal information like credit card, phone number, etc. related details will be stored in Database only for security reasons.

2. **Subscription Management Function**
   The user can subscribe to a new plan or modify an existing one using the user interface. The subscription details of the customer and details of new available plans can be stored in the cache for faster retrieval and improve the performance of the platform.

3. **Device Management Function**
   The users can manage and authenticate the devices through the user interface. The details of authorised devices can be stored in database as this information is only retrieved on the request of the user. The Device Restriction Service will fetch the number of devices currently authorised by the user and then fetch user's subscription details from Subscription Management Service and enforce the platform policy for limiting the number of devices that can be used simultaneously to access the platform.

4. **Content Restrictions Management Function**
   The content distribution system will restrict access to content that is not available in the country in which the customer has purchased the subscription. This will be done on the basis of IP address or other location data. The Content Filter service filters content based on the user's country of residency and the available content in that country. This service communicates with the database and CDN to ensure users can only access appropriate content

5. **Viewing History Management Function**
   The users can get their watch history and new recommendations from the user interface. The details of user's watch history and recommendations will be stored in cache as user frequently accesses this data allowing for fast retrieval of history data when needed thereby increasing the overall performance of the system.

Conclusion

In conclusion, the architecture proposed for the content streaming platform includes various services and components such as User Account Management, Subscription Management, Device Restriction, Content Filter, View History, Content Distribution Network, Caches, and Database, to ensure high performance, scalability, availability, and reliability. The load balancer and UX Service also play important roles in distributing traffic and handling user requests. Overall, this architecture provides a solid foundation for building a successful content streaming platform.

References

- Serving 200 Million Online Subscribers - The Netflix Way - Smart Studios
- Understanding Design of Microservices Architecture at Netflix (techaheadcorp.com)
- Designing Netflix - High Scalability –
- A Design Analysis of Cloud-based Microservices Architecture at Netflix | by Cao Duc Nguyen | The Startup | Medium
- System Design Netflix - A Complete Architecture - GeeksforGeeks
- The Human Side of Airbnb's Microservice Architecture (infoq.com)
- How does Netflix scale push messaging for millions of devices?