Event-Driven Microservices with Serverless Technologies

Project Proposal FCJ Internship

Họ tên: Nguyễn Quốc Huy

Mentor: Văn Hoàng Kha

TP.Hồ Chí Minh, tháng 07 , năm 2025

Executive Summary

Tổng quan Dự án

Dự án "Hệ thống Đặt vé Xem phim Trực tuyến" được thiết kế để giải quyết các vấn đề hiện tại trong việc đặt vé xem phim truyền thống thông qua việc xây dựng một nền tảng số hiện đại, scalable và cost-effective sử dụng kiến trúc microservices serverless trên AWS Cloud.

Vấn đề Cần Giải quyết

Ngành công nghiệp giải trí Việt Nam đang phát triển mạnh mẽ với doanh thu phòng vé đạt 1,2 tỷ USD năm 2023. Tuy nhiên, các rạp chiếu phim hiện tại đang gặp phải những thách thức:

- **Hệ thống legacy**: 70% rạp chiếu phim sử dụng hệ thống đặt vé cũ, không linh hoạt
- **Trải nghiệm người dùng kém**: Thời gian đặt vé trung bình 5-7 phút, tỷ lệ bỏ giỏ hàng 35%
- **Khả năng mở rộng hạn chế**: Hệ thống thường crash trong các đợt phim hot (peak time)
- **Chi phí vận hành cao**: Infrastructure maintenance chiếm 30% budget IT

Giải pháp Đề xuất

Xây dựng hệ thống đặt vé xem phim hiện đại với kiến trúc microservices serverless bao gồm:

Kiến trúc Cốt lõi

- **6 Microservices** độc lập: User, Upload, Product, Order, Payment Services
- **API Gateway** với Kong Gateway để routing và security
- **Database per Service** pattern đảm bảo data consistency
- **Event-driven architecture** cho real-time communication

Công nghệ AWS Chính

- **AWS Lambda**: Serverless compute cho business logic
- **Amazon EC2:** a scalable, secure, and resizable compute capacity service in the AWS cloud
- **Amazon API Gateway**: API management và throttling
- **Amazon Elasticache**: in-memory data store and caching service designed to improve the performance of applications by reducing latency and database load
- **Amazon RDS**: Relational database cho complex queries
- **AWS MSK**: Event-driven messaging

Lợi ích Kinh doanh

Lợi ích Ngắn hạn (0-6 tháng)

- Giảm 50% thời gian đặt vé (từ 5-7 phút xuống 2-3 phút)
- Tăng 25% conversion rate
- Giảm 60% chi phí infrastructure so với traditional hosting

Lợi ích Trung hạn (6-18 tháng)

- Tăng 40% customer satisfaction score
- Giảm 70% system downtime
- Tăng 30% revenue do improved user experience

Lợi ích Dài hạn (18+ tháng)

- Khả năng mở rộng ra các dịch vụ giải trí khác
- Foundation cho AI/ML recommendations
- Competitive advantage trong thị trường

Timeline Triển khai

- **Phase 1**: Infrastructure & Core Services (2 tháng)
- **Phase 2**: Business Logic & Integration (2 tháng)
- **Phase 3**: Testing & Optimization (1 tháng)
- **Phase 4**: Deployment & Go-live (1 tháng)

Chỉ số Thành công

- **Technical KPIs**: Thời gian hoạt động 99,9%, thời gian phản hồi <2 giây, 10.000 người dùng đồng thời
- **Business KPIs**: Tỷ lệ chuyển đổi 65%, sự hài lòng của khách hàng 90%
- **Operational KPIs**: Giảm 80% chi phí, triển khai tự động 95%

---

1. Problem Statement

Tình trạng Hiện tại

Thị trường rạp chiếu phim Việt Nam đang phát triển nhanh chóng với hơn 1,000 rạp chiếu phim trên toàn quốc. Tuy nhiên, phần lớn các rạp vẫn đang sử dụng hệ thống đặt vé truyền thống với nhiều hạn chế về mặt kỹ thuật và trải nghiệm người dùng.

Các Thách thức Chính

1. Hệ thống Kỹ thuật Lạc hậu

Hiện trạng: 70% rạp chiếu phim sử dụng hệ thống monolithic architecture được phát triển từ 5-10 năm trước.

