# CAFE MANAGEMENT SYSTEM

## **Technical Documentation**

### **A Comprehensive JavaFX Desktop Application**

**Version:** 1.0.0  
**Author:** Team 2\_C2406L  
**Date:** 2025  
**Technology Stack:** Java 21, JavaFX 24, MySQL 8.0, Maven

## **TABLE OF CONTENTS**

### **PART I: PROJECT OVERVIEW & INTRODUCTION**

1. [Executive Summary](#X102b2e9de4183984340b81b0bb402a984d3925a)
2. [System Introduction](#X40d60e48906b95bd60b963fdec2c88f63d17c65)
3. [Technology Stack](#Xd4be06774eb8fef4566a4cdca4cb8ee01c79793)
4. [Project Structure Overview](#X1ec74576fc66a3adc97386b1525e94d8db9c7e0)

### **PART II: SYSTEM ARCHITECTURE & DESIGN**

1. [Architectural Overview](#Xb09d22497c780cdd62108db31a0c95c3e51f010)
2. [Database Design](#Xdf3b3627d88400041a9e3e6eb491f39baf9d5b4)
3. [Security Architecture](#X5cdd4290724e4563dd66f08e609ef95ed53719f)
4. [UI/UX Design Architecture](#X79157078afb0d70f50ffd011b8e5283f6c29ce1)

### **PART III: CORE MODULES & COMPONENTS**

1. [Authentication & User Management](#Xb979a9253082fedfd766c8e340a9fe5b67d0705)
2. [Menu & Product Management](#X618c9cfe6d857d1b01a4d9874ff60ff1436463f)
3. [Table & Area Management](#X18c5c64dd2f5f38e77b9df1b1b81b25f612092d)
4. [Order Processing System](#Xd910de30ef7510157cdd9ee1cb4a3a7a5411981)
5. [Payment Processing System](#Xa5bd4a76e4e94d3ff7bdaaacb196fea98d6612f)

### **PART IV: ADVANCED FEATURES & INTEGRATIONS**

1. [Customer Management System](#Xd449faab11f278c813bf35be4db35d5ce602d8f)
2. [Promotion & Discount System](#X3412d14567a0de5bd29ae5507e6510b7e64655e)
3. [Reporting & Analytics](#X4795deac424b54050e746b9d5909c5c87947161)
4. [QR Code & Receipt System](#X5caba3cec53360735db1d372b8fd5ad22dd3f89)

### \*\*PART V: DATA ACCESS & BUSINESS LOGIC\*

1. [Data Access Layer (DAO Pattern)](#Xb9c9c1dda6b988f02052d40f387319645c61c78)
2. [Service Layer Business Logic](#X12f184a16e9ec51f4f1511fb2a47c4cbc64778e)

### **PART VI: USER INTERFACES & CONTROLLERS**

1. [Controller Architecture](#X400cb5c47357a9faffde72a348adfea660443a2)
2. [Main Application Controllers](#Xfc530c041f09f0ff584793c4c3c3b71fd82872f)

### **PART VII: TECHNICAL IMPLEMENTATION**

1. [Development Environment & Setup](#Xb21037092a15a56b27bb6916fba6ff6ebfbf948)
2. [Configuration Management](#X34d60823bedbbbdd708217992cbe264739f13b8)
3. [Error Handling & Logging](#X4e327c88836e3f8917aab747e30fa88e5e2eb38)
4. [Testing Strategy](#X89fadffb4442af4ef00f604b1b1f03707816a5a)

### **PART VIII: DEPLOYMENT & MAINTENANCE**

1. [Build & Deployment](#Xf74816945f758a79c208edc08735a2d8d527e42)
2. [Performance Optimization](#Xc867345504d68adc77b9e1003cd0a9e6be3723b)
3. [Maintenance & Monitoring](#X730fefa0da02b975afe977936723fb2851760e8)

### **PART IX: APPENDICES**

1. [API Documentation](#X89000ec693977513e7ca20426dab45d05028eaf)
2. [Database Schema Reference](#X740a79aae7e2fdcbd61f0b27936d338281cc352)
3. [Configuration Reference](#X796db31be63bfe842db4c0e723cca202bca95f7)
4. [Troubleshooting Guide](#X4afda1cadb9fc124f0ffaf2bb1b095611803178)

# **PART I: PROJECT OVERVIEW & INTRODUCTION**

## **1. Executive Summary**

### **1.1 Project Overview**

The Cafe Management System is a comprehensive desktop application designed to streamline operations in coffee shops and small restaurants. Built using modern Java technologies, this system provides an intuitive interface for managing daily operations including order processing, inventory management, customer relationships, and financial reporting.

**Key Objectives:**

* Digitize and automate cafe operations
* Improve customer service efficiency
* Provide real-time business insights
* Ensure data security and reliability
* Support scalable business growth

### **1.2 Key Features and Functionalities**

**Core Business Functions:**

* **User Management**: Role-based access control with Admin and Staff roles
* **Menu Management**: Dynamic product catalog with categories and pricing
* **Table Management**: Real-time table status tracking and reservation system
* **Order Processing**: Streamlined order creation, modification, and completion
* **Payment Processing**: Multiple payment methods including cash, card, and mobile payments
* **Customer Management**: Customer profiles with loyalty program integration
* **Inventory Tracking**: Stock management with automated alerts
* **Reporting & Analytics**: Comprehensive business intelligence dashboard

**Technical Features:**

* **Cross-platform Compatibility**: Desktop application supporting Windows, macOS, and Linux
* **Real-time Updates**: Live synchronization of order status and table availability
* **Data Security**: Encrypted password storage and secure database connections
* **Backup & Recovery**: Automated database backup with point-in-time recovery
* **Extensible Architecture**: Modular design supporting future enhancements

### **1.3 Technology Stack Overview**

**Frontend Technologies:**

* **JavaFX 24**: Modern UI framework for rich desktop applications
* **FXML**: Declarative markup for user interface design
* **CSS**: Advanced styling and theming capabilities

**Backend Technologies:**

* **Java 21**: Latest LTS version with enhanced performance features
* **Maven**: Dependency management and build automation
* **HikariCP**: High-performance database connection pooling

**Database Technologies:**

* **MySQL 8.0**: Enterprise-grade relational database management system
* **JDBC**: Database connectivity and transaction management

**Supporting Libraries:**

* **BCrypt**: Advanced password hashing and security
* **Apache POI**: Excel document generation and export
* **iText PDF**: Professional PDF document creation
* **ZXing**: QR code generation for mobile payments
* **SLF4J/Logback**: Comprehensive logging framework

### **1.4 Target Users and Use Cases**

**Primary Users:**

1. **Cafe Staff (Cashiers/Waiters)**
   * Process customer orders efficiently
   * Manage table assignments and status
   * Handle payment transactions
   * Access basic reporting functions
2. **Cafe Managers/Owners (Admins)**
   * Oversee all system operations
   * Manage staff accounts and permissions
   * Configure menu items and pricing
   * Generate detailed business reports
   * Monitor system performance and health

**Use Case Scenarios:**

1. **Daily Operations**
   * Morning setup and system initialization
   * Peak hour order processing and management
   * Table turnover optimization
   * End-of-day reporting and reconciliation
2. **Customer Service**
   * Order customization and special requests
   * Payment processing with multiple methods
   * Customer loyalty program management
   * Issue resolution and refund processing
3. **Business Management**
   * Menu planning and pricing strategy
   * Staff performance monitoring
   * Financial analysis and reporting
   * Inventory management and forecasting

### **1.5 Business Value and Benefits**

**Operational Efficiency:**

* **50% reduction** in order processing time
* **30% improvement** in table turnover rate
* **Automated inventory tracking** reducing manual errors
* **Real-time reporting** enabling quick decision making

**Customer Experience:**

* **Faster service delivery** with streamlined ordering
* **Accurate order processing** minimizing mistakes
* **Flexible payment options** improving convenience
* **Loyalty program integration** encouraging repeat business

**Business Intelligence:**

* **Comprehensive analytics** for data-driven decisions
* **Sales trend analysis** for inventory optimization
* **Customer behavior insights** for targeted marketing
* **Financial reporting** for accurate business planning

**Technology Benefits:**

* **Scalable architecture** supporting business growth
* **Cross-platform compatibility** reducing hardware constraints
* **Secure data management** ensuring business continuity
* **Easy maintenance** with modular design approach

## **2. System Introduction**

### **2.1 Problem Statement and Business Requirements**

**Current Industry Challenges:**

Modern cafe and restaurant operations face numerous challenges that traditional manual systems cannot adequately address:

1. **Order Management Complexity**
   * Manual order tracking leads to errors and delays
   * Difficulty managing peak hour rushes
   * Lack of order customization flexibility
   * Poor coordination between kitchen and service staff
2. **Inventory and Menu Management**
   * Manual inventory tracking prone to errors
   * Difficulty updating prices and menu items
   * Limited ability to track popular products
   * Waste due to poor inventory forecasting
3. **Customer Service Limitations**
   * Slow payment processing
   * Limited payment method options
   * No customer history or loyalty tracking
   * Difficulty handling special requests or modifications
4. **Business Intelligence Gaps**
   * Limited visibility into sales performance
   * Manual reporting processes are time-consuming
   * Difficulty identifying trends and patterns
   * Lack of real-time business metrics

**Specific Business Requirements:**

1. **Functional Requirements**
   * User authentication and role-based access control
   * Complete order lifecycle management
   * Multi-table and area management
   * Flexible payment processing system
   * Customer relationship management
   * Comprehensive reporting and analytics
2. **Non-Functional Requirements**
   * System response time under 2 seconds
   * 99.9% uptime during business hours
   * Support for concurrent users (up to 10)
   * Data backup and recovery capabilities
   * Cross-platform desktop compatibility
3. **Security Requirements**
   * Secure user authentication
   * Role-based access control
   * Encrypted sensitive data storage
   * Audit trail for critical operations
   * Protection against common security vulnerabilities

### **2.2 Solution Approach and Methodology**

**Development Methodology:**

The project follows an **Agile Development Approach** with the following characteristics:

1. **Iterative Development**
   * Modular component development
   * Continuous integration and testing
   * Regular stakeholder feedback incorporation
   * Incremental feature delivery
2. **Design Patterns Implementation**
   * **Model-View-Controller (MVC)**: Clear separation of concerns
   * **Data Access Object (DAO)**: Abstracted database operations
   * **Service Layer**: Encapsulated business logic
   * **Factory Pattern**: Flexible object creation
   * **Observer Pattern**: Real-time UI updates
3. **Quality Assurance**
   * Unit testing with JUnit 5
   * Integration testing for database operations
   * User acceptance testing for UI components
   * Performance testing for concurrent operations

**Technical Solution Architecture:**

1. **Layered Architecture Approach**
   * **Presentation Layer**: JavaFX-based user interfaces
   * **Business Logic Layer**: Service classes with core business rules
   * **Data Access Layer**: DAO pattern with MySQL integration
   * **Cross-Cutting Concerns**: Security, logging, and error handling
2. **Database-Centric Design**
   * Normalized relational database schema
   * Optimized queries and indexing strategy
   * Transaction management for data consistency
   * Connection pooling for performance
3. **User-Centric Interface Design**
   * Intuitive navigation and workflow
   * Responsive design principles
   * Role-based interface customization
   * Accessibility considerations

### **2.3 System Scope and Limitations**

**Project Scope:**

**Included Features:**

* Complete order management lifecycle
* User authentication and authorization
* Menu and product catalog management
* Table and area management
* Payment processing with multiple methods
* Customer relationship management
* Basic inventory tracking
* Reporting and analytics dashboard
* Receipt generation and printing
* QR code integration for payments

**Technical Scope:**

* Desktop application for Windows, macOS, and Linux
* MySQL database backend
* Local network deployment
* Single cafe location support
* English and Vietnamese language support

**Current Limitations:**

1. **Deployment Limitations**
   * Single-location deployment (no multi-branch support)
   * Local network only (no cloud deployment)
   * Desktop application only (no web or mobile interfaces)
2. **Integration Limitations**
   * No external payment gateway integration
   * No accounting software integration
   * No supplier management system
   * No employee scheduling system
3. **Scalability Limitations**
   * Maximum 10 concurrent users
   * Single database instance
   * No horizontal scaling capability
   * Limited to small to medium-sized cafes
4. **Functional Limitations**
   * Basic inventory management (no advanced forecasting)
   * No kitchen display system integration
   * No online ordering capability
   * No delivery management system

**Future Enhancement Opportunities:**

* Cloud-based deployment options
* Mobile application development
* Multi-location support
* Advanced analytics and AI integration
* Third-party system integrations
* Kitchen automation integration

### **2.4 Success Criteria and Metrics**

**Primary Success Criteria:**

1. **Functional Success Metrics**
   * 100% of core features implemented and tested
   * User acceptance rate > 90%
   * System uptime > 99% during business hours
   * Order processing time reduction > 40%
2. **Performance Success Metrics**
   * Average response time < 2 seconds
   * Database query performance < 500ms
   * Memory usage < 512MB under normal load
   * Startup time < 30 seconds
3. **User Experience Success Metrics**
   * User training time < 2 hours
   * Task completion rate > 95%
   * User satisfaction score > 4.0/5.0
   * Error rate < 5% of transactions
4. **Business Impact Success Metrics**
   * Operational efficiency improvement > 30%
   * Customer service time reduction > 25%
   * Order accuracy improvement > 95%
   * Daily reporting automation 100%

**Key Performance Indicators (KPIs):**

1. **Operational KPIs**
   * Orders processed per hour
   * Average order processing time
   * Table turnover rate
   * System availability percentage
2. **Financial KPIs**
   * Daily revenue tracking accuracy
   * Payment processing success rate
   * Inventory variance reduction
   * Cost per transaction
3. **Customer KPIs**
   * Customer satisfaction score
   * Repeat customer rate
   * Average order value
   * Customer service response time
4. **Technical KPIs**
   * System response time
   * Database performance metrics
   * Error rate and resolution time
   * User adoption rate

## **3. Technology Stack**

### **3.1 Frontend Technologies**

**JavaFX 24 - Rich Client Platform**

JavaFX serves as the primary UI framework, providing modern desktop application capabilities:

**Core Features:**

* **Scene Graph Architecture**: Hierarchical structure for UI components
* **CSS Styling Support**: Advanced theming and visual customization
* **FXML Integration**: Declarative UI design with separation of concerns
* **Animation Framework**: Smooth transitions and visual effects
* **Media Support**: Image, audio, and video integration capabilities

**Benefits for Cafe Management System:**

* **Native Look and Feel**: Consistent user experience across platforms
* **High Performance Rendering**: Smooth UI interactions even during peak usage
* **Responsive Design**: Adaptive layouts for different screen resolutions
* **Rich Controls**: Advanced components like TableView, Charts, and Custom Controls

**FXML - Declarative UI Framework**

FXML enables separation of UI design from business logic:

**Key Advantages:**

* **Design-Developer Separation**: UI designers can work independently
* **Maintainable Code**: Clean separation between presentation and logic
* **Scene Builder Integration**: Visual design tools support
* **Internationalization**: Easy localization and multi-language support

**Implementation in Project:**

* Login interface with form validation
* Dashboard layouts with dynamic content areas
* Admin panels for system configuration
* Order processing interfaces with real-time updates

**CSS Styling Framework**

Custom CSS implementation provides professional theming:

**Styling Strategy:**

* **Modular CSS Architecture**: Separate stylesheets for different modules
* **Color Scheme Management**: Consistent branding across all interfaces
* **Responsive Design**: Adaptive layouts for different screen sizes
* **Theme Customization**: Easy modification of visual appearance

### **3.2 Backend Technologies**

**Java 21 - Core Programming Language**

Latest LTS version providing enhanced performance and security:

**New Features Utilized:**

* **Enhanced Performance**: Improved JVM optimizations
* **Security Enhancements**: Better cryptographic support
* **Modern Syntax**: Pattern matching and sealed classes
* **Memory Management**: Improved garbage collection algorithms

**Project-Specific Benefits:**

* **Concurrent Processing**: Better handling of multiple user sessions
* **Database Performance**: Optimized connection handling
* **Security Features**: Enhanced password encryption and session management
* **Cross-Platform Compatibility**: Consistent behavior across operating systems

**Maven - Build and Dependency Management**

Comprehensive project management and build automation:

**Configuration Management:**

* **Dependency Resolution**: Automatic library management
* **Build Lifecycle**: Standardized compilation and packaging
* **Profile Management**: Environment-specific configurations
* **Plugin Integration**: Extended functionality for specialized tasks

**Project Build Structure:**

* **Development Profile**: Debug mode with enhanced logging
* **Production Profile**: Optimized performance settings
* **Testing Profile**: Specialized testing configurations
* **Deployment Profile**: Distribution-ready packaging

**HikariCP - Database Connection Pooling**

High-performance JDBC connection pool management:

**Performance Benefits:**

* **Fast Connection Acquisition**: Minimal overhead for database access
* **Efficient Resource Management**: Optimized connection lifecycle
* **Monitoring Capabilities**: Real-time connection pool metrics
* **Fault Tolerance**: Automatic connection recovery

**Configuration for Cafe System:**

* Maximum pool size: 50 connections
* Minimum idle connections: 5
* Connection timeout: 10 seconds
* Idle timeout: 10 minutes

### **3.3 Database Technologies**

**MySQL 8.0 - Enterprise Database Management**

Robust relational database supporting complex business operations:

**Advanced Features:**

* **JSON Support**: Flexible data structures for configuration
* **Window Functions**: Advanced analytical queries
* **Common Table Expressions**: Complex query optimization
* **Enhanced Security**: Improved authentication and encryption

**Database Design for Cafe System:**

* **Normalized Schema**: Efficient data organization
* **Optimized Indexes**: Fast query performance
* **Foreign Key Constraints**: Data integrity maintenance
* **Stored Procedures**: Complex business logic implementation

**JDBC - Database Connectivity**

Standard Java database connectivity with optimized performance:

**Implementation Strategy:**

* **Prepared Statements**: SQL injection prevention
* **Transaction Management**: ACID compliance for critical operations
* **Batch Processing**: Efficient bulk data operations
* **Connection Pooling**: Resource optimization

### **3.4 Supporting Libraries and Frameworks**

**Security Framework - BCrypt**

Advanced password hashing and authentication:

**Security Features:**

* **Adaptive Hashing**: Configurable cost factors
* **Salt Generation**: Automatic salt creation
* **Timing Attack Resistance**: Secure comparison operations
* **Future-Proof Algorithm**: Scalable security levels

**Document Generation - Apache POI & iText**

Professional document creation and export capabilities:

**Apache POI for Excel:**

* **Spreadsheet Generation**: Dynamic report creation
* **Formula Support**: Automated calculations
* **Styling Options**: Professional formatting
* **Large Dataset Handling**: Memory-efficient processing

**iText for PDF:**

* **Professional Layout**: Complex document structures
* **Image Integration**: Logo and graphics support
* **Table Formatting**: Structured data presentation
* **Digital Signatures**: Document authentication

**QR Code Generation - ZXing**

Mobile payment integration and modern customer experience:

**Implementation Features:**

* **Payment QR Codes**: Mobile payment integration
* **Table Identification**: Quick table access
* **Order Tracking**: Customer order status
* **Marketing Integration**: Promotional campaigns

**Logging Framework - SLF4J/Logback**

Comprehensive logging and monitoring system:

**Logging Strategy:**

* **Structured Logging**: Consistent log formats
* **Level-Based Filtering**: Appropriate detail levels
* **File Rotation**: Automatic log management
* **Performance Monitoring**: System health tracking

### **3.5 Development and Testing Tools**

**Testing Framework - JUnit 5 & Mockito**

Comprehensive testing strategy for reliability:

**Testing Approach:**

* **Unit Testing**: Individual component validation
* **Integration Testing**: Database and service layer testing
* **Mock Testing**: Isolated component testing
* **Performance Testing**: Load and stress testing

**IDE and Development Tools**

**Supported Development Environments:**

* **IntelliJ IDEA**: Professional Java development
* **Eclipse IDE**: Open-source development platform
* **Visual Studio Code**: Lightweight development option
* **Scene Builder**: Visual FXML design tool

**Version Control and Collaboration:**

* **Git**: Distributed version control
* **GitHub**: Repository hosting and collaboration
* **Maven Central**: Dependency repository access
* **Continuous Integration**: Automated build and testing

This comprehensive technology stack provides a solid foundation for building a reliable, scalable, and maintainable cafe management system while ensuring modern development practices and industry standards compliance.

## **4. Project Structure Overview**

### **4.1 Directory Organization**

The Cafe Management System follows a well-organized directory structure that separates concerns and facilitates maintainability, scalability, and team collaboration.

