🍴 Food Delivery App - Complete Backend Specification

Express.js MVP + Security + Deployment Ready

Executive Summary

Enterprise-grade food delivery platform optimized for the Cameroon market with Mobile Money integration, SMS/USSD support, and low-internet adaptation. Built with Express.js, featuring comprehensive security, real-time capabilities, and production deployment on Render.com.

---

1. 🏗️ System Architecture & Technical Stack

Core Technology Stack

- **Backend Framework**: Express.js (Node.js 18+)
- **Database**: SQLite (MVP) → PostgreSQL (Production)
- **Authentication**: JWT + Refresh Tokens + OAuth2
- **Real-time**: Socket.io with Redis adapter
- **File Storage**: Cloudinary / AWS S3
- **Caching**: Redis (sessions, rate limiting, caching)
- **Queue System**: Bull Queue with Redis
- **Monitoring**: Winston + Morgan logging
- **Testing**: Jest + Supertest + Artillery (load testing)

Security Infrastructure

- **Encryption**: AES-256-GCM for data at rest
- **Hashing**: Argon2id for passwords
- **Rate Limiting**: Express-rate-limit + Redis
- **Input Validation**: Joi + express-validator
- **CSRF Protection**: csurf middleware
- **CORS**: Configured for production domains
- **Helmet.js**: Security headers
- **Content Security Policy**: Strict CSP rules

Payment Integration Stack

- **Mobile Money**: CamPay API + NouPay API (dual provider)
- **SMS Gateway**: Twilio + Local Cameroon provider
- **Push Notifications**: Firebase Cloud Messaging (FCM)
- **Maps & Geolocation**: Yandex Maps API
- **AI Recommendations**: Custom lightweight model (<50MB) + Gemini integration

---

2. 👥 Enhanced User Roles & Permissions

2.1 Visitor (Guest - Not Authenticated)

Permissions: READ_ONLY, CART_MANAGEMENT, GUEST_ORDER

- Browse restaurants & menus with search/filtering
- Add items to cart (session-based, 30min expiry)
- Place orders with mandatory phone verification (OTP)
- Track delivery via public tracking link
- Access limited promotions
- View restaurant ratings & reviews
- Convert to customer account post-order

Security Context: Rate-limited API access, session tracking, phone verification required

2.2 Customer (Authenticated User)

Permissions: All Visitor + LOYALTY_PROGRAM, WALLET_MANAGEMENT, PROFILE_MANAGEMENT

- **Account Management**: Multiple delivery addresses, payment methods, preferences
- **Ordering**: Quick reorder, scheduled orders, bulk orders
- **Wallet System**: Top-up via Mobile Money, refunds, bonus credits, transaction history
- **Loyalty Program**: Points earning (1 point per 100 XAF), tier progression, reward redemption
- **Social Features**: Rate/review restaurants & drivers, favorite lists, referral rewards
- **Notifications**: Push, SMS, email preferences with granular control
- **Privacy**: Data export, account deletion (GDPR compliant)

Security Context: 2FA optional, login attempt monitoring, sensitive action confirmation

2.3 Restaurant Owner

Permissions: RESTAURANT_MANAGEMENT, MENU_CRUD, ORDER_PROCESSING, FINANCIAL_REPORTS

- **Restaurant Profile**: Hours, contact info, cuisine types, delivery radius
- **Menu Management**: Categories, items, modifiers, availability scheduling
- **Order Processing**: Real-time notifications, accept/decline with reasons, preparation time estimates
- **Financial Dashboard**:
  - Revenue tracking with commission breakdown
  - Payout requests to Mobile Money
  - Tax report generation
  - Promotional campaign ROI
- **Analytics**: Peak hours, best-selling items, customer demographics, delivery performance
- **Promotions**: Time-based discounts, combo deals, loyalty integration

Security Context: Business verification required, IP restrictions for admin functions, audit trail

2.4 Delivery Driver (Agent)

Permissions: ORDER_ACCEPTANCE, LOCATION_SHARING, EARNINGS_TRACKING

- **Order Management**: Accept/reject with smart matching algorithm
- **Navigation**: Integrated Yandex Maps with optimized routes
- **Real-time Tracking**: Live location sharing with ETA updates
- **Delivery Proof**: OTP verification, photo confirmation, digital signatures
- **Earnings Dashboard**: Daily/weekly/monthly reports, tip tracking, fuel reimbursements
- **Performance Metrics**: Rating trends, delivery success rate, customer feedback
- **Vehicle Management**: Registration, insurance tracking, maintenance reminders