Vấn đề cụ thể:

- Khó khăn trong việc maintenance và upgrade
- Scalability hạn chế, không thể handle traffic cao
- Single point of failure - khi một component lỗi, toàn bộ hệ thống bị ảnh hưởng
- Thời gian phát triển feature mới kéo dài (3-6 tháng/feature)

2. Trải nghiệm Người dùng Kém

Thống kê hiện tại:

- Thời gian đặt vé trung bình: 5-7 phút
- Tỷ lệ bỏ giỏ hàng: 35%
- Điểm hài lòng của khách hàng: 6.2/10
- Xếp hạng trải nghiệm di động: 5.8/10

Nguyên nhân:

- Giao diện người dùng không thân thiện
- Quá trình thanh toán phức tạp
- Thiếu tính năng real-time (ghế đã được đặt không update kịp thời)
- Không có notification system hiệu quả

3. Vấn đề Hiệu suất và Độ tin cậy

Peak time performance:

- 40% hệ thống gặp sự cố trong giờ cao điểm
- Average downtime: 2-3 giờ/tháng
- Response time trong peak: 8-12 giây
- Concurrent user limit: 500-1000 users

Tác động kinh doanh:

- Doanh thu mất mát: $50,000/tháng do system downtime
- Tỷ lệ khách hàng mất khách hàng: 15%/năm
- Tác động của đánh giá tiêu cực: 20% giảm khách hàng mới

Tác động đến Stakeholders

Khách hàng (End Users)

- Thất vọng với quá trình đặt phòng chậm
- Mất cơ hội xem phim do không đặt được vé
- Thiếu các đề xuất được cá nhân hóa
- Trải nghiệm di động kém

Nhà điều hành Rạp chiếu phim

- Mất doanh thu do trải nghiệm người dùng kém
- Chi phí hoạt động cao
- Bất lợi cạnh tranh
- Khó khăn trong việc mở rộng quy mô kinh doanh

Nhà đầu tư

- ROI thấp hơn do chi phí hoạt động cao
- Tăng trưởng kinh doanh chậm lại
- Rủi ro cao hơn do hệ thống không ổn định
- Những cơ hội bị bỏ lỡ trong chuyển đổi số

Hậu quả của Việc Không Hành động

Tác động Ngắn hạn (6 tháng)

- Tiếp tục mất 30% doanh thu tiềm năng
- Sự hài lòng của khách hàng giảm xuống 5.5/10
- Lợi thế của đối thủ cạnh tranh tăng 25%

Tác động Trung hạn (1-2 năm)

- Tổng doanh thu mất mát: $500,000/năm
- Thiệt hại danh tiếng thương hiệu: không thể khôi phục

Tác động Dài hạn (3+ năm)

- Có thể bị loại khỏi thị trường
- Không thể cạnh tranh với digital-native competitors
- Missed opportunities trong AI/ML integration
- Sự gián đoạn hoàn toàn mô hình kinh doanh

Cơ hội Thị trường

Ngày nay, nhu cầu giải trí của người dân, đặc biệt là việc xem phim tại rạp, ngày càng tăng cao. Việc đặt vé trực tuyến đã trở thành xu hướng phổ biến do người dùng mong muốn có trải nghiệm nhanh chóng, tiện lợi và chủ động chọn chỗ ngồi, suất chiếu.

Tại Việt Nam và các nước Đông Nam Á, tỷ lệ người dùng sử dụng smartphone và internet đang tăng mạnh. Điều này tạo điều kiện thuận lợi để triển khai các nền tảng đặt vé

phim trực tuyến hiện đại. Đặc biệt, sau dịch COVID-19, hành vi người dùng đã thay đổi rõ rệt – họ ưu tiên mua vé online thay vì đến trực tiếp quầy, vừa tiết kiệm thời gian, vừa đảm bảo an toàn.

Bên cạnh đó, nhiều cụm rạp, đặc biệt là rạp tư nhân hoặc rạp quy mô vừa và nhỏ, vẫn chưa có hệ thống đặt vé online hiệu quả. Đây là cơ hội để cung cấp giải pháp phần mềm có thể dễ dàng tích hợp, giúp các rạp tối ưu vận hành, quản lý suất chiếu, phòng chiếu, doanh thu và nâng cao trải nghiệm khách hàng.

