CITS4401 Software Requirements and Design

Personalised video recommendation system

Group 7
Part 2

Group Members

23895642 Yi Ren
23957505 Zongqi Wu
24121014 Rui Qin
24125299 Yu Xia
24146595 Yechang Wu

Contents

Task 1

User Management System

The user management system is essential for handling all aspects of user information and activities on the platform. This subsystem facilitates user registration, login, profile management, and setup customisation. It is responsible for ensuring that user data is stored securely and easily accessible for updating and retrieval. Key features include password recovery, email authentication, and user preference management.

Browser and Search System

Browsers and search systems provide users with the capabilities they need to effectively find and interact with video content. The subsystem includes advanced search capabilities that allow users to query videos by keyword, tag, or category. It also includes filtering options to optimise search results based on criteria such as date, popularity, or relevance. The system is designed to dynamically process a large number of search queries, ensuring fast response times and the accuracy of search results. Integration with recommendation systems can enhance the search experience, recommending content based on user preferences and previous interactions.

Recommendation System

A recommendation system is a complex component that uses algorithms to analyse user behaviour and provide personalised video recommendations. The subsystem leverages data collected from user interactions, such as viewing history, likes, and shares, to tailor content recommendations to individual tastes. It aims to adapt over time, improving its recommendations based on ongoing user feedback and changing preferences. The System works closely with the content management system to ensure that suggestions are current and relevant, and it will also be linked to the Social Network System to learn what users like.

Notification System

The Notification System is essential for keeping users informed and engaged with the platform. It manages the delivery of alerts and updates regarding new video content, comments, likes, and other user interactions. This subsystem allows users to customise their notification settings, choosing which updates they receive and how they are notified (e.g., via email, or mobile notifications). It is designed to be scalable to handle a high volume of notifications without delay, ensuring users receive timely updates that enhance their interaction with the platform.

Content Moderation System

The Content Moderation System is responsible for maintaining the quality and safety of content on the platform. It employs automated tools and human oversight to review uploaded videos and user comments, ensuring compliance with community guidelines and legal standards. This subsystem features tools for flagging and removing inappropriate content, managing user reports, and providing feedback to content creators about moderation decisions. It is critical for protecting users and upholding the platform's reputation, facilitating a respectful and safe viewing environment.

Content Delivery Network System

The Content Delivery Network System optimises video content distribution to users worldwide. It reduces latency by caching content on multiple servers located in different geographical regions, ensuring that videos are streamed efficiently regardless of user location. This subsystem enhances user experience by enabling fast, reliable access to high-quality video content and supports scalable content delivery to accommodate growing numbers of users and data demands.

Content Management System

The Content Management System of the platform manages a range of essential functions, supporting various video formats for uploads and ensuring compatibility across different devices and bandwidths. It provides content creators with advanced tools for editing videos directly within the platform and options to schedule, publish, or unpublish their content as needed. Creators can also modify or delete videos and efficiently manage metadata, including titles, descriptions, tags, and categories, which boosts the searchability and accessibility of content. These comprehensive features ensure that creators can effortlessly control their content, while users can have an enjoyable viewing experience.

Channel Management System

The Channel Management System of the platform provides content creators with extensive customization options for their channel profiles. Creators can personalise their channel's appearance by selecting a profile picture and cover photo and crafting a compelling channel description to attract viewers. These features allow creators to establish their unique brand identity within the platform, making their channels more inviting and recognizable to users.

Monetization System

The Monetization System of the platform provides content creators with multiple methods to generate revenue from their work. Creators can monetize their content through advertising, by enabling ads to be displayed before, during, or after their videos, and earn based on ad views or clicks. They can also opt for subscription models, offering exclusive content to subscribers for a fee. Additionally, the system supports direct donations, allowing viewers to financially support their favourite creators directly through the platform. The Monetization

System should be directly linked with the Analytics and Reporting System to provide content creators with insights into their earnings, viewer demographics, and engagement statistics.