**Root Directory Structure:**

Project2/  
├── src/ # Source code directory  
│ ├── main/ # Main application code  
│ │ ├── java/ # Java source files  
│ │ └── resources/ # Application resources  
│ └── test/ # Test code directory  
├── database/ # Database-related files  
├── docs/ # Documentation  
├── scripts/ # Build and deployment scripts  
├── target/ # Build output directory  
├── lib/ # External libraries  
├── config/ # Configuration files  
├── logs/ # Application logs  
├── temp/ # Temporary files  
├── backup/ # Database backups  
├── pom.xml # Maven configuration  
└── README.md # Project documentation

**Source Code Organization:**

The source code follows Java package conventions with clear separation of responsibilities:

src/main/java/com/cafe/  
├── CafeManagementApplication.java # Main application entry point  
├── config/ # Configuration management  
│ └── DatabaseConfig.java # Database connection configuration  
├── controller/ # MVC Controllers (Presentation Layer)  
│ ├── admin/ # Administrative controllers  
│ ├── auth/ # Authentication controllers  
│ ├── base/ # Base controller classes  
│ ├── customer/ # Customer management controllers  
│ ├── dashboard/ # Dashboard controllers  
│ ├── menu/ # Menu management controllers  
│ ├── order/ # Order processing controllers  
│ ├── payment/ # Payment processing controllers  
│ ├── promotion/ # Promotion management controllers  
│ └── table/ # Table management controllers  
├── dao/ # Data Access Objects (Data Layer)  
│ └── base/ # DAO implementations  
├── exception/ # Custom exception classes  
├── model/ # Data Models (Domain Layer)  
│ ├── dto/ # Data Transfer Objects  
│ ├── entity/ # Database entities  
│ └── enums/ # Enumeration types  
├── service/ # Business Logic (Service Layer)  
├── util/ # Utility classes  
└── view/ # View-related utilities

### **4.2 Package Structure Analysis**

**Configuration Package (config/)**

**Purpose**: Centralized configuration management

* **DatabaseConfig.java**: Database connection pooling and configuration
* **ApplicationConfig.java**: Application-wide settings and properties
* **SecurityConfig.java**: Security-related configurations

**Responsibilities:**

* Database connection management
* Environment-specific configurations
* Security parameter initialization
* **Logging configuration setup**

**Controller Package (controller/)**

**Purpose**: User interface interaction handling and request processing

**Sub-packages:**

1. **admin/** - Administrative Interface Controllers
   * **AdminDashboardController**: Main admin interface coordination
   * **AdminMenuController**: Product and category management
   * **AdminUserController**: User account management
   * **AdminTableController**: Table and area configuration
   * **AdminReportController**: Business reporting interfaces
2. **auth/** - Authentication Controllers
   * **LoginController**: User authentication handling
   * **SessionController**: Session management
3. **dashboard/** - Main Dashboard Controllers
   * **DashboardController**: Staff dashboard coordination
   * **OverviewController**: System overview and metrics
4. **order/** - Order Processing Controllers
   * **OrderPanelController**: Order creation and management
   * **OrderHistoryController**: Order tracking and history
5. **payment/** - Payment Processing Controllers
   * **PaymentController**: Payment method handling
   * **ReceiptController**: Receipt generation and printing

**Model Package (model/)**

**Purpose**: Data representation and business domain modeling

**Sub-packages:**

1. **entity/** - Database Entity Classes
   * **User**: User account information
   * **Product**: Menu item details
   * **Category**: Product categorization
   * **Order**: Order information
   * **OrderDetail**: Order line items
   * **Customer**: Customer profiles
   * **TableCafe**: Table management
   * **Area**: Cafe area organization
   * **Promotion**: Discount and promotion rules
2. **dto/** - Data Transfer Objects
   * **PaymentRequest**: Payment processing data
   * **PaymentResponse**: Payment result information
   * **SalesData**: Reporting data structures
3. **enums/** - Enumeration Types
   * **UserRole**: User permission levels
   * **OrderStatus**: Order lifecycle states
   * **PaymentMethod**: Payment type options
   * **TableStatus**: Table availability states

**DAO Package (dao/)**

**Purpose**: Database access and data persistence management

**Structure:**

* **Interface Definitions**: Abstract DAO contracts
* **Implementation Classes**: Concrete database operations
* **Base DAO**: Common database operations

**Key Components:**

* **UserDAO/UserDAOImpl**: User data access
* **ProductDAO/ProductDAOImpl**: Product data management
* **OrderDAO/OrderDAOImpl**: Order data operations
* **CustomerDAO/CustomerDAOImpl**: Customer data handling

**Service Package (service/)**

**Purpose**: Business logic implementation and operation coordination

**Key Services:**

* **UserService**: User management business logic
* **MenuService**: Product and category business rules
* **OrderService**: Order processing workflows
* **PaymentService**: Payment processing logic
* **CustomerService**: Customer relationship management
* **TableService**: Table management operations
* **PromotionService**: Discount calculation logic
* **ReportService**: Business analytics and reporting

**Utility Package (util/)**

**Purpose**: Common functionality and helper classes

**Utility Classes:**

* **AlertUtils**: User notification management
* **DateUtils**: Date and time formatting
* **PriceFormatter**: Currency and price formatting
* **ValidationUtils**: Input validation utilities
* **SessionManager**: User session management
* **PasswordUtil**: Password encryption utilities
* **ExcelExporter**: Excel document generation
* **PDFExporter**: PDF document creation

### **4.3 Resource Management**

**Resource Directory Structure:**

src/main/resources/  
├── fxml/ # FXML layout files  
│ ├── admin/ # Administrative interfaces  
│ ├── auth/ # Authentication screens  
│ ├── dashboard/ # Dashboard layouts  
│ ├── order/ # Order processing screens  
│ └── payment/ # Payment interfaces  
├── css/ # Stylesheet files  
│ ├── admin-dashboard.css # Admin interface styling  
│ ├── dashboard.css # Main dashboard styling  
│ ├── login.css # Login screen styling  
│ └── common.css # Shared styling  
├── images/ # Image resources  
│ ├── icons/ # UI icons  
│ ├── logo/ # Brand logos  
│ ├── placeholders/ # Placeholder images  
│ └── products/ # Product images  
│ ├── coffee/ # Coffee product images  
│ ├── tea/ # Tea product images  
│ ├── cake/ # Cake product images  
│ └── juice/ # Juice product images  
└── properties/ # Configuration files  
 └── database\_config.properties

**FXML File Organization:**

Each FXML file corresponds to a specific user interface component: - **Modular Design**: Each screen is a separate FXML file - **Controller Binding**: FXML files are bound to specific controller classes - **Reusable Components**: Common UI elements are componentized - **Consistent Layout**: Standardized layout patterns across screens

**CSS Styling Strategy:**

* **Modular Stylesheets**: Separate CSS files for different application areas
* **Theme Consistency**: Unified color schemes and typography
* **Responsive Design**: Flexible layouts for different screen sizes
* **Custom Controls**: Styled components for specific business needs

### **4.4 Build and Deployment Process**

**Maven Build Configuration:**

The project uses Maven for dependency management and build automation:

**Key Configuration Elements:**

* **Java 21 Compatibility**: Source and target compatibility settings
* **JavaFX Dependencies**: UI framework integration
* **Database Dependencies**: MySQL connector and connection pooling
* **Security Libraries**: Password hashing and encryption
* **Testing Framework**: JUnit and Mockito integration
* **Utility Libraries**: PDF generation, Excel export, QR code creation

**Build Profiles:**

1. **Development Profile**
   * Debug logging enabled
   * Test database configuration
   * Hot reload capabilities
   * Detailed error reporting
2. **Production Profile**
   * Optimized performance settings
   * Production database configuration
   * Minimized logging
   * Security hardening
3. **Testing Profile**
   * Test-specific configurations
   * Mock database integration
   * Comprehensive test coverage
   * Performance benchmarking

**Deployment Strategy:**

**Desktop Application Packaging:**

* **JAR File Generation**: Self-contained executable JAR
* **Dependencies Inclusion**: All required libraries bundled
* **Configuration Management**: Environment-specific settings
* **Installation Scripts**: Automated setup procedures

**Database Deployment:**

* **Schema Creation**: Automated database structure setup
* **Sample Data**: Initial data population scripts
* **Migration Scripts**: Version update procedures
* **Backup Procedures**: Data protection strategies

This well-organized project structure ensures maintainability, scalability, and ease of development while following industry best practices for Java enterprise applications.

# **PART II: SYSTEM ARCHITECTURE & DESIGN**

## **5. Architectural Overview**

### **5.1 MVC Pattern Implementation**

The Cafe Management System is built on a robust Model-View-Controller (MVC) architecture that ensures clean separation of concerns, maintainability, and scalability.

**MVC Architecture Benefits:**

* **Separation of Concerns**: Clear distinction between data, presentation, and control logic
* **Maintainability**: Easy to modify individual components without affecting others
* **Testability**: Isolated components can be tested independently
* **Reusability**: Components can be reused across different parts of the application
* **Team Development**: Different developers can work on different layers simultaneously

**Model Layer Responsibilities:**

The Model layer represents the core business data and logic of the cafe management system:

**Entity Classes:**

* **User Entity**: Represents staff members and administrators
  + Attributes: userId, username, fullName, email, phone, role, isActive
  + Business Rules: Password encryption, role validation, activity status
* **Product Entity**: Represents menu items and products
  + Attributes: productId, productName, categoryId, price, description, imageUrl
  + Business Rules: Price validation, availability status, category relationships
* **Order Entity**: Represents customer orders
  + Attributes: orderId, tableId, customerId, orderDate, totalAmount, orderStatus
  + Business Rules: Order total calculations, status transitions, payment validation
* **Customer Entity**: Represents customer information
  + Attributes: customerId, fullName, phone, email, loyaltyPoints, totalSpent
  + Business Rules: Loyalty point calculations, contact validation, spending tracking

**Data Transfer Objects (DTOs):**

* **PaymentRequest**: Encapsulates payment processing data
* **PaymentResponse**: Contains payment processing results
* **SalesData**: Aggregated sales information for reporting

**Enumeration Classes:**

* **UserRole**: ADMIN, STAFF role definitions
* **OrderStatus**: PENDING, PREPARING, READY, COMPLETED, CANCELLED
* **PaymentMethod**: CASH, CARD, MOMO, VNPAY payment options
* **TableStatus**: AVAILABLE, OCCUPIED, RESERVED, CLEANING states

**View Layer (JavaFX) Structure:**

The View layer handles all user interface components and user interactions:

**FXML-based Interface Design:**

* **Declarative UI Definition**: FXML files define interface structure
* **CSS Styling**: Professional appearance with consistent theming
* **Component Binding**: Automatic data binding between UI and controllers
* **Event Handling**: User interaction management

**Key Interface Components:**

1. **Login Interface** (login.fxml)
   * User authentication form
   * Password field with security masking
   * Remember me functionality
   * Error message display area
2. **Dashboard Interface** (dashboard.fxml)
   * Navigation tab structure
   * Content area for dynamic loading
   * User information display
   * Quick action buttons
3. **Admin Dashboard** (admin-dashboard.fxml)
   * Administrative navigation menu
   * System overview widgets
   * Management panels access
   * Reporting dashboard
4. **Order Panel** (order\_panel.fxml)
   * Product selection grid
   * Order item list
   * Total calculation display
   * Payment processing area

**Controller Layer Organization:**

The Controller layer manages user interactions and coordinates between View and Model:

**Base Controller Architecture:**

* **Abstract Base Controllers**: Common functionality for similar controllers
* **Interface Definitions**: Standardized controller contracts
* **Event Handling**: Centralized event management
* **Navigation Control**: Coordinated screen transitions

**Specialized Controllers:**

1. **Authentication Controllers**
   * **LoginController**: User authentication and session initialization
   * Validates user credentials
   * Manages session creation
   * Handles authentication errors
2. **Dashboard Controllers**
   * **DashboardController**: Main application navigation
   * **AdminDashboardController**: Administrative interface management
   * Coordinates content loading
   * Manages user context
3. **Business Process Controllers**
   * **OrderPanelController**: Order creation and management
   * **PaymentController**: Payment processing workflows
   * **MenuController**: Product display and selection
   * **TableController**: Table status management

### **5.2 Layered Architecture**

The system implements a multi-layered architecture that promotes separation of concerns and maintainability:

**Presentation Layer (JavaFX Controllers)**

**Responsibilities:**

* User interface interaction handling
* Input validation and formatting
* Navigation and workflow management
* Error display and user feedback

**Key Components:**

* FXML-based user interfaces
* Controller classes for interaction handling
* CSS stylesheets for visual presentation
* Resource management for images and assets

**Business Logic Layer (Service Classes)**

**Responsibilities:**

* Core business rule implementation
* Transaction coordination
* Data validation and processing
* Business workflow orchestration

**Service Classes:**

1. **OrderService**: Order processing business logic
   * Order creation and validation
   * Product addition and removal
   * Price calculation and discount application
   * Order status management
2. **PaymentService**: Payment processing logic
   * Payment method validation
   * Transaction processing
   * Receipt generation
   * Payment status tracking
3. **MenuService**: Product management logic
   * Product CRUD operations
   * Category management
   * Price management
   * Availability tracking
4. **CustomerService**: Customer relationship management
   * Customer profile management
   * Loyalty point calculations
   * Purchase history tracking
   * Communication preferences

**Data Access Layer (DAO Classes)**

**Responsibilities:**

* Database connection management
* CRUD operation implementation
* Query optimization
* Data mapping and transformation

**DAO Implementation Pattern:**

* Interface-based design for flexibility
* Concrete implementations for specific databases
* Connection pooling for performance
* Transaction management for consistency

**Cross-Cutting Concerns**

**Security Management:**

* User authentication and authorization
* Password encryption and validation
* Session management and timeout
* Role-based access control

**Logging and Monitoring:**

* Application event logging
* Performance monitoring
* Error tracking and reporting
* Audit trail maintenance

**Configuration Management:**

* Environment-specific settings
* Database connection parameters
* Application feature toggles
* User preference management

### **5.3 Design Patterns Used**

**Data Access Object (DAO) Pattern**

**Purpose**: Abstraction of data access logic from business logic

**Implementation:**

* Interface definitions for each entity
* Concrete implementations for MySQL database
* Connection management and resource cleanup
* Exception handling and error management

**Benefits:**

* Database independence
* Testability with mock implementations
* Consistent data access patterns
* Centralized query management

**Service Layer Pattern**

**Purpose**: Encapsulation of business logic and transaction management

**Implementation:**

* Service interfaces defining business operations
* Service implementations with business rules
* Transaction boundary management
* Cross-service coordination

**Benefits:**

* Business logic centralization
* Transaction consistency
* Service reusability
* Clear API definitions

**Factory Pattern**

**Purpose**: Object creation abstraction and configuration

**Implementation:**

* DAO factory for database access objects
* Service factory for business services
* UI component factories for interface elements
* Configuration factories for setup management

**Benefits:**

* Flexible object creation
* Configuration-driven instantiation
* Easy testing with mock objects
* Reduced coupling between components

**Observer Pattern**

**Purpose**: Real-time UI updates and event notification

**Implementation:**

* Table status change notifications
* Order status update broadcasts
* Inventory level alerts
* User interface refresh triggers

**Benefits:**

* Automatic UI synchronization
* Event-driven architecture
* Loose coupling between components
* Real-time data consistency

**Command Pattern**

**Purpose**: Request encapsulation and undo functionality

**Implementation:**

* Order modification commands
* Payment processing commands
* User action commands
* Batch operation commands

**Benefits:**

* Undo/redo functionality
* Request queuing capability
* Macro command composition
* Audit trail generation

### **5.4 Component Integration**

**Inter-Layer Communication**

**Controller to Service Communication:**

* Controllers call service methods for business operations
* Service layer returns DTOs or domain objects
* Error handling through exception propagation
* Transaction boundaries managed at service level

**Service to DAO Communication:**

* Services coordinate multiple DAO operations
* Transaction management across multiple data sources
* Data transformation between business and persistence models
* Connection resource management

**Real-time Update Mechanism**

**Event-Driven Updates:**

* Table status changes trigger UI refreshes
* Order completions update dashboard metrics
* Inventory changes affect product availability
* Payment completions update order status

**Data Consistency:**

* Database transactions ensure ACID compliance
* Optimistic locking for concurrent updates
* Validation at multiple layers
* Error recovery and rollback procedures

This comprehensive architectural design ensures that the Cafe Management System is robust, maintainable, scalable, and follows industry best practices for enterprise application development.

## **6. Database Design**

### **6.1 Database Schema Overview**

The Cafe Management System utilizes a well-normalized MySQL 8.0 database designed to efficiently handle all aspects of cafe operations while maintaining data integrity and optimal performance.

**Database Design Principles:**

1. **Normalization**: Tables are normalized to 3NF (Third Normal Form) to eliminate redundancy
2. **Referential Integrity**: Foreign key constraints ensure data consistency
3. **Performance Optimization**: Strategic indexing for frequently accessed data
4. **Scalability**: Schema design supports future growth and feature additions
5. **Data Security**: Sensitive information protection through proper field design

**Core Database Tables (12 Primary Tables):**

1. **users** - User accounts and authentication
2. **categories** - Product categorization
3. **products** - Menu items and products
4. **areas** - Physical cafe areas
5. **tables** - Individual table management
6. **customers** - Customer information
7. **orders** - Order records
8. **order\_details** - Order line items
9. **promotions** - Discount and promotion rules
10. **order\_promotions** - Applied promotions tracking
11. **attendance** - Staff attendance records
12. **system\_settings** - Application configuration

### **6.2 Entity Relationships and Data Model**

**Primary Entity Relationships:**

**User Management Relationships:**

users (1) ←→ (N) orders  
users (1) ←→ (N) attendance

* Each user can create multiple orders
* Each user can have multiple attendance records
* Users are authenticated staff members with role-based permissions

**Product Catalog Relationships:**

categories (1) ←→ (N) products  
products (1) ←→ (N) order\_details

* Each category contains multiple products
* Each product can appear in multiple order details
* Supports hierarchical product organization

**Order Management Relationships:**

orders (1) ←→ (N) order\_details  
customers (1) ←→ (N) orders  
tables (1) ←→ (N) orders  
areas (1) ←→ (N) tables

* Each order contains multiple order details (line items)
* Each customer can have multiple orders
* Each table can host multiple orders over time
* Each area contains multiple tables

**Promotion System Relationships:**

promotions (1) ←→ (N) order\_promotions  
orders (1) ←→ (N) order\_promotions

* Each promotion can be applied to multiple orders
* Each order can have multiple promotions applied
* Supports complex discount scenarios

**Detailed Table Specifications:**

**Users Table Structure:**

users (  
 user\_id INT PRIMARY KEY AUTO\_INCREMENT,  
 username VARCHAR(50) UNIQUE NOT NULL,  
 password VARCHAR(255) NOT NULL,  
 full\_name VARCHAR(100) NOT NULL,  
 email VARCHAR(100),  
 phone VARCHAR(20),  
 role ENUM('ADMIN', 'STAFF') NOT NULL,  
 is\_active BOOLEAN DEFAULT TRUE,  
 image\_url VARCHAR(255),  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  
)

**Products Table Structure:**

products (  
 product\_id INT PRIMARY KEY AUTO\_INCREMENT,  
 product\_name VARCHAR(100) NOT NULL,  
 category\_id INT NOT NULL,  
 price DECIMAL(10,2) NOT NULL,  
 cost\_price DECIMAL(10,2),  
 description TEXT,  
 image\_url VARCHAR(255),  
 is\_available BOOLEAN DEFAULT TRUE,  
 is\_active BOOLEAN DEFAULT TRUE,  
 stock\_quantity INT DEFAULT 0,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 FOREIGN KEY (category\_id) REFERENCES categories(category\_id)  
)

**Orders Table Structure:**

orders (  
 order\_id INT PRIMARY KEY AUTO\_INCREMENT,  
 order\_number VARCHAR(20) UNIQUE NOT NULL,  
 table\_id INT,  
 customer\_id INT,  
 user\_id INT NOT NULL,  
 order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 total\_amount DECIMAL(10,2) DEFAULT 0.00,  
 discount\_amount DECIMAL(10,2) DEFAULT 0.00,  
 final\_amount DECIMAL(10,2) DEFAULT 0.00,  
 payment\_method ENUM('CASH', 'CARD', 'MOMO', 'VNPAY'),  
 payment\_status ENUM('PENDING', 'PAID', 'REFUNDED') DEFAULT 'PENDING',  
 order\_status ENUM('PENDING', 'PREPARING', 'READY', 'SERVED', 'COMPLETED', 'CANCELLED') DEFAULT 'PENDING',  
 notes TEXT,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 FOREIGN KEY (table\_id) REFERENCES tables(table\_id),  
 FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),  
 FOREIGN KEY (user\_id) REFERENCES users(user\_id)  
)

### **6.3 Database Optimization Strategies**

**Index Strategy:**

**Primary Indexes:**

-- Performance-critical indexes  
CREATE INDEX idx\_user\_role ON users(role);  
CREATE INDEX idx\_user\_active ON users(is\_active);  
CREATE INDEX idx\_product\_category ON products(category\_id);  
CREATE INDEX idx\_product\_available ON products(is\_available);  
CREATE INDEX idx\_order\_date ON orders(order\_date);  
CREATE INDEX idx\_order\_status ON orders(order\_status);  
CREATE INDEX idx\_order\_user ON orders(user\_id);  
CREATE INDEX idx\_table\_status ON tables(status);  
CREATE INDEX idx\_customer\_phone ON customers(phone);

**Composite Indexes:**

-- Multi-column indexes for complex queries  
CREATE INDEX idx\_order\_date\_status ON orders(order\_date, order\_status);  
CREATE INDEX idx\_product\_category\_active ON products(category\_id, is\_active);  
CREATE INDEX idx\_order\_detail\_order\_product ON order\_details(order\_id, product\_id);

**Query Optimization Techniques:**

1. **Prepared Statements**: Prevention of SQL injection and query plan caching
2. **Connection Pooling**: Efficient database connection management
3. **Batch Processing**: Optimized bulk data operations
4. **Transaction Management**: Proper ACID compliance and performance

**Database Views for Complex Queries:**

**Sales Report View:**

CREATE VIEW view\_sales\_report AS  
SELECT   
 DATE(o.order\_date) as sale\_date,  
 COUNT(o.order\_id) as total\_orders,  
 SUM(o.final\_amount) as total\_revenue,  
 AVG(o.final\_amount) as average\_order\_value  
FROM orders o  
WHERE o.order\_status = 'COMPLETED'  
GROUP BY DATE(o.order\_date);

**Menu Items View:**

CREATE VIEW view\_menu\_items AS  
SELECT   
 p.product\_id,  
 p.product\_name,  
 c.category\_name,  
 p.price,  
 p.description,  
 p.image\_url,  
 p.is\_available  
FROM products p  
JOIN categories c ON p.category\_id = c.category\_id  
WHERE p.is\_active = TRUE AND c.is\_active = TRUE;

### **6.4 Connection Management and Performance**

**HikariCP Configuration:**

The system uses HikariCP, a high-performance JDBC connection pool, for optimal database connectivity:

**Connection Pool Settings:**

**HikariCP Configuration Implementation:**

The system utilizes HikariCP as the primary connection pooling solution, implementing enterprise-grade performance optimization and resource management strategies:

**Database Connection Configuration:**

* **Connection String Setup**: MySQL 8.0 JDBC driver configuration with database-specific connection parameters including timezone handling and character encoding
* **Authentication Management**: Secure credential configuration with environment-variable based password management for production security
* **Driver Optimization**: Latest MySQL Connector/J driver with enhanced performance features and SSL support

**Performance Optimization Parameters:**

* **Connection Pool Sizing**: Optimal pool configuration with 50 maximum connections and 5 minimum idle connections for balanced resource utilization
* **Timeout Management**: Sophisticated timeout configuration including 10-second connection timeout, 10-minute idle timeout, and 30-minute maximum lifetime for connection freshness
* **Leak Detection**: Advanced connection leak detection with 1-minute threshold for proactive resource monitoring and debugging

**Advanced Performance Features:**

* **Prepared Statement Caching**: Comprehensive prepared statement optimization with 250-statement cache size and 2KB SQL limit for enhanced query performance
* **Server-side Statement Preparation**: MySQL-specific optimizations utilizing server-side prepared statements for reduced parsing overhead and improved execution efficiency

**Transaction Management:**

**ACID Compliance Implementation:**

* **Atomicity**: All-or-nothing transaction execution
* **Consistency**: Database constraints maintained
* **Isolation**: Concurrent transaction separation
* **Durability**: Committed data persistence

**Transaction Patterns and Implementation:**

The system implements sophisticated transaction management patterns ensuring data integrity across complex business operations:

**Order Processing Transaction Workflow:**

* **Connection Management**: Automatic resource management using try-with-resources pattern for guaranteed connection cleanup and optimal resource utilization
* **Transaction Scope Control**: Manual transaction control with disabled autocommit for precise transaction boundary management and rollback capabilities
* **Multi-step Operation Coordination**: Coordinated execution of order creation, detail insertion, and inventory updates within single transaction scope
* **Error Handling and Recovery**: Comprehensive exception handling with automatic rollback mechanisms and detailed error reporting for operational transparency

**Business Logic Integration:**

* **DAO Pattern Implementation**: Coordinated use of multiple DAO objects within single transaction context for consistent data access patterns
* **Inventory Management**: Real-time stock quantity updates synchronized with order processing to maintain accurate inventory levels
* **Service Layer Exception Management**: Custom ServiceException handling providing meaningful error messages and proper exception propagation for debugging and monitoring

**Data Backup and Recovery Strategy:**

**Automated Backup Schedule:**

* **Daily Backups**: Complete database backup at 2 AM
* **Incremental Backups**: Every 6 hours during business operations
* **Backup Retention**: 30 days of backup history
* **Backup Verification**: Automated backup integrity checks

**Recovery Procedures:**

* **Point-in-time Recovery**: Restore to specific timestamp
* **Partial Recovery**: Restore specific tables or data
* **Disaster Recovery**: Complete system restoration
* **Data Migration**: Upgrade and migration procedures

This comprehensive database design ensures data integrity, optimal performance, and scalability while supporting all business requirements of the cafe management system.

