Restaurant CRM Platform - System Architecture Documentation

1. Architecture Overview

The Restaurant CRM Platform follows a **microservices architecture** with event-driven communication, designed for scalability, maintainability, and real-time operations. The system is cloud-native and container-based, supporting multi-tenant restaurant operations.

2. Client Layer

2.1 Mobile Application (Flutter)

- **Target Users**: Procurement staff, Kitchen staff, POS operators
- **Key Features**:
    - Offline-first architecture with sync capabilities
    - Real-time notifications via WebSocket
    - Role-based UI adaptation
    - Camera integration for inventory scanning
    - Push notifications

2.2 Web Dashboard (React/Next.js)

- **Target Users**: Super Admin, Management, Reporting
- **Key Features**:
    - Responsive design for tablets and desktops
    - Advanced reporting and analytics
    - User management and system configuration
    - Real-time dashboard updates

2.3 Progressive Web App (PWA)

- **Purpose**: Backup access for mobile users
- **Features**: Offline capability, push notifications, app-like experience

3. API Gateway & Load Balancing

3.1 Load Balancer (Nginx/HAProxy)

- **Purpose**: Distribute incoming requests across multiple instances
- **Features**: SSL termination, request routing, health checks

3.2 API Gateway

- **Functions**:
    - Request routing and load balancing
    - Authentication and authorization
    - Rate limiting and throttling
    - Request/response transformation
    - API versioning

4. Application Layer (Microservices)

4.1 Authentication Service

- **Technology**: NestJS with Passport.js
- **Responsibilities**:
    - JWT token generation and validation
    - Role-based access control (RBAC)
    - Session management
    - Password security and policies

4.2 User Management Service

- **Responsibilities**:
    - User profile management
    - Role assignment and permissions
    - Super admin functions
    - User activity tracking

4.3 Procurement Service

- **Core Functions**:
    - Purchase order creation and management
    - Supplier management
    - Cost tracking and budgeting
    - Approval workflows

4.4 Inventory Service

- **Key Features**:
    - Real-time stock tracking
    - Automatic reorder points
    - Expiry date management
    - Stock validation workflows
    - Barcode/QR code scanning

4.5 Kitchen Service

- **Capabilities**:
    - Recipe management
    - Ingredient allocation
    - Cooking status tracking
    - Kitchen workflow optimization
    - Food safety compliance

4.6 POS Service

- **Functions**:
    - Order processing and payment
    - Menu management
    - Customer management
    - Sales reporting
    - Receipt generation

4.7 Accounting Service

- **Features**:
    - Automated bookkeeping
    - Invoice generation
    - Expense tracking
    - Financial reporting
    - Tax calculation
    - Integration with external accounting systems

4.8 AI Analytics Service

- **Capabilities**:
    - Cost optimization recommendations
    - Demand forecasting
    - Menu profitability analysis
    - Waste reduction insights
    - Dynamic pricing suggestions

4.9 Notification Service

- **Channels**:
    - In-app notifications (WebSocket)
    - Push notifications (FCM/APNS)
    - SMS notifications (Twilio)
    - Email notifications (SendGrid)

4.10 Reporting Service

- **Features**:
    - Real-time dashboards
    - Custom report generation

    - Data visualization
    - Scheduled reports
    - Export capabilities (PDF, Excel, CSV)

5. Message Broker & Real-time Communication

5.1 RabbitMQ

- **Purpose**: Event-driven communication between microservices
- **Features**:
    - Message queuing and routing
    - Dead letter queues for error handling
    - Message persistence and durability
    - Clustering for high availability

5.2 WebSocket Server

- **Functionality**:
    - Real-time updates to client applications
    - Bidirectional communication
    - Connection management and scaling
    - Room-based messaging for different user roles

5.3 Redis

- **Use Cases**:
    - Session storage and management
    - Application-level caching
    - Rate limiting data
    - Real-time data caching
    - Pub/sub for real-time notifications

6. Database Layer

6.1 PostgreSQL (Primary Database)

- **Usage**: Transactional data storage
- **Data Types**:
    - User accounts and authentication
    - Inventory and stock data
    - Financial transactions
    - Order and customer data
- **Features**: ACID compliance, complex queries, JSON support