Social Network System

The Social Network System focuses on user interactions, such as commenting on videos, replying to comments, or giving a thumbs-up. Isolating the social network subsystem allows for a clear separation of concerns. The functionalities related to user interactions (commenting, replying, reacting) are contained within its module, making the overall architecture easier to understand. This will reduce dependencies between different system parts, although it is related to the Recommendation System. The Social Network System continuously feeds data such as likes and comments to the Recommendation System to enhance user experience. With the growth of user interactions, it could achieve better resource allocation and performance optimisation.

Administrator System

The Administrator System stands as a pivotal subsystem within the video platform, overseeing critical tasks such as user management, content monitoring, backup, and recovery. Equipped with specialized tools tailored for administrators, it empowers them to efficiently manage users and content, monitor system metrics, and execute backup and recovery operations. Besides, in concern of the importance of security to an administrator system, some extra boundaries should be established. To ensure comprehensive tracking of administrative actions and system events, an auditing and logging service will be included in this system.

User Feedback System

The User Feedback System is dedicated to capturing and managing user input regarding their platform experience. This subsystem allows users to report bugs, provide suggestions for improvements, and offer general feedback through surveys or direct input forms. It acts as a critical channel for user engagement, giving users a voice and helping to improve platform satisfaction and usability. The feedback collected is systematically categorised and analysed to identify common issues or trends, which are then used to prioritise development tasks and enhancements.

Analytics and Reporting System

The Analytics and Reporting System is a comprehensive data processing and visualisation tool designed to support decision-making across the platform. This subsystem aggregates data from various sources within the platform, including user interactions, content performance, system usage, and financial transactions. It processes and analyses this data to produce actionable insights, which are presented through dashboards and detailed reports. For content creators, it provides metrics on viewership, engagement, and revenue, aiding in the optimisation of content and monetisation strategies. For administrators, it offers operational insights, such as system performance metrics, user engagement statistics, and compliance monitoring.

Task 2

Architectural Pattern: Microservices Architecture

When choosing an architectural pattern for a complex interactive platform such as a video recommendation system, several factors must be considered. The microservices architecture is particularly suitable for this project because of its modular nature, which integrates well with the various functions required by the platform, such as user management, content delivery, and personalized recommendations.

Advantages

Individual Service Scalability

Each microservice, such as a recommendation system or content delivery network, can scale independently based on demand and load. This allows developers to use resources efficiently and scale specific components without scaling the entire application, which reduces operational costs.

Agile Development and Deployment

This platform benefits from microservices by enabling different teams to manage distinct parts of the application, such as user authentication, video processing, or data analytics. Each team can deploy updates to their service without disrupting others, facilitating faster updates and feature rollouts, crucial for staying competitive in the fast-paced media content industry.

Service Isolation

Fault isolation in microservices means that an issue in one service, such as a bug in the user feedback system, does not crash or disrupt the entire platform. Services can be designed to handle failures gracefully, using techniques like circuit breakers and fallbacks, which enhances the overall reliability of the platform.

Disadvantages

Complex Service Management

Managing multiple microservices on a video platform increases the complexity of overseeing numerous small components instead of a single, unified application. This includes challenges in orchestrating services, maintaining ample communication between them, and ensuring that they function cohesively as a part of the total system.

Data Consistency

The platform must synchronise user data across services like user profiles, viewing preferences, and interaction histories. Ensuring consistency across these services, particularly when they operate independent databases, requires implementing sophisticated synchronisation mechanisms, which can complicate system architecture.

Increased Network Latency

Communication between microservices might introduce latency, which is particularly detrimental in a video recommendation platform where speed is crucial for features like video loading times and recommendation delivery.

Microservices architecture offers a flexible, scalable, and resilient framework that is particularly suitable for a video recommendation platform, which requires rapid content updates, high availability, and a personalised user experience. The ability to independently develop and scale services allows the platform to remain innovative and responsive to user needs. Despite the challenges such as increased complexity and data management issues, the benefits significantly outweigh the disadvantages, particularly when appropriate technological strategies and tools are employed to mitigate these challenges.

UML Diagram See Appendix

Task 3: Addressing Security Vulnerabilities

Issue

Each microservice operates independently and communicates over a network in a microservice architecture. This setup increases the number of potential entry points for cyber attackers, increases the complexity of data exchange, and ultimately increases the risk of security breaches. Such vulnerabilities can lead to unauthorised access and data breaches and can compromise the integrity of the entire platform, as well as the reputation of the platform.