## **7. Security Architecture**

### **7.1 Authentication System**

The Cafe Management System implements a robust authentication framework designed to ensure secure access while maintaining user convenience and system performance.

**Authentication Framework Components:**

**Password Security Implementation:**

* **BCrypt Hashing**: Industry-standard adaptive password hashing
* **Salt Generation**: Automatic salt creation for each password
* **Cost Factor Configuration**: Adjustable complexity for future security needs
* **Timing Attack Protection**: Constant-time password verification

**BCrypt Implementation Details:**

// Password Hashing Configuration  
private static final int BCRYPT\_COST\_FACTOR = 12;  
  
public class PasswordUtil {  
 public static String hashPassword(String plainPassword) {  
 return BCrypt.hashpw(plainPassword, BCrypt.gensalt(BCRYPT\_COST\_FACTOR));  
 }  
   
 public static boolean verifyPassword(String plainPassword, String hashedPassword) {  
 return BCrypt.checkpw(plainPassword, hashedPassword);  
 }  
}

**Authentication Process Flow:**

1. **User Credential Submission**
   * Username and password input validation
   * Protection against empty or malformed inputs
   * Rate limiting for repeated failed attempts
2. **Credential Verification**
   * Database user lookup by username
   * Password hash comparison using BCrypt
   * Account status verification (active/inactive)
3. **Session Initialization**
   * Secure session token generation
   * User context establishment
   * Role-based permission initialization
4. **Authentication Response**
   * Success: Dashboard navigation with user context
   * Failure: Error message with security logging
   * Account lockout: Temporary access suspension

**Session Management Implementation:**

**Session Security Features:**

* **Secure Session IDs**: Cryptographically strong session identifiers
* **Session Timeout**: Automatic logout after inactivity period
* **Session Invalidation**: Proper cleanup on logout
* **Concurrent Session Control**: Multiple login prevention

**SessionManager Class Features:**

The SessionManager implements sophisticated session management capabilities with comprehensive security features:

**Session State Management:**

* **User Context Tracking**: Advanced user session tracking with comprehensive user information storage and activity monitoring
* **Timestamp Management**: Sophisticated timestamp tracking for login time and last activity with automatic timeout calculation
* **Session Validation**: Intelligent session validity checking with null safety and timeout enforcement ensuring secure session management
* **Activity Updates**: Real-time activity tracking with automatic timestamp updates maintaining current session state

### **7.2 Authorization Framework**

**Role-Based Access Control (RBAC)**

The system implements a two-tier role structure designed for cafe operations:

**Role Definitions:**

1. **ADMIN Role Permissions**
   * **User Management**: Create, modify, deactivate user accounts
   * **System Configuration**: Modify application settings and parameters
   * **Menu Management**: Full CRUD operations on products and categories
   * **Financial Reports**: Access to all financial and business reports
   * **System Monitoring**: View system logs and performance metrics
   * **Data Management**: Backup, restore, and data export capabilities
2. **STAFF Role Permissions**
   * **Order Processing**: Create, modify, and complete customer orders
   * **Table Management**: Update table status and assignments
   * **Customer Service**: Access customer information and order history
   * **Basic Reporting**: View daily sales and performance summaries
   * **Payment Processing**: Handle all payment methods and transactions
   * **Menu Viewing**: Read-only access to product information

**Permission Matrix:**

| **Feature** | **Admin** | **Staff** | **Description** |
| --- | --- | --- | --- |
| User Management | ✓ | ✗ | Create/modify staff accounts |
| Menu Management | ✓ | ✓ | Product and category management |
| Order Processing | ✓ | ✓ | Complete order lifecycle |
| Payment Processing | ✓ | ✓ | All payment methods |
| Table Management | ✓ | ✓ | Table status and assignments |
| Customer Management | ✓ | ✓ | Customer profiles and history |
| Financial Reports | ✓ | Limited | Business intelligence access |
| System Settings | ✓ | ✗ | Application configuration |
| Data Export | ✓ | ✗ | Backup and export functions |
| Audit Logs | ✓ | ✗ | Security and system logs |

**Authorization Implementation:**

**Controller-Level Security:**

The system implements sophisticated controller-level security with comprehensive role-based access control:

**Role-Based Controller Protection:**

* **Annotation-Based Security**: Advanced role requirement annotations ensuring automatic access control enforcement
* **Method-Level Authorization**: Sophisticated method-level security checking with real-time role validation
* **Unauthorized Access Prevention**: Comprehensive unauthorized access prevention with user-friendly error messaging
* **Security Context Integration**: Advanced security context integration ensuring consistent authorization across all controller operations

**Method-Level Security:**

The system provides comprehensive method-level security with sophisticated permission and role management:

**Permission-Based Method Protection:**

* **Permission Annotation System**: Advanced permission requirement annotations ensuring granular access control
* **Security Context Validation**: Sophisticated security context checking with real-time permission validation
* **Role-Based Method Access**: Comprehensive role-based method access control with automatic authorization enforcement
* **Service Layer Security**: Advanced service layer security integration ensuring consistent authorization across business operations

### **7.3 Data Protection Strategies**

**SQL Injection Prevention:**

**Prepared Statement Implementation:**

The system implements comprehensive SQL injection prevention with sophisticated prepared statement patterns:

**SQL Injection Prevention Framework:**

* **Parameterized Query Management**: Advanced parameterized query implementation with automatic parameter binding and type safety
* **Resource Management**: Sophisticated resource management using try-with-resources patterns ensuring automatic cleanup
* **Query Optimization**: Intelligent query optimization with prepared statement caching and performance enhancement
* **Security Validation**: Comprehensive security validation with automatic parameter sanitization and injection prevention

**Input Validation Framework:**

**Validation Utilities:**

The system provides comprehensive input validation with sophisticated security and data integrity features:

**Input Validation Framework:**

* **Email Validation**: Advanced email format validation using comprehensive regex patterns ensuring proper email structure verification
* **Phone Number Validation**: Sophisticated phone number validation supporting Vietnamese phone formats with international prefix support
* **Password Strength Validation**: Comprehensive password strength checking with multiple criteria including length, character types, and complexity requirements
* **Input Sanitization**: Advanced input sanitization with automatic dangerous character removal and length limiting ensuring data safety

**Data Encryption:**

**Sensitive Data Protection:**

* **Password Storage**: BCrypt hashed with salt
* **Database Connection**: SSL encryption for production
* **Configuration Files**: Encrypted sensitive configuration data
* **Audit Logs**: Hashed user identifiers for privacy

### **7.4 Security Monitoring and Logging**

**Security Event Logging:**

**Audit Trail Implementation:**

The system implements comprehensive audit trail functionality with sophisticated logging and security monitoring:

**Security Logging Framework:**

* **Login Attempt Tracking**: Advanced login attempt logging with success/failure status, username tracking, and IP address recording for security analysis
* **Unauthorized Access Monitoring**: Sophisticated unauthorized access detection with detailed resource access logging and user identification
* **Data Modification Auditing**: Comprehensive data modification tracking with user attribution, operation details, and entity identification for compliance
* **Logging Infrastructure**: Advanced logging infrastructure using SLF4J framework with structured logging patterns and security event categorization

**Security Monitoring Features:**

1. **Failed Login Detection**
   * Track repeated failed login attempts
   * Automatic account lockout after threshold
   * Administrator notification of suspicious activity
2. **Session Monitoring**
   * Track active user sessions
   * Detect unusual session patterns
   * Automatic session termination on security events
3. **Data Access Monitoring**
   * Log all sensitive data access
   * Track data modification operations
   * Monitor export and backup operations
4. **System Health Monitoring**
   * Database connection monitoring
   * Application performance tracking
   * Resource usage monitoring

**Security Configuration Management:**

**Security Settings:**

The system implements comprehensive security configuration management with sophisticated parameter control:

**Security Configuration Framework:**

* **Password Policy Management**: Advanced password policy configuration with length requirements, character type mandates, and complexity enforcement
* **Session Management**: Sophisticated session timeout configuration with automatic logout and security enforcement
* **Login Security**: Comprehensive login security settings with attempt limiting, lockout mechanisms, and account protection
* **Encryption Configuration**: Advanced encryption configuration with BCrypt cost factor management for optimal security-performance balance

**Environment-Specific Security:**

* **Development**: Relaxed security for testing
* **Production**: Maximum security configuration
* **Testing**: Isolated security context

This comprehensive security architecture ensures that the Cafe Management System maintains the highest standards of data protection, user privacy, and system security while providing a smooth user experience for legitimate users.

## **8. UI/UX Design Architecture**

### **8.1 JavaFX Component Structure**

The Cafe Management System leverages JavaFX’s powerful UI framework to create an intuitive, responsive, and visually appealing user interface that enhances productivity and user satisfaction.

**JavaFX Architecture Benefits:**

* **Scene Graph**: Hierarchical structure for efficient rendering
* **CSS Styling**: Professional appearance with easy customization
* **FXML Integration**: Separation of design and logic
* **Rich Controls**: Advanced UI components for complex business needs
* **Animation Support**: Smooth transitions and visual feedback

**Core UI Component Hierarchy:**

**Main Application Structure:**

Stage (Primary Window)  
├── Scene (Application Container)  
│ ├── BorderPane (Main Layout)  
│ │ ├── Top: MenuBar/Toolbar  
│ │ ├── Left: Navigation Sidebar  
│ │ ├── Center: Content Area (StackPane)  
│ │ ├── Right: Context Panel (Optional)  
│ │ └── Bottom: Status Bar  
│ └── CSS Stylesheets  
└── Icons and Resources

**FXML File Organization and Structure:**

**Login Interface (login.fxml):**

The login interface implements sophisticated FXML-based layout with comprehensive user interaction design:

**Login Interface Architecture:**

* **Container Structure**: Advanced VBox container layout with comprehensive styling and controller binding for seamless user experience
* **Branding Integration**: Sophisticated logo section with image display and application title presentation for professional appearance
* **Form Design**: Comprehensive login form with username/password fields, remember me functionality, and action button integration
* **Error Handling**: Advanced error display system with visibility management and user feedback for authentication issues

**Dashboard Interface (dashboard.fxml):**

The dashboard interface implements sophisticated BorderPane layout with comprehensive navigation and content management:

**Dashboard Layout Architecture:**

* **BorderPane Structure**: Advanced BorderPane layout with comprehensive positioning for navigation, content, and user information
* **Top Navigation Bar**: Sophisticated top navigation with user information display, role indication, and logout functionality
* **Side Navigation**: Advanced left sidebar navigation with menu, table, and order management buttons for efficient workflow
* **Content Management**: Intelligent center content area with StackPane for dynamic content loading and order panel integration

**Advanced UI Components:**

**Custom Product Card Component:**

The ProductCard component implements sophisticated custom UI design with comprehensive product display and interaction capabilities:

**Product Card Architecture:**

* **Component Structure**: Advanced VBox-based component design with comprehensive product information display and user interaction
* **Visual Elements**: Sophisticated visual elements including product images, names, pricing, and interactive controls with proper sizing and styling
* **User Interaction**: Comprehensive user interaction with add-to-order functionality, quantity selection, and product detail access through double-click
* **Event Management**: Advanced event handling with custom event firing for order integration and user experience optimization

### **8.2 CSS Styling and Theme Management**

**Modular CSS Architecture:**

The application uses a sophisticated CSS architecture that promotes maintainability and consistent theming:

**CSS File Structure:**

src/main/resources/css/  
├── common/  
│ ├── base.css # Base styles and resets  
│ ├── typography.css # Font definitions and text styles  
│ ├── colors.css # Color palette and variables  
│ └── animations.css # Transition and animation definitions  
├── components/  
│ ├── buttons.css # Button styles and states  
│ ├── forms.css # Form controls and validation  
│ ├── tables.css # TableView and data display  
│ └── cards.css # Card components and layouts  
├── layouts/  
│ ├── login.css # Login screen specific styles  
│ ├── dashboard.css # Main dashboard layout  
│ ├── admin-dashboard.css # Administrative interface  
│ └── order-panel.css # Order processing interface  
└── themes/  
 ├── default-theme.css # Default color scheme  
 ├── dark-theme.css # Dark mode support  
 └── high-contrast.css # Accessibility theme

**CSS Variable System:**

The system implements sophisticated CSS variable management with comprehensive design system integration:

**Design System Architecture:**

* **Color Palette Management**: Advanced color palette system with primary, secondary, neutral, and status colors supporting consistent branding
* **Spacing System**: Sophisticated spacing system with standardized spacing values ensuring consistent layout and visual hierarchy
* **Typography Framework**: Comprehensive typography system with font families, sizes, and weights supporting professional text presentation
* **Visual Effects**: Advanced shadow and border radius systems providing depth and visual polish across all interface components

**Component-Specific Styling:**

**Button System:**

The system implements sophisticated button styling with comprehensive interaction states and visual feedback:

**Button Design Framework:**

* **Base Button Styling**: Advanced base button styling with consistent typography, spacing, and visual effects across all button types
* **Primary Button Design**: Sophisticated primary button design with brand color integration, hover effects, and pressed state feedback
* **Secondary Button Styling**: Advanced secondary button styling with transparent backgrounds and border-based design for visual hierarchy
* **Danger Button Integration**: Comprehensive danger button styling with error color integration for critical actions and user warning

**Table Styling:**

The system implements sophisticated table styling with comprehensive data presentation and user interaction:

**Table Design Framework:**

* **Table Container Styling**: Advanced table container styling with surface colors, borders, and rounded corners for professional appearance
* **Column Header Design**: Sophisticated column header styling with brand color integration, typography emphasis, and proper spacing
* **Row Interaction States**: Comprehensive row styling with selection highlighting, hover effects, and border management for clear data separation
* **Visual Hierarchy**: Advanced visual hierarchy management with color differentiation and spacing ensuring optimal data readability

### **8.3 Responsive Design Implementation**

**Adaptive Layout System:**

**Screen Size Breakpoints:**

* **Small Screens**: 1024x768 (minimum supported)
* **Medium Screens**: 1366x768 (standard)
* **Large Screens**: 1920x1080 (optimal)
* **Extra Large**: 2560x1440+ (full featured)

**Responsive Layout Strategies:**

**Dynamic Content Sizing:**

The system implements sophisticated responsive layout management with comprehensive screen size adaptation:

**Responsive Layout Framework:**

* **Container Size Binding**: Advanced container size binding with automatic window size synchronization ensuring optimal layout utilization
* **Breakpoint Management**: Sophisticated breakpoint system with intelligent layout switching based on screen width for optimal user experience
* **Layout State Management**: Comprehensive layout state management with automatic style class switching for different screen sizes
* **Adaptive Design**: Advanced adaptive design implementation supporting compact, standard, and expanded layouts for diverse screen environments

**CSS Responsive Rules:**

The system implements sophisticated CSS responsive rules with comprehensive layout adaptation:

**Responsive CSS Framework:**

* **Layout-Specific Scaling**: Advanced scaling system with spacing and font size adjustments for different layout modes
* **Component Adaptation**: Sophisticated component adaptation with size adjustments for product cards and interactive elements
* **Proportional Scaling**: Comprehensive proportional scaling system ensuring consistent visual hierarchy across all screen sizes
* **Layout Mode Support**: Advanced layout mode support with compact, standard, and expanded configurations for optimal user experience

### **8.4 User Experience Flow Design**

**Navigation Flow Architecture:**

**Primary User Journeys:**

1. **Staff Order Processing Flow:**

* Login → Dashboard → Table Selection → Menu Browse →   
  Product Selection → Order Review → Payment Processing →   
  Receipt Generation → Order Completion

1. **Admin Management Flow:**

* Login → Admin Dashboard → User Management →   
  Menu Configuration → Sales Reports → System Settings

**UI State Management:**

**Context-Aware Interface Updates:**

The system implements sophisticated context-aware interface management with comprehensive state synchronization:

**UI State Management Framework:**

* **Order Panel State Management**: Advanced order panel state management with intelligent state detection and appropriate UI updates
* **Table Status Synchronization**: Sophisticated table status synchronization with real-time visual updates and status-based styling
* **Platform-Safe Updates**: Comprehensive platform-safe UI updates using Platform.runLater ensuring thread-safe interface modifications
* **Visual State Management**: Advanced visual state management with automatic style class updates and status-based visual feedback

**Animation and Transition System:**

**Smooth UI Transitions:**

The system implements sophisticated animation and transition systems with comprehensive visual feedback:

**Animation Framework:**

* **Fade Transitions**: Advanced fade transition system with smooth opacity changes and configurable timing for elegant content reveals
* **Slide Transitions**: Sophisticated slide transition system with horizontal movement and cubic-bezier easing for natural motion
* **Scale Transitions**: Comprehensive scale transition system with smooth size changes and optimized timing for interactive feedback
* **Transition Timing**: Advanced transition timing management with appropriate duration and easing functions for optimal user experience

**Accessibility Considerations:**

**Accessibility Features:**

* **Keyboard Navigation**: Full keyboard accessibility for all functions
* **Screen Reader Support**: Proper ARIA labels and descriptions
* **High Contrast Mode**: Alternative color schemes for visual impairments
* **Font Size Scaling**: Adjustable text size for readability
* **Focus Indicators**: Clear visual focus indicators for navigation

This comprehensive UI/UX design architecture ensures that the Cafe Management System provides an intuitive, efficient, and accessible user experience while maintaining professional appearance and performance across different screen sizes and user capabilities.

# **PART III: CORE MODULES & COMPONENTS**

## **9. Authentication & User Management**

### **9.1 Login System Architecture**

The authentication system serves as the primary gateway to the Cafe Management System, implementing secure user verification while maintaining simplicity and user-friendliness.

**LoginController Implementation**

The LoginController manages the complete authentication workflow, from user input validation to successful login redirection:

**Core Responsibilities:**

* User credential collection and validation
* Authentication service integration
* Error handling and user feedback
* Session initialization and management
* Role-based dashboard routing

**Authentication Flow Process:**

1. **Input Validation Phase**
   * Username format verification
   * Password strength checking
   * Empty field detection
   * Special character sanitization
2. **Credential Verification Phase**
   * Database user lookup
   * Password hash comparison
   * Account status verification
   * Login attempt logging
3. **Session Establishment Phase**
   * User context creation
   * Role permission initialization
   * Session timeout configuration
   * Activity tracking setup
4. **Navigation Phase**
   * Role-based dashboard selection
   * UI state initialization
   * Welcome message display
   * Previous session cleanup

**LoginController Key Methods:**

The LoginController implements sophisticated authentication workflow management with comprehensive user interaction and security features:

**Authentication Workflow Implementation:**

* **FXML Component Integration**: Advanced FXML component binding with comprehensive field management including username, password, remember me functionality, and error display
* **Input Validation Framework**: Sophisticated input validation with real-time error checking, field focus management, and user feedback systems
* **Authentication Service Integration**: Advanced authentication service coordination with comprehensive error handling and security logging
* **Session Management**: Intelligent session initialization with user context creation, credential management, and role-based navigation routing

**Authentication Service Implementation:**

The AuthenticationService provides sophisticated user authentication with comprehensive security and audit capabilities:

**Authentication Service Architecture:**

* **Database Integration**: Advanced database connection management with automatic resource cleanup and connection pooling optimization
* **User Lookup and Validation**: Sophisticated user lookup with comprehensive account status verification and security logging
* **Password Verification**: Advanced password verification using BCrypt hashing with comprehensive security logging for audit trails
* **Session Management**: Intelligent session management with automatic login timestamp updates and error handling for operational continuity

### **9.2 User Entity Model and Management**

**User Entity Structure:**

The User entity represents all system users (both Admin and Staff) with comprehensive profile information and security attributes:

**User Entity Attributes:**

The User entity implements comprehensive user representation with sophisticated security and profile management:

**User Entity Architecture:**

* **Identity Management**: Advanced identity management with unique user identifiers, display names, and comprehensive profile information
* **Security Integration**: Sophisticated security integration with BCrypt password hashing and account status management
* **Contact Information**: Comprehensive contact information management with optional email and phone number support
* **Audit Trail**: Advanced audit trail with creation and modification timestamps for compliance and operational tracking

**User Role System:**

The UserRole enumeration implements sophisticated role-based access control with comprehensive permission management:

**Role-Based Access Control Framework:**

* **Administrative Role**: Advanced administrative role with comprehensive system management permissions including user management, system configuration, financial reporting, and audit access
* **Staff Role**: Sophisticated staff role with operational permissions including order processing, table management, customer service, and payment processing capabilities
* **Permission Management**: Advanced permission management with granular permission checking and role-based access control enforcement
* **Display Integration**: Comprehensive display integration with human-readable role names and permission-based UI customization

**User Management Business Logic:**

**UserService Implementation:**

The UserService provides comprehensive user management with sophisticated business logic and security integration:

**User Management Service Architecture:**

* **User Creation Workflow**: Advanced user creation with comprehensive validation, password hashing, timestamp management, and security logging
* **User Update Operations**: Sophisticated user update capabilities with validation, timestamp updates, and comprehensive audit trail maintenance
* **User Deactivation**: Intelligent user deactivation with business rule enforcement preventing last admin deactivation and comprehensive security logging
* **Data Validation Framework**: Advanced data validation with comprehensive field checking, format validation, and business rule enforcement

### **9.3 SessionManager and User Context**

**Session Management Implementation:**

The SessionManager provides centralized user session management across the application:

**SessionManager Features:**

The SessionManager implements sophisticated session management with comprehensive security and user context tracking:

**Session Management Architecture:**

* **User Context Management**: Advanced user context management with comprehensive user information storage and activity tracking
* **Session Validation**: Sophisticated session validation with timeout enforcement and automatic session cleanup for security
* **Activity Tracking**: Intelligent activity tracking with automatic timestamp updates and session timeout management
* **Role and Permission Management**: Advanced role and permission management with real-time access control and security enforcement

### **9.4 User Interface Components**

**Login Screen Design:**

The login interface provides a clean, professional authentication experience:

**UI Components:**

* **Logo Display**: Brand identity and visual appeal
* **Username Field**: Text input with validation
* **Password Field**: Secure password entry
* **Remember Me**: Credential persistence option
* **Login Button**: Authentication trigger
* **Error Display**: User feedback area

**Login Screen CSS Styling:**

The login interface implements sophisticated CSS styling with comprehensive visual design and user experience optimization:

**Login Interface Design Framework:**

* **Container Styling**: Advanced container styling with gradient backgrounds, proper spacing, and center alignment for professional appearance
* **Logo and Branding**: Sophisticated logo section styling with proper sizing, spacing, and typography for brand identity
* **Form Design**: Comprehensive form styling with consistent field design, focus states, and visual feedback for optimal user interaction
* **Interactive Elements**: Advanced button styling with hover effects, pressed states, and visual feedback for enhanced user experience
* **Error Handling**: Sophisticated error message styling with appropriate colors, borders, and visual indicators for clear user feedback

**User Management Interface (Admin):**

The admin user management interface provides comprehensive user administration capabilities:

**User Management Features:**

* **User List Display**: TableView with user information
* **Add User Dialog**: Modal form for creating new users
* **Edit User Dialog**: Modal form for modifying user details
* **User Status Management**: Activate/deactivate user accounts
* **Role Assignment**: Change user roles and permissions

**UserFormDialog Implementation:**

The UserFormDialog implements sophisticated modal dialog functionality with comprehensive form management and validation:

**Dialog Management Architecture:**

* **Modal Dialog Design**: Advanced modal dialog design with comprehensive title management, button configuration, and user experience optimization
* **Form Content Creation**: Sophisticated form content creation with grid layout, field management, and comprehensive component integration
* **Field Population**: Intelligent field population with edit mode detection and comprehensive data binding for seamless user experience
* **Real-time Validation**: Advanced real-time validation with comprehensive field checking, error messaging, and dynamic button state management
* **Result Processing**: Sophisticated result processing with comprehensive data extraction, validation, and user object creation

This comprehensive authentication and user management system provides secure access control, efficient user administration, and a professional user experience while maintaining the highest security standards throughout the application.