---

2. Solution Architecture

Tổng quan Kiến trúc

Hệ thống được thiết kế theo kiến trúc microservices serverless trên AWS Cloud, đảm bảo scalability, reliability, và cost-effectiveness. Kiến trúc tuân thủ các nguyên tắc:

- **Separation of Concerns**: Mỗi service chịu trách nhiệm một domain cụ thể
- **Database per Service**: Đảm bảo loose coupling
- **Event-driven Communication**: Asynchronous processing
- **API-first Design**: Consistent integration patterns

Các AWS Services Sử dụng

Compute Services

- **AWS Lambda**: Serverless compute cho business logic
- **Amazon ECS Fargate**: Container orchestration cho Kong Gateway
- **AWS Step Functions**: Workflow orchestration cho complex processes

Storage Services

- **Amazon DynamoDB**: NoSQL database cho user profiles, sessions
- **Amazon RDS (PostgreSQL)**: Relational database cho transactional data
- **Amazon S3**: Object storage cho media files, backups
- **Amazon ElastiCache**: In-memory caching cho performance

Networking & Security

- **Amazon VPC**: Network isolation
- **AWS Certificate Manager**: SSL/TLS certificates

Integration Services

- **Amazon MSK**: Event-driven messaging
- **Amazon SQS**: Message queuing cho async processing
- **Amazon SNS**: Notification service
- **AWS API Gateway**: API management (alternative to Kong)

Thiết kế Chi tiết các Components

1. User Service

Chức năng: Quản lý người dùng, authentication, authorization

Technical Stack:

- **Runtime**: AWS Lambda (Node.js 18.x)
- **Database**: Amazon RDS (PostgreSQL)
- **Caching**: Amazon ElastiCache (Redis)
- **Authentication**: AWS Cognito

2. Product Service

Chức năng: Quản lý phim, rạp, suất chiếu

Technical Stack:

- **Runtime**: AWS Lambda (Node.js 18.x)
- **Database**: Amazon RDS (PostgreSQL)
- **Caching**: Amazon ElastiCache

3. Order Service

Chức năng: Xử lý đơn hàng, booking logic

Technical Stack:

- **Runtime**: AWS Lambda (Node.js 18.x)
- **Database**: Amazon RDS (PostgreSQL)
- **State Management**: AWS Step Functions
- **Caching**: Amazon ElastiCache

Workflow:

1. Create order (PENDING status)
2. Reserve seats (10 minutes timeout)
3. Wait for payment confirmation
4. Confirm booking or release seats

4. Payment Service

Chức năng: Xử lý thanh toán, integration với payment gateways

Technical Stack:

- **Runtime**: AWS Lambda (Node.js 18.x)
- **Database**: Amazon RDS (PostgreSQL)
- **Queue**: Amazon MSK
- **Encryption**: AWS KMS

Supported Payment Methods:

- MoMo

5. Upload Service

Chức năng: Quản lý media files (posters, trailers)

Technical Stack:

- **Runtime**: AWS Lambda (Node.js 18.x)
- **Storage**: Amazon S3
- **Image Processing**: AWS Lambda

Features:

- Image resize và optimization
- Multiple format support
- Secure upload URLs
- Automatic compression

6. Notification Service

Chức năng: Gửi thông báo, email, SMS

Technical Stack:

- **Runtime**: AWS Lambda (Node.js 18.x)
- **Email**: Amazon SES

Kiến trúc Bảo mật

Network Security

- **VPC**: Isolated network environment
- **Security Groups**: Firewall rules
- **NACLs**: Network-level access control
- **Private Subnets**: Database isolation

Application Security

- **WAF**: Protection against common attacks
- **Rate Limiting**: DDoS protection
- **Input Validation**: Prevent injection attacks
- **HTTPS**: End-to-end encryption

Data Security

- **Encryption at Rest**: RDS, S3
- **Encryption in Transit**: TLS 1.3
- **KMS**: Key management
- **IAM**: Role-based access control

Compliance

- **PCI DSS**: Payment card industry standards
- **GDPR**: Data protection regulation
- **SOC 2**: Security controls
- **ISO 27001**: Information security

Thiết kế Scalability

Horizontal Scaling

- **Lambda**: Automatic scaling (0-15,000 concurrent)
- **RDS**: Read replicas for read-heavy workloads
- **ElastiCache**: Cluster mode for Redis

Performance Optimization

- **Caching Strategy**: Multi-layer caching
- **Database Optimization**: Indexing, query optimization
- **Connection Pooling**: Efficient database connections

Integration Points

External Systems

- **Payment Gateways**: MoMo APIs
- **Cinema Management**: Existing POS systems
- **Email Service**: Amazon SES

---

3. Technical Implementation

Các Giai đoạn Triển khai

Phase 1: Infrastructure Setup

Sprint 1.1: Core Infrastructure

Deliverables:

- AWS account setup và organization
- VPC configuration với public/private subnets
- Security groups và NACLs
- IAM roles và policies
- CloudFormation templates