Security Context: Background check required, real-time location verification, fraud detection

2.5 Support Agent

Permissions: TICKET_MANAGEMENT, CHAT_SUPPORT, ESCALATION_RIGHTS

- **Ticket System**: Multi-channel (chat, email, SMS) with priority levels
- **Live Chat**: Real-time customer support with chat history
- **Knowledge Base**: FAQ management, canned responses, escalation procedures

- **Dispute Resolution**: Order issues, refund processing, mediation tools
- **Performance Tracking**: Response time, resolution rate, customer satisfaction

Security Context: Limited data access, action logging, supervisor oversight

2.6 Admin (Super Admin)

Permissions: FULL_SYSTEM_ACCESS, USER_MANAGEMENT, FINANCIAL_OVERSIGHT, SECURITY_MONITORING

- **User Management**: CRUD operations, role assignments, bulk actions
- **Financial Control**:
  - Platform wallet management
  - Commission settings per restaurant
  - Payout approvals and fraud detection
  - Revenue forecasting and reporting
- **Security Dashboard**: Login attempts, fraud alerts, system health monitoring
- **Content Management**: Newsletter campaigns, partnership requests, promotional content
- **System Configuration**: Feature flags, maintenance mode, API rate limits
- **Audit & Compliance**: Activity logs, data retention, regulatory reporting

Security Context: Multi-factor authentication mandatory, IP whitelisting, session timeout

---

3. 🔐 Comprehensive Security Implementation

3.1 Authentication & Authorization

```javascript
// JWT Strategy with Refresh Tokens
- Access Token: 15 minutes expiry, contains user role & permissions
- Refresh Token: 7 days expiry, secure httpOnly cookie
- Token Rotation: New refresh token on each access token renewal
- Logout: Blacklist tokens in Redis with TTL
- Brute Force Protection: Progressive delays after failed attempts
```

3.2 Password Security

- **Hashing**: Argon2id with salt (memory: 64MB, time: 3, parallelism: 4)

- **Password Policy**: Min 8 chars, complexity requirements, breach detection
- **Reset Flow**: Secure token (32 bytes), 1-hour expiry, rate limited

3.3 Input Validation & Sanitization

- **Request Validation**: Joi schemas for all endpoints
- **SQL Injection Prevention**: Parameterized queries, ORM validation
- **XSS Protection**: Input sanitization, output encoding, CSP headers
- **File Upload Security**: Type validation, size limits, virus scanning

3.4 API Security

- **Rate Limiting**:
  - General: 100 requests/15 minutes per IP
  - Auth: 5 attempts/15 minutes
  - Payment: 3 attempts/5 minutes
- **API Keys**: Rotation every 90 days, encrypted storage
- **CORS**: Strict origin validation for production
- **Request Signing**: HMAC-SHA256 for sensitive operations

3.5 Data Protection

- **Encryption at Rest**: AES-256-GCM for PII data
- **Encryption in Transit**: TLS 1.3 minimum
- **Data Masking**: Log sanitization, debug mode restrictions
- **Privacy Controls**: Data anonymization, retention policies

---

4. 📊 Enhanced Database Schema

Core Tables

sql

```
-- Users with enhanced security
Users: id, email, phone, passwordHash, salt, role, status,
       lastLogin, failedAttempts, lockedUntil, twoFactorSecret,
       emailVerified, phoneVerified, createdAt, updatedAt

-- Restaurants with business validation
Restaurants: id, ownerId, businessName, legalName, taxId,
             location, latitude, longitude, deliveryRadius,
             status, verificationStatus, avgRating, totalOrders

-- Enhanced menu system
MenuItems: id, restaurantId, categoryId, name, description,
           price, costPrice, discountPrice, images, allergens,
           nutritionalInfo, availability, preparationTime

-- Comprehensive order tracking
Orders: id, customerId, restaurantId, driverId, status,
        items, subtotal, deliveryFee, tax, discount, total,
        paymentMethod, paymentStatus, estimatedTime, actualTime,
        customerNotes, restaurantNotes, driverNotes

-- Advanced wallet system
Wallets: id, ownerType, ownerId, balance, frozenAmount,
         currency, status, dailyLimit, monthlyLimit

-- Transaction audit trail
Transactions: id, walletId, type, amount, description,
              referenceId, status, metadata, ipAddress,
              userAgent, createdAt
```