## **10. Menu & Product Management**

### **10.1 Product Entity Model and Business Logic**

The Menu and Product Management system serves as the foundation for the cafe’s digital catalog, enabling dynamic product management, categorization, and pricing control.

**Product Entity Architecture:**

The Product entity represents individual menu items with comprehensive attributes supporting business operations:

**Product Entity Structure:**

The Product entity implements comprehensive product representation with sophisticated business logic and inventory management:

**Product Entity Architecture:**

* **Identity Management**: Advanced product identity management with unique identifiers, display names, and comprehensive product information
* **Financial Integration**: Sophisticated financial integration with pricing, cost management, and automatic profit margin calculation
* **Inventory Management**: Advanced inventory management with stock quantity tracking, availability status, and ordering capability assessment
* **Business Logic Integration**: Comprehensive business logic integration with automatic profit calculations, stock status checking, and order eligibility determination

**Category Entity Model:**

The Category entity implements sophisticated category management with comprehensive organization and display capabilities:

**Category Entity Architecture:**

* **Category Identity**: Advanced category identity management with unique identifiers, display names, and comprehensive description support
* **Display Management**: Sophisticated display management with configurable display order and image integration for visual presentation
* **Status Management**: Advanced status management with active/inactive state tracking and comprehensive lifecycle management
* **Product Integration**: Comprehensive product integration with automatic product counting and active product filtering capabilities

**Product-Category Relationship:**

* **One-to-Many**: Each category can contain multiple products
* **Foreign Key**: Products reference categories via categoryId
* **Cascade Operations**: Category deactivation affects product visibility
* **Hierarchical Display**: Categories organize menu presentation

### **10.2 MenuService Business Logic Implementation**

**MenuService Architecture:**

The MenuService encapsulates all business logic related to product and category management, providing a clean interface between controllers and data access layers.

**Core MenuService Responsibilities:**

* Product CRUD operations with business rule validation
* Category management and hierarchical organization
* Price management and profit margin calculations
* Inventory tracking and availability management
* Image handling and asset management

**MenuService Implementation:**

The MenuService provides comprehensive menu management with sophisticated business logic and data access coordination:

**Service Layer Architecture:**

* **DAO Integration Strategy**: Advanced DAO integration with automatic connection management using try-with-resources patterns ensuring optimal database performance and resource cleanup
* **Service Layer Abstraction**: Sophisticated service layer abstraction implementing facade pattern providing simplified interface for complex menu management operations while maintaining separation of concerns
* **Exception Handling Framework**: Comprehensive exception management with custom ServiceException wrapping for meaningful error propagation and debugging support

**Product Management Operations:**

**Active Product Retrieval and Filtering:**

* **Comprehensive Product Listing**: Advanced filtering mechanisms combining active status verification with availability checking for consistent product presentation
* **Category-Based Organization**: Sophisticated category-specific product retrieval with automatic filtering for active products and streamlined data organization for efficient UI population
* **Availability Assessment**: Real-time availability calculation utilizing business logic integration for accurate product status determination and ordering optimization

**Product Creation and Management:**

* **Multi-Layer Validation**: Comprehensive product data validation including name verification, price validation, and category existence checking ensuring operational integrity
* **SKU Generation**: Intelligent SKU generation using category prefixes and timestamp components with fallback mechanisms for unique product identification
* **Timestamp Management**: Sophisticated timestamp management with automatic creation and update timestamp assignment for audit trail maintenance
* **Security Integration**: Advanced security integration with comprehensive operation logging through SecurityLogger for user attribution and compliance tracking

**Product Update Operations:**

* **Existence Validation**: Sophisticated product existence validation with comprehensive error handling and user feedback for operational transparency
* **Availability Management**: Advanced availability management with real-time status updates and comprehensive audit trail maintenance
* **Stock Management**: Intelligent stock management with quantity tracking, change logging, and automatic timestamp updates for inventory accuracy

**Category Management Operations:**

**Category Retrieval and Organization:**

* **Active Category Filtering**: Advanced active category filtering with automatic display order sorting for consistent menu presentation
* **Hierarchical Organization**: Sophisticated hierarchical organization supporting complex category relationships and display optimization

**Category Creation and Validation:**

* **Duplicate Prevention**: Advanced duplicate prevention with comprehensive category name validation ensuring unique category identification
* **Display Order Management**: Intelligent display order management with automatic next order calculation and fallback mechanisms
* **Timestamp Integration**: Sophisticated timestamp integration with automatic creation and update timestamp assignment for lifecycle tracking

**Business Logic and Validation Framework:**

**Product Validation System:**

* **Comprehensive Field Validation**: Advanced field validation including name requirements, length constraints, and price validation ensuring data integrity
* **Business Rule Enforcement**: Sophisticated business rule enforcement with negative value prevention and category relationship validation
* **Error Messaging**: Comprehensive error messaging with user-friendly descriptions and specific validation feedback

**Category Validation Framework:**

* **Name Validation**: Advanced name validation with length constraints and format requirements ensuring consistent category presentation
* **Business Rule Compliance**: Sophisticated business rule compliance with comprehensive validation ensuring operational integrity

**Utility Functions and Inventory Management:**

**SKU Generation System:**

* **Intelligent SKU Creation**: Advanced SKU creation using category prefixes and timestamp components with fallback mechanisms for collision prevention
* **Category Integration**: Sophisticated category integration with automatic category lookup and prefix generation for consistent product identification

**Display Order Management:**

* **Automatic Order Calculation**: Intelligent automatic order calculation with database integration and fallback mechanisms for consistent category presentation
* **Error Handling**: Comprehensive error handling with default value assignment ensuring operational continuity

**Inventory Management System:**

**Low Stock Monitoring:**

* **Threshold-Based Filtering**: Advanced threshold-based filtering with comprehensive stock level monitoring and alert generation
* **Active Product Focus**: Sophisticated active product focus ensuring only relevant products are included in monitoring systems

**Inventory Analytics:**

* **Comprehensive Summary Generation**: Advanced summary generation including total products, active counts, availability statistics, and inventory valuation
* **Financial Integration**: Sophisticated financial integration with automatic inventory value calculation using cost prices and stock quantities
* **Performance Metrics**: Comprehensive performance metrics including out-of-stock tracking and availability percentages for business intelligence

### **10.3 AdminMenuController Implementation**

**Controller Architecture:**

The AdminMenuController manages the administrative interface for product and category management, providing comprehensive CRUD operations and business intelligence.

**Core Controller Responsibilities:**

* Product management interface coordination
* Category management and organization
* Image upload and asset management
* Bulk operations for efficiency
* Real-time inventory monitoring

**AdminMenuController Key Features:**

The AdminMenuController serves as the central hub for menu management operations, providing a comprehensive interface for administrators to manage products, categories, and inventory efficiently. This controller implements sophisticated JavaFX patterns and follows enterprise-level design principles to ensure scalability and maintainability.

**User Interface Components and Architecture:**

The controller manages a sophisticated tabbed interface that separates product management from category management concerns. The product management section features a comprehensive data table displaying essential product information including names, categories, pricing, stock levels, and availability status. The interface incorporates the following key components:

* **Interactive Search and Filtering Controls**: Advanced text-based search functionality with real-time filtering capabilities, category-specific dropdown filters, and toggle controls for viewing inactive products
* **Product Management Action Buttons**: Comprehensive CRUD operation controls including add, edit, delete, and bulk update functionalities with confirmation dialogs and validation feedback
* **Category Management Interface**: Dedicated category administration section with hierarchical organization, display order management, and product count tracking
* **Image Management System**: Integrated image preview functionality with upload capabilities, format validation, and automatic optimization for consistent display
* **Real-time Statistics Dashboard**: Live inventory metrics including total product counts, active inventory levels, low stock alerts, and total inventory valuation

**Service Integration and Data Management:**

The controller implements a comprehensive data management strategy utilizing JavaFX’s reactive programming patterns. The MenuService integration provides seamless access to business logic operations while maintaining clean separation of concerns. Key data management features include:

* **Observable Collections**: Real-time data synchronization using ObservableList and FilteredList patterns for immediate UI updates
* **Service Layer Integration**: Centralized business logic access through the MenuService facade pattern
* **Initialization Workflow**: Systematic component setup including table configurations, event handler registration, search functionality setup, initial data loading, and statistics calculation
* **Error Handling**: Comprehensive exception management with user-friendly error messaging and automatic recovery mechanisms

**Table Configuration and Cell Factories:**

The product table configuration utilizes advanced JavaFX cell factory patterns to provide sophisticated data presentation and interaction capabilities. The implementation includes:

* **Product Name Display**: Standard property binding with text-based representation for efficient rendering and sorting capabilities
* **Category Resolution**: Dynamic category name lookup with fallback error handling to display readable category names instead of internal ID references
* **Currency Formatting**: Specialized price formatting cell factory that automatically applies locale-specific currency formatting with consistent decimal precision
* **Visual Stock Indicators**: Intelligent color-coded stock level display using conditional styling - red for out-of-stock items, orange for low inventory (≤10 units), and green for adequate stock levels
* **Interactive Availability Controls**: Embedded checkbox controls with real-time availability toggling, including automatic service layer updates and rollback functionality on failure
* **Action Button Integration**: Comprehensive action column with styled button controls for edit, duplicate, and delete operations, each with proper event handling and user confirmation workflows

**Category Management Configuration:**

The category management interface implements sophisticated table configuration with dynamic data calculation and real-time updates:

* **Category Information Display**: Standard property binding for category names with automatic sorting and filtering capabilities
* **Dynamic Product Counting**: Real-time calculation of product counts per category using service layer integration with error handling fallbacks
* **Display Order Management**: Configurable ordering system for category presentation with drag-and-drop reordering support
* **Status Tracking**: Active/inactive status indicators with visual styling for immediate status recognition

**Event Handling and User Interaction:**

The controller implements comprehensive event handling patterns to provide responsive user interaction across all interface components:

* **Product Management Operations**: Complete CRUD operation handling including creation, editing, deletion, and bulk update workflows with proper validation and confirmation dialogs
* **Category Administration**: Full category lifecycle management with creation, modification, and deletion capabilities including dependency validation
* **Asset Management**: Integrated image upload functionality with format validation, compression, and preview capabilities
* **Selection Management**: Advanced selection tracking with automatic UI state updates and context-sensitive action availability
* **Accessibility Features**: Double-click editing support, keyboard navigation, and touch-friendly interaction patterns for enhanced usability

**Advanced Search and Filtering System:**

The filtering system implements sophisticated real-time search capabilities using JavaFX’s FilteredList pattern for optimal performance:

* **Real-time Text Search**: Instantaneous filtering based on product name matching with case-insensitive string comparison and partial matching support
* **Category-based Filtering**: Dynamic category selection filtering with dropdown interface for focused product viewing
* **Status-based Filtering**: Toggle controls for viewing active/inactive products with immediate filter application
* **Composite Filtering Logic**: Multiple filter criteria combination with AND logic for precise product selection
* **Performance Optimization**: Efficient predicate-based filtering that minimizes UI update overhead and maintains responsive user experience

**Comprehensive Event Handler Implementation:**

The controller implements sophisticated event handling patterns that provide robust product management capabilities with comprehensive error handling and user feedback:

**Product Creation Workflow:**

* Utilizes modal dialog interface (ProductFormDialog) for comprehensive product data input with real-time validation
* Integrates with MenuService for business logic execution and data persistence
* Implements automatic UI updates with Observable collections for immediate reflection of changes
* Provides success confirmation with user-friendly messaging and automatic statistics refresh

**Product Modification Operations:**

* Supports in-place editing through dedicated form dialogs pre-populated with existing product data
* Implements optimistic UI updates with rollback capability on service layer failures
* Maintains referential integrity through comprehensive validation before persistence
* Includes automatic index-based collection updates for consistent data synchronization

**Product Deletion Management:**

* Features confirmation dialog systems with detailed deletion impact information
* Implements cascade validation to prevent orphaned data relationships
* Provides comprehensive error handling with user-friendly error messaging
* Includes automatic cleanup of related UI components and statistics recalculation

**Data Loading and Synchronization:**

The data management system implements efficient loading strategies with comprehensive error handling:

* **Service Integration**: Utilizes MenuService facade for centralized data access with connection pooling optimization
* **Observable Collection Setup**: Configures reactive data binding using ObservableList patterns for automatic UI synchronization
* **Filter Integration**: Establishes FilteredList relationships for real-time search and filtering capabilities
* **UI Component Population**: Systematically populates all interface components including tables, dropdowns, and filter controls
* **Error Recovery**: Implements graceful error handling with user-friendly error messaging and component state preservation

**Real-time Statistics and Business Intelligence:**

The statistics system provides comprehensive business insights with automatic calculation and display:

* **Inventory Metrics**: Real-time calculation of total products, active inventory levels, and stock status indicators
* **Financial Analysis**: Automatic inventory valuation with currency formatting and trending analysis
* **Performance Monitoring**: Low stock alerts and availability tracking for proactive inventory management
* **Data Visualization**: Dynamic label updates with color-coded indicators for immediate visual feedback
* **Background Processing**: Asynchronous statistics calculation to maintain responsive user interface performance

### **10.4 Customer Menu Interface**

**Customer-Facing Menu Design:**

The customer menu interface provides an intuitive, visually appealing way for staff to browse and select products during order creation.

**MenuController Architecture:**

The MenuController provides a sophisticated customer-facing interface designed for efficient product browsing and order management. This controller implements advanced UI patterns optimized for touch-screen environments and high-volume transaction processing.

**Interface Design and Layout:**

The controller utilizes a tabbed interface architecture that organizes products by category for intuitive navigation. Each category tab contains a dynamically generated product grid that displays items in an aesthetically pleasing card-based layout. The interface components include:

* **Category Navigation System**: Dynamic tab generation based on active categories with automatic tab ordering and responsive design adaptation
* **Product Display Grid**: Responsive grid layout with configurable column counts (4 products per row by default) that automatically adjusts based on screen resolution and device orientation
* **Search Integration**: Real-time search functionality with autocomplete capabilities and category-specific filtering options
* **Scrollable Content Areas**: Optimized scroll panes with smooth scrolling behavior and touch-friendly interaction patterns

**Data Management and Caching:**

The controller implements sophisticated data management strategies optimized for performance and user experience:

* **Category-based Data Organization**: Products are organized in memory using HashMap structures indexed by category ID for efficient retrieval and display
* **Active Content Filtering**: Only active and orderable products are displayed to customers, with automatic filtering of discontinued or unavailable items
* **Service Layer Integration**: Centralized data access through MenuService with connection pooling and caching optimization
* **Lazy Loading**: Dynamic content loading as users navigate between categories to minimize initial load times

**Product Card Generation and Interaction:**

The system generates interactive product cards with comprehensive information display and order integration:

* **Visual Product Representation**: Each product card includes high-quality product images, pricing information, descriptions, and availability status
* **Interactive Order Controls**: Embedded quantity selectors and “Add to Order” functionality with immediate feedback and validation
* **Real-time Availability Updates**: Dynamic availability checking with automatic card state updates based on current inventory levels
* **Responsive Design**: Product cards automatically adapt to different screen sizes while maintaining consistent spacing and visual hierarchy

**Order Integration and Workflow:**

The controller seamlessly integrates with the order management system through callback-based communication patterns: - **Callback-based Communication**: Direct integration with OrderPanelController through method callbacks for immediate order updates - **Quantity Validation**: Automatic quantity validation and adjustment based on product availability and ordering constraints - **Real-time Order Updates**: Immediate order panel updates with product additions, including pricing calculations and availability verification - **Controller Dependency Injection**: Flexible controller coupling through setter injection pattern for maintainable architecture

This sophisticated customer menu interface ensures optimal user experience while maintaining robust backend integration for reliable order processing and inventory management.

This comprehensive Menu and Product Management system provides powerful tools for managing the cafe’s digital catalog while ensuring an intuitive user experience for both administrators and staff members.

## **11. Table & Area Management**

### **11.1 Table Entity Model and Business Logic**

The Table and Area Management system provides comprehensive functionality for organizing and managing the physical layout of the cafe, enabling efficient table tracking, reservation management, and area organization.

**TableCafe Entity Structure:**

The TableCafe entity represents individual tables with comprehensive attributes designed to support complex operational workflows and spatial management requirements:

**Core Entity Attributes:**

* **Identification System**: Unique table identifiers with customizable display names supporting various naming conventions (numeric, alphanumeric, or custom schemes like “VIP-A”, “Patio-3”)
* **Capacity Management**: Configurable seating capacity with support for different party sizes and accommodation logic
* **Spatial Positioning**: Coordinate-based location tracking (X/Y coordinates) enabling accurate floor plan representation and layout optimization
* **Status Tracking**: Real-time status management with timestamp tracking for creation and modification history
* **Operational Metadata**: Descriptive information and activation status for comprehensive table management

**Business Logic Integration:**

* **Availability Calculation**: Sophisticated availability determination combining active status and current table state for accurate booking decisions
* **Party Accommodation Logic**: Intelligent party size matching with capacity constraints and special seating requirements
* **Display Name Generation**: Flexible naming conventions with fallback logic for consistent presentation across all interface components
* **Maintenance Status Tracking**: Cleaning and maintenance requirement identification for operational workflow optimization

**Area Entity Model:**

The Area entity provides comprehensive zone management capabilities for organizing tables into logical groupings with sophisticated operational analytics:

**Structural Components:**

* **Area Identification**: Unique area identifiers with descriptive naming conventions supporting diverse establishment layouts (Main Floor, VIP Section, Outdoor Patio, Private Dining)
* **Capacity Planning**: Maximum capacity constraints for area-level booking management and occupancy optimization
* **Layout Configuration**: JSON-based layout storage supporting complex spatial arrangements and custom configuration parameters
* **Administrative Metadata**: Creation and modification timestamps with activation status for comprehensive area lifecycle management

**Operational Intelligence Functions:**

* **Real-time Availability Tracking**: Dynamic calculation of available tables within each area with intelligent filtering based on current status and reservation constraints
* **Occupancy Analytics**: Continuous monitoring of current occupancy levels with percentage-based occupancy rate calculations for performance optimization
* **Capacity Utilization Metrics**: Comprehensive analysis of area utilization patterns supporting data-driven operational decisions and staff allocation strategies

**TableStatus Enumeration:**

The TableStatus enumeration provides comprehensive status tracking with visual representation and operational workflow support:

**Status Categories and Visual Design:**

* **AVAILABLE (Green #4caf50)**: Tables ready for immediate seating with complete setup and cleaning verification
* **OCCUPIED (Red #f44336)**: Currently in use by customers with active orders or ongoing service
* **RESERVED (Orange #ff9800)**: Pre-booked tables with confirmed reservations and specific timing requirements
* **CLEANING (Purple #9c27b0)**: Tables requiring maintenance, cleaning, or setup preparation before next service

**Integration Features:**

* **Visual Consistency**: Color-coded status representation for immediate recognition across all interface components
* **Display Name Mapping**: Human-readable status descriptions for customer communication and staff coordination
* **Workflow Integration**: Status-based business logic support for automated table management and service optimization

**Accessor Methods and Business Logic:**

* **Display Name Retrieval**: Standardized method for obtaining human-readable status descriptions for UI presentation
* **Color Code Access**: Centralized color code management for consistent visual theming across application components
* **Order Availability Logic**: Business rule implementation determining which table statuses permit new order creation and customer seating

### **11.2 TableService Business Logic Implementation**

**TableService Architecture:**

The TableService encapsulates all business logic related to table and area management, providing comprehensive functionality for table operations.