Technical Tasks:

- Setup AWS Organizations với multiple accounts (dev, staging, prod)
- Configure VPC với 3 AZ availability
- Implement least privilege IAM policies
- Setup AWS Config cho compliance monitoring
- Create CloudFormation templates cho infrastructure

Sprint 1.2: Database Layer

Deliverables:

- RDS PostgreSQL setup với multi-AZ
- DynamoDB tables với proper indexing
- ElastiCache Redis cluster
- Database migration scripts
- Backup và restore procedures

Technical Tasks:

- Configure RDS với automated backups
- Design DynamoDB tables với GSI
- Setup ElastiCache với cluster mode
- Create database schemas và seed data
- Implement backup strategies

Phase 2: Core Services Development

Sprint 2.1: User Service

Deliverables:

- User registration/login APIs
- JWT token management
- Profile management
- Password reset functionality
- Unit tests (80% coverage)

Technical Implementation:

```
// User Service Lambda Function
exports.handler = async (event) => {
    const { httpMethod, path, body } = event;

    switch (httpMethod) {
        case 'POST':
            if (path === '/users/register') {
                return await registerUser(JSON.parse(body));
            }
            break;
        case 'GET':
            if (path === '/users/profile') {
                return await getUserProfile(event.headers.authorization);
            }
            break;
    }
};
```

Sprint 2.2: Product Service

Deliverables:

- Movie catalog APIs
- Theater management
- Showtime scheduling
- Seat availability tracking
- Search functionality

Database Optimization:

- Indexing strategies cho fast queries
- Read replicas cho heavy read workloads
- Query optimization
- Connection pooling

Sprint 2.3: Order Service

Deliverables:

- Order creation và management

- Seat reservation logic
- Order state machine
- Timeout handling
- Integration với payment service

State Machine Design:

```
{
  "StartAt": "CreateOrder",
  "States": {
    "CreateOrder": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:region:account:function:createOrder",
      "Next": "ReserveSeats"
    },
    "ReserveSeats": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:region:account:function:reserveSeats",
      "Next": "WaitForPayment"
    },
    "WaitForPayment": {
      "Type": "Wait",
      "Seconds": 600,
      "Next": "CheckPayment"
    }
  }
}
```

Phase 3: Integration & Advanced Features

Sprint 3.1: Payment Integration

Deliverables:

- VNPay integration
- MoMo wallet integration
- Credit card processing
- Refund handling
- Payment webhooks

Sprint 3.2: Upload Service

Deliverables:

- Image upload to S3
- Image resizing và optimization
- CDN integration
- Secure upload URLs

Sprint 3.3: Notification Service

Deliverables:

- Email notifications
- SMS notifications
- Push notifications
- Notification preferences

Phase 4: Testing & Deployment

Sprint 4.1: Testing

Deliverables:

- Unit tests (80% coverage)
- Integration tests
- Load testing
- Security testing
- User acceptance testing

Sprint 4.2: Production Deployment

Deliverables:

- Production environment setup
- CI/CD pipeline
- Monitoring và alerting
- Documentation
- Go-live support

Yêu cầu Kỹ thuật

Compute Requirements

- **Lambda Functions**: 512MB-1GB memory, 30s timeout
- **RDS**: db.t3.medium (2 vCPU, 4GB RAM) initially
- **DynamoDB**: On-demand pricing model
- **ElastiCache**: cache.t3.micro (1 vCPU, 0.5GB RAM)

Storage Requirements

- **RDS Storage**: 100GB GP2 SSD
- **S3 Storage**: 1TB standard storage
- **DynamoDB**: Estimated 50GB
- **ElastiCache**: 1GB memory

Network Requirements

- **Bandwidth**: 1Gbps connection
- **Latency**: <50ms within region
- **Availability**: 99.9% uptime SLA
- **Security**: TLS 1.3, encryption at rest

Monitoring Requirements

- **Metrics**: CPU, memory, disk, network
- **Logs**: Application logs, access logs, error logs
- **Alerts**: Response time, error rate, availability
- **Dashboards**: Real-time monitoring

Phương pháp Phát triển

Development Methodology

- **Agile/Scrum**: 2-week sprints
- **DevOps**: CI/CD pipeline
- **Test-Driven Development**: Unit tests first
- **Code Reviews**: Pull request workflow

Technology Stack

- **Languages**: Node.js, Python, Java
- **Frameworks**: Express.js, FastAPI, Spring Boot
- **Testing**: Jest, pytest, JUnit
- **CI/CD**: AWS CodePipeline, GitHub Actions