Security Tables

sql

```
-- Session management
Sessions: id, userId, token, refreshToken, ipAddress,
        userAgent, lastActivity, expiresAt

-- Security audit
AuditLogs: id, userId, action, resource, oldValue,
         newValue, ipAddress, userAgent, timestamp

-- Login attempts tracking
LoginAttempts: id, identifier, ipAddress, success,
             attemptedAt, userAgent

-- Two-factor authentication
TwoFactorAuth: id, userId, secret, backupCodes,
             lastUsed, createdAt
```

---

5. 🚀 Core Functionalities & Enhanced Flows

5.1 Advanced Customer Journey

mermaid

```
graph TD
A[Guest Browse] --> B{Register?}
B -->|Yes| C[Account Creation + OTP]
B -->|No| D[Guest Checkout]
C --> E[Profile Setup]
E --> F[Browse with Personalization]
F --> G[Add to Cart]
G --> H[Loyalty Points Applied]
H --> I[Payment Selection]
I --> J[Order Confirmation]
J --> K[Real-time Tracking]
K --> L[Delivery + Rating]
L --> M[Loyalty Points Earned]
```

5.2 Restaurant Operations Flow

- **Order Reception**: Real-time notifications with sound alerts
- **Preparation Time**: Dynamic estimation based on kitchen load
- **Inventory Management**: Auto-disable items when out of stock

- **Peak Hour Management**: Surge pricing and delay warnings
- **Quality Control**: Photo verification for order accuracy

5.3 Driver Optimization System

- **Smart Matching**: AI-powered assignment based on location, rating, vehicle type
- **Route Optimization**: Multi-order batching with optimal sequencing
- **Earnings Maximization**: Peak hour incentives, bonus challenges
- **Safety Features**: Emergency button, incident reporting, speed monitoring

5.4 Admin Control Center

- **Real-time Dashboard**: Live orders, active drivers, system health
- **Fraud Detection**: Unusual patterns, fake orders, suspicious accounts
- **Revenue Optimization**: Dynamic pricing, commission adjustments
- **Customer Insights**: Behavior analytics, churn prediction, LTV calculation

---

6. 💳 Payment & Financial System

6.1 Multi-Provider Payment Integration

```javascript
// Payment Providers Configuration
const paymentProviders = {
  campay: {
    api: 'https://api.campay.net/collect',
    methods: ['MTN_MOMO', 'ORANGE_MONEY'],
    fees: { mtn: 2.5, orange: 2.0 }
  },
  noupay: {
    api: 'https://api.noupay.com/payments',
    methods: ['MTN_MOMO', 'ORANGE_MONEY', 'EU_MOBILE'],
    fallback: true
  }
}
```

6.2 Wallet System Features

- **Multi-currency Support**: XAF primary, USD for international

- **Auto-topup**: Scheduled refills when balance drops below threshold
- **Spending Controls**: Daily/monthly limits, category restrictions
- **Transaction Categorization**: Food, delivery, tips, refunds
- **Escrow System**: Hold payments until delivery confirmation

6.3 Commission & Payout Structure

```javascript
const commissionStructure = {
  restaurants: {
    tier1: 15, // New restaurants (first 30 days)
    tier2: 12, // Established (>100 orders)
    tier3: 10, // Premium partners (>500 orders)
  },
  drivers: {
    baseFee: 500, // XAF per delivery
    distanceFee: 100, // Per km
    tips: 100, // Driver keeps 100%
    bonuses: 'performance-based'
  }
}
```

---

7. 🤖 AI & Recommendation Engine

7.1 Custom Lightweight AI Model (<50MB)

```javascript
// Local AI Model Specifications
const aiModel = {
  size: '49.8MB',
  framework: 'TensorFlow.js',
  training: 'Transfer learning from Gemini base model',
  features: [
    'Dish recommendations based on order history',
    'Peak time predictions',
    'Cuisine preference learning',
    'Price sensitivity analysis'
  ]
}
```

7.2 Recommendation Logic

- **Collaborative Filtering**: Users with similar tastes
- **Content-Based**: Dish characteristics, cuisine types
- **Time-Based**: Breakfast/lunch/dinner recommendations
- **Weather Integration**: Hot soups on rainy days, cold drinks in heat
- **Location-Based**: Popular dishes in user's area

7.3 Gemini AI Integration

- **Menu Description Enhancement**: Auto-generate appealing descriptions
- **Customer Service**: Smart reply suggestions for support agents
- **Fraud Detection**: Pattern recognition for suspicious activities
- **Business Intelligence**: Market trend analysis, competitor insights

