TECHNICAL SPECIFICATION DOCUMENT
Distributed Microservices Architecture Implementation

PROJECT CODENAME: Project Nebula

TECHNICAL OVERVIEW:
We require the development of a cloud-native, containerized microservices architecture utilizing Kubernetes orchestration with Istio service mesh for inter-service communication. The system must implement event-driven architecture patterns using Apache Kafka for asynchronous message processing and maintain ACID compliance through distributed transaction management with the Saga pattern.

TECHNOLOGY STACK:
• Backend: Node.js (NestJS framework), Go (gRPC services)
• Frontend: React 18 with TypeScript, Next.js for SSR
• State Management: Redux Toolkit with RTK Query
• Database: PostgreSQL (primary), MongoDB (document store), Redis (caching layer)
• Message Queue: Apache Kafka with Schema Registry
• API Gateway: Kong with rate limiting and JWT authentication
• Container Orchestration: Kubernetes (EKS)
• Service Mesh: Istio for traffic management and observability
• CI/CD: GitHub Actions, ArgoCD for GitOps
• Monitoring: Prometheus + Grafana, ELK stack for log aggregation
• Infrastructure as Code: Terraform, Helm charts

ARCHITECTURAL COMPONENTS:

1. API Gateway Layer
   - Kong API Gateway with OAuth 2.0 / OpenID Connect
   - Rate limiting (token bucket algorithm)
   - Request validation using JSON Schema
   - Circuit breaker pattern implementation

2. Service Layer (Microservices)
   - User Authentication Service (OAuth 2.0, JWT)
   - Authorization Service (RBAC + ABAC)
   - Payment Processing Service (PCI DSS compliant)
   - Notification Service (WebSocket + FCM)
   - Analytics Service (real-time data processing)
   - File Storage Service (S3-compatible object storage)
   - Search Service (Elasticsearch with custom analyzers)
   - Audit Logging Service (write-ahead logging)

3. Data Layer
   - CQRS pattern implementation
   - Event Sourcing for audit trail
   - Database sharding strategy (consistent hashing)
   - Read replicas for query optimization
   - Connection pooling (pgBouncer)

4. Infrastructure
   - Multi-region deployment (active-active)
   - Auto-scaling policies (HPA + VPA)
   - Blue-green deployment strategy
   - Disaster recovery (RTO: 1 hour, RPO: 15 minutes)
   - Backup strategy (incremental + full)

DELIVERABLES:

Technical Documentation:
- System Architecture Diagram (C4 model)
- API Documentation (OpenAPI 3.0 specification)

- Database Schema (ERD + migration scripts)
- Infrastructure Architecture (Terraform modules)
- Security Architecture Document
- Disaster Recovery Plan
- Performance Benchmarking Report

Source Code Repositories:
- Microservices (8 repositories)
- Frontend Application (1 repository)
- Infrastructure as Code (1 repository)
- CI/CD Pipelines (GitHub Actions workflows)
- Helm Charts (Kubernetes deployment configs)

Testing Artifacts:
- Unit Tests (>80% code coverage)
- Integration Tests (contract testing with Pact)
- End-to-End Tests (Cypress/Playwright)
- Performance Tests (k6 load testing scripts)
- Security Tests (OWASP ZAP automation)

DevOps Deliverables:
- Kubernetes Manifests (production-ready)
- Monitoring Dashboards (Grafana)
- Alerting Rules (Prometheus AlertManager)
- Log Parsing Rules (Logstash configurations)
- Service Mesh Configuration (Istio policies)

PERFORMANCE REQUIREMENTS:
- API Response Time: p95 < 200ms, p99 < 500ms
- Throughput: 10,000 requests/second sustained
- Database Query Performance: p95 < 50ms
- Message Processing Latency: < 100ms
- System Availability: 99.95% uptime SLA
- Data Consistency: Eventual consistency with 5-second window

SECURITY REQUIREMENTS:
- TLS 1.3 for all communications
- Secrets management (HashiCorp Vault)
- Container security scanning (Trivy)
- Runtime security monitoring (Falco)
- Compliance: SOC 2 Type II, GDPR, HIPAA

ESTIMATED EFFORT:
Development: 240 man-days (6 senior engineers × 2 months)
DevOps: 60 man-days (2 DevOps engineers × 1.5 months)
QA: 40 man-days (2 QA engineers × 1 month)
Architecture/Leadership: 30 man-days

Total Duration: 4 months (with parallel workstreams)

COMPLEXITY LEVEL: Complex (Enterprise-grade distributed system)

LICENSING: One-Time Use License (exclusive deployment rights)

USAGE RIGHTS: Commercial Use (production deployment + modifications)

BUDGET ESTIMATE: $180,000 - $220,000
(Based on team composition and technology stack complexity)