Code Quality Standards

- **Linting**: ESLint, Pylint, CheckStyle
- **Code Coverage**: Minimum 80%
- **Security Scanning**: SonarQube
- **Documentation**: JSDoc, Swagger

Chiến lược Testing

Unit Testing

- **Coverage**: 80% minimum
- **Framework**: Jest, pytest, JUnit
- **Mocking**: AWS SDK mocking
- **Automation**: Run on every commit

Integration Testing

- **API Testing**: Postman, Newman
- **Database Testing**: Test containers
- **Event Testing**: LocalStack
- **End-to-end**: Cypress

Load Testing

- **Tools**: Artillery, JMeter
- **Scenarios**: Peak load, stress testing
- **Targets**: 10,000 concurrent users
- **Duration**: 30 minutes sustained load

Security Testing

- **OWASP**: Top 10 vulnerabilities
- **Penetration Testing**: Third-party assessment
- **Dependency Scanning**: Snyk, OWASP dependency check
- **Static Analysis**: SonarQube security rules

Deployment Plan

Environment Strategy

- **Development**: Local development + AWS dev account
- **Staging**: Production-like environment
- **Production**: Live environment

CI/CD Pipeline

```
stages:
  - build
  - test
  - security-scan
  - deploy-staging
  - integration-test
  - deploy-production
```

Rollback Procedures

- **Database**: Point-in-time recovery
- **Application**: Blue-green deployment
- **Lambda**: Version aliases
- **Infrastructure**: CloudFormation rollback

Configuration Management

- **Environment Variables**: AWS Parameter Store

- **Secrets**: AWS Secrets Manager
- **Infrastructure**: CloudFormation/Terraform
- **Application Config**: Config files in S3

---

4. Timeline & Milestones

Tổng quan Timeline

Dự án được chia thành 4 phases chính trong 6 tháng với các milestone rõ ràng và measurable success criteria.

Phase 1: Foundation Setup (Tuần 1)

Milestone 1.1: Infrastructure Ready

Success Criteria:

- VPC setup hoàn tất với 3 AZ
- Security groups và NACLs configured
- IAM roles và policies implemented
- CloudFormation templates created
- Basic monitoring dashboard active

Deliverables:

- Network architecture diagram
- Security configuration document
- IAM policy documentation
- Infrastructure automation scripts
- Monitoring setup guide

Dependencies:

- AWS account approval
- Team onboarding completion
- Development environment setup

Milestone 1.2: Database Layer Complete

Success Criteria:

- RDS PostgreSQL cluster running
- DynamoDB tables created với proper indexing
- ElastiCache Redis cluster active

- Database schemas deployed
- Backup procedures tested

Deliverables:

- Database schema documentation
- Migration scripts
- Backup và restore procedures
- Performance tuning results
- Data model diagrams

Dependencies:

- Infrastructure setup completion
- Database design approval
- Security clearance for data handling

Milestone 1.3: Security & Monitoring

Success Criteria:

- WAF rules deployed và tested
- CloudWatch dashboards functional
- X-Ray tracing enabled
- Security scanning automated
- Compliance monitoring active

Deliverables:

- Security configuration guide
- Monitoring runbook
- Compliance reports
- Incident response procedures
- Security assessment results

Dependencies:

- Security team approval
- Compliance requirements documentation
- Monitoring tools selection

Phase 2: Core Services Development (Tuần 2)

Milestone 2.1: User Service

Success Criteria:

- User registration/login APIs functional
- JWT authentication working

- Profile management complete
- Unit test coverage >80%
- Load testing passed (1000 concurrent users)

Deliverables:

- User service API documentation
- Authentication flow diagrams
- Test reports
- Performance benchmarks
- Security audit results

Dependencies:

- Database layer completion
- Security framework approval
- API design review

Milestone 2.2: Product Service Deployed

Success Criteria:

- Movie catalog APIs functional
- Theater management working
- Showtime scheduling active
- Search functionality implemented
- Database queries optimized (<200ms)

Deliverables:

- Product service documentation
- API specification
- Database performance reports
- Search implementation guide
- Integration test results

Dependencies:

- User service completion
- Data migration completion
- Search engine configuration

Milestone 2.3: Order Service Ready

Success Criteria:

- Order creation flow working
- Seat reservation logic implemented
- State machine functional

- Timeout handling tested
- Integration với payment service

Deliverables:

- Order service documentation
- State machine diagrams
- Business logic specifications
- Integration test reports
- Performance analysis

Dependencies:

- Product service completion
- Payment gateway integration plan
- Business rules approval

Phase 3: Integration & Advanced Features (Tuần 3)

Milestone 3.1: Payment Integration

Success Criteria:

- VNPay integration functional
- MoMo wallet working
- Credit card processing active
- Refund functionality tested
- Payment webhooks configured

Deliverables:

- Payment integration guide
- Transaction flow documentation
- Security compliance report
- Testing results
- Merchant configuration

Dependencies:

- Order service completion
- Payment gateway approvals
- PCI DSS compliance verification

Milestone 3.2: Media & Notification Services

Success Criteria:

- Image upload functionality working
- CDN integration active

- Email notifications sending
- SMS notifications functional
- Push notifications configured

Deliverables:

- Media service documentation
- Notification system guide
- CDN configuration
- Message templates
- Delivery reports

Dependencies:

- S3 bucket configuration
- CDN setup completion
- Notification service approvals

Milestone 3.3: System Integration

Success Criteria:

- All services integrated
- End-to-end workflows functional
- API Gateway configured
- Kong Gateway deployed
- Load balancing active

Deliverables:

- System integration report
- API Gateway configuration
- Load balancer setup
- Integration test results
- Performance benchmarks

Dependencies:

- All individual services completion
- Integration testing environment
- Performance testing tools

Phase 4: Testing & Go-Live

Milestone 4.1: Testing Complete (Tuần 21-22)

Success Criteria:

- Unit tests >80% coverage

- Integration tests passed
- Load testing successful (10,000 users)
- Security testing cleared
- UAT approved

Deliverables:

- Test execution reports
- Performance test results
- Security assessment
- UAT sign-off
- Bug fix documentation

Dependencies:

- System integration completion
- Test environment setup
- User acceptance criteria

Milestone 4.2: Production Deployment

Success Criteria:

- Production environment ready
- CI/CD pipeline functional
- Monitoring alerts active
- Documentation complete
- Go-live successful

Deliverables:

- Production deployment guide
- Operations manual
- Monitoring setup
- User documentation
- Go-live report

Dependencies:

- Testing completion
- Production environment approval
- Operations team training

Dependencies Matrix

Critical Path Dependencies

1. Infrastructure → Database → Services → Integration → Testing → Deployment

2. Security Setup → Compliance → Payment Integration
3. User Service → Order Service → Payment Service

External Dependencies

- **AWS Account Setup**: 1 tuần
- **Payment Gateway Approvals**: 2-3 tuần
- **Security Compliance Review**: 1 tuần
- **Third-party Service Integration**: 2 tuần

Risk Mitigation Timeline

- **Buffer Time**: 10% added to each phase
- **Parallel Development**: Non-dependent services
- **Early Testing**: Continuous integration
- **Backup Plans**: Alternative solutions identified

Resource Allocation

Dự án chủ yếu do một cá nhân thực hiện nên việc phân bổ thời gian cần hợp lý. Mỗi tuần cần tối thiểu 10–15 giờ để đảm bảo chất lượng. Các công cụ như GitHub Project hoặc Notion sẽ được dùng để theo dõi tiến độ và quản lý công việc cá nhân theo mô hình Kanban.

---

5. Budget Estimation

Tổng quan Chi phí

Chi phí của dự án được tính dựa trên mô hình Pay-as-you-go của AWS, kết hợp với chính sách Free Tier dành cho tài khoản mới. Vì dự án có quy mô vừa phải và tận dụng dịch vụ serverless như Lambda, MSK, và RDS, nên chi phí vận hành được tối ưu ở mức rất thấp trong giai đoạn phát triển và thử nghiệm.

Các nhóm chi phí được chia làm ba loại chính:

- Chi phí hạ tầng AWS (Infra)
- Chi phí phát triển (Dev Effort)
- Chi phí vận hành dài hạn (Ops)

Chi phí hạ tầng AWS

Dịch vụ	Mô tả sử dụng	Ước tính chi phí	Ghi chú
AWS Lambda	~1 triệu invocations/tháng	$0	Nằm trong Free Tier
MSK	Dùng cluster 2 broker, chạy 24/7	~$50–100/tháng	Tính theo giờ, không nằm trong Free Tier
RDS	Dùng instance t3.micro/t4g.micro	$0	Free Tier có 750 giờ/tháng + 20 GB storage (chỉ áp dụng với db.t2.micro, t3.micro, t4g.micro cho PostgreSQL/MySQL)
EC2	~5GB logs = ~10 alarm	~$1-2	Tính theo ~$0.5/GB
S3	~1-2GB lưu trữ file tài liệu	~$0.05	Có thể miễn phí nếu dưới 5GB

Tổng chi phí hạ tầng ước tính/tháng: $50–120, nằm trong mức rất thấp nhờ tận dụng Free Tier và quy mô nhỏ gọn.