---

8. 📱 Real-time Communication & Notifications

8.1 WebSocket Namespaces

```javascript
const socketNamespaces = {
  '/customer': ['order-updates', 'driver-location', 'promotions'],
  '/restaurant': ['new-orders', 'driver-assigned', 'customer-messages'],
  '/driver': ['order-assignments', 'route-updates', 'earnings-updates'],
  '/admin': ['system-alerts', 'live-analytics', 'support-tickets'],
  '/support': ['new-tickets', 'customer-chat', 'escalations']
}
```

8.2 Multi-Channel Notification System

- **Push Notifications**: FCM with custom sounds, images, actions
- **SMS Fallback**: When app is uninstalled or push fails
- **Email Campaigns**: Newsletter, promotional, transactional
- **WhatsApp Business**: Order confirmations, delivery updates (future)
- **In-App Notifications**: Persistent notification center

8.3 Offline-First Approach

- **Menu Caching**: 7-day local cache with delta updates
- **Order Queue**: Store orders locally, sync when online
- **Critical Data Sync**: Priority sync for orders, payments
- **Progressive Web App**: Full offline functionality

---

9. 🔧 Missing Features & Enhancements

9.1 Advanced Search & Discovery

```javascript
const searchFeatures = {
  elastic: {
    fuzzySearch: true,
    autocomplete: true,
    filters: ['cuisine', 'price', 'rating', 'delivery-time'],
    sorting: ['relevance', 'rating', 'price', 'distance'],
    facetedSearch: true
  },
  ai: {
    voiceSearch: true,
    imageSearch: 'search by food photo',
    semanticSearch: 'natural language queries'
  }
}
```

9.2 Social & Community Features

- **Review System**: Verified purchase reviews, helpful votes
- **User Profiles**: Public profiles, badges, achievement system
- **Social Sharing**: Share favorite dishes, restaurants
- **Community Challenges**: Monthly food challenges, leaderboards
- **Recipe Sharing**: Restaurant-customer recipe exchange

9.3 Business Intelligence & Analytics

- **Customer Segmentation**: RFM analysis, behavior clustering
- **Predictive Analytics**: Demand forecasting, churn prediction

- **A/B Testing Framework**: UI/UX experiments, pricing tests
- **Market Analysis**: Competitor monitoring, trend identification
- **Restaurant Success Metrics**: Health scores, improvement suggestions

9.4 Advanced Logistics

```javascript
const logisticsFeatures = {
  routing: {
    multiStop: true,
    trafficAware: true,
    weatherConsideration: true,
    vehicleOptimization: ['motorcycle', 'car', 'bicycle']
  },
  inventory: {
    realTimeUpdates: true,
    autoReordering: true,
    wastageTracking: true,
    expiryManagement: true
  }
}
```

9.5 Customer Service Evolution

- **AI Chatbot**: 24/7 first-line support with escalation
- **Video Support**: Screen sharing for complex issues
- **Proactive Support**: Issue prediction and prevention
- **Multilingual Support**: English, French, local languages
- **Voice Support**: Phone support integration

---

10. 🚀 Production Deployment (Render.com)

10.1 Render Configuration

```yaml
```

```yaml
# render.yaml
services:
  - type: web
    name: food-delivery-api
    env: node
    plan: starter
    buildCommand: npm ci && npm run build
    startCommand: npm start
    envVars:
      - key: NODE_ENV
        value: production
      - key: DATABASE_URL
        fromDatabase:
          name: food-delivery-db
          property: connectionString
      - key: REDIS_URL
        fromDatabase:
          name: food-delivery-redis
          property: connectionString

databases:
  - name: food-delivery-db
    databaseName: fooddelivery
    user: admin
    plan: starter

  - name: food-delivery-redis
    plan: starter
```

10.2 Docker Configuration

dockerfile

```
FROM node:18-alpine

# Security: Run as non-root user
RUN addgroup -g 1001 -S nodejs
RUN adduser -S nodejs -u 1001

# Install dependencies
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production && npm cache clean --force

# Copy application
COPY --chown=nodejs:nodejs . .

# Security headers and optimizations
ENV NODE_ENV=production
ENV NODE_OPTIONS="--max-old-space-size=1024"

USER nodejs
EXPOSE 3000

# Health check
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
  CMD node healthcheck.js

CMD ["npm", "start"]
```

