Kitchen Management System - Development Tasks

This file tracks the implementation progress of 24 prioritized tasks to transform our JWT backend into a comprehensive kitchen management system.

📊 Progress Overview

- **Total Tasks**: 24
- **Completed**: 0
- **In Progress**: 0
- **Pending**: 24

🧪 Testing Tasks (3/24)

T1: Add unit tests for core business logic

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 3-5 days
- **Dependencies**: None
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement comprehensive unit tests for authentication services, menu CRUD operations, and user management functions

Acceptance criteria:

- [ ] 90%+ code coverage for core modules
- [ ] Tests for authentication flows, menu validation, and user role management
- [ ] CI integration with automated test execution

Implementation Notes:

- Create test modules in `src/core/*/tests.rs`
- Use `tokio-test` for async testing
- Mock database connections for isolated testing
- Add `cargo test` to CI pipeline

T2: Implement integration tests for API endpoints

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 4-6 days
- **Dependencies**: T1

- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Create end-to-end integration tests for all REST API endpoints including authentication, menu management, and user operations

Acceptance criteria:

- [ ] Full API endpoint coverage
- [ ] Database integration testing
- [ ] Mock external service dependencies

Implementation Notes:

- Use `reqwest` for HTTP client testing
- Set up test database with Docker
- Create test fixtures and data factories
- Test error scenarios and edge cases

T3: Set up end-to-end test scenarios for critical user flows

- **Status**: 🔴 Pending
- **Priority**: Medium
- **Estimated effort**: 5-7 days
- **Dependencies**: T1, T2
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement E2E tests for complete user journeys: staff login → order creation → kitchen workflow → completion

Acceptance criteria:

- [ ] Automated browser testing for frontend
- [ ] Complete workflow validation
- [ ] Performance benchmarking integration

Implementation Notes:

- Use Selenium or Playwright for browser automation
- Create realistic test scenarios
- Integrate with load testing tools
- Set up test data management

📚 Documentation Tasks (3/24)

D1: Add comprehensive inline documentation for all public APIs

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 2-3 days
- **Dependencies**: None
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Document all public functions, structs, and API endpoints with comprehensive rustdoc comments

Acceptance criteria:

- ☐ All public APIs documented with examples
- ☐ OpenAPI/Swagger documentation generation
- ☐ Code examples for common use cases

Implementation Notes:

- Use `///` comments for all public items
- Add `#[doc]` attributes for complex examples
- Integrate `utoipa` for OpenAPI generation
- Create documentation build process

D2: Create Architecture Decision Records (ADRs) for key technical decisions

- **Status**: 🔴 Pending
- **Priority**: Medium
- **Estimated effort**: 3-4 days
- **Dependencies**: None
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Document architectural decisions, technology choices, and design patterns used in the kitchen management system

Acceptance criteria:

- ☐ ADRs for database design, authentication strategy, and microservices architecture
- ☐ Decision rationale and alternatives considered
- ☐ Template for future ADRs

Implementation Notes:

- Create `docs/adr/` directory structure
- Use ADR template format

- Document current architecture decisions retroactively
- Set up ADR review process

D3: Develop API usage examples and update README with setup instructions

- **Status**: 🔴 Pending
- **Priority**: Medium
- **Estimated effort**: 2-3 days
- **Dependencies**: D1
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Create comprehensive examples for API usage, including kitchen-specific workflows and integration patterns

Acceptance criteria:

- [ ] Complete setup guide for development environment
- [ ] API usage examples with curl and code samples
- [ ] Deployment documentation for production

Implementation Notes:

- Update README with kitchen management focus
- Create `examples/` directory with code samples
- Document Docker development setup
- Add troubleshooting guide

⚡ Performance Tasks (3/24)

P1: Implement connection pooling for gRPC services

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 3-4 days
- **Dependencies**: None
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Optimize gRPC communication with connection pooling for better performance under load

