Movie Catalog MicroserviceA movie catalog management system built on Spring Boot 3 and JHipster, optimized for modern microservices architecture.□ Project OverviewThis project is a core component service within a microservices ecosystem, responsible for managing movie data sourced from the sample_mflix collection on MongoDB Atlas. The service is fully integrated with enterprise-grade standards.Key Features:Movie Management: Full CRUD operations with Partial Update (Patch) support for optimized data updates.Service Discovery: Service registration and management via Consul.Database Migration: Automated data schema versioning with Mongock.Observability: Integrated Prometheus (metrics) and Zipkin (distributed tracing) for performance monitoring.Resilience: Utilizes Feign Client combined with Circuit Breakers to prevent cascading system failures.□ Technology StackComponentTechnologyFrameworkSpring Boot 3.x, JHipster 8LanguageJava 17DatabaseMongoDB Atlas (Cloud NoSQL)Service MeshSpring Cloud ConsulMonitoringMicrometer, Prometheus, GrafanaSecurityOAuth2, JWT□ System ArchitectureThe service operates on a Stateless model, connecting directly to a MongoDB Atlas Cluster.Web Layer: REST Controllers providing APIs for clients.Service Layer: MovieService handling business logic and mapping optimization.Data Layer: MovieRepository extending MongoRepository for Atlas interaction.□□ Configuration & Setup1. PrerequisitesJava 17+Docker (to run Consul/Zipkin locally)MongoDB Atlas Account2. Critical Environment VariablesThe application uses configurations in application-dev.yml for the development environment:MongoDB URI: Connection string to Cluster0 on Atlas.Server Port: 8081.JWT Secret: Base64 encoded for enhanced security.3. Run CommandBash./mvnw □ Monitoring & API DocumentationSwagger UI: Access http://localhost:8081/swagger-ui/index.html to view API documentation.Health Check: http://localhost:8081/management/healthMetrics: http://localhost:8081/management/prometheus□ Technical Highlights & Problem SolvingDuring development, the following technical challenges were addressed:Hazelcast vs. DevTools: Disabled DevTools' automatic restart feature to prevent conflicts with Hazelcast’s caching mechanism.Fluent API: Implemented a Fluent-style Domain Model to make object initialization and Unit Testing more concise.CORS & Security: Configured a strict Content Security Policy (CSP) to mitigate XSS (Cross-Site Scripting) attacks.