Proposal

- P1: Avoid the Use of Privileged Containers
    - Ensure that containers do not run-in privileged mode to prevent them from having full access to host system resources, which attackers could exploit.
- P2: Create Immutable Containers
    - Deploy containers in an immutable configuration, where no changes are allowed once the containers are in production. Any necessary updates, patches, or configuration changes are made by rebuilding the image and redeploying the container.
- P3: Use Container Native Monitoring Tools
    - Utilise monitoring tools designed for container environments to keep a close watch on all container activities, identifying and responding to threats in real time.

Argument

P1: Avoid the Use of Privileged Containers

+ Limits the privileges of containers, reducing the risk of attackers exploiting them to gain broader access to the system.
+ Even if a security breach occurs within one service, the entire platform is not severely compromised, preventing attackers from escalating privileges to access other critical

parts of the infrastructure. This is particularly important in a microservices environment where numerous services interact over a network, each potentially exposed to external threats.
- Requires careful management of container privileges to ensure that each container has only the necessary permissions.
- May add complexity to the deployment and management process, especially in large-scale environments with many containers.
- Some applications may require elevated privileges for certain operations, which could conflict with this security measure.

P2: Create Immutable Containers

+ Maintains the integrity of services throughout their lifecycle by preventing unauthorised changes to running application environments.
+ Ensures that all deployments are predictable and consistent, originating from secure images thoroughly scanned for vulnerabilities.
+ Facilitates direct rollback to previous versions in case of unexpected behaviour or new security flaws, minimising potential damage and downtime.
- Requires a disciplined approach to container management to ensure that containers are not modified after deployment.
- This may increase storage requirements due to the need to store multiple versions of container images.
- Continuous deployment processes may need to be adjusted to accommodate the immutability of containers.

P3: Use Container Native Monitoring Tools

+ Allows real-time detection and response to security threats in container environments.
+ Provides detailed information about container behaviour, network traffic patterns, and system calls, enabling the identification of potentially compromised activity.
+ Enables quick response to threats, minimising potential damage by providing visibility into container environments.
- May require additional resources and infrastructure to implement and maintain container-native monitoring tools.
- Monitoring a large number of containers can generate a significant amount of data, requiring efficient storage and analysis solutions.
- The effectiveness of monitoring tools may depend on their configuration and the expertise of the personnel managing them.

Criteria for Evaluation

Security Robustness:

- Goal: The security measures must substantially decrease the risk of unauthorised access and data breaches. This involves ensuring that the protections are comprehensive and address all known types of vulnerabilities associated with containerised environments.
- Evaluation: Measure the decrease in security incidents and breaches post-implementation compared to historical data. Regular penetration testing and vulnerability scanning will be used to assess the security posture continuously.

Performance Metrics:

- Goal: The security measures should not negatively impact the system's performance. Security enhancements mustn’t degrade the user experience or the efficiency of the microservice operations.
- Evaluation: Track key performance indicators such as response times, system latency, and resource utilisation before and after implementation. Any negative impacts should be mitigated by optimising the configurations or re-evaluating the solutions.

Cost-effectiveness:

- Goal: Security enhancements should provide optimal protection without incurring prohibitive costs. Budget considerations are essential to maintain the overall financial health of the platform operations.
- Evaluation: Calculate the return on investment of the security measures by comparing the cost of implementation and maintenance against the costs saved from avoiding security incidents and breaches.

Operational Impact:

- Goal: Implement security enhancements with minimal disruption to existing operational workflows. The solutions should integrate smoothly with current development and deployment processes without introducing significant delays or complexities.
- Evaluation: Monitor the deployment cycle times before and after implementation to ensure they remain within acceptable thresholds. Feedback from the operations team will also be collected to assess any operational challenges or improvements.

Resolution

Avoid the Use of Privileged Containers:

- Action: Modify container deployment configurations to ensure none run with elevated privileges that could potentially give wide access if compromised.
- Outcome: Containers will operate with minimal permissions necessary for their specific functions, significantly reducing the risk of privilege escalation attacks.