**Core TableService Responsibilities:**

* Table status management and transitions
* Area organization and capacity planning
* Reservation handling and scheduling
* Real-time availability tracking
* Layout management and optimization

**TableService Implementation:**

The TableService provides comprehensive business logic implementation for table and area management operations, implementing sophisticated operational workflows and data management strategies:

**Service Architecture and Design Patterns:**

The TableService follows enterprise-level design patterns with comprehensive dependency injection and resource management:

* **DAO Integration Strategy**: Utilizes try-with-resources pattern for automatic connection management and resource cleanup, ensuring optimal database connection handling
* **Service Layer Abstraction**: Implements facade pattern providing simplified interface for complex table management operations while maintaining separation of concerns
* **Exception Handling Framework**: Comprehensive exception management with custom ServiceException wrapping for meaningful error propagation and debugging support

**Core Table Management Operations:**

**Active Table Retrieval and Filtering:**

* **Comprehensive Table Listing**: Advanced filtering mechanisms combining active status verification with customizable sorting options based on table names for consistent presentation
* **Area-Based Organization**: Sophisticated area-specific table retrieval with automatic filtering for active tables and streamlined data organization for efficient UI population
* **Availability Assessment**: Real-time availability calculation utilizing business logic integration for accurate table status determination and booking optimization
* **Capacity-Based Matching**: Intelligent party size accommodation logic with optimal capacity sorting to provide best-fit table recommendations for different group sizes

**Advanced Table Status Management:**

The service implements sophisticated table status management with comprehensive validation and audit trail functionality:

* **Status Transition Validation**: Implements business rule validation ensuring only valid status transitions are permitted (Available→Occupied/Reserved/Cleaning, Occupied→Cleaning/Available, etc.)
* **Audit Trail Integration**: Comprehensive logging of all status changes through SecurityLogger integration with user attribution and timestamp tracking
* **Real-time Notification System**: Event-based notification system for UI synchronization using Observer pattern for immediate status updates across all connected interfaces
* **Transaction Safety**: Database transaction management with rollback capabilities ensuring data consistency during complex status change operations

**Order Assignment and Table Coordination:**

The service provides sophisticated order-to-table assignment functionality with comprehensive validation and transaction management:

* **Multi-Entity Transaction Management**: Coordinated updates across table and order entities within single transaction scope ensuring data consistency and referential integrity
* **Availability Validation**: Real-time availability checking with business rule validation preventing double-booking and ensuring optimal table utilization
* **Bidirectional Relationship Management**: Maintains synchronized relationships between orders and tables with automatic status updates and cross-referential validation
* **Comprehensive Audit Logging**: Detailed operation logging including user attribution, operation timestamps, and entity relationship tracking for compliance and debugging
* **Error Recovery and Rollback**: Sophisticated error handling with automatic transaction rollback ensuring database consistency in case of operation failures

**Table Release and Lifecycle Management:**

The service provides comprehensive table release functionality with automated cleaning workflow and scheduling capabilities:

* **Active Order Validation**: Comprehensive order status checking preventing premature table release while active orders exist, ensuring operational integrity
* **Automated Cleaning Workflow**: Intelligent cleaning status assignment with scheduled availability restoration, supporting configurable cleaning periods for different table types
* **Scheduled Status Updates**: Timer-based automatic status transitions ensuring tables return to availability after appropriate cleaning periods without manual intervention
* **Audit Trail Maintenance**: Complete operation logging with user attribution and timestamp tracking for regulatory compliance and operational analytics

**Area Management Operations:**

The service implements sophisticated area management capabilities supporting complex restaurant layout configurations:

**Area Creation and Validation:**

* **Comprehensive Area Administration**: Full lifecycle management for dining areas including creation, validation, and organization with support for diverse layout configurations
* **Duplicate Prevention**: Advanced validation preventing area name conflicts with existing areas while supporting flexible naming conventions
* **Hierarchical Organization**: Support for complex area relationships and capacity management with automatic table count calculation and capacity optimization

**Table Creation and Management:**

* **Intelligent Table Provisioning**: Automated table creation with smart default value assignment, area validation, and naming convention support
* **Area Integration Validation**: Comprehensive area existence and status verification ensuring tables are only created in active, valid areas
* **Automated Naming Systems**: Intelligent table name generation using area-based prefixes and sequential numbering with fallback mechanisms for unique identification

**Business Logic and Validation Framework:**

The service implements comprehensive business rule validation and helper functions supporting operational integrity:

**Status Transition Management:**

* **Business Rule Enforcement**: Sophisticated state machine implementation defining valid table status transitions (Available→Occupied/Reserved/Cleaning, Occupied→Cleaning/Available, etc.)
* **Active Order Checking**: Real-time order status validation preventing table operations that would conflict with ongoing service delivery
* **Data Validation Framework**: Comprehensive input validation for both table and area entities ensuring data integrity and business rule compliance

**Supporting Infrastructure:**

* **Automated Name Generation**: Intelligent naming systems using area prefixes and sequential numbering with error handling and fallback mechanisms
* **Real-time Event Notification**: Platform-based event publishing for UI synchronization using EventBus pattern for immediate status updates
* **Scheduled Task Management**: Timer-based task scheduling for automated table status updates supporting configurable cleaning periods and maintenance workflows **Analytics and Performance Monitoring:**

The service provides comprehensive analytics and reporting capabilities for operational intelligence and business decision support:

* **Utilization Analytics**: Real-time calculation of table utilization metrics including occupancy rates, availability percentages, and area-specific performance statistics
* **Revenue Analysis**: Integration with order data to provide table-level revenue analytics and performance metrics for business optimization
* **Historical Reporting**: Comprehensive historical data analysis supporting trend identification and capacity planning decisions
* **Performance Metrics**: Key performance indicators including average table turnover time, peak usage periods, and efficiency metrics for operational optimization

This comprehensive TableService implementation provides robust, scalable table and area management capabilities supporting complex restaurant operations while maintaining data integrity and providing rich business intelligence capabilities.

### **11.3 TableController Implementation**

**Controller Architecture:**

The TableController manages the table layout interface, providing real-time table status visualization and interaction capabilities.

**Core Controller Responsibilities:**

* Real-time table status display
* Table assignment and status management
* Area-based table organization
* Interactive table selection
* Status change coordination

**TableController Key Features:**

The TableController provides sophisticated table layout management with real-time visualization and comprehensive interaction capabilities designed for high-volume restaurant operations:

**User Interface Architecture and Design:**

The controller implements a sophisticated multi-view interface supporting diverse operational requirements:

* **Flexible Layout Management**: Advanced tabbed interface organization with area-based table grouping supporting multiple dining zones and specialized seating areas
* **Multiple View Modes**: Comprehensive view options including grid-based visual layout and list-based tabular presentation with seamless switching capabilities
* **Interactive Control Elements**: Comprehensive control suite including area filtering, table operations, and real-time statistics display with responsive design principles
* **Real-time Statistics Dashboard**: Live metrics display showing total tables, availability counts, occupancy status, and operational performance indicators

**Service Integration and Data Management:**

The controller implements comprehensive data management strategies with service layer integration:

* **Multi-Service Coordination**: Integrated service layer communication combining TableService and OrderService for coordinated operations and data consistency
* **Hierarchical Data Organization**: Sophisticated area-based data organization using HashMap structures for efficient table grouping and retrieval optimization
* **Real-time Data Synchronization**: Observable data patterns ensuring immediate UI updates with table status changes and operational events
* **Inter-Controller Communication**: Advanced communication patterns through DashboardCommunicator interface supporting system-wide coordination and event propagation

**Initialization and Component Setup:**

The controller follows comprehensive initialization patterns ensuring reliable system startup and configuration:

* **Sequential Initialization**: Systematic component setup including service initialization, UI component configuration, event handler registration, and data loading workflows
* **Error Handling Framework**: Comprehensive exception management with user-friendly error messaging and graceful degradation for operational continuity
* **Real-time Update Configuration**: Event-based update system setup ensuring immediate reflection of table status changes across all interface components
* **Performance Monitoring**: Integrated performance tracking with initialization timing and operational metrics for system optimization

**Advanced View Management and Layout Systems:**

The controller implements sophisticated view management supporting multiple presentation modes and dynamic layout generation:

**Dynamic View Toggle System:**

* **Flexible View Switching**: Advanced toggle mechanism supporting seamless transitions between grid-based visual layouts and list-based tabular presentations with intelligent state management
* **State Persistence**: View preference management maintaining user selections across sessions with automatic restoration capabilities
* **Default Configuration**: Strategic default view selection optimizing initial user experience with grid-based visual presentation

**Comprehensive Event Handler Configuration:**

The controller implements advanced event handling patterns ensuring responsive user interaction and real-time operational coordination:

* **Area Filtering Integration**: Dynamic area-based filtering with automatic table display updates ensuring focused view management for specific operational zones
* **CRUD Operation Support**: Complete Create, Read, Update, Delete operation handling through button-based interfaces with context-sensitive availability management
* **Context Menu Integration**: Sophisticated right-click menu systems providing quick access to table operations, status changes, and administrative functions
* **Real-time Data Refresh**: Automatic and manual data refresh capabilities ensuring current information display with minimal operational disruption

**Data Loading and Organization Strategy:**

The controller implements sophisticated data loading patterns optimized for performance and user experience:

* **Hierarchical Data Loading**: Strategic service layer integration loading areas and associated tables with optimized query patterns for minimal database interaction overhead
* **Observable Collection Management**: Advanced observable list configuration for UI component population enabling automatic updates and reactive data binding
* **Area-Based Organization**: Intelligent data organization using HashMap structures indexed by area ID for efficient retrieval and display optimization
* **Error Handling and Recovery**: Comprehensive exception management during data loading with user-friendly error messaging and graceful degradation for operational continuity

// ===================================================== // TABLE MANAGEMENT UI ARCHITECTURE // =====================================================

**Area Tab Management System:**

* **Dynamic Tab Generation**: Hierarchical navigation structure creating scalable table organization across multiple physical zones with modular area-based categorization
* **Tabbed Interface Architecture**: Advanced tabbed interface system supporting multiple cafe areas with associated table collections and seamless navigation
* **Scalable Organization**: Modular approach enabling efficient table management across diverse operational zones with consistent user experience

**Dual Layout Rendering System:**

* **Flexible View Switching**: Advanced view switching mechanism supporting seamless transitions between grid and list layouts with optimal visualization preferences
* **Grid Layout Optimization**: Spatial organization system providing visual table cards with status indicators and interactive elements for intuitive management
* **List Layout Enhancement**: Traditional table view implementation with sortable columns, filtering capabilities, and comprehensive data display

**Grid Layout Architecture:**

* **Responsive Grid System**: Configurable column-based visualization (default 6 per row) with automatic wrapping and scrollable content areas for optimal display
* **Visual Card Rendering**: Individual table cards with integrated status indicators, capacity displays, and interactive event handling capabilities
* **Context-Sensitive Styling**: Dynamic styling system adapting visual presentation based on current table status and operational requirements

**List Layout Architecture:**

* **Traditional Table View**: Comprehensive table view implementation with sortable columns, filtering capabilities, and detailed information display
* **Structured Data Format**: Built-in selection and action management providing organized table data presentation with enhanced user interaction
* **Column-Based Organization**: Systematic column structure supporting efficient table information display and management operations

**Table View Component System:**

* **Individual Component Generation**: Visual component creation with integrated status indicators, capacity displays, and interactive event handling
* **Context-Sensitive Styling**: Dynamic styling adaptation based on current table status with comprehensive user interaction support
* **Interactive Element Integration**: Advanced event handling system providing responsive user interface with real-time status updates

**Column Configuration Framework:**

* **Comprehensive Column Structure**: Advanced column configuration including table identification, capacity information, and status visualization with color coding
* **Action Management Integration**: Built-in action management buttons with custom cell factories providing dynamic content rendering
* **Dynamic Content Rendering**: Custom cell factory implementation supporting flexible content display and interactive element management

**Context Menu System:**

* **Right-Click Menu Implementation**: Comprehensive context menu system providing table management actions including assignment, status modification, and editing
* **Dynamic Menu Management**: Menu items dynamically enabled/disabled based on table state with intelligent action availability
* **Release Operation Support**: Integrated table release operations with confirmation dialogs and safety checks for operational integrity

// ===================================================== // EVENT HANDLING ARCHITECTURE // =====================================================

**Double-Click Interaction Handler:**

* **Intelligent Table Selection**: Advanced selection behavior automatically determining appropriate actions based on table availability status with context-aware responses
* **Assignment Dialog Integration**: Available table triggering assignment dialogs while occupied tables display detailed information for comprehensive user guidance
* **Status-Based Action Routing**: Dynamic action routing system providing appropriate responses based on current table operational state

**Table Assignment Workflow:**

* **Complete Assignment Process**: Orchestrated table-to-order assignment process including order creation for new assignments and existing order association
* **Comprehensive State Management**: Advanced state management system maintaining data integrity through transaction-based operations with real-time dashboard communication
* **Synchronized Updates**: Real-time dashboard communication ensuring synchronized updates across all application components for operational consistency

**Status Management System:**

* **Dynamic Status Modification**: Advanced table status modification capabilities with immediate visual feedback and statistical updates for real-time operational awareness
* **Data Consistency Assurance**: Service layer validation ensuring data consistency with comprehensive user feedback through success/error messaging
* **Visual Feedback Integration**: Immediate visual feedback system providing real-time status updates with statistical information for operational decision support

**Table Release Protocol:**

* **Confirmation-Based Release**: Advanced confirmation-based table release system with safety checks and cleaning status transition for operational security
* **User Confirmation Dialogs**: Comprehensive user confirmation dialogs with service layer validation and comprehensive state management for operational integrity
* **Visual Update Integration**: Integrated visual update system providing immediate feedback for all table release operations with status transition management

// ===================================================== // DASHBOARD INTEGRATION ARCHITECTURE // =====================================================

**Table Selection Communication:**

* **Observer Pattern Implementation**: Advanced observer pattern implementation for real-time table selection synchronization across the application with consistent state management
* **UI Component Synchronization**: Consistent state management between different UI components maintaining data integrity throughout the user interface
* **Real-Time Synchronization**: Immediate synchronization system ensuring operational consistency across all application components

**Order Creation Integration:**

* **Automatic Table Refresh**: Advanced automatic table refresh capabilities when new orders are created ensuring real-time synchronization between order management and table status
* **Real-Time Synchronization**: Seamless synchronization system between order management and table status with comprehensive error handling
* **Graceful Degradation**: Error handling and graceful degradation for failed refresh operations ensuring continuous operational capability

**Communication Interface Management:**

* **Communication Bridge Establishment**: Advanced communication bridge establishment between table management components and main dashboard enabling real-time updates
* **Synchronized State Management**: Comprehensive synchronized state management across the entire application architecture with real-time update capabilities
* **Cross-Component Communication**: Seamless communication system enabling real-time updates and synchronized state management across all application components

**Utility Methods and Real-time Updates:**

The controller implements comprehensive utility functionality and real-time update capabilities ensuring responsive and current information display:

**Real-time Display Management:**

* **Platform-Based UI Updates**: Thread-safe UI update mechanisms using Platform.runLater for safe cross-thread UI manipulation ensuring responsive interface updates
* **Targeted Refresh Strategies**: Intelligent refresh algorithms updating specific table displays rather than complete interface refreshes for optimal performance
* **Area-Specific Updates**: Focused update mechanisms targeting specific operational areas minimizing unnecessary UI redraw operations

**Statistics and Performance Monitoring:**

* **Live Statistics Calculation**: Real-time calculation and display of table utilization metrics including total tables, availability counts, and occupancy statistics
* **Service Integration**: Direct integration with TableService analytics providing accurate and current operational data for decision support
* **Error Handling**: Comprehensive exception management during statistics updates ensuring continuous operation even with data access issues

**Event-Based Real-time Updates:**

* **EventBus Integration**: Advanced event subscription patterns enabling real-time response to table status changes across the entire application
* **Automatic Synchronization**: Seamless synchronization between backend data changes and frontend display ensuring immediate reflection of operational events
* **Performance Optimization**: Efficient event handling minimizing UI update overhead while maintaining real-time responsiveness for critical operational information

This sophisticated TableController implementation provides comprehensive table layout management with real-time visualization, advanced interaction capabilities, and robust performance optimization designed for high-volume restaurant operations.

This comprehensive Table and Area Management system provides efficient tools for organizing the cafe’s physical layout, managing table assignments, and tracking real-time table status while ensuring optimal space utilization and customer service.

## **12. Order Processing System**

### **12.1 Order Entity Model and Lifecycle Management**

The Order Processing System forms the core of the cafe’s operational workflow, managing the complete order lifecycle from creation to completion with comprehensive tracking and status management.

**Order Entity Architecture:**

The Order entity represents customer orders with detailed attributes supporting complex business operations:

**Order Entity Architecture:**

The Order entity provides comprehensive order representation supporting complex restaurant operations with sophisticated business logic integration:

**Core Entity Attributes:**

* **Identification and Tracking**: Unique order identifiers with customizable order numbering system supporting various formats and automatic generation capabilities
* **Relationship Management**: Comprehensive association management linking orders to tables, customers (optional for walk-in orders), and staff members for complete operational tracking
* **Financial Management**: Sophisticated pricing structure supporting pre-discount totals, discount applications, final amounts, and multiple payment method tracking
* **Status and Workflow**: Advanced status tracking with payment status monitoring and comprehensive notes system for special instructions and operational communication
* **Audit Trail**: Complete timestamp tracking for creation and modification supporting compliance requirements and operational analysis

**Business Logic Integration:**

* **Modification Control**: Intelligent modification rules allowing changes only during appropriate lifecycle stages (PENDING, PREPARING) ensuring operational integrity
* **Payment Readiness Assessment**: Sophisticated payment status evaluation determining when orders are ready for payment processing based on preparation and service status
* **Completion Tracking**: Advanced completion status management supporting complex workflow requirements and operational reporting
* **Financial Calculations**: Integrated tax calculation capabilities with configurable tax rates supporting diverse taxation requirements
* **Display Formatting**: Intelligent display number generation with fallback formatting ensuring consistent order identification across all interface components

**OrderDetail Entity Model:**

The OrderDetail entity provides granular order line item management with comprehensive product and customization tracking:

**Entity Structure and Relationships:**

* **Hierarchical Order Association**: Robust parent-child relationship linking individual line items to orders with referential integrity constraints
* **Product Integration**: Comprehensive product relationship management with snapshot capabilities preserving product information at time of order
* **Quantity and Pricing Management**: Sophisticated quantity tracking with unit price preservation and automatic line total calculation capabilities
* **Customization Support**: Advanced special request handling supporting custom modifications, dietary restrictions, and preparation instructions
* **Audit and Modification Tracking**: Complete creation and modification timestamp management for operational analysis and compliance requirements

**Business Logic and Display Management:**

* **Financial Calculations**: Intelligent line total calculation with quantity-price multiplication and validation ensuring accurate order totals
* **Special Request Detection**: Advanced special request validation and detection supporting conditional display logic and operational workflow optimization
* **Display Text Generation**: Sophisticated display formatting combining quantity, product names, and special requests for consistent presentation across all interface components

**Order Status Lifecycle:**

The OrderStatus enumeration provides comprehensive order workflow management with sophisticated state transitions and visual representation:

**Status Categories and Workflow:**

* **PENDING (Yellow #ffc107)**: Initial order state representing newly created orders awaiting kitchen preparation with full modification capabilities
* **PREPARING (Blue #17a2b8)**: Active preparation state indicating kitchen/bar staff are actively working on order items
* **READY (Green #28a745)**: Completion state signaling order is prepared and ready for service delivery to customers
* **SERVED (Purple #6f42c1)**: Service completion state indicating successful delivery to customer table with payment readiness
* **COMPLETED (Green #28a745)**: Final successful state representing fully paid and closed orders for reporting and analysis
* **CANCELLED (Red #dc3545)**: Terminal cancellation state for orders that cannot be completed due to various operational reasons

**State Transition Management:**

* **Workflow Validation**: Sophisticated state machine implementation ensuring only valid status transitions are permitted, maintaining operational integrity
* **Terminal State Protection**: Advanced protection mechanisms preventing modification of completed or cancelled orders ensuring data consistency
* **Cancellation Flexibility**: Comprehensive cancellation capabilities available from most active states supporting operational flexibility and customer service requirements

**Visual Integration and Display:**

* **Color-Coded Representation**: Consistent color scheme across all interface components providing immediate visual status recognition for staff efficiency
* **Descriptive Information**: Comprehensive status descriptions supporting staff training and customer communication requirements
* **Display Name Management**: Standardized display name handling ensuring consistent presentation across all system components and interfaces

### **12.2 OrderService Business Logic Implementation**

**OrderService Architecture:**

The OrderService orchestrates all order-related business operations, ensuring data consistency and business rule enforcement throughout the order lifecycle.

