Key Non-Functional Requirements and Design Solutions

1. Scalability **3**
- Requirement 3
- Design Decisions 3
  - Microservices Architecture 3
  - Horizontal Scaling 3
  - Event-Driven Architecture 4
  - Distributed Caching 4

2. Availability **4**
- Requirement 4
- Design Decisions 4
  - Replication 4
  - Circuit Breaker Pattern 4
  - Load Balancing 4
  - Multi-Region Deployment 4

3. Performance **4**
- Requirement 4
- Design Decisions 5
  - Database Optimization 5
  - Concurrency Control 5
  - Content Delivery Network (CDN) 5
  - Service Profiling 5

4. Reliability **5**
- Requirement 5
- Design Decisions 5
  - Distributed Transactions 5
  - Idempotent APIs 5
  - Data Backup and Recovery 6
  - Chaos Engineering 6

5. Security **6**
- Requirement 6
- Design Decisions 6
  - Authentication and Authorization 6
  - Data Encryption 6
  - Rate Limiting and DDoS Protection 6
  - Audit Logging 6

6. Maintainability **6**
- Requirement 6
- Design Decisions 7

Key Non-Functional Requirements and Design Solutions

Service Discovery 7
Centralized Logging 7
CI/CD Pipeline 7
7. Observability **7**
Requirement 7
Design Decisions 7
Distributed Tracing 7
Health Checks 7
Alerting Systems 7
8. Transactional Scenarios and Handling **7**
Seat Booking 8
Challenge 8
Solution 8
Locking Mechanisms 8
Retry Logic 8
Compensation Transactions 8
Payment Processing 8
Challenge 8
Solution 8
Two-Phase Commit (2PC) 8
Eventual Consistency 8
Refund and Cancellation 9
Challenge 9
Solution 9
Asynchronous Updates 9
9. OWASP **9**
Broken Access Control 9
Cryptographic Failures 9
Injection 9
Security Misconfiguration 9
Vulnerable and Outdated Components 10
Server-Side Request Forgery (SSRF) 10
10. Monetize Platform **10**
Ticket Sales (Primary Revenue) 10
Service Fees 10
Transaction Fees 10
Advertising 10
Display Ads 10
Premium Memberships or Subscriptions 10

Key Non-Functional Requirements and Design Solutions

VIP Membership 10
Subscription Models 11
Corporate Tie-ups and Bulk Booking 11
Corporate Sales 11
Group Discounts 11
Selling Merchandise 11
Affiliate Marketing 11
11. Compliance 11
Data Privacy and Protection 11
Payment and Financial Compliance 11
Consumer Protection Laws 12
Tax Compliance 12
Copyright and Intellectual Property 12
Key Non-Functional Requirements and Design Solutions 1

1. Scalability

Requirement

The system must handle high traffic during events (e.g., blockbuster movie releases).

Design Decisions

Microservices Architecture

- Each service (e.g., Theatre/Show/movie listings, ticket booking, payments, Notification, user) is independently deployable and scalable.

Horizontal Scaling

- Create AWS ECS Cluster per Region.
- Use container orchestration platforms like **AWS ECS** to scale services dynamically based on traffic.
- Shard Database based on City id, and then shard by show Id,Booking Id inside a city.

Key Non-Functional Requirements and Design Solutions

Event-Driven Architecture

- Use message brokers like **Kafka** to decouple services and process asynchronous requests during peak loads.

Distributed Caching

- Use **Redis** to cache frequently accessed data (e.g., movie details, seating availability).

2. Availability

Requirement

Ensure minimal downtime for seamless user experience.

Design Decisions

Replication

- Use database replication and service redundancy to avoid single points of failure.

Circuit Breaker Pattern

- Implement tools like **Resilience 4j** to handle service failures gracefully.

Load Balancing

- Use tools like **AWS ELB** to distribute traffic across multiple instances.

Multi-Region Deployment

- Deploy the system across multiple regions for geo-redundancy using cloud platforms like **AWS**.

3. Performance

Requirement

Low latency for transactional operations like seat booking.

Key Non-Functional Requirements and Design Solutions

Design Decisions

Database Optimization

- Use optimized queries, indexing,views,stored procedures and read replicas for high-frequency reads.

Concurrency Control

- Use pessimistic locking mechanisms to handle simultaneous seat bookings.

Content Delivery Network (CDN)

- Use a CDN for serving static content like images and movie trailers.

Service Profiling

- observability tools (Amazon Cloud watch) to identify metrics to monitor, CPUUtilization, Percentage of CPU used,MemoryUtilization, Percentage of memory used. set alarm and optimize bottlenecks.

4. Reliability

Requirement

Ensure data consistency and fault tolerance during transactions.

Design Decisions

Distributed Transactions

- Implement the **Saga pattern** for handling distributed transactions across microservices.

Idempotent APIs

- Ensure API endpoints can handle retries without unintended side effects.

Key Non-Functional Requirements and Design Solutions

Data Backup and Recovery

- Schedule periodic backups and maintain versioned snapshots of critical data.

Chaos Engineering

- Use tools like **Gremlin** to simulate failures and test system reliability under stress and **Jmeter** for load test.

5. Security

Requirement

Protect user data and transactions.

Design Decisions

Authentication and Authorization

- Use OAuth 2.0 with a token-based system (e.g., **JWT**) for secure access.

Data Encryption

- Encrypt sensitive data in transit (TLS) and at rest (AES-256).

Rate Limiting and DDoS Protection

- Use AWS API Gateway and rate-limiting algorithms like the **token bucket**.

Audit Logging

- Maintain detailed logs for sensitive transactions and monitor with tools like **ELK Stack**.