10.3 Environment Variables

bash

```
# Core Configuration
NODE_ENV=production
PORT=3000
API_VERSION=v1

# Database
DATABASE_URL=postgresql://user:pass@host:5432/db
REDIS_URL=redis://host:6379

# Security
JWT_SECRET=your-super-secure-secret-key
JWT_REFRESH_SECRET=your-refresh-secret-key
ENCRYPTION_KEY=32-byte-encryption-key

# Payment Providers
CAMPAY_API_KEY=your-campay-api-key
CAMPAY_SECRET=your-campay-secret
NOUPAY_API_KEY=your-noupay-api-key
NOUPAY_SECRET=your-noupay-secret

# External Services
YANDEX_MAPS_API_KEY=your-yandex-api-key
FCM_SERVER_KEY=your-fcm-server-key
TWILIO_ACCOUNT_SID=your-twilio-sid
TWILIO_AUTH_TOKEN=your-twilio-token
GEMINI_API_KEY=your-gemini-api-key

# Email & SMS
SMTP_HOST=smtp.gmail.com
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password

# Security
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100
SESSION_TIMEOUT=3600000
```

11. ✅ Comprehensive Testing Strategy

11.1 Test Pyramid

javascript

```
const testStrategy = {
  unit: {
    coverage: '95%',
    tools: ['Jest', 'Sinon'],
    focus: ['utils', 'validators', 'middleware', 'services']
  },
  integration: {
    coverage: '85%',
    tools: ['Supertest', 'TestContainers'],
    focus: ['API endpoints', 'database operations', 'external services']
  },
  e2e: {
    coverage: 'critical paths',
    tools: ['Playwright', 'Cucumber'],
    focus: ['user journeys', 'payment flows', 'real-time features']
  },
  performance: {
    tools: ['Artillery', 'K6'],
    scenarios: ['peak load', 'stress testing', 'spike testing']
  }
}
```

11.2 Security Testing

- **SAST**: SonarQube for static analysis
- **DAST**: OWASP ZAP for dynamic testing
- **Dependency Scanning**: npm audit, Snyk
- **Container Scanning**: Trivy for Docker images
- **Penetration Testing**: Quarterly external audits

---

12. 📊 Monitoring & Observability

12.1 Application Monitoring

javascript

```
const monitoring = {
  logs: {
    tool: 'Winston + Morgan',
    levels: ['error', 'warn', 'info', 'debug'],
    storage: 'ELK Stack / CloudWatch',
    retention: '90 days'
  },
  metrics: {
    tool: 'Prometheus + Grafana',
    dashboards: ['system', 'business', 'security'],
    alerts: ['response time', 'error rate', 'payment failures']
  },
  tracing: {
    tool: 'Jaeger',
    coverage: 'all API calls',
    sampling: '10% in production'
  }
}
```

12.2 Business Metrics Dashboard

- **Real-time Orders**: Active orders, completion rate
- **Revenue Tracking**: Hourly/daily revenue, commission breakdown
- **User Activity**: Active users, new registrations, churn rate
- **Driver Performance**: Average delivery time, driver utilization
- **Restaurant Analytics**: Order volume, popular items, customer ratings

---

13. 🎯 AI Agent Task Breakdown

Phase 1: Foundation & Security (Sprint 1-2)

markdown

```
## Task 1.1: Project Setup
- [ ] Initialize Express.js project with TypeScript
- [ ] Configure ESLint, Prettier, Husky pre-commit hooks
- [ ] Setup project structure (MVC pattern)
- [ ] Configure environment variables and validation
- [ ] Setup Docker configuration with security best practices

## Task 1.2: Database & ORM Setup
- [ ] Configure Sequelize ORM with PostgreSQL
- [ ] Create all database models with validation
- [ ] Setup database migrations and seeders
- [ ] Configure connection pooling and query optimization
- [ ] Implement database backup strategies

## Task 1.3: Core Security Implementation
- [ ] JWT authentication with refresh token rotation
- [ ] Argon2id password hashing implementation
- [ ] Rate limiting with Redis backend
- [ ] Input validation using Joi schemas
- [ ] CORS, Helmet, CSP security headers
- [ ] Request/response logging with PII masking
```

Phase 2: User Management & Authentication (Sprint 3-4)

markdown

```
## Task 2.1: Authentication System
- [ ] User registration with email/phone verification
- [ ] Login with brute force protection
- [ ] Password reset flow with secure tokens
- [ ] Two-factor authentication (TOTP)
- [ ] Social login (Google, Facebook) - OAuth2
- [ ] Role-based access control middleware

## Task 2.2: User Profile Management
- [ ] Profile CRUD operations with validation
- [ ] Multiple address management
- [ ] Preference settings (notifications, cuisine, etc.)
- [ ] Account deletion with data anonymization
- [ ] Data export functionality (GDPR compliance)
```

