Roadmap Videoflix

Phase 1: Planification et Préparation

Définition des Exigences

✅Langage Backend → Java (Spring Boot) pour la robustesse et l'évolutivité.
✅Langage Frontend → React.js ou Next.js (SEO, SSR) ou Vue.js (facilité de prise en main).
✅Base de données → PostgreSQL/MySQL (SQL pour utilisateurs/abonnements) + MongoDB (NoSQL pour métadonnées vidéos).
✅Stockage des vidéos → AWS S3, Google Cloud Storage ou MinIO (solution self-hosted).
✅Auth & Sécurité → OAuth2, JWT, Spring Security.

Architecture et Design

✅Concevoir l'architecture microservices.
✅Définir une base de données SQL (utilisateurs, abonnements) => MySQL

✅Définir une base de données NoSQL (métadonnées des vidéos). => MongoDB
✅Créer des schémas UML pour les flux de données et relations entre services.
✅Prévoir l'intégration d'un CDN pour optimiser la distribution des vidéos.

✅Microservices → Spring Cloud (Eureka, Config Server, Gateway, Circuit Breaker).
✅Communication inter-services → Kafka ou RabbitMQ pour la gestion des événements asynchrones.
✅CDN → CloudFront (AWS), Akamai ou Cloudflare pour distribuer les vidéos efficacement.
✅Cache → Redis pour accélérer la récupération des métadonnées et gérer les sessions.

Mise en Place de l'Environnement de Développement

✅Configurer les environnements (IDE, CI/CD, serveurs, monitoring).
✅Créer un dépôt GitHub/GitLab et définir les branches principales.
✅Prévoir un système de gestion des logs et monitoring (Prometheus, Grafana, ELK Stack).

✅IDE → IntelliJ IDEA, VS Code (pour frontend), Cursor.
✅CI/CD → GitHub Actions, GitLab CI/CD, Jenkins.
✅Monitoring & Logs → Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana).
✅Containerisation & Orchestration → Docker + Kubernetes (K8s, AWS EKS, GKE, AKS).

Phase 2: Développement des Fonctionnalités de Base

Backend

✅Implémenter l'authentification et l'inscription des utilisateurs (OAuth2, JWT).
✅Créer le service de gestion des vidéos (upload, stockage, récupération).
✅Développer le service de gestion des métadonnées des vidéos.

Backend

✅Auth & Utilisateurs → Spring Security + JWT/OAuth2 + Keycloak (IAM).
✅Upload des vidéos → AWS S3 avec SDK, Google Cloud Storage API ou MinIO.
✅Gestion des métadonnées → MongoDB (flexibilité) + Elasticsearch (recherche rapide).
✅API REST → Spring Boot (Spring WebFlux pour réactivité si nécessaire).

Frontend

✅Framework UI → React.js (Next.js pour SEO) ou Vue.js.
✅UI/UX → TailwindCSS, Material UI ou Chakra UI.
✅Gestion de l'état → Redux Toolkit (React) ou Pinia (Vue).
✅Streaming Player → Video.js, Plyr.js ou Shaka Player (support DRM & HLS/DASH).

Streaming et Qualité Vidéo

✅Encodage vidéo → FFmpeg pour convertir les vidéos en formats adaptatifs (HLS, DASH).
✅Streaming adaptatif → HLS.js pour React/Vue, ExoPlayer pour Android, AVPlayer pour iOS.
✅Génération de miniatures → FFmpeg + Lambda (AWS) ou service backend dédié.

Frontend

✅Concevoir l'UI/UX de la plateforme.
✅Développer les pages de connexion et d'inscription.
✅Créer les pages de gestion et de lecture des vidéos avec un design adaptatif.

Streaming et Qualité Vidéo

✅Implémenter le streaming adaptatif (HLS, DASH).
✅Ajouter des contrôles de qualité vidéo en fonction du réseau.
✅Générer automatiquement des miniatures et des prévisualisations des vidéos.

Phase 3: Développement des Fonctionnalités Avancées

Système de Recommandations

✅Intégrer un moteur de recommandations basé sur l'historique de visionnage.
✅Utiliser du machine learning (collaborative filtering, TensorFlow, Scikit-learn).

Fonctionnalités Sociales

✅Ajouter la possibilité de commenter et d'évaluer les vidéos.
✅Intégrer le partage sur les réseaux sociaux.

Gestion des Abonnements

✅Mettre en place différents plans d'abonnement (mensuel, annuel, free, premium ultra).
✅Intégrer une plateforme de paiement sécurisée (Stripe, PayPal).