6.2 MongoDB (Document Store)

- **Usage**: Analytics and logging data
- **Data Types**:
    - User activity logs
    - System performance metrics
    - AI model training data
    - Flexible schema data

6.3 Elasticsearch

- **Purpose**: Search and analytics
- **Features**:
    - Full-text search capabilities
    - Real-time analytics
    - Log aggregation and analysis
    - Business intelligence queries

7. External Service Integration

7.1 Payment Gateway

- **Providers**: Stripe, PayPal, Square
- **Features**: Secure payment processing, refunds, recurring payments

7.2 Communication Services

- **SMS**: Twilio for SMS notifications
- **Email**: SendGrid for email notifications
- **Push**: Firebase Cloud Messaging (FCM) for mobile notifications

7.3 Cloud Storage

- **Provider**: AWS S3, Google Cloud Storage
- **Purpose**: Document storage, image uploads, backup files
- **CDN**: CloudFront for global content delivery

7.4 AI/ML Services

- **Providers**: TensorFlow, OpenAI, AWS SageMaker
- **Use Cases**: Predictive analytics, natural language processing, image recognition

8. Security & Monitoring

8.1 Security Measures

- **Firewall**: Network-level security
- **SSL/TLS**: End-to-end encryption

- **JWT Authentication**: Stateless authentication
- **Role-based Access Control**: Granular permissions
- **Data Encryption**: At-rest and in-transit encryption
- **Audit Logging**: Complete activity tracking

8.2 Monitoring & Observability

- **Prometheus**: Metrics collection and alerting
- **Grafana**: Visualization and dashboards
- **ELK Stack**: Centralized logging and analysis
- **Health Checks**: Service availability monitoring
- **Performance Monitoring**: Response time and throughput tracking

8.3 Backup & Disaster Recovery

- **Automated Backups**: Regular database backups
- **Point-in-time Recovery**: Database restoration capabilities
- **Cross-region Replication**: Data redundancy
- **Disaster Recovery Plan**: Business continuity procedures

9. Cloud Infrastructure

9.1 Container Orchestration

- **Kubernetes**: Container management and orchestration
- **Docker**: Application containerization
- **Helm Charts**: Application deployment templates

9.2 Auto Scaling

- **Horizontal Pod Autoscaler**: Dynamic scaling based on CPU/memory
- **Vertical Pod Autoscaler**: Resource optimization
- **Cluster Autoscaler**: Node scaling based on demand

9.3 CI/CD Pipeline

- **Source Control**: Git with branching strategy
- **Build Pipeline**: Automated testing and building
- **Deployment**: Blue-green deployments
- **Monitoring**: Deployment health checks

10. Data Flow Architecture

10.1 Procurement to Stock Flow

1. Procurement creates purchase order

2. Event published to RabbitMQ
3. Stock service receives notification
4. Physical validation and approval
5. Inventory updated in real-time
6. Notifications sent to relevant users

10.2 Kitchen to POS Flow

1. Kitchen updates cooking status
2. Real-time updates via WebSocket
3. POS receives order ready notification
4. Payment processing and completion
5. Accounting service updates financial records
6. Analytics service processes data for insights

10.3 Real-time Notification Flow

1. Service publishes event to RabbitMQ
2. Notification service processes event
3. Determines notification channels and recipients
4. Sends notifications via appropriate channels
5. Tracks delivery status and retries if needed

11. Scalability Considerations

11.1 Horizontal Scaling

- Microservices can be scaled independently
- Load balancing across multiple instances
- Database read replicas for query performance

11.2 Caching Strategy

- Redis for application-level caching
- CDN for static content delivery
- Database query result caching

11.3 Performance Optimization

- Database indexing and query optimization
- Asynchronous processing for heavy operations
- Connection pooling for database connections
- Message queuing for decoupled processing

12. Deployment Strategy

12.1 Environment Setup

- **Development**: Local development with Docker Compose
- **Staging**: Kubernetes cluster with reduced resources
- **Production**: Full Kubernetes deployment with HA

12.2 Release Management

- **Feature Flags**: Gradual feature rollout
- **Blue-Green Deployment**: Zero-downtime deployments
- **Canary Releases**: Risk mitigation for new features
- **Rollback Strategy**: Quick recovery from issues

This architecture provides a robust, scalable, and maintainable foundation for the Platform, supporting real-time operations, multi-tenant capabilities, and future growth requirements.