Create Immutable Containers:

- Action: Implement a strict policy where containers are treated as immutable objects. This involves setting up automated pipelines that build, test, and deploy containers from a secured image repository.
- Outcome: Any changes to the running applications require a full rebuild and redeploy of the container, thereby preventing unauthorised runtime modifications and ensuring consistency across all environments.

Use Container Native Monitoring Tools:

- Action: Deploy advanced monitoring solutions tailored for containerised environments.
- Outcome: Real-time detection and alerts on abnormal activities or security breaches, enabling rapid response and mitigation, thereby reducing potential damage from attacks.

Integration and Continuous Evaluation:

Security configurations and best practices will be audited annually for the latest standards and best practices of security measures based on emerging threats and vulnerabilities. The benchmarks related to security incidents and system performance are to be met both pre- and post-implementation. Security audits and performance evaluations must be conducted periodically to ensure the effectiveness of the measures already put into place. More importantly, they point to areas that need greater improvement in updating the framework of security so that it remains strong and adaptable to the new challenges.

Task 4

1. Facade Pattern

The problem: The video platform contains multiple complex subsystems (e.g., video processing, metadata management, content delivery), which can be overwhelming for client applications or user interfaces to interact with directly due to their complexity and interdependencies.

The solution: Implement a Facade to provide a simplified interface to these subsystems, which abstracts the complexities involved in the operations of uploading, encoding, and metadata handling of videos. The Facade handles all interactions with the subsystems, allowing clients to perform actions without knowing the inner workings.

Reasons for Using Facade Pattern:

- **Simplification of Dependencies:** Users of the platform do not need to understand or interact with the complex underlying systems directly. For example, when a user uploads a video, the Facade can handle the intricacies of file encoding, metadata extraction, and thumbnail generation without exposing these complexities to the user.

- **Improved Maintainability:** By isolating dependencies in a single interface, the system becomes easier to manage and update. Changes in the subsystems, like modifications in the content management process or updates in the video encoding algorithms, can be implemented without affecting the user-facing parts of the platform.

- **Enhanced Scalability:** The Facade Pattern allows the system to scale more gracefully by decoupling the components used by clients from the actual implementations of these components, making it easier to modify or replace one without affecting others.

2. Observer Pattern

The problem: The platform needs to notify multiple components about changes in the system state, such as new video uploads, changes in user preferences, or updates in video recommendations. Directly linking these components can lead to a tightly coupled system that is hard to manage and scale.

The solution: Implement the Observer pattern by defining a subject (publisher) and multiple observers (subscribers). The subject maintains a list of observers and notifies them of any state changes, typically by calling one of their methods defined in a common interface. This allows observers to update their state in response to changes without being directly coupled to the source of the changes.