Chi phí phát triển

Phần lớn thời gian phát triển do thực tập sinh đảm nhận nên không tính chi phí nhân công. Tuy nhiên, nếu tính theo chuẩn thị trường, effort khoảng 60–70 giờ làm việc tương đương:

- Trung bình $10–15/giờ → tổng chi phí phát triển ước lượng: $600–$1,000

Khoản này chỉ mang tính tham khảo trong trường hợp mở rộng nhóm hoặc tính toán ROI về sau.

Phân tích ROI và điểm hòa vốn

Mục tiêu Đầu tư

Dự án hướng đến việc số hóa quy trình đặt vé, giảm thiểu chi phí vận hành, tăng trải nghiệm người dùng và mở rộng khả năng phục vụ trên toàn quốc. Việc sử dụng kiến trúc microservices serverless không chỉ giúp giảm chi phí hạ tầng mà còn hỗ trợ mở rộng hệ thống linh hoạt mà không cần đầu tư thêm phần cứng.

Tác động Chiến lược

- **Lợi thế cạnh tranh**: Giúp các rạp nhỏ và vừa bắt kịp công nghệ, tăng khả năng cạnh tranh với hệ thống lớn.
- **Nền tảng mở rộng**: Có thể triển khai cho nhiều ngành giải trí khác như sự kiện, ca nhạc, thể thao.
- **Khả năng thương mại hóa**: Mô hình SaaS (Software as a Service) cho phép cung cấp giải pháp cho nhiều rạp với chi phí linh hoạt theo lưu lượng sử dụng.

Chiến lược tối ưu chi phí

Một số phương án tối ưu chi phí sẽ được áp dụng bao gồm:

- Tận dụng tối đa Free Tier AWS trong giai đoạn thử nghiệm
- Dùng mô hình on-demand thay vì provisioned cho RDS/Lambda
- Chỉ dùng 1–2 môi trường chính (dev, prod), tránh phân mảnh

---

6. Risk Assessment

Triển khai một hệ thống phân tán hiện đại luôn đi kèm nhiều rủi ro liên quan đến kỹ thuật, vận hành, bảo mật và thị trường. Việc đánh giá và chuẩn bị cho các rủi ro giúp giảm thiểu gián đoạn, đảm bảo tiến độ và chất lượng dự án.

Risk Matrix

Mức độ ảnh hưởng	Cao	Trung bình	Thấp
Cao (Khả năng xảy ra cao)	R1. Mất dữ liệu do lỗi hệ thống	R2. Độ trễ cao khi tải dữ liệu lớn	R3. Lỗi cấu hình IAM gây từ chối truy cập
Trung bình (Khả năng xảy ra trung bình)	R4. Tăng đột biến chi phí AWS	R5. Vấn đề tích hợp với cổng thanh toán	R6. Trễ tiến độ do resource cá nhân
Thấp (Khả năng xảy ra thấp)	R7. Vi phạm tuân thủ bảo mật (GDPR, PCI DSS)	R8. Phản hồi tiêu cực từ người dùng thử nghiệm	R9. Tốc độ phát triển chậm do thiếu tài liệu

 = Critical | = Moderate | = Low | = Very Low

Mitigation Strategies (Chiến lược Giảm thiểu Rủi ro)

Rủi ro	Chiến lược giảm thiểu
R1	Thiết lập backup tự động hàng ngày (RDS, S3); bật **point-in-time recovery**; dùng versioning với Lambda và S3
R2	Dùng CDN cho media (CloudFront); cache bằng ElastiCache; tối ưu truy vấn và nén dữ liệu
R3	Áp dụng chính sách **IAM theo nguyên tắc Least Privilege**; kiểm thử quyền bằng staging environment trước production
R4	

	Cấu hình cảnh báo **AWS Budgets** và CloudWatch Billing; dùng **throttling/quota** cho API Gateway
R5	Thiết lập môi trường staging với test credentials từ MoMo/VNPay; kiểm thử kỹ các payment webhook và timeout
R6	Quản lý tiến độ bằng công cụ Kanban (GitHub Projects/Notion); phân bổ buffer thời gian 10–15% cho mỗi phase
R7	Tuân thủ mã hóa dữ liệu (KMS, TLS 1.3); kiểm tra bảo mật định kỳ; dùng dịch vụ đạt chuẩn GDPR/PCI-DSS (RDS, SES)
R8	Tổ chức đợt user testing nhỏ; khảo sát phản hồi người dùng đầu tiên để điều chỉnh sớm
R9	Viết tài liệu nội bộ (Notion/Markdown); tạo sơ đồ hệ thống; giữ log kỹ thuật trong quá trình phát triển