Acceptance criteria:

- [ ] Configurable connection pool size
- [ ] Connection health monitoring
- [ ] Performance benchmarks showing improvement

Implementation Notes:

- Use `tonic` connection pooling features
- Implement health check service
- Add connection pool metrics
- Load test before/after implementation

P2: Integrate Redis for caching frequently accessed data

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 4-5 days
- **Dependencies**: None
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement Redis caching for menu items, user sessions, and frequently accessed restaurant data

Acceptance criteria:

- [ ] Cache invalidation strategies
- [ ] TTL configuration for different data types
- [ ] Cache hit ratio monitoring

Implementation Notes:

- Add `redis` crate dependency
- Create cache abstraction layer
- Implement cache-aside pattern
- Add cache metrics and monitoring

P3: Add request batching for bulk operations

- **Status**: 🔴 Pending
- **Priority**: Medium
- **Estimated effort**: 3-4 days
- **Dependencies**: P2
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement batching for bulk menu updates, order processing, and inventory operations

Acceptance criteria:

- [ ] Batch processing for menu updates

- ☐ Bulk order status updates
- ☐ Performance improvements for large datasets

Implementation Notes:

- Design batch API endpoints
- Implement async batch processing
- Add batch size limits and validation
- Create batch operation monitoring

🔒 Security Tasks (3/24)

S1: Implement request/response validation middleware

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 3-4 days
- **Dependencies**: None
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Add comprehensive input validation and sanitization for all API endpoints

Acceptance criteria:

- ☐ Schema validation for all request payloads
- ☐ Input sanitization to prevent injection attacks
- ☐ Detailed validation error responses

Implementation Notes:

- Use `validator` crate for validation rules
- Create validation middleware for Axum
- Implement custom validation functions
- Add validation error response formatting

S2: Add rate limiting per endpoint with Redis

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 2-3 days
- **Dependencies**: P2
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement granular rate limiting per endpoint and user role using Redis

Acceptance criteria:

- ☐ Configurable rate limits per endpoint
- ☐ Different limits for different user roles
- ☐ Rate limit monitoring and alerting

Implementation Notes:

- Implement token bucket algorithm with Redis
- Create rate limiting middleware
- Add rate limit headers in responses
- Configure different limits per endpoint/role

S3: Configure security headers and CORS policies

- **Status**: 🔴 Pending
- **Priority**: Medium
- **Estimated effort**: 1-2 days
- **Dependencies**: None
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement comprehensive security headers and CORS policies for web security

Acceptance criteria:

- ☐ Security headers (CSP, HSTS, etc.)
- ☐ Configurable CORS policies
- ☐ Security header testing and validation

Implementation Notes:

- Use `tower-http` for security headers
- Configure CORS for different environments
- Add security header middleware
- Implement security header testing

🍽️ Core Features Tasks (5/24)

CF1: Develop menu management system (CRUD operations)

- **Status**: 🔴 Pending
- **Priority**: Critical
- **Estimated effort**: 8-10 days
- **Dependencies**: S1
- **Assignee**: TBD
- **Start Date**: TBD

- **Completion Date**: TBD

Description: Build comprehensive menu management with items, categories, pricing, ingredients, and nutritional information

Acceptance criteria:

- [ ] Full CRUD operations for menu items
- [ ] Category and subcategory management
- [ ] Ingredient tracking and allergen information
- [ ] Pricing and availability management

Implementation Notes:

- Design menu database schema
- Create menu API endpoints
- Implement menu item validation
- Add image upload for menu items
- Create menu search and filtering

CF2: Build inventory tracking system

- **Status**: 🔴 Pending
- **Priority**: Critical
- **Estimated effort**: 10-12 days
- **Dependencies**: CF1
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement real-time inventory tracking with automatic reorder points and supplier integration

Acceptance criteria:

- [ ] Real-time stock level tracking
- [ ] Low-stock alerts and automatic reordering
- [ ] Supplier management and purchase orders
- [ ] Inventory reports and analytics

Implementation Notes:

- Design inventory database schema
- Implement stock level tracking
- Create supplier management system
- Add automated reorder point calculations
- Build inventory reporting dashboard

CF3: Implement order management workflow

- **Status**: 🔴 Pending
- **Priority**: Critical
- **Estimated effort**: 12-15 days
- **Dependencies**: CF1, CF2
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Create complete order lifecycle from creation to completion with status tracking

Acceptance criteria:

- ☐ Order creation and modification
- ☐ Kitchen workflow integration
- ☐ Status tracking and updates
- ☐ Order history and reporting

Implementation Notes:

- Design order state machine
- Create order API endpoints
- Implement order status transitions
- Add order timing and tracking
- Build kitchen workflow integration

CF4: Create staff management with role-based access

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 6-8 days
- **Dependencies**: S1
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement comprehensive staff management with roles, permissions, and shift scheduling

Acceptance criteria:

- ☐ Role-based access control (Chef, Server, Manager, etc.)
- ☐ Shift scheduling and management
- ☐ Staff performance tracking
- ☐ Permission management system

Implementation Notes:

- Extend existing user system with roles

- Implement permission-based middleware
- Create shift scheduling system
- Add staff performance metrics
- Build role management interface

CF5: Design table/reservation system

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 8-10 days
- **Dependencies**: CF4
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Build table management and reservation system with real-time availability

Acceptance criteria:

- [ ] Table layout and capacity management
- [ ] Reservation booking and management
- [ ] Real-time table status updates
- [ ] Waitlist and notification system

Implementation Notes:

- Design table and reservation schema
- Create reservation API endpoints
- Implement real-time table status
- Add waitlist management
- Build notification system

🔧 Technical Enhancement Tasks (4/24)

TE1: Add WebSockets for real-time order updates

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 5-7 days
- **Dependencies**: CF3
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement WebSocket connections for real-time order status updates across kitchen and front-of-house

Acceptance criteria:

- [ ] Real-time order status broadcasting
- [ ] Kitchen display system integration
- [ ] Connection management and reconnection logic
- [ ] Scalable WebSocket architecture

Implementation Notes:

- Add WebSocket support to Axum
- Implement order event broadcasting
- Create WebSocket authentication
- Add connection management
- Build scalable WebSocket clustering

TE2: Develop mobile app API endpoints

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 6-8 days
- **Dependencies**: CF1, CF3, CF4
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Create mobile-optimized API endpoints for staff applications and management tools

Acceptance criteria:

- [ ] Mobile-optimized response formats
- [ ] Offline capability support
- [ ] Push notification integration
- [ ] Mobile authentication flows

Implementation Notes:

- Design mobile-specific API endpoints
- Implement optimized data formats
- Add offline sync capabilities
- Integrate push notification service
- Create mobile authentication flows

TE3: Build kitchen display system interface

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 7-9 days
- **Dependencies**: CF3, TE1
- **Assignee**: TBD

- **Start Date**: TBD
- **Completion Date**: TBD

Description: Create dedicated interface for kitchen display systems showing orders, timing, and preparation status

Acceptance criteria:

- [ ] Real-time order display
- [ ] Preparation time tracking
- [ ] Kitchen workflow optimization
- [ ] Multi-screen support

Implementation Notes:

- Build dedicated kitchen UI
- Implement real-time order updates
- Add preparation time tracking
- Create kitchen workflow tools
- Support multiple display configurations

TE4: Implement reporting and analytics dashboard

- **Status**: 🔴 Pending
- **Priority**: Medium
- **Estimated effort**: 8-10 days
- **Dependencies**: CF1, CF2, CF3
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Build comprehensive analytics dashboard with sales, inventory, and performance metrics

Acceptance criteria:

- [ ] Sales reporting and analytics
- [ ] Inventory turnover analysis
- [ ] Staff performance metrics
- [ ] Customizable dashboard views

Implementation Notes:

- Design analytics data models
- Create reporting API endpoints
- Build dashboard frontend
- Implement data visualization
- Add custom report generation

🚀 Operational Improvement Tasks (3/24)

OI1: Set up CI/CD pipeline with GitHub Actions

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 3-4 days
- **Dependencies**: T1, T2
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Implement comprehensive CI/CD pipeline with automated testing, security scanning, and deployment

Acceptance criteria:

- ☐ Automated testing on pull requests
- ☐ Security vulnerability scanning
- ☐ Automated deployment to staging/production
- ☐ Rollback capabilities

Implementation Notes:

- Create GitHub Actions workflows
- Set up automated testing pipeline
- Add security scanning tools
- Configure deployment automation
- Implement rollback procedures

OI2: Implement feature flags for gradual rollouts

- **Status**: 🔴 Pending
- **Priority**: Medium
- **Estimated effort**: 4-5 days
- **Dependencies**: OI1
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Add feature flag system for safe deployment of new features and A/B testing

Acceptance criteria:

- ☐ Runtime feature toggle system
- ☐ User-based feature rollouts
- ☐ A/B testing capabilities
- ☐ Feature flag management interface

Implementation Notes:

- Implement feature flag service
- Create feature toggle middleware
- Add user-based feature targeting
- Build feature flag management UI
- Integrate with deployment pipeline

OI3: Configure monitoring and alerting with Prometheus/Grafana

- **Status**: 🔴 Pending
- **Priority**: High
- **Estimated effort**: 5-6 days
- **Dependencies**: None
- **Assignee**: TBD
- **Start Date**: TBD
- **Completion Date**: TBD

Description: Set up comprehensive monitoring, alerting, and observability for production systems

Acceptance criteria:

- ☐ Application metrics collection
- ☐ Business metrics dashboards
- ☐ Alerting for critical issues
- ☐ Log aggregation and analysis

Implementation Notes:

- Set up Prometheus metrics collection
- Create Grafana dashboards
- Configure alerting rules
- Implement log aggregation
- Add distributed tracing

📈 Progress Tracking

Week 1-2 Progress

- ☐ T1: Unit tests implementation
- ☐ S1: Validation middleware
- ☐ D1: API documentation

Week 3-4 Progress

- ☐ T2: Integration tests
- ☐ P1: gRPC connection pooling
- ☐ P2: Redis caching integration

Week 5-8 Progress

- [ ] CF1: Menu management system
- [ ] CF4: Staff management
- [ ] S2: Rate limiting

Week 9-12 Progress

- [ ] CF2: Inventory tracking
- [ ] CF3: Order management
- [ ] TE1: WebSocket implementation

Week 13-16 Progress

- [ ] CF5: Table/reservation system
- [ ] TE2: Mobile API endpoints
- [ ] TE3: Kitchen display system

Final Weeks Progress

- [ ] OI1: CI/CD pipeline
- [ ] OI2: Feature flags
- [ ] OI3: Monitoring setup
- [ ] T3: E2E testing
- [ ] TE4: Analytics dashboard

📝 Notes and Decisions

Technical Decisions

- Using Axum for web framework (already established)
- PostgreSQL for primary database (already established)
- Redis for caching and sessions
- WebSockets for real-time communication
- gRPC for internal service communication

Architecture Decisions

- Modular architecture with clear separation of concerns
- Event-driven design for real-time features
- Microservices approach for scalability
- API-first design for mobile and web clients

Next Steps

1. Review and prioritize tasks based on business requirements
2. Assign tasks to team members
3. Set up project tracking and communication

4. Begin with foundational tasks (Testing, Security, Documentation)
5. Regular progress reviews and roadmap adjustments

Last Updated: August 6, 2025
Next Review: TBD
Project Manager: TBD