**Core OrderService Responsibilities:**

* Order creation and modification management
* Order detail management and calculation
* Status transition validation and tracking
* Promotion and discount application
* Integration with inventory and table management

**OrderService Implementation:**

The OrderService provides comprehensive order management capabilities implementing sophisticated business logic and integration patterns designed for high-volume restaurant operations:

**Service Architecture and Dependency Management:**

The service implements a comprehensive dependency injection pattern with multiple DAO and service integrations:

* **Multi-DAO Integration**: Coordinated data access through OrderDAO, OrderDetailDAO, ProductDAO, TableDAO, and CustomerDAO for complete order lifecycle management
* **Service Layer Coordination**: Advanced service integration with PromotionService and MenuService for cross-functional business logic execution
* **Resource Management**: Sophisticated connection management and transaction coordination ensuring data consistency across complex operations

**Core Business Operations:**

**Order Creation and Management Operations:**

The service implements comprehensive order creation workflows with sophisticated validation and business rule enforcement:

* **Table Assignment Validation**: Advanced table availability checking with real-time status verification ensuring accurate table-order associations
* **Order Initialization**: Sophisticated order entity creation with comprehensive default value assignment, unique order number generation, and audit trail establishment
* **Multi-Entity Coordination**: Coordinated creation involving order entities, table assignments, customer associations, and staff attribution for complete order tracking
* **Security and Audit Integration**: Comprehensive logging and security tracking for all order creation operations ensuring compliance and operational transparency

**Product Management and Order Detail Operations:**

The service provides advanced product addition and modification capabilities with real-time calculation and validation:

* **Dynamic Product Addition**: Intelligent product addition logic supporting quantity management, special requests, and existing item consolidation
* **Stock Management Integration**: Real-time inventory checking and automatic stock adjustment ensuring accurate inventory tracking
* **Price Calculation Engine**: Sophisticated pricing calculation with unit price preservation, line total calculation, and order total updates
* **Modification Control**: Advanced modification validation ensuring changes are only permitted during appropriate order lifecycle stages

**Order Total Management and Financial Operations:**

The service implements comprehensive financial calculation and management capabilities:

* **Real-time Total Calculation**: Dynamic order total calculation with automatic updates reflecting product additions, removals, and modifications
* **Discount and Promotion Integration**: Sophisticated promotion application with automatic discount calculation and promotion usage tracking
* **Tax Calculation Support**: Integrated tax calculation capabilities with configurable tax rates and comprehensive tax management
* **Financial Audit Trail**: Complete financial transaction tracking with detailed logging for compliance and analysis requirements

**Order Status Management and Workflow Control:**

The service provides advanced order status management with sophisticated workflow validation:

* **Status Transition Validation**: Comprehensive business rule enforcement ensuring only valid status transitions maintaining operational integrity
* **Workflow Coordination**: Advanced coordination between order status changes, table management, and kitchen operations
* **Real-time Status Updates**: Event-based status update system ensuring immediate reflection of status changes across all system components
* **Terminal State Management**: Sophisticated handling of completed and cancelled orders with appropriate business rule enforcement

**Query and Retrieval Operations:**

The service implements advanced order retrieval and filtering capabilities supporting operational and analytical requirements:

* **Multi-Criteria Filtering**: Comprehensive filtering capabilities by status, date range, table, customer, and staff member for operational efficiency
* **Performance Optimization**: Optimized query patterns with efficient indexing and caching strategies for high-volume operations
* **Real-time Data Access**: Current data retrieval with immediate reflection of recent changes and updates
* **Analytical Support**: Advanced querying capabilities supporting business intelligence and operational reporting requirements **Advanced Implementation Features:**

The OrderService implementation provides comprehensive business logic supporting complex restaurant operations with sophisticated integration patterns:

**Transaction Management and Data Consistency:**

* **Multi-Entity Transaction Coordination**: Advanced transaction management ensuring atomicity across order creation, table assignment, and inventory updates
* **Connection Pooling Optimization**: Efficient database connection utilization with automatic resource cleanup and performance optimization
* **Rollback and Recovery**: Sophisticated error handling with automatic transaction rollback ensuring data consistency in failure scenarios

**Business Rule Enforcement and Validation:**

* **Table Availability Validation**: Real-time table status checking preventing double-booking and ensuring accurate table assignments
* **Product Availability Checking**: Comprehensive product validation including stock levels, active status, and ordering capability assessment
* **Order Modification Control**: Advanced lifecycle-based modification rules ensuring changes are only permitted during appropriate order stages

**Integration and Event Management:**

* **Kitchen Coordination**: Advanced notification systems for kitchen operations with real-time order status communication
* **Inventory Management Integration**: Automatic stock level updates coordinated with product additions and order modifications
* **Promotion System Integration**: Sophisticated discount application with automatic promotion calculation and usage tracking

This comprehensive OrderService implementation provides robust, scalable order management capabilities supporting complex restaurant operations while maintaining data integrity and providing rich business intelligence for operational optimization. This comprehensive OrderService implementation continues with advanced product management, order modification, and status management capabilities designed for complex restaurant operations:

**Product Addition and Order Management Operations:**

The service implements sophisticated product addition workflows with comprehensive validation and business rule enforcement:

* **Multi-DAO Coordination**: Advanced coordination between OrderDAO, OrderDetailDAO, and ProductDAO ensuring data consistency across multiple entities during product addition operations
* **Comprehensive Validation Framework**: Multi-layer validation including order modification status checking, product availability verification, and quantity validation ensuring operational integrity
* **Order and Product Validation**: Sophisticated validation logic ensuring orders exist and can be modified while verifying product availability and ordering capability
* **Business Rule Enforcement**: Advanced business rule implementation preventing modifications to orders in inappropriate states and ensuring product availability before addition

**Intelligent Product Addition and Consolidation:**

The service implements sophisticated product addition logic with advanced consolidation and inventory management:

* **Quantity Validation**: Comprehensive quantity validation ensuring positive values and preventing invalid order entries
* **Duplicate Product Detection**: Advanced logic for detecting existing products in orders with special request differentiation, automatically consolidating quantities for identical items while maintaining separate entries for customized orders
* **Order Detail Creation**: Sophisticated order detail entity creation with comprehensive attribute assignment including product snapshots, pricing information, special requests, and audit timestamps
* **Real-time Inventory Management**: Automatic stock level updates coordinated with product additions ensuring accurate inventory tracking and prevention of overselling
* **Transaction Coordination**: Advanced transaction management with automatic total recalculation and commit/rollback handling ensuring data consistency

**Product Removal and Stock Restoration:**

The service provides comprehensive product removal capabilities with intelligent stock management:

* **Order Detail Validation**: Multi-layer validation ensuring order details exist and can be modified based on order status and business rules
* **Stock Restoration Logic**: Intelligent stock restoration automatically returning removed quantities to inventory maintaining accurate stock levels
* **Total Recalculation**: Automatic order total recalculation following product removal ensuring accurate financial tracking
* **Audit Trail Maintenance**: Comprehensive logging of all removal operations with user attribution and detailed operation descriptions for compliance and analysis

**Order Status Management and Workflow Control:**

The service provides comprehensive order status management with sophisticated workflow validation and coordination:

* **Status Transition Validation**: Advanced business rule enforcement using state machine patterns ensuring only valid status transitions are permitted throughout the order lifecycle
* **Status Update Coordination**: Sophisticated status update mechanisms with automatic timestamp management and comprehensive validation before persistence
* **Security and Audit Integration**: Complete logging and security tracking for all status changes with user attribution ensuring compliance and operational transparency
* **Status-Specific Action Handling**: Intelligent handling of status-specific operations including kitchen notifications, service alerts, and workflow coordination

**Financial Calculation and Pricing Engine:**

The service implements sophisticated financial calculation capabilities with automated promotion integration and real-time total management:

**Order Total Calculation System:**

* **Dynamic Subtotal Calculation**: Real-time calculation of order subtotals using stream processing for efficient aggregation of order detail totals
* **Promotion Integration**: Advanced promotion application system with automatic discount calculation and validation ensuring accurate promotional pricing
* **Final Amount Computation**: Sophisticated final amount calculation combining subtotals with discount applications for accurate billing amounts
* **Database Synchronization**: Automatic order total updates with timestamp management ensuring data consistency and audit trail maintenance

**Promotion Application Framework:**

* **Promotion Eligibility Assessment**: Comprehensive promotion evaluation considering order total, customer eligibility, and product-specific criteria
* **Discount Calculation Engine**: Advanced discount calculation with support for percentage-based, fixed-amount, and complex promotional algorithms
* **Usage Tracking and Analytics**: Comprehensive promotion usage recording supporting analytics, campaign effectiveness measurement, and promotional optimization
* **Error Handling and Fallback**: Robust error handling ensuring order processing continues even if promotion application fails with graceful degradation

**Order Retrieval and Query Operations:**

The service provides comprehensive order retrieval capabilities supporting diverse operational and analytical requirements:

**Multi-Criteria Order Filtering:**

* **Status-Based Retrieval**: Efficient order filtering by status with automatic sorting by order date ensuring operational workflow optimization
* **Table-Specific Queries**: Specialized table-based order retrieval supporting table management and service coordination workflows
* **Active Order Management**: Advanced active order filtering supporting kitchen operations and service delivery with real-time status assessment
* **Individual Order Access**: Direct order retrieval by ID with comprehensive error handling and validation ensuring data integrity

**Performance-Optimized Query Processing:**

* **Stream Processing Integration**: Advanced stream processing patterns for efficient filtering and sorting of large order datasets
* **Connection Management**: Sophisticated database connection handling with automatic resource cleanup ensuring optimal performance
* **Error Handling and Recovery**: Comprehensive exception management with meaningful error messages and graceful degradation for operational continuity

**Utility Operations and Support Functions:**

The service provides comprehensive utility functions supporting operational efficiency and system integration:

**Order Detail Management:**

* **Detail Retrieval Operations**: Comprehensive order detail retrieval with automatic error handling and validation ensuring accurate line item access for display and analysis

**Unique Order Number Generation:**

* **Intelligent Number Generation**: Advanced order number generation using date-based prefixes with daily sequence counting ensuring unique, readable order identification
* **Fallback Mechanisms**: Robust fallback number generation using timestamp-based algorithms providing collision-resistant order numbering even during database connectivity issues
* **Format Standardization**: Consistent order number formatting (ORD-YYYYMMDD-NNNN) supporting easy identification and operational workflow integration

**Status-Specific Action Coordination:**

* **Kitchen and Service Coordination**: Advanced notification systems for kitchen operations and service staff with real-time order status communication through EventBus patterns
* **Table Management Integration**: Sophisticated coordination with table management systems including automatic table release upon order completion ensuring efficient table turnover
* **Stock Restoration Management**: Intelligent stock restoration for cancelled orders with automatic inventory adjustment ensuring accurate stock levels and preventing inventory discrepancies

**Event-Based Communication System:**

* **Real-time Notification Framework**: Comprehensive event-based notification system using Platform.runLater for thread-safe UI updates and EventBus patterns for system-wide communication
* **Kitchen Display Integration**: Advanced kitchen notification system supporting real-time order communication for efficient food preparation workflows
* **Service Staff Coordination**: Sophisticated service staff notification system ensuring timely order delivery and customer service excellence

This comprehensive OrderService implementation provides robust, scalable order management capabilities supporting complex restaurant operations while maintaining data integrity and providing rich business intelligence for operational optimization.

### **12.3 OrderPanelController Implementation**

**Controller Architecture:**

The OrderPanelController manages the order creation and modification interface, providing real-time order management capabilities integrated with the main dashboard.

**Core Controller Responsibilities:**

* Order creation and item management
* Real-time order total calculation
* Product addition and removal
* Order status tracking and updates
* Integration with menu and table controllers

**OrderPanelController Key Features:**

The OrderPanelController provides a sophisticated order management interface implementing advanced JavaFX patterns and real-time data synchronization for efficient restaurant operations:

**User Interface Architecture and Component Design:**

The controller implements a comprehensive FXML-based interface supporting complete order lifecycle management:

* **Order Information Display**: Advanced order header management displaying order numbers, table assignments, customer information, and status indicators with real-time updates
* **Interactive Order Items Table**: Sophisticated table view implementation with custom cell factories for item names, quantities, pricing, totals, and action controls supporting in-line editing and modification
* **Financial Summary Section**: Comprehensive financial display including subtotals, discounts, taxes, and final totals with automatic calculation and real-time updates
* **Order Action Controls**: Complete action button suite supporting order creation, saving, printing, payment processing, and cancellation with context-sensitive availability
* **Customer Integration Panel**: Advanced customer search and selection interface with autocomplete functionality and quick customer association capabilities

**Service Integration and Data Management:**

The controller implements comprehensive service layer integration supporting complex order operations:

* **Multi-Service Coordination**: Advanced integration with OrderService, CustomerService, and TableService ensuring coordinated operations and data consistency
* **State Management**: Sophisticated state management using currentOrder tracking, table selection management, and observable order item collections for real-time UI updates
* **Inter-Controller Communication**: Advanced communication patterns through DashboardCommunicator interface supporting system-wide coordination and event propagation

**Initialization and Component Setup:**

The controller follows comprehensive initialization patterns ensuring reliable system startup and configuration:

* **Sequential Service Initialization**: Systematic service instantiation with comprehensive error handling and validation ensuring reliable service availability
* **UI Component Configuration**: Advanced UI component setup including table configuration, event handler registration, and customer search functionality
* **Empty State Management**: Intelligent empty state display providing clear user guidance and interface consistency during order creation workflows

**Advanced Table Configuration and Data Management:**

The controller implements sophisticated table configuration patterns supporting interactive order item management:

* **Dynamic Column Configuration**: Advanced column setup with custom cell factories for item display, quantity editing, price formatting, and action controls
* **Interactive Editing Capabilities**: In-line quantity editing with real-time validation and automatic item removal for zero quantities
* **Financial Display Formatting**: Specialized currency formatting for price and total columns ensuring consistent financial presentation
* **Action Control Integration**: Embedded remove buttons with stylized presentation and event handling for order item management
* **Observable Data Binding**: Advanced observable list configuration enabling real-time UI updates and reactive data synchronization

**Event Handler Configuration and User Interaction Management:**

The controller implements comprehensive event handling patterns ensuring responsive user interaction and real-time operational coordination:

* **Order Management Action Handlers**: Complete CRUD operation support through button-based interfaces including order creation, saving, printing, payment processing, and cancellation with context-sensitive availability
* **Status Change Management**: Advanced order status monitoring with automatic change detection and validation ensuring proper workflow progression
* **Customer Assignment Integration**: Real-time customer selection and assignment with automatic order updates supporting customer relationship management
* **Notes Management**: Automatic note saving with text change detection ensuring order information is preserved without manual intervention

**Customer Search and Selection System:**

The controller provides advanced customer search capabilities with real-time filtering and selection:

* **Customer Data Loading**: Comprehensive customer list loading with observable collection management for real-time UI updates
* **Real-time Search Filtering**: Advanced search functionality supporting name and phone number filtering with case-insensitive matching
* **Dynamic List Updates**: Intelligent customer list filtering with automatic dropdown updates providing immediate search feedback
* **Fallback to Complete List**: Smart reset functionality returning to complete customer list when search criteria are cleared

**Order Management Operations and Business Logic:**

The controller implements comprehensive order management capabilities supporting complete order lifecycle operations with sophisticated user interaction patterns:

**Order Creation and Initialization Framework:**

* **Dynamic Order Creation Logic**: Advanced order creation supporting both table-assigned and take-away orders with comprehensive state management and service integration
* **UI State Synchronization**: Sophisticated UI updates including order display refresh, total calculations, and control state management ensuring consistent user experience
* **Table Assignment Management**: Intelligent table association with automatic display updates and status tracking for efficient table management workflows
* **User Feedback Systems**: Comprehensive success and error feedback mechanisms with detailed operation messaging for operational transparency

**Product Addition and Management Operations:**

* **Automatic Order Initialization**: Smart order creation triggering when products are added to non-existent orders ensuring seamless operational workflow
* **Special Request Integration**: Advanced special request dialog system for customizable products supporting detailed customer requirement capture
* **Service Layer Coordination**: Comprehensive OrderService integration for product addition with automatic order detail creation and real-time inventory management
* **Real-time Interface Updates**: Immediate UI synchronization including order item refresh and financial total recalculation ensuring current information display

**Order Item Management and User Interaction:**

* **Interactive Item Removal**: Sophisticated confirmation dialog system with comprehensive validation ensuring user intent verification before irreversible operations
* **Service Integration Patterns**: Advanced OrderService integration for item removal with automatic stock restoration and order total recalculation
* **User Experience Optimization**: Streamlined removal workflow with immediate UI updates and comprehensive success feedback for operational efficiency

**Item Quantity Management and Update Operations:**

The controller provides sophisticated item quantity management with intelligent update algorithms and comprehensive error handling:

* **Dynamic Quantity Calculation**: Advanced quantity difference calculation supporting both increment and decrement operations with special request preservation
* **Service Integration Patterns**: Sophisticated OrderService coordination for quantity updates using remove-and-readd strategies ensuring data consistency
* **Real-time UI Synchronization**: Immediate order item refresh and total recalculation with automatic error recovery and state reversion capabilities
* **Error Handling and Recovery**: Comprehensive exception management with automatic UI state restoration ensuring operational continuity

**UI Update Methods and State Management:**

The controller implements comprehensive UI update mechanisms ensuring consistent information display and user experience:

**Order Item Refresh Operations:**

* **Real-time Data Synchronization**: Advanced order detail retrieval with automatic observable collection updates ensuring current information display
* **Service Layer Integration**: Comprehensive OrderService integration for order detail access with comprehensive error handling and user feedback

**Order Display Management:**

* **Comprehensive Order Information Display**: Advanced order display updates including order numbers, status indicators, notes, and customer information with automatic formatting
* **Customer Integration**: Sophisticated customer information loading and display with optional customer relationship management and error handling

**Financial Total Calculation and Display:**

* **Real-time Financial Calculation**: Advanced total calculation including subtotals, discounts, taxes, and final amounts with configurable tax rates
* **Database Synchronization**: Intelligent order refresh from database ensuring latest financial information display with automatic currency formatting
* **Empty State Handling**: Comprehensive empty state management with proper zero value display and error handling

**Control State Management:**

* **Dynamic Control Enablement**: Sophisticated control state management enabling and disabling order controls based on order existence and operational state
* **Comprehensive Control Coordination**: Advanced control state synchronization across all order management interfaces ensuring consistent user experience

**Dashboard Communication and Inter-Controller Coordination:**

The controller implements sophisticated inter-controller communication patterns supporting seamless system integration and coordinated operations:

**Table Selection and Order Loading:**

* **Dynamic Table Selection Handling**: Advanced table selection processing with automatic order status checking and intelligent workflow determination
* **Existing Order Detection**: Sophisticated active order detection using stream filtering and order completion status validation ensuring accurate order state management
* **Conditional Order Loading**: Smart order loading logic supporting both existing order continuation and new order preparation based on table status
* **Error Handling and Recovery**: Comprehensive error management during table order loading with user feedback and graceful degradation

**Order Loading and State Management:**

* **Comprehensive Order State Restoration**: Advanced order loading with complete UI state restoration including order items, display information, totals, and control enablement
* **Multi-Component Synchronization**: Coordinated updates across all UI components ensuring consistent order information display and user experience

**Dashboard Integration Framework:**

* **Communicator Pattern Implementation**: Advanced inter-controller communication using DashboardCommunicator interface supporting system-wide coordination and event propagation
* **Controller Coupling Management**: Sophisticated controller integration maintaining loose coupling while enabling rich communication capabilities

## This comprehensive OrderPanelController implementation provides robust, scalable order management capabilities with advanced user interaction patterns, real-time data synchronization, and seamless integration with other system components, ensuring efficient order processing workflows and superior user experience.

This comprehensive Order Processing System provides robust functionality for managing the complete order lifecycle, from creation through completion, with real-time updates, comprehensive validation, and seamless integration with other system components.

## **13. Payment Processing System**

### **13.1 Payment Entity Model and Business Logic**

The Payment Processing System handles all financial transactions within the cafe management system, supporting multiple payment methods and ensuring secure, accurate financial record-keeping.

**Payment Entity Architecture:**

Payment processing is integrated into the Order entity with additional support structures for complex payment scenarios:

**Payment Method Enumeration:**

The PaymentMethod enumeration provides comprehensive payment method support with sophisticated business logic and external processing capabilities:

**Payment Method Categories and Features:**

* **CASH (💵)**: Traditional physical cash payment supporting partial payments and immediate processing with change calculation capabilities
* **CARD (💳)**: Credit/Debit card payments via terminal integration supporting partial payments and secure transaction processing
* **MOMO (📱)**: MoMo mobile payment integration requiring external processing with QR code generation and real-time transaction validation
* **VNPAY (🏦)**: VNPay digital wallet integration providing secure online payment processing with comprehensive transaction tracking
* **ZALOPAY (💰)**: ZaloPay mobile payment system supporting instant mobile transactions with advanced security features

**Business Logic Integration:**

* **External Processing Detection**: Intelligent detection of payment methods requiring external processing enabling appropriate workflow selection
* **Partial Payment Support**: Advanced support for partial payments enabling flexible payment scenarios and installment capabilities
* **Display and UI Integration**: Comprehensive display name, icon, and description management supporting consistent user interface presentation

**Payment Status Enumeration:**

The PaymentStatus enumeration provides comprehensive payment status tracking with visual representation and workflow management:

**Status Categories with Visual Indicators:**

* **PENDING (Yellow #ffc107)**: Initial payment state awaiting processing with appropriate user interface styling
* **PROCESSING (Blue #17a2b8)**: Active processing state indicating payment is being handled by external systems
* **PAID (Green #28a745)**: Successful completion state representing confirmed payment with positive visual feedback
* **FAILED (Red #dc3545)**: Failed payment state with error indication and retry capabilities
* **REFUNDED (Gray #6c757d)**: Refund completion state for processed refunds with audit trail integration
* **PARTIAL (Orange #fd7e14)**: Partial payment state supporting complex payment scenarios and outstanding balance tracking

**Status Business Logic:**

* **Completion Detection**: Advanced completion status detection supporting workflow decisions and order processing logic
* **Refund Eligibility**: Intelligent refund eligibility checking enabling appropriate refund processing workflows

**PaymentRequest Data Transfer Object:**

The PaymentRequest DTO provides comprehensive payment request management with advanced validation and calculation capabilities:

**Core Payment Information Management:**

* **Order Association**: Direct order relationship management linking payments to specific orders with comprehensive metadata support
* **Payment Method Integration**: Advanced payment method selection with external transaction ID support for complex payment scenarios
* **Amount Management**: Sophisticated amount handling including received amounts, due amounts, and comprehensive validation logic

**Validation and Calculation Framework:**

* **Amount Validation**: Advanced amount validation ensuring positive values and sufficient payment amounts with business rule enforcement
* **Change Calculation**: Intelligent change calculation with automatic excess amount determination supporting cash transaction workflows
* **Payment Requirement Assessment**: Advanced requirement checking determining change necessity for appropriate user interface updates

**PaymentResponse Data Transfer Object:**

The PaymentResponse DTO provides comprehensive payment result management with detailed transaction information and error handling:

**Transaction Result Management:**

* **Success Status Tracking**: Comprehensive success/failure status management with detailed payment status integration
* **Transaction Identification**: Advanced transaction ID management supporting audit trails and external system integration
* **Financial Information**: Detailed amount tracking including paid amounts, change amounts, and receipt number generation
* **Error Handling**: Comprehensive error message management supporting user feedback and troubleshooting workflows
* **Processing Timestamps**: Automatic timestamp management supporting audit trails and performance monitoring

**Factory Method Pattern Implementation:**

* **Success Response Generation**: Advanced factory method for successful payment responses with comprehensive transaction information including transaction IDs, payment amounts, change calculation, and receipt number generation
* **Failure Response Generation**: Sophisticated failure response creation with error message management and automatic timestamp assignment for audit trail maintenance

## **Automatic Status Assignment**: Intelligent status assignment with automatic timestamp generation ensuring consistent response creation patterns

### **13.2 PaymentService Business Logic Implementation**

**PaymentService Architecture:**

The PaymentService orchestrates payment processing across multiple payment methods while ensuring transaction integrity and comprehensive audit trails.