Contingency Plans (Kế hoạch Dự phòng)

Kịch bản	Phản ứng dự phòng
Mismatch giữa OpenAPI và code	Kích hoạt **blue-green deployment** để rollback phiên bản trước; phục hồi dữ liệu từ snapshot
Lỗi Lambda do xử lý hoặc timeout	Cho phép retry thủ công trong 5 phút; gửi thông báo lỗi có hướng dẫn; fallback sang phương thức thanh toán khác (nếu có)
Chi phí AWS vượt ngưỡng dự kiến	Tạm ngưng các service không cần thiết (MSK nếu chưa dùng nhiều); chuyển sang plan tiết kiệm hơn (RDS reserved instance)
Không đạt được KPIs sau triển khai	Xem lại luồng người dùng, A/B testing giao diện; thu thập session logs để tối ưu trải nghiệm
Resource cá nhân không đủ thời gian triển khai	Ưu tiên phát triển các chức năng cốt lõi trước; có thể chia nhỏ thành các giai đoạn và dùng mock API cho demo

---

7. Expected Outcomes

Success Metrics (Chỉ số Thành công)

Danh mục	Chỉ số cụ thể
Hiệu suất kỹ thuật	Thời gian phản hồi trung bình < 2 giây Tỷ lệ uptime hệ thống > 99.9% Hỗ trợ tối thiểu 10,000 người dùng đồng thời
Chất lượng người dùng	Tỷ lệ chuyển đổi (conversion rate) đạt 65% Điểm hài lòng người dùng (CSAT) > 90% Tỷ lệ bỏ giỏ hàng giảm từ 35% xuống <20%

Quản trị vận hành	Triển khai CI/CD tự động > 95% Giảm chi phí hạ tầng > 60% so với hệ thống truyền thống

Business Benefits

Lợi ích	Mô tả
Tăng doanh thu	Tăng 25–30% doanh thu từ vé nhờ trải nghiệm người dùng được tối ưu
Tối ưu chi phí vận hành	Cắt giảm đến 60% chi phí so với hệ thống on-premise nhờ kiến trúc serverless
Rút ngắn thời gian phát triển	Triển khai chức năng mới nhanh hơn 50% nhờ kiến trúc microservices và CI/CD
Cải thiện uy tín thương hiệu	Giao diện mượt mà, tốc độ nhanh, độ ổn định cao → tăng sự tin tưởng và trung thành
Mở rộng thị phần	Có thể cung cấp giải pháp cho nhiều cụm rạp khác, đặc biệt là các rạp nhỏ và vừa

Technical Improvements

Khu vực	Cải tiến đạt được
Kiến trúc hệ thống	Chuyển đổi từ monolith sang microservices serverless, tăng khả năng mở rộng và bảo trì
Quy trình CI/CD	Tích hợp GitHub Actions / AWS CodePipeline giúp triển khai tự động và rollback nhanh

Bảo mật	Áp dụng các best practices: IAM chi tiết, mã hóa end-to-end, kiểm soát truy cập mạnh mẽ
Giám sát & Logging	Tích hợp CloudWatch, X-Ray, thiết lập cảnh báo theo real-time metrics
Tối ưu dữ liệu	Sử dụng ElastiCache để giảm tải RDS; phân tán dữ liệu theo service riêng biệt

Long-term Value

Khía cạnh	Giá trị đạt được
Tính bền vững	Dễ dàng bảo trì, mở rộng, thay thế từng service độc lập mà không ảnh hưởng toàn hệ thống
Khả năng mở rộng	Có thể tích hợp thêm loyalty program, quảng cáo, AI recommendation trong tương lai
Thích nghi thị trường	Nhanh chóng cập nhật giao diện, phương thức thanh toán mới hoặc tích hợp với đối tác
Tận dụng dữ liệu	Cơ sở dữ liệu người dùng phục vụ cho phân tích hành vi, AI/ML và chiến lược marketing
Tiềm năng thương mại hóa	Có thể cung cấp như giải pháp SaaS cho rạp chiếu phim khác, tạo doanh thu dài hạn

---