6. Maintainability

Requirement

Ensure ease of updates and debugging.

Key Non-Functional Requirements and Design Solutions

Design Decisions

Service Discovery

- Use service registries like **ECS Managed Service discovery** for managing microservice endpoints.

Centralized Logging

- Use logging frameworks (e.g.**Logstash**) to consolidate logs.

CI/CD Pipeline

- Automate testing, deployment, and rollback using tools like **Jenkins** or **GitHub Actions**.

7. Observability

Requirement

Monitor system health and performance in real-time.

Design Decisions

Distributed Tracing

- Use tools like **Spring cloud Sleuth** to add common message id across micro services and unique id for specific micro services to trace requests across services.

Health Checks

- Implement health endpoints in services for proactive monitoring by the orchestrator.

Alerting Systems

- Integrate SNS with the AWS service generating alerts, such as CloudWatch Alarms Configure alerts in **Email/Slack** for anomalies detected by monitoring tools.

8. Transactional Scenarios and Handling

Key Non-Functional Requirements and Design Solutions

Here's how design solutions align with specific transactional scenarios:

Seat Booking

Challenge

Concurrent users might try to book the same seat.

Solution

Locking Mechanisms

- Use PostgresQL advanced locking mechanisms (e.g., advisory locks) and better concurrency handling via MVCC (Multiversion Concurrency Control). to avoid double booking.

Retry Logic

- Implement retry with exponential backoff for failed transactions.

Compensation Transactions

- Allow rollbacks for unsuccessful bookings using the Saga pattern.

Payment Processing

Challenge

Ensuring consistency in a multi-step process involving third-party systems.

Solution

Two-Phase Commit (2PC)

- For tightly coupled systems, but avoid overuse due to performance impact.

Eventual Consistency

- Use Message Queue to reconcile transactions asynchronously.

Key Non-Functional Requirements and Design Solutions

Refund and Cancellation

Challenge

Ensure quick refunds while reconciling with external payment gateways.

Solution

Asynchronous Updates

- Notify users and payment gateways via message queues.
- Payment failed/Cancelled kafka queue for Refund processing.

9. OWASP

Broken Access Control

- Use Spring Security to enforce access control rules.
- Implement role-based or attribute-based access control policies.
- Validate @PreAuthorize and @PostAuthorize annotations to secure methods.

Cryptographic Failures

- Use strong algorithms like AES-256, RSA-2048, and PBKDF2 for encryption and hashing.
- Always store passwords using BCryptPasswordEncoder in Spring Security.
- Enforce HTTPS to secure data in transit

Injection

- Use Spring Data JPA or Hibernate, which provides protection through parameterized queries.
- Sanitize and validate user inputs rigorously.
- Avoid dynamic query building using string concatenation; use @Query with placeholders in Spring Data JPA.
- Use libraries like Hibernate Validator to validate input data.

Security Misconfiguration

- Disable directory listing, stack traces, and verbose error messages in production.
- Use @ConfigurationProperties to securely bind configurations.
- Regularly review and harden security settings, such as CORS and CSRF.

Key Non-Functional Requirements and Design Solutions

Vulnerable and Outdated Components

- Use dependency management tools like Maven or Gradle to track versions.
- Remove unused dependencies.

Server-Side Request Forgery (SSRF)

- Validate and whitelist URLs in any server-to-server HTTP requests.
- Disable access to local or private networks from your server.
- Implement rate-limiting and authentication for APIs handling such requests.

10. Monetize Platform

Ticket Sales (Primary Revenue)

Service Fees

- The platform charges a service fee or convenience fee on each ticket sold. This can be a fixed amount or a percentage of the ticket price.

Transaction Fees

- Platform can also earn a small fee for processing payments, either as a percentage or a flat rate.

Advertising

Display Ads

- The platform can run display advertising on the website or mobile app. These ads could be for upcoming events, movies, or other entertainment-related products.

Premium Memberships or Subscriptions

VIP Membership

- Platform can offer a paid membership program, giving users perks like early access to tickets, discounts, or exclusive events.

Key Non-Functional Requirements and Design Solutions

Subscription Models

- Monthly or yearly subscription plans for frequent users offering benefits like discounted tickets, priority booking, etc.

Corporate Tie-ups and Bulk Booking

Corporate Sales

- Platform can collaborate with businesses for bulk booking of tickets for corporate events, employee engagement, or incentive programs.

Group Discounts

- Offer group discounts or corporate packages for large bookings, often generating higher volumes of sales.

Selling Merchandise

Affiliate Marketing

- Partnering with merchandise sellers and earning a commission on items sold through affiliate links.

11. Compliance

Data Privacy and Protection

- General Data Protection Regulation (GDPR) in the EU.
- California Consumer Privacy Act (CCPA) in the US.
- Personal Data Protection Bill (PDPB) or Digital Personal Data Protection Act (DPDP Act) in India.

Payment and Financial Compliance

- Payment Card Industry Data Security Standard (PCI DSS): For handling credit/debit card transactions securely.
- Reserve Bank of India (RBI) guidelines for digital payment systems.

Key Non-Functional Requirements and Design Solutions

Consumer Protection Laws

- Clearly state refund, cancellation, and rescheduling policies.
- Display accurate event information (pricing, location, time)

Tax Compliance

- Collect and remit applicable taxes, such as Goods and Services Tax (GST) in India or VAT in the EU, for ticket sales

Copyright and Intellectual Property

- Obtain necessary rights for event details, images, or promotional material displayed on the platform.
- Avoid hosting or distributing copyrighted content without permission.

By aligning these solutions with NFRs, the backend becomes robust, scalable, and user-friendly.