Phase 3: Core Business Logic (Sprint 5-7)

markdown

```
## Task 3.1: Restaurant Management
- [ ] Restaurant registration and verification
- [ ] Menu management with categories and modifiers
- [ ] Business hours and availability system
- [ ] Restaurant dashboard with analytics
- [ ] Inventory management integration

## Task 3.2: Order Processing System
- [ ] Shopping cart with session management
- [ ] Order placement and validation
- [ ] Order status management workflow
- [ ] Real-time order updates via WebSocket
- [ ] Order cancellation and modification rules

## Task 3.3: Payment Integration
- [ ] CamPay API integration for Mobile Money
- [ ] NouPay API as backup payment provider
- [ ] Wallet system with transaction history
- [ ] Automatic retry logic for failed payments
- [ ] Payment reconciliation and reporting
```

Phase 4: Advanced Features (Sprint 8-10)

markdown

```
## Task 4.1: Real-time Communication
- [ ] Socket.io setup with Redis adapter
- [ ] Real-time order tracking system
- [ ] Live chat support system
- [ ] Push notification service (FCM)
- [ ] SMS notification fallback

## Task 4.2: Loyalty & Rewards System
- [ ] Point calculation and tier management
- [ ] Reward redemption system
- [ ] Referral program implementation
- [ ] Promotional campaigns management
- [ ] Coupon and discount code system

## Task 4.3: AI Recommendations
- [ ] Train custom lightweight AI model (<50MB)
- [ ] Implement recommendation algorithms
- [ ] Gemini AI integration for enhanced features
- [ ] A/B testing framework for recommendations
- [ ] Performance monitoring for AI features
```

Phase 5: Operations & Analytics (Sprint 11-12)

markdown

```
## Task 5.1: Admin Dashboard
- [ ] Real-time analytics dashboard
- [ ] User management interface
- [ ] Financial reporting system
- [ ] System monitoring and health checks
- [ ] Audit log and activity tracking

## Task 5.2: Driver Management
- [ ] Driver onboarding and verification
- [ ] Route optimization algorithms
- [ ] Earnings calculation and payout system
- [ ] Performance tracking and rating system
- [ ] Vehicle and document management
```

Phase 6: Testing & Deployment (Sprint 13-14)

markdown

```
## Task 6.1: Comprehensive Testing
- [ ] Unit tests for all services and utilities
- [ ] Integration tests for API endpoints
- [ ] End-to-end testing for critical user journeys
- [ ] Performance testing with load simulation
- [ ] Security testing and vulnerability assessment

## Task 6.2: Production Deployment
- [ ] Render.com deployment configuration
- [ ] Database migration scripts
- [ ] Environment-specific configurations
- [ ] Health check endpoints
- [ ] Monitoring and alerting setup
- [ ] Documentation and deployment guide
```

---

14. 📋 Success Criteria & KPIs

Technical Metrics

- **Performance**: API response time < 200ms (95th percentile)
- **Reliability**: 99.9% uptime, zero data loss
- **Security**: No critical vulnerabilities, SOC 2 compliance ready
- **Scalability**: Handle 10,000 concurrent users, 1M daily orders

Business Metrics

- **User Growth**: 50% month-over-month growth in first 6 months
- **Order Volume**: 1000 orders/day by month 3
- **Revenue**: $10K monthly revenue by month 6
- **Customer Satisfaction**: 4.5+ star average rating

Operational Metrics

- **Code Quality**: 90%+ test coverage, A-grade code quality
- **Deployment**: Zero-downtime deployments, 15-minute deployment cycle
- **Support**: < 2 hour response time, 95% first-call resolution
- **Monitoring**: 100% critical path monitoring, proactive alerting

---

15. 🔄 Maintenance & Future Roadmap

Short-term (3-6 months)

- Machine learning model improvements
- Advanced analytics and reporting
- Mobile app API optimization
- Third-party logistics integration

Medium-term (6-12 months)

- Multi-city expansion
- B2B catering services
- Subscription meal plans
- Advanced inventory management

Long-term (12+ months)

- International expansion
- Franchise management system
- AI-powered kitchen optimization
- Blockchain-based supply chain tracking

This comprehensive specification provides AI agents with clear, actionable tasks while ensuring enterprise-grade security, scalability, and maintainability. Each phase builds upon the previous one, allowing for incremental development and testing.