**Core PaymentService Responsibilities:**

* Multi-method payment processing
* Transaction validation and verification
* Receipt generation and management
* Payment status tracking and updates
* Refund processing and management
* Integration with external payment providers

**PaymentService Implementation:**

The PaymentService provides comprehensive payment processing capabilities implementing sophisticated multi-method payment handling with enterprise-level security and integration patterns:

**Service Architecture and Dependency Management:**

The service implements comprehensive dependency injection patterns with multiple service integrations:

* **Multi-DAO Coordination**: Advanced integration with OrderDAO and CustomerDAO ensuring coordinated data access and transaction management
* **Specialized Service Integration**: Comprehensive integration with ReceiptService for receipt generation and QRCodeService for mobile payment support
* **Validation Framework**: Advanced PaymentValidator integration ensuring comprehensive payment request validation and business rule enforcement
* **Service Initialization**: Sophisticated service initialization with automatic dependency setup ensuring reliable service availability

**Core Payment Processing Operations:**

The service implements advanced payment processing workflows with comprehensive validation and transaction management:

**Payment Request Validation and Processing:**

* **Multi-Layer Validation**: Comprehensive payment request validation including amount validation, order status checking, and payment method verification
* **Order Status Verification**: Advanced order validation ensuring orders exist, are ready for payment, and have not been previously paid
* **Transaction Coordination**: Sophisticated database transaction management with automatic commit/rollback handling ensuring data consistency
* **Method-Specific Processing**: Intelligent payment processing delegation based on payment method with appropriate workflow selection

**Payment Method Integration and Processing:**

* **Multi-Method Support**: Advanced payment method processing supporting cash, card, and mobile payment methods with method-specific business logic
* **External Payment Integration**: Sophisticated integration with external payment providers for mobile payments including MoMo, VNPay, and ZaloPay
* **Transaction Status Management**: Comprehensive transaction status tracking with real-time status updates and error handling
* **Receipt Generation Integration**: Automatic receipt generation upon successful payment with comprehensive receipt number management

**Customer Loyalty and Analytics Integration:**

* **Loyalty Points Management**: Advanced customer loyalty point calculation and assignment for successful payments
* **Analytics Integration**: Comprehensive payment analytics tracking supporting business intelligence and financial reporting requirements **Security and Audit Integration:**
* **Comprehensive Audit Logging**: Advanced security logging for all payment operations including successful payments and failed attempts with detailed user attribution
* **Transaction Management**: Sophisticated database transaction coordination with automatic commit for successful payments and rollback for failures
* **Error Handling and Recovery**: Comprehensive exception management with detailed error logging and graceful degradation

**Payment Method Selection and Delegation:**

* **Method-Specific Processing**: Intelligent payment method detection and delegation to appropriate processing workflows
* **Comprehensive Method Support**: Complete coverage of all payment methods including cash, card, and mobile payment options
* **Extensible Architecture**: Sophisticated architecture supporting easy addition of new payment methods with minimal code changes
* **Error Handling**: Comprehensive validation for unsupported payment methods ensuring robust operation

**Payment Method Implementation Framework:**

The service provides comprehensive payment method implementations supporting diverse payment scenarios with sophisticated processing workflows:

**Cash Payment Processing:**

* **Amount Validation**: Advanced cash amount validation ensuring sufficient funds with comprehensive insufficiency handling
* **Change Calculation**: Sophisticated change calculation with automatic excess amount determination supporting accurate cash transaction workflows
* **Transaction ID Generation**: Unique transaction identifier generation with method-specific prefixes ensuring audit trail accuracy
* **Immediate Processing**: Instantaneous cash payment processing without external dependencies ensuring rapid transaction completion

**Card Payment Processing:**

* **Terminal Integration Simulation**: Advanced card processing simulation with real-world processing delay modeling and balance validation
* **Transaction Coordination**: Sophisticated card transaction processing with timeout handling and interruption management
* **Error Handling**: Comprehensive error management for card declines, processing failures, and communication issues
* **No-Change Processing**: Intelligent processing for card payments eliminating change calculation requirements

**Mobile Payment Integration (MoMo, VNPay, ZaloPay):**

**MoMo Payment Processing:**

* **QR Code Generation**: Advanced QR code generation using QRCodeService integration for mobile payment initiation
* **External API Integration**: Comprehensive MoMo API integration patterns with transaction ID management and payment URL generation
* **Payment Details Enhancement**: Sophisticated payment response enhancement with QR code paths and payment URL integration for user experience optimization

**VNPay Payment Processing:**

* **Payment URL Generation**: Advanced VNPay payment URL generation with order-specific parameters and transaction tracking
* **Redirect Management**: Sophisticated redirect-based payment flow with comprehensive URL management and user experience optimization
* **External Integration**: Complete VNPay API integration supporting secure payment processing and transaction validation

**ZaloPay Payment Processing:**

* **Order Data Generation**: Advanced ZaloPay order data generation with comprehensive metadata management and transaction tracking
* **Payment Details Integration**: Sophisticated payment response enhancement with ZaloPay-specific data for mobile payment processing
* **Transaction Coordination**: Complete ZaloPay integration supporting instant mobile payments and real-time transaction validation

**Common Payment Processing Features:**

* **Method-Specific Transaction IDs**: Intelligent transaction ID generation with payment method prefixes ensuring unique identification and audit trail accuracy
* **Success Response Generation**: Comprehensive success response creation with appropriate amount tracking and change calculation
* **Error Handling Framework**: Advanced error management with method-specific error messages and graceful failure handling
* **Payment Details Enhancement**: Sophisticated payment response enhancement with method-specific details supporting user experience and integration requirements

**Payment Utilities and Helper Functions:**

The service provides comprehensive utility functions supporting payment processing operations with sophisticated validation and integration capabilities:

**Payment Request Validation Framework:**

* **Comprehensive Null Checking**: Advanced null validation ensuring payment requests are properly initialized before processing
* **Order ID Validation**: Sophisticated order ID validation ensuring valid positive integer values for order association
* **Payment Method Validation**: Comprehensive payment method validation ensuring valid payment method selection
* **Amount Validation**: Advanced amount validation for both due amounts and received amounts ensuring positive values and business rule compliance
* **External Validator Integration**: Advanced PaymentValidator integration providing additional business rule validation and comprehensive request verification

**Order Status Management and Updates:**

* **Payment Status Synchronization**: Sophisticated order payment status updates coordinating payment method information and status tracking
* **Order Completion Logic**: Intelligent order completion handling automatically transitioning successful payments to completed status
* **Timestamp Management**: Comprehensive timestamp tracking for payment completion ensuring accurate audit trails
* **Database Coordination**: Advanced database update coordination ensuring consistent order status across all system components

**Customer Loyalty Integration:**

* **Loyalty Point Calculation**: Advanced loyalty point calculation system awarding points based on spending amounts (1 point per 10,000 VND)
* **Customer Data Updates**: Sophisticated customer data updates including loyalty points accumulation and total spending tracking
* **Error Isolation**: Comprehensive error handling ensuring payment processing continues even if loyalty point updates fail
* **Transaction Coordination**: Advanced customer data updates within payment transactions ensuring data consistency

**Receipt Generation and Management:**

* **Receipt Service Integration**: Comprehensive ReceiptService integration for receipt generation with automatic fallback mechanisms
* **Fallback Receipt Numbers**: Intelligent fallback receipt number generation ensuring receipt identification even during service failures
* **Error Handling**: Advanced error management with graceful degradation for receipt generation failures

**Transaction and Payment Data Generation:**

* **Unique Transaction ID Generation**: Sophisticated transaction ID generation using timestamp and random components with method-specific prefixes ensuring unique identification
* **MoMo QR Data Generation**: Advanced MoMo QR code data generation with order information and amount formatting for mobile payment integration
* **VNPay URL Generation**: Comprehensive VNPay payment URL generation with sandbox integration and proper amount formatting for external payment processing
* **ZaloPay Data Generation**: Sophisticated ZaloPay payment data generation with comprehensive metadata and order information for mobile payment integration

**Payment Method Integration Utilities:**

* **Method-Specific Data Formatting**: Advanced data formatting for each payment method ensuring proper integration with external payment providers
* **URL and QR Code Management**: Comprehensive URL and QR code generation supporting mobile payment workflows and user experience optimization
* **External API Preparation**: Sophisticated data preparation for external payment API integration ensuring proper format and validation requirements **Refund Processing Framework:**

The service provides comprehensive refund processing capabilities supporting complex refund scenarios with sophisticated validation and method-specific processing:

**Refund Validation and Processing:**

* **Order Validation**: Comprehensive order existence and payment status validation ensuring refunds are only processed for paid orders
* **Amount Validation**: Advanced refund amount validation ensuring positive values within original payment limits preventing fraudulent refund attempts
* **Original Payment Method Detection**: Intelligent original payment method detection enabling appropriate refund processing workflows
* **Transaction Coordination**: Sophisticated database transaction management with automatic commit/rollback handling ensuring refund data consistency

**Method-Specific Refund Processing:**

* **Cash Refund Processing**: Immediate cash refund processing with transaction ID generation for audit trail maintenance
* **Electronic Payment Refunds**: Advanced electronic payment refund processing supporting card, MoMo, VNPay, and ZaloPay refunds with external API integration patterns
* **Transaction ID Management**: Sophisticated refund transaction ID generation with method-specific prefixes ensuring unique identification and audit trail accuracy
* **Status Update Coordination**: Comprehensive order status updates marking orders as refunded with timestamp management

**Audit Trail and Security Integration:**

* **Security Logging**: Advanced security logging for all refund operations including amount details, reasons, and user attribution
* **Reason Tracking**: Comprehensive refund reason tracking supporting operational analytics and fraud prevention
* **Transaction Logging**: Complete transaction logging with detailed refund information for compliance and audit requirements

## This comprehensive PaymentService implementation provides robust, scalable payment processing capabilities supporting diverse payment methods, comprehensive validation, secure transaction handling, and advanced refund management for complex restaurant operations while maintaining data integrity and regulatory compliance.

### **13.3 PaymentController Implementation**

**Controller Architecture:**

The PaymentController manages the payment interface, providing intuitive payment processing with support for multiple payment methods and real-time feedback.

**Core Controller Responsibilities:**

* Payment method selection and validation
* Payment amount calculation and display
* Payment processing coordination
* Receipt generation and display
* Error handling and user feedback

**PaymentController Key Features:**

**PaymentController Implementation Architecture:**

The PaymentController implements a comprehensive payment processing system with multi-method payment support, real-time validation, and asynchronous processing capabilities:

**FXML Component Injection System:**

* **Container Management**: Advanced BorderPane container system providing structured layout organization for payment interface components with hierarchical component management
* **Payment Method Selection**: Comprehensive ToggleGroup implementation supporting multiple payment methods including cash, card, and mobile payment options with dynamic UI switching
* **Payment-Specific UI Components**: Specialized UI components for each payment method including cash amount input, card terminal instructions, and mobile QR code display with context-sensitive visibility management
* **Action Button Integration**: Strategic button placement including process payment, cancel, and print receipt functionality with intelligent state management and user interaction control
* **Status Display System**: Advanced status display components including progress indicators and status labels providing real-time feedback during payment processing operations

**Service Layer Integration Architecture:**

* **Multi-Service Coordination**: Comprehensive service integration including PaymentService, OrderService, and ReceiptService with dependency injection and lifecycle management
* **Data State Management**: Advanced state management system maintaining current order, payment response, and dialog stage references with proper memory management
* **Initialization Protocol**: Sophisticated initialization sequence including service instantiation, UI component setup, and event handler configuration with comprehensive error handling

**Payment Method Management System:**

* **Dynamic UI Switching**: Advanced UI switching mechanism supporting seamless transitions between different payment method interfaces with intelligent component visibility control
* **Payment Method Configuration**: Comprehensive payment method configuration including radio button setup, default selection management, and user data association for method identification
* **Change Calculation Engine**: Real-time change calculation system for cash payments with automatic amount validation and button state management based on payment adequacy
* **Method-Specific Instructions**: Context-sensitive instruction display system providing appropriate guidance for each payment method including card terminal and mobile app integration

**Event Handling and Validation Framework:**

* **Comprehensive Event Management**: Advanced event handling system including payment processing, cancellation, receipt printing, and real-time amount calculation with keyboard shortcut support
* **Payment Validation Protocol**: Sophisticated validation system ensuring order selection, amount adequacy, and payment method compatibility with comprehensive error messaging
* **Asynchronous Processing Integration**: Advanced asynchronous payment processing with background task execution, progress monitoring, and thread-safe UI updates using Platform.runLater

**Payment Processing Workflow Architecture:**

* **Dialog Management System**: Comprehensive payment dialog lifecycle management including stage creation, modal behavior, and parent window centering with proper cleanup protocols
* **Order Display Integration**: Advanced order information display system including order number, total amount, and final amount presentation with automatic UI updates
* **Payment Request Creation**: Sophisticated payment request creation system supporting multiple payment methods with appropriate amount handling and request object construction
* **Response Handling Protocol**: Advanced payment response handling including success/failure state management, receipt button activation, and automatic dialog closure with user experience optimization

**UI Status Management and User Feedback:**

The controller implements comprehensive UI status management providing real-time feedback and user interaction coordination:

**Payment Processing Status Display:**

* **Processing State Indication**: Advanced processing status display with progress indicators and button state management during payment processing
* **Status Visibility Management**: Sophisticated UI component visibility control coordinating status display elements and progress feedback
* **User Interaction Control**: Intelligent control disabling during processing preventing duplicate payment attempts and ensuring processing integrity

**Payment Result Communication:**

* **Success State Display**: Comprehensive success status presentation with positive visual feedback, change amount display, and payment method control updates
* **Failure State Management**: Advanced failure status display with error message integration and clear failure indication for user understanding
* **Payment Method Control**: Sophisticated payment method control management disabling selection after successful payment completion

**Receipt Management and Cleanup:**

* **Receipt Printing Integration**: Advanced receipt service integration with success feedback and comprehensive error handling for printing operations
* **Payment Dialog Management**: Intelligent payment dialog lifecycle management with proper cleanup and user experience optimization

## This comprehensive PaymentController implementation provides sophisticated user interface management with real-time payment processing feedback, comprehensive status indication, and seamless user experience optimization for efficient payment processing workflows.

This comprehensive Payment Processing System provides secure, multi-method payment handling with real-time processing, comprehensive validation, receipt generation, and seamless integration with the overall cafe management workflow, ensuring accurate financial transactions and customer satisfaction.

# **PART IV: ADVANCED FEATURES & INTEGRATIONS**

## **14. Customer Management System**

### **14.1 Customer Entity Model and Business Logic**

The **Customer** entity supports comprehensive customer relationship management, enabling loyalty programs, purchase tracking, and targeted promotions.

**Entity Attributes:** - **customerId**: Unique identifier - **fullName**: Customer name (mandatory) - **phone**: Primary contact number (unique, validated) - **email**: Optional contact for digital receipts & promotions - **loyaltyPoints**: Accumulated points - **totalSpent**: Aggregated spending history - **createdAt / updatedAt**: Timestamps for lifecycle tracking

**Business Logic Functions:** - **Loyalty Point Accumulation**: 1 point per 10,000 VND spent - **Tier-based Benefits**: Bronze, Silver, Gold levels (future upgrade) - **Customer Validation**: Duplicate prevention (phone uniqueness) - **Purchase History Tracking**: Aggregates orders linked to each customer

### **14.2 CustomerService Business Logic**

**Core Responsibilities:** - Customer registration & profile management - Loyalty point calculation and update after payments - Search/filter functions (by name, phone, loyalty level) - Integration with **OrderService** for history tracking

**Advanced Functions:** - **Auto-detection**: Phone number lookup for repeat customers - **Profile Enrichment**: Automatic update of totalSpent and loyalty points - **Data Analytics**: Identify top customers for targeted campaigns

### **14.3 CustomerController Implementation**

**Features:** - **Customer List Table**: Displays ID, name, phone, email, loyalty points, total spent - **Add/Edit Dialog**: Modal form for profile management - **Search & Filter**: Real-time search by phone/name - **Order Integration**: Customer info automatically linked during checkout - **Validation**: Prevents duplicate phone number registrations

**UI Design:** - Tab-based navigation in admin panel - Real-time loyalty point updates after payment - Context menu for quick actions (View History, Edit, Delete)

## **15. Promotion & Discount System**

### **15.1 Promotion Entity Model**

**Entity Attributes:** - **promotionId**: Unique identifier - **code**: Promotion code (e.g., WELCOME10) - **description**: Details of the discount rule - **discountType**: ENUM(‘PERCENTAGE’, ‘FIXED’) - **discountValue**: Discount rate or amount - **startDate / endDate**: Validity period - **isActive**: Boolean flag - **usageLimit**: Optional cap on usage

### **15.2 PromotionService Business Logic**

**Core Responsibilities:** - Validate promotion code during checkout - Calculate applicable discount - Ensure promotion eligibility (date, usage count, min order value) - Apply multiple promotions with priority rules

**Discount Calculation Engine:** - **Percentage Discounts**: e.g., 10% off total bill - **Fixed Amount Discounts**: e.g., -50,000 VND - **Combination Rules**: Highest-value discount prioritized

**Integration:** - Works with **OrderService** for final amount calculation - Links with **CustomerService** for personalized campaigns

### **15.3 PromotionController Implementation**

**Features:** - **Promotion List Table**: Code, discount type, value, validity - **CRUD Operations**: Add, edit, delete promotions - **Search/Filter**: Active promotions, expired, upcoming - **Bulk Import/Export**: Excel upload via Apache POI - **Usage Analytics**: Tracks redemption rate per promotion

**UI Design:** - Admin panel tab for promotions - Form validation: prevents overlapping promotion dates - Real-time calculation preview during checkout

## **16. Reporting & Analytics**

### **16.1 SalesData Entity Model**

**Attributes:** - **reportId**: Unique ID - **dateRange**: Report time span - **totalOrders**: Number of completed orders - **totalRevenue**: Sum of final amounts - **averageOrderValue**: Computed metric - **topProducts**: List of best-selling items - **customerStats**: New vs returning customers

### **16.2 ReportService Business Logic**

**Core Reports:** - **Daily Sales Summary** - **Monthly Revenue Trends** - **Product Performance Report** - **Customer Loyalty Report**

**Technical Features:** - **Excel Export (Apache POI)** for financial reports - **PDF Export (iText)** for official documents - **Predefined Database Views** for sales and menu analytics

**KPIs Monitored:** - Orders/hour, Average Table Turnover - Revenue by category (Coffee, Tea, Cake…) - Customer retention rate

### **16.3 AdminReportController Implementation**

**Features:** - **Date Range Picker**: Select report period - **Report Generation Buttons**: Excel, PDF export - **Charts & Graphs**: JavaFX charts (bar, line, pie) - **Interactive Dashboard**: Filter by product/category

## **17. QR Code & Receipt System**

### **17.1 QR Code Integration**

**Features:** - **Table QR Codes**: Each table has a unique QR for quick check-in - **Payment QR Codes**: Integration with MoMo, VNPay, ZaloPay - **Order Tracking**: Customers can scan to check order status

**Implementation:** - ZXing library for QR generation - Stored in DB & printed on receipts

### **17.2 Receipt System**

**Receipt Features:** - **Auto-generated on Payment** - **Content**: Cafe logo, order details, discount, total, payment method, QR code - **Formats**: PDF, print-friendly text - **Multi-language**: English & Vietnamese

**ReceiptController Implementation:** - Generates receipts after successful payment - Exports PDF with iText - Supports reprinting via admin panel

# **PART V: DATA ACCESS & BUSINESS LOGIC**

## **18. Data Access Layer (DAO Pattern)**

### **18.1 DAO Design Principles**

The Cafe Management System applies the **Data Access Object (DAO) pattern** to abstract and encapsulate all database access logic. This approach ensures separation of concerns, easier maintenance, and improved testability.

**Key Benefits:** - **Centralized Database Access Logic**: All database operations are contained within DAO classes - **SQL Injection Prevention**: Prepared statements ensure secure database interactions - **Database Independence**: Easy transition between different database systems - **Testability**: Mock DAO implementations simplify unit testing - **Maintainability**: Clear separation between business logic and data access

### **18.2 DAO Architecture**

**DAO Layer Components:**

* **Interfaces**: Define common operations (CRUD, queries) for each entity
* **Implementations**: Contain JDBC code with MySQL-specific queries
* **BaseDAO**: Shared utility functions (connection handling, error logging)

**Interface Design Pattern:** The ProductDAO interface defines a comprehensive contract for product data access operations. It includes methods for basic CRUD operations (Create, Read, Update, Delete) as well as specialized queries for category-based filtering and stock management. The interface design follows the DAO pattern principles, providing a clean abstraction layer between business logic and data persistence.

**Implementation Architecture:** The ProductDAOImpl class implements the ProductDAO interface using JDBC with MySQL-specific optimizations. The implementation uses prepared statements to prevent SQL injection attacks and ensure secure database interactions. The class includes comprehensive error handling with logging, automatic resource management using try-with-resources patterns, and efficient data mapping from database results to domain objects.

### **18.3 DAO Components by Entity**

**UserDAO:** - Handles authentication queries and user lookup - Supports user creation, update, deactivation - Manages role-based filtering and user status

**ProductDAO:** - CRUD operations for menu items - Price and availability updates - Category-based filtering and stock management

**OrderDAO:** - Create new orders and retrieve order history - Manage order status updates and transitions - Handle order detail relationships and calculations

**CustomerDAO:** - Store and retrieve customer profiles - Loyalty points and purchase history tracking - Phone-based quick lookup and duplicate prevention

**PromotionDAO:** - Manage promotions and validity checks - Track usage limits and redemption logs - Active promotion filtering and expiration handling

### **18.4 Transaction Management**

The DAO layer integrates with the Service layer for comprehensive transaction management:

**Transaction Principles:** - Uses Connection.setAutoCommit(false) for multi-step operations - Ensures rollback on failure to maintain data consistency - Guarantees ACID compliance across complex operations

**Example Transaction:**

try (Connection conn = DatabaseConfig.getConnection()) {  
 conn.setAutoCommit(false);  
   
 // Create order  
 orderDAO.create(order, conn);  
   
 // Add order details  
 for (OrderDetail detail : orderDetails) {  
 orderDetailDAO.add(detail, conn);  
 productDAO.decreaseStock(detail.getProductId(), detail.getQuantity(), conn);  
 }  
   
 // Update table status  
 tableDAO.updateStatus(order.getTableId(), TableStatus.OCCUPIED, conn);  
   
 conn.commit();  
} catch (Exception e) {  
 if (conn != null) {  
 conn.rollback(); // Rollback on failure  
 }  
 throw new ServiceException("Failed to process order", e);  
}

## **19. Service Layer Business Logic**

### **19.1 Service Layer Principles**

The Service Layer provides a higher abstraction above DAO, encapsulating business rules, workflows, and transaction logic.

**Benefits:** - **Lightweight Controllers**: Controllers focus on UI interaction, not business logic - **Centralized Validation**: Business rules are enforced at the service layer - **Reusable Logic**: Business operations can be reused across different controllers - **Future Integration**: Easy integration with external APIs or cloud services - **Transaction Management**: Complex multi-step operations are handled atomically

### **19.2 Service Architecture**

**Key Services:**

* **UserService**: Authentication, role validation, user lifecycle management
* **MenuService**: Product CRUD, category management, stock validation
* **OrderService**: Order creation, modification, discount application
* **PaymentService**: Multi-method payments, refunds, receipt generation
* **CustomerService**: Loyalty points updates, purchase tracking
* **PromotionService**: Validation and application of discounts
* **TableService**: Table/area assignment, status transitions
* **ReportService**: Sales analytics, export to PDF/Excel

### **19.3 Business Logic Examples**

**Order Processing Workflow:** The OrderService implements a comprehensive order creation workflow that ensures data consistency and business rule enforcement. The process begins with order validation to ensure all required data is present and valid. The service then creates the order record and processes each order detail, updating inventory levels accordingly. The workflow includes table status updates to reflect the new order assignment. All operations are wrapped in a database transaction to ensure atomicity and data integrity.

**Promotion Application Logic:** The promotion application system provides flexible discount calculation capabilities supporting both percentage-based and fixed-amount discounts. The system validates promotion eligibility based on active status and order requirements before applying any discounts. The calculation engine ensures that discounts never result in negative order amounts, maintaining business rule compliance. The promotion system integrates seamlessly with the order processing workflow to provide real-time discount application.