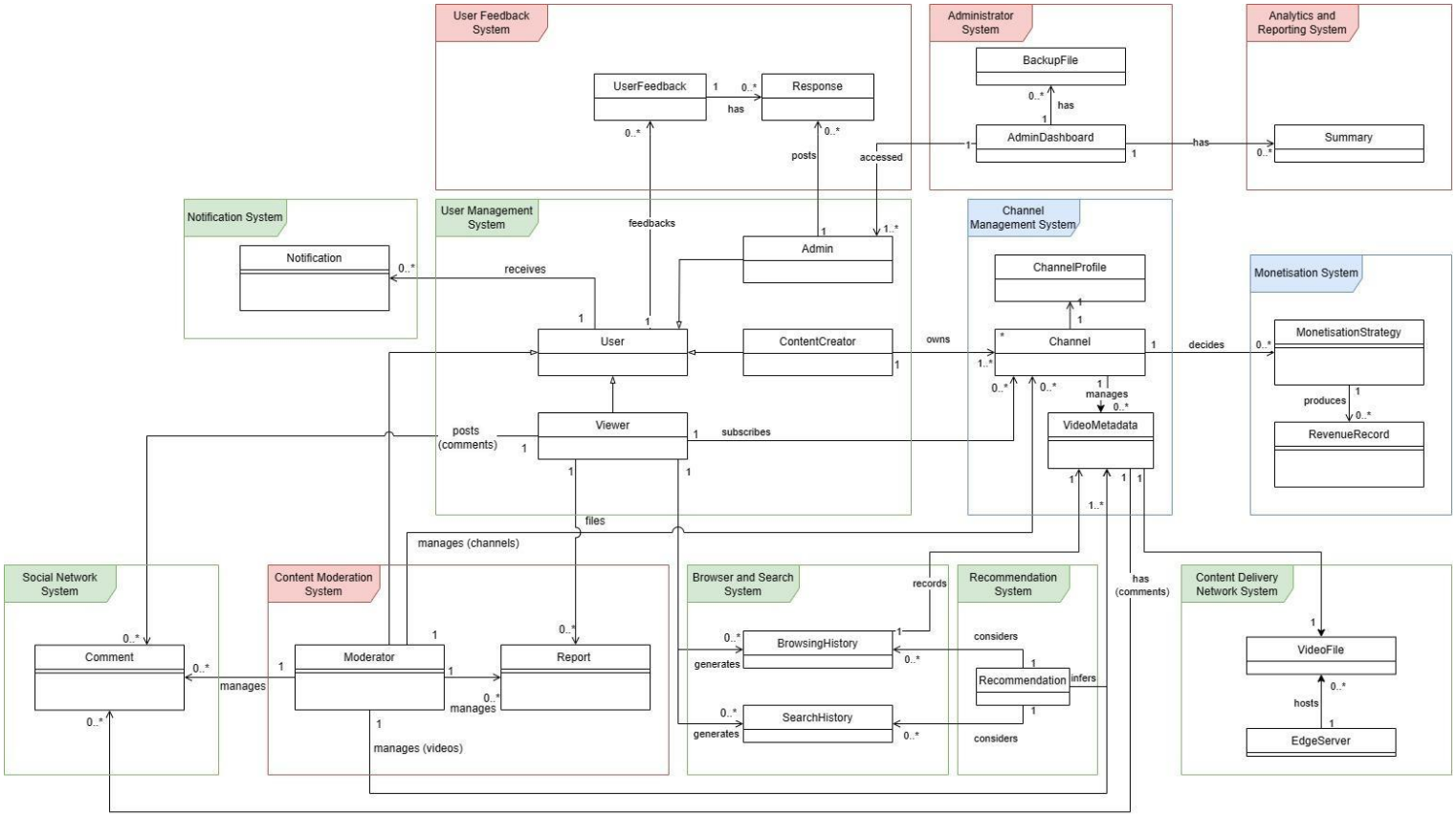
Reasons for Using Observer Pattern:

- **Real-time Interactivity:** This pattern supports the dynamic nature of the platform by enabling real-time updates to users as soon as new content is available or preferences are updated. It enhances user engagement by keeping the content fresh and aligned with user interests.

- **Decoupling of Components:** The Observer Pattern separates components that manage data from those that display data, making the system more modular and robust. For instance, when a new video is uploaded, the system automatically notifies all subscribed users, without needing to know the details about each subscriber.

- **Flexibility and Scalability:** Adding new types of notifications or new subscribers (observers) becomes straightforward, as the pattern allows for easy extension without significant changes to existing code.

Appendix


User Feedback System
UserFeedback
Response
has
posts
accessed
Administrator System
BackupFile
has
AdminDashboard
has
Summary
Analytics and Reporting System
Notification System
Notification
receives
User Management System
feedbacks
Admin
User
ContentCreator
owns
Channel Management System
ChannelProfile
Channel
decides
Monetisation System
MonetisationStrategy
produces
RevenueRecord
Viewer
posts (comments)
subscribes
manages
VideoMetadata
files
manages (channels)
Social Network System
Comment
manages
Content Moderation System
Moderator
Report
manages
manages (videos)
Browser and Search System
records
BrowsingHistory
generates
SearchHistory
generates
Recommendation System
considers
Recommendation
infers
considers
has (comments)
Content Delivery Network System
VideoFile
hosts
EdgeServer