Movie Booking Platform – High-Level System Design

# 1. Introduction

This document outlines the high-level system architecture for the movie booking platform. It defines the system components, microservices architecture, communication protocols, cloud resource usage, and non-functional aspects such as scalability, availability, and observability.

# 2. System Overview

The platform is a microservices-based distributed system. It handles user registration, movie browsing, ticket booking, payments, and notifications. The system is built using Java (Spring Boot) and is cloud-native, designed to run on Microsoft Azure Cloud infrastructure.

# 3. Microservices Architecture

1. 1. User Service – Manages user registration, login, and profiles.
2. 2. Movie Service – Stores and serves movie metadata.
3. 3. Theatre Service – Manages theatres, screens, and locations.
4. 4. Showtime Service – Maps movies to specific showtimes and screens.
5. 5. Booking Service – Handles seat selection, locking, and booking.
6. 6. Payment Service – Integrates with payment gateways for processing.
7. 7. Notification Service – Sends booking confirmations via email/SMS.
8. 8. API Gateway – Routes traffic to respective microservices and enforces security policies.
9. 9. Auth Service – Issues JWTs for authentication and supports RBAC.

# 4. Technology Stack

• Backend Language: Java 17

• Framework: Spring Boot

• Database: PostgreSQL

• Caching: Redis

• Messaging: Kafka / Azure Service Bus

• Authentication: JWT (or Keycloak)

• Cloud: Microsoft Azure (AKS, Azure PostgreSQL, Redis, Event Grid)

• Observability: Spring Boot Actuator, ELK stack, Prometheus, Grafana

• Containerization: Docker, Docker Compose, Kubernetes (AKS)

• CI/CD: GitHub Actions / Azure DevOps Pipelines

# 5. Communication Protocols

• REST APIs between microservices via API Gateway.

• Async messaging using Kafka or Azure Service Bus for decoupled communication (e.g., booking confirmed event).

# 6. Cloud Architecture Mapping (Azure)

• AKS: Host microservices

• Azure PostgreSQL: Store user, movie, booking, and theatre data

• Azure Redis: Cache user sessions and showtime data

• Azure Blob Storage: Store images, tickets, documents

• Azure Key Vault: Secure credentials and secrets

• Azure Service Bus: Async communication and event handling

• Azure Monitor: Logs and metrics collection

# 7. Key Architecture Diagrams

• Logical Architecture Diagram – Shows the interaction between frontend (later), API Gateway, microservices, databases, and message queues.

• Component Diagram – Illustrates individual services and their internal layers.

• Sequence Diagram – Represents booking flow from user login to ticket confirmation.

• Entity Relationship Diagram – Shows tables such as User, Movie, Theatre, Screen, Booking, Payment.

# 8. Security Design

• All APIs secured via API Gateway using JWT-based Auth.

• Services validate roles and permissions (RBAC).

• Secrets stored in Azure Key Vault.

• Input validation, rate limiting, and circuit breakers enabled.

# 9. Scalability & Reliability

• Horizontal scaling via AKS for all services.

• Redis caching for high read throughput.

• Retry, timeout, and fallback patterns for inter-service calls (Resilience4J).

• Active-active deployment strategy with auto-scaling.