Autres Fonctionnalités Avancées

✅Ajouter un mode hors ligne pour le téléchargement de vidéos.
✅Intégrer la diffusion en direct (Live Streaming - WebRTC, HLS, RTMP).

Système de Recommandations

✅Moteur de recommandations → Scikit-learn (collaborative filtering), TensorFlow (deep learning), ou Apache Spark ML.
✅Base de données pour recommandations → Neo4j (graphe) ou PostgreSQL (relationnel).

Fonctionnalités Sociales

✅Commentaires et avis → MongoDB (NoSQL, stockage flexible) + API REST.
✅Partage réseaux sociaux → OAuth2 avec API Google

Gestion des Abonnements

✅Paiement sécurisé → Stripe, PayPal API.
✅Gestion des plans → Spring Boot + PostgreSQL (gestion des abonnements et transactions).

Autres Fonctionnalités Avancées

✅Mode hors ligne → PWA (Progressive Web App) avec IndexedDB pour stocker les vidéos téléchargées.
✅Live Streaming → WebRTC pour faible latence, RTMP pour diffusion classique.

Phase 4: Tests et Validation

Tests Unitaires et d'Intégration

✅Écrire des tests unitaires pour chaque service.
✅Effectuer des tests d'intégration pour vérifier la communication entre les services.

Tests de Performance et Scalabilité

✅Réaliser des tests de charge sous différentes conditions.
✅Optimiser la scalabilité en configurant Kubernetes, auto-scaling AWS ECS/EKS.

Tests Utilisateurs et Sécurité

✅Faire tester la plateforme à un panel d'utilisateurs et recueillir des retours.
✅Effectuer un test d'intrusion pour vérifier la robustesse de la sécurité.
✅Implémenter un DRM ou un système de protection des vidéos (token sécurisé, watermarking).
✅Réaliser des A/B tests pour améliorer l'UX.

Tests Unitaires et d'Intégration

✅Tests unitaires → JUnit 5 + Mockito (Backend), Jest + Testing Library (Frontend).
✅Tests API → Postman/Newman, RestAssured (Java).

Tests de Performance et Scalabilité

✅Tests de charge → JMeter, k6.
✅Auto-scaling → Kubernetes HPA (Horizontal Pod Autoscaler), AWS Auto Scaling Groups.

Tests Utilisateurs et Sécurité

✅Tests d'intrusion → OWASP ZAP, Burp Suite.
✅DRM et protection → Widevine (Google), PlayReady (Microsoft), FairPlay (Apple).
✅A/B Testing → Google Optimize, Optimizely.

Phase 5: Déploiement et Maintenance

Déploiement Initial

✅Déployer la plateforme sur AWS, Google Cloud, Azure.
✅Assurer la haute disponibilité et la performance du service.

Surveillance et Maintenance

✅Mettre en place des outils de monitoring avancés (Prometheus, Grafana, ELK).
✅Automatiser les mises à jour et le déploiement avec GitOps (ArgoCD, Flux).
✅Gérer la maintenance continue et appliquer les mises à jour de sécurité.

Améliorations Futures

✅Ajouter de nouvelles fonctionnalités en fonction des retours utilisateurs.
✅Améliorer constamment l'expérience utilisateur et l'efficacité du système.

Déploiement Initial

✅Cloud Provider → AWS (EKS, S3, CloudFront, RDS, Lambda) ou Google Cloud (GKE, Firebase, Cloud Storage).
✅Infrastructure as Code → Terraform, AWS CloudFormation.

Surveillance et Maintenance

✅Logs et Monitoring → Grafana + Prometheus, ELK Stack.
✅Automatisation des mises à jour → GitOps avec ArgoCD ou FluxCD.

Stack Technologique Résumée

Catégorie	Technologies Recommandées
Backend	Spring Boot, Spring Cloud, PostgreSQL, MongoDB, Redis
Frontend	React.js (Next.js) ou Vue.js
Streaming	FFmpeg, HLS.js, DASH.js, WebRTC, RTMP
Stockage	AWS S3, Google Cloud Storage, MinIO
Base de données	PostgreSQL/MySQL (SQL), MongoDB (NoSQL), Neo4j (Recommandations)
Auth & Sécurité	OAuth2, JWT, Spring Security, Keycloak, DRM (Widevine, PlayReady)
CI/CD	GitHub Actions, GitLab CI/CD, Jenkins
Monitoring	Prometheus, Grafana, ELK Stack
Tests	JUnit, Jest, Postman, OWASP ZAP
Cloud & Orchestration	Docker, Kubernetes (EKS, GKE), Terraform

---