### **19.4 Error Handling in Service Layer**

**Custom Exceptions:** - **ServiceException**: Business logic errors with meaningful messages - **DAOException**: Data access errors with technical details - **ValidationException**: Input validation errors with field-specific messages

**Logging Strategy:** - **SLF4J Integration**: Comprehensive logging for traceability - **Error Context**: Detailed error information for debugging - **Security Logging**: Sensitive operations are logged for audit purposes

**Graceful Error Handling:** - **User-Friendly Messages**: Errors are translated to user-friendly messages - **Rollback Protection**: Failed transactions are automatically rolled back - **Recovery Mechanisms**: System can recover from temporary failures

### **19.5 Integration with Controllers**

Controllers only call Service methods, never DAO directly:

**Benefits:** - **Clean MVC Separation**: Clear boundaries between presentation and business logic - **Validation at Service Layer**: Business rules are enforced before data access - **Maintainability**: Changes to business logic don’t affect controllers - **Testability**: Services can be tested independently of UI components

**Example (Controller → Service):**

@FXML  
private void handleLogin() {  
 try {  
 // Validate input  
 if (usernameField.getText().isEmpty() || passwordField.getText().isEmpty()) {  
 AlertUtils.showError("Please enter both username and password");  
 return;  
 }  
   
 // Authenticate user  
 User user = userService.authenticate(usernameField.getText(), passwordField.getText());  
 if (user != null) {  
 SessionManager.startSession(user);  
 navigateToDashboard(user);  
 } else {  
 AlertUtils.showError("Invalid credentials!");  
 }  
 } catch (ServiceException e) {  
 AlertUtils.showError("Login failed: " + e.getMessage());  
 logger.error("Login error for user: " + usernameField.getText(), e);  
 }  
}  
  
### \*\*19.6 Security Implementation Examples\*\*  
  
\*\*Password Hashing with BCrypt:\*\*  
```java  
public class SecurityUtils {  
 private static final int BCRYPT\_ROUNDS = 12;  
   
 public static String hashPassword(String plainPassword) {  
 return BCrypt.hashpw(plainPassword, BCrypt.gensalt(BCRYPT\_ROUNDS));  
 }  
   
 public static boolean verifyPassword(String plainPassword, String hashedPassword) {  
 return BCrypt.checkpw(plainPassword, hashedPassword);  
 }  
}

**SQL Injection Prevention:**

public class UserDAOImpl implements UserDAO {  
 @Override  
 public User findByUsername(String username) {  
 String query = "SELECT \* FROM users WHERE username = ? AND is\_active = true";  
 try (Connection conn = DatabaseConfig.getConnection();  
 PreparedStatement stmt = conn.prepareStatement(query)) {  
 stmt.setString(1, username); // Parameterized query prevents SQL injection  
 ResultSet rs = stmt.executeQuery();  
 if (rs.next()) {  
 return mapResultSetToUser(rs);  
 }  
 } catch (SQLException e) {  
 logger.error("Error finding user by username: " + username, e);  
 }  
 return null;  
 }  
}

**Role-Based Access Control:**

public class AuthorizationManager {  
 public static boolean hasPermission(User user, String permission) {  
 if (user == null || user.getRole() == null) {  
 return false;  
 }  
   
 switch (user.getRole()) {  
 case ADMIN:  
 return true; // Admin has all permissions  
 case MANAGER:  
 return MANAGER\_PERMISSIONS.contains(permission);  
 case STAFF:  
 return STAFF\_PERMISSIONS.contains(permission);  
 default:  
 return false;  
 }  
 }  
}

### **19.6 Advanced Business Logic Examples**

**Promotion Application with Complex Rules:**

public class PromotionService {  
 public double applyPromotion(Order order, Promotion promo) throws ServiceException {  
 // Validate promotion eligibility  
 if (!isPromotionEligible(order, promo)) {  
 throw new ServiceException("Promotion not eligible for this order");  
 }  
   
 // Calculate discount based on promotion type  
 double discount = 0.0;  
 switch (promo.getType()) {  
 case PERCENTAGE:  
 discount = order.getSubtotal() \* (promo.getValue() / 100.0);  
 break;  
 case FIXED\_AMOUNT:  
 discount = Math.min(promo.getValue(), order.getSubtotal());  
 break;  
 case BUY\_X\_GET\_Y:  
 discount = calculateBuyXGetYDiscount(order, promo);  
 break;  
 }  
   
 // Ensure discount doesn't exceed order total  
 return Math.max(order.getSubtotal() - discount, 0.0);  
 }  
   
 private boolean isPromotionEligible(Order order, Promotion promo) {  
 return promo.isActive() &&   
 order.getSubtotal() >= promo.getMinimumAmount() &&  
 !promo.isExpired() &&  
 promo.getUsageCount() < promo.getMaxUsage();  
 }  
}

**Inventory Management with Stock Validation:**

public class InventoryService {  
 public void processOrder(Order order) throws ServiceException {  
 try (Connection conn = DatabaseConfig.getConnection()) {  
 conn.setAutoCommit(false);  
   
 // Validate stock availability  
 for (OrderDetail detail : order.getOrderDetails()) {  
 Product product = productDAO.findById(detail.getProductId());  
 if (product.getStockQuantity() < detail.getQuantity()) {  
 throw new ServiceException("Insufficient stock for " + product.getName());  
 }  
 }  
   
 // Update inventory  
 for (OrderDetail detail : order.getOrderDetails()) {  
 productDAO.decreaseStock(detail.getProductId(), detail.getQuantity(), conn);  
 }  
   
 // Create order  
 orderDAO.create(order, conn);  
   
 conn.commit();  
 } catch (SQLException e) {  
 throw new ServiceException("Failed to process order", e);  
 }  
 }  
}

# **PART VI: USER INTERFACES & CONTROLLERS**

## **20. Controller Architecture**

### **20.1 MVC Pattern Implementation**

The Cafe Management System implements a comprehensive Model-View-Controller (MVC) architecture with JavaFX controllers managing user interactions and coordinating between the UI and business logic.

**Controller Responsibilities:** - **User Input Handling**: Capture and validate user interactions - **UI State Management**: Maintain interface state and synchronization - **Service Coordination**: Orchestrate business operations through service layer - **Navigation Control**: Manage screen transitions and routing - **Error Presentation**: Display user-friendly error messages

### **20.2 Controller Hierarchy**

**Base Controllers:** - **BaseController**: Common functionality for all controllers - **AdminBaseController**: Administrative controller base class - **DialogController**: Modal dialog management patterns

**Specialized Controllers:** - **Authentication Controllers**: Login and session management - **Business Process Controllers**: Order, payment, and table management - **Administrative Controllers**: User, menu, and system management - **Reporting Controllers**: Analytics and data export

### **20.3 Controller Lifecycle Management**

**Initialization Pattern:** The OrderPanelController follows a structured initialization pattern that ensures proper component setup and service integration. The controller extends BaseController to inherit common functionality and implements the JavaFX Initializable interface for proper lifecycle management. During initialization, the controller establishes service layer connections, configures UI components including data tables and event handlers, and loads initial data to populate the interface. This pattern ensures consistent controller behavior and proper resource management.

**Event Handling Patterns:** - **FXML Event Binding**: Direct method binding for simple interactions - **Custom Event Handlers**: Complex logic with validation and error handling - **Asynchronous Operations**: Background processing with UI updates

## **21. Main Application Controllers**

### **21.1 Authentication Controllers**

**LoginController Implementation:** - **Credential Validation**: Real-time input validation and formatting - **Authentication Flow**: Secure login process with session management - **Error Handling**: User-friendly error messages and retry mechanisms - **Remember Me**: Optional credential persistence for user convenience

**Session Management:** - **User Context**: Maintains user information throughout the session - **Role-Based Access**: Enforces permissions based on user roles - **Session Timeout**: Automatic logout after inactivity periods - **Security Logging**: Comprehensive audit trail for login attempts

### **21.2 Dashboard Controllers**

**DashboardController Features:** - **Navigation Management**: Centralized routing between application sections - **User Context Display**: Shows current user information and role - **Quick Actions**: Direct access to common operations - **Real-time Updates**: Live synchronization with backend data

**AdminDashboardController:** - **System Overview**: Key performance indicators and metrics - **Administrative Functions**: User management, system configuration - **Reporting Access**: Business intelligence and analytics - **System Monitoring**: Performance and health indicators

### **21.3 Business Process Controllers**

**OrderPanelController:** - **Order Creation**: Interactive order building with real-time validation - **Product Selection**: Integrated menu browsing and item addition - **Order Modification**: In-line editing and quantity management - **Payment Integration**: Seamless transition to payment processing

**OrderPanelController Implementation:**

public class OrderPanelController extends BaseController implements Initializable {  
 @FXML private TableView<OrderDetail> orderTable;  
 @FXML private TableView<Product> productTable;  
 @FXML private Label totalLabel;  
 @FXML private ComboBox<Table> tableComboBox;  
   
 private OrderService orderService;  
 private ProductService productService;  
 private ObservableList<OrderDetail> orderDetails;  
   
 @Override  
 public void initialize(URL url, ResourceBundle rb) {  
 orderService = new OrderService();  
 productService = new ProductService();  
 orderDetails = FXCollections.observableArrayList();  
   
 setupTables();  
 loadProducts();  
 loadTables();  
 setupEventHandlers();  
 }  
   
 @FXML  
 private void handleAddToOrder() {  
 Product selectedProduct = productTable.getSelectionModel().getSelectedItem();  
 if (selectedProduct != null) {  
 OrderDetail detail = new OrderDetail();  
 detail.setProduct(selectedProduct);  
 detail.setQuantity(1);  
 detail.setPrice(selectedProduct.getPrice());  
   
 orderDetails.add(detail);  
 updateTotal();  
 }  
 }  
   
 @FXML  
 private void handlePlaceOrder() {  
 if (orderDetails.isEmpty()) {  
 AlertUtils.showError("Please add items to the order");  
 return;  
 }  
   
 Table selectedTable = tableComboBox.getValue();  
 if (selectedTable == null) {  
 AlertUtils.showError("Please select a table");  
 return;  
 }  
   
 try {  
 Order order = new Order();  
 order.setTable(selectedTable);  
 order.setOrderDetails(orderDetails);  
 order.setTotalAmount(calculateTotal());  
   
 orderService.createOrder(order);  
 clearOrder();  
 AlertUtils.showInfo("Order placed successfully!");  
 } catch (ServiceException e) {  
 AlertUtils.showError("Failed to place order: " + e.getMessage());  
 }  
 }  
}

**TableController:** - **Visual Layout**: Interactive table status display and management - **Status Transitions**: Real-time table state updates - **Assignment Logic**: Intelligent table assignment and reservation - **Area Management**: Multi-zone table organization

**PaymentController:** - **Multi-Method Support**: Cash, card, and mobile payment processing - **Amount Validation**: Real-time payment amount verification - **Receipt Generation**: Automatic receipt creation and printing - **Transaction Security**: Secure payment processing with audit trails

**PaymentController Implementation:**

public class PaymentController extends BaseController {  
 @FXML private Label amountLabel;  
 @FXML private TextField cashAmountField;  
 @FXML private ComboBox<PaymentMethod> paymentMethodComboBox;  
 @FXML private Label changeLabel;  
   
 private Order currentOrder;  
 private PaymentService paymentService;  
   
 @FXML  
 private void handlePayment() {  
 PaymentMethod method = paymentMethodComboBox.getValue();  
 if (method == null) {  
 AlertUtils.showError("Please select a payment method");  
 return;  
 }  
   
 try {  
 Payment payment = new Payment();  
 payment.setOrder(currentOrder);  
 payment.setMethod(method);  
 payment.setAmount(currentOrder.getTotalAmount());  
   
 if (method == PaymentMethod.CASH) {  
 double cashAmount = Double.parseDouble(cashAmountField.getText());  
 if (cashAmount < currentOrder.getTotalAmount()) {  
 AlertUtils.showError("Cash amount must be greater than or equal to total");  
 return;  
 }  
 payment.setCashAmount(cashAmount);  
 }  
   
 paymentService.processPayment(payment);  
 generateReceipt(payment);  
 closePaymentWindow();  
 } catch (ServiceException e) {  
 AlertUtils.showError("Payment failed: " + e.getMessage());  
 }  
 }  
   
 @FXML  
 private void handleCashAmountChange() {  
 try {  
 double cashAmount = Double.parseDouble(cashAmountField.getText());  
 double change = cashAmount - currentOrder.getTotalAmount();  
 changeLabel.setText(String.format("Change: $%.2f", Math.max(0, change)));  
 } catch (NumberFormatException e) {  
 changeLabel.setText("Change: $0.00");  
 }  
 }  
}

### **21.4 Administrative Controllers**

**AdminMenuController:** - **Product Management**: Comprehensive CRUD operations for menu items - **Category Organization**: Hierarchical product categorization - **Inventory Control**: Stock level monitoring and management - **Image Management**: Product photo upload and optimization

**AdminUserController:** - **User Administration**: Staff account creation and management - **Role Assignment**: Permission-based access control - **Account Status**: User activation and deactivation - **Security Management**: Password policies and account security

**AdminReportController:** - **Report Generation**: Automated report creation and export - **Data Visualization**: Charts and graphs for business insights - **Export Functions**: PDF and Excel export capabilities - **Analytics Dashboard**: Real-time business intelligence

### **21.5 Controller Communication Patterns**

**Inter-Controller Communication:** - **Event Bus Pattern**: Decoupled communication between controllers - **Callback Mechanisms**: Direct method calls for tight integration - **Shared State Management**: Common data access through service layer - **Navigation Coordination**: Centralized screen transition management

**Service Integration:** - **Dependency Injection**: Service instances injected into controllers - **Error Propagation**: Consistent error handling across all controllers - **Transaction Coordination**: Service layer transaction management - **Data Synchronization**: Real-time data updates and UI refresh

# **PART VII: TECHNICAL IMPLEMENTATION**

## **22. Development Environment & Setup**

### **22.1 Prerequisites and System Requirements**

**Development Environment:** - **Java Development Kit (JDK) 21**: Latest LTS version with enhanced features - **Maven 3.8+**: Dependency management and build automation - **MySQL 8.0**: Database server with advanced features - **IDE Support**: IntelliJ IDEA, Eclipse, or Visual Studio Code

**System Requirements:** - **Operating System**: Windows 10+, macOS 10.15+, or Linux (Ubuntu 20.04+) - **Memory**: Minimum 8GB RAM, recommended 16GB for development - **Storage**: 10GB free space for development environment - **Network**: Internet connection for dependency downloads

### **22.2 Project Setup and Configuration**

**Repository Setup:** The project setup process begins with cloning the repository from the version control system. The setup includes installing all required dependencies through Maven’s dependency management system, which automatically downloads and configures all necessary libraries and frameworks. The database setup involves executing SQL scripts to create the database schema and populate it with initial sample data, ensuring the application has a proper foundation for development and testing.

**Configuration Files:** - **database\_config.properties**: Database connection parameters - **application.properties**: Application-specific settings - **logback.xml**: Logging configuration and levels

### **22.3 Development Workflow**

**Code Organization:** - **Feature Branches**: Separate branches for new features - **Pull Request Reviews**: Code review process for quality assurance - **Continuous Integration**: Automated build and testing - **Documentation Updates**: Synchronized documentation with code changes

**Testing Strategy:** - **Unit Tests**: Individual component testing with JUnit 5 - **Integration Tests**: Database and service layer testing - **UI Tests**: User interface automation testing - **Performance Tests**: Load and stress testing

## **23. Configuration Management**

### **23.1 Environment-Specific Configuration**

**Configuration Hierarchy:** - **Default Properties**: Base configuration values - **Environment Overrides**: Environment-specific modifications - **User Preferences**: Individual user customizations - **Runtime Configuration**: Dynamic configuration changes

**Configuration Sources:**

# Database Configuration  
db.url=jdbc:mysql://localhost:3306/cafe\_management  
db.username=cafe\_user  
db.password=${DB\_PASSWORD}  
db.pool.size=50  
db.pool.timeout=10  
  
# Application Configuration  
app.name=Cafe Management System  
app.version=1.0.0  
app.language=en  
app.timezone=Asia/Ho\_Chi\_Minh  
  
# Security Configuration  
security.password.min.length=8  
security.session.timeout=30  
security.max.login.attempts=5

### **23.2 Dynamic Configuration Management**

**Configuration Loading:** - **Property File Loading**: Hierarchical property file resolution - **Environment Variable Integration**: External configuration support - **Database Configuration**: Runtime configuration from database - **Hot Reload**: Configuration changes without application restart

**Configuration Manager Implementation:**

public class ConfigurationManager {  
 private static final Properties config = new Properties();  
 private static final String CONFIG\_FILE = "application.properties";  
 private static final Logger logger = LoggerFactory.getLogger(ConfigurationManager.class);  
   
 static {  
 loadConfiguration();  
 }  
   
 private static void loadConfiguration() {  
 try (InputStream input = ConfigurationManager.class.getClassLoader()  
 .getResourceAsStream(CONFIG\_FILE)) {  
 if (input != null) {  
 config.load(input);  
 // Override with environment variables  
 overrideWithEnvironmentVariables();  
 logger.info("Configuration loaded successfully");  
 }  
 } catch (IOException e) {  
 logger.error("Failed to load configuration", e);  
 }  
 }  
   
 private static void overrideWithEnvironmentVariables() {  
 // Database password from environment variable  
 String dbPassword = System.getenv("DB\_PASSWORD");  
 if (dbPassword != null) {  
 config.setProperty("db.password", dbPassword);  
 }  
   
 // Application environment  
 String appEnv = System.getenv("APP\_ENV");  
 if (appEnv != null) {  
 config.setProperty("app.environment", appEnv);  
 }  
 }  
   
 public static String getProperty(String key) {  
 return config.getProperty(key);  
 }  
   
 public static int getIntProperty(String key, int defaultValue) {  
 try {  
 return Integer.parseInt(config.getProperty(key));  
 } catch (NumberFormatException e) {  
 logger.warn("Invalid integer property: " + key + ", using default: " + defaultValue);  
 return defaultValue;  
 }  
 }  
   
 public static boolean getBooleanProperty(String key, boolean defaultValue) {  
 try {  
 return Boolean.parseBoolean(config.getProperty(key));  
 } catch (Exception e) {  
 logger.warn("Invalid boolean property: " + key + ", using default: " + defaultValue);  
 return defaultValue;  
 }  
 }  
}

**Configuration Validation:** - **Schema Validation**: Configuration structure validation - **Value Validation**: Configuration value range and format checking - **Dependency Validation**: Cross-configuration dependency verification - **Default Value Fallback**: Automatic fallback to default values

## **24. Error Handling & Logging**

### **24.1 Exception Handling Strategy**

**Exception Hierarchy:** - **ApplicationException**: Base exception for application errors - **ServiceException**: Business logic and service layer errors - **DAOException**: Data access and database errors - **ValidationException**: Input validation and format errors - **SecurityException**: Authentication and authorization errors

**Exception Handling Patterns:** The ExceptionHandler class implements a centralized exception management strategy that categorizes different types of exceptions and provides appropriate user feedback. The system distinguishes between business logic errors, database connectivity issues, and unexpected system errors, ensuring users receive meaningful error messages while technical details are logged for debugging purposes. This pattern maintains a clean separation between user-facing error messages and technical error information.

**Exception Handler Implementation:**

public class ExceptionHandler {  
 private static final Logger logger = LoggerFactory.getLogger(ExceptionHandler.class);  
   
 public static void handleException(Exception e, String context) {  
 if (e instanceof ServiceException) {  
 handleServiceException((ServiceException) e, context);  
 } else if (e instanceof DAOException) {  
 handleDAOException((DAOException) e, context);  
 } else if (e instanceof ValidationException) {  
 handleValidationException((ValidationException) e, context);  
 } else {  
 handleUnexpectedException(e, context);  
 }  
 }  
   
 private static void handleServiceException(ServiceException e, String context) {  
 logger.error("Service error in " + context + ": " + e.getMessage(), e);  
 AlertUtils.showError("Operation failed: " + e.getMessage());  
 }  
   
 private static void handleDAOException(DAOException e, String context) {  
 logger.error("Database error in " + context + ": " + e.getMessage(), e);  
 AlertUtils.showError("Database operation failed. Please try again.");  
 }  
   
 private static void handleValidationException(ValidationException e, String context) {  
 logger.warn("Validation error in " + context + ": " + e.getMessage());  
 AlertUtils.showError("Invalid input: " + e.getMessage());  
 }  
   
 private static void handleUnexpectedException(Exception e, String context) {  
 logger.error("Unexpected error in " + context + ": " + e.getMessage(), e);  
 AlertUtils.showError("An unexpected error occurred. Please contact support.");  
 }  
}

### **24.2 Logging Framework**

**Logging Configuration:** The logging framework utilizes Logback with a comprehensive configuration that includes file-based logging with automatic log rotation. The system implements time-based log file management, automatically creating new log files daily and maintaining a 30-day history of log files. The logging pattern includes timestamps, thread information, log levels, and logger names to provide comprehensive debugging information while maintaining efficient log file management.

**Logging Levels:** - **ERROR**: System errors requiring immediate attention - **WARN**: Potential issues that should be monitored - **INFO**: General application flow and important events - **DEBUG**: Detailed debugging information - **TRACE**: Very detailed debugging information

### **24.3 Error Recovery and Resilience**

**Error Recovery Mechanisms:** - **Automatic Retry**: Transient error recovery with exponential backoff - **Circuit Breaker**: Protection against cascading failures - **Graceful Degradation**: Reduced functionality during partial failures - **Data Recovery**: Automatic data restoration and consistency checks

**Monitoring and Alerting:** - **Error Tracking**: Comprehensive error monitoring and analysis - **Performance Monitoring**: System performance and resource usage - **User Experience Monitoring**: Error impact on user experience - **Proactive Alerting**: Early warning systems for potential issues

## **25. Testing Strategy**

### **25.1 Testing Pyramid**

**Unit Testing (70%):** - **Service Layer Tests**: Business logic validation - **Utility Function Tests**: Helper method verification - **Model Tests**: Entity and DTO validation - **Configuration Tests**: Configuration loading and validation

**Integration Testing (20%):** - **Database Integration**: DAO layer and transaction testing - **Service Integration**: Cross-service interaction testing - **External Service Integration**: Payment and notification testing - **UI Integration**: Controller and FXML integration testing

**End-to-End Testing (10%):** - **User Workflow Testing**: Complete user journey validation - **System Integration**: Full system integration testing - **Performance Testing**: Load and stress testing - **Security Testing**: Authentication and authorization testing

### **25.2 Test Implementation**

**Unit Test Example:** The unit test for order creation demonstrates the Arrange-Act-Assert pattern, where test data is prepared, the service method is executed, and the results are verified. The test validates that order creation properly assigns an order ID, sets the correct initial status, and interacts correctly with the data access layer. This approach ensures that individual components function correctly in isolation.

**Integration Test Example:** The integration test for payment processing validates the complete payment workflow from request to response. The test verifies that payment processing returns successful results with appropriate status codes and transaction identifiers. This type of test ensures that different system components work together correctly and that the complete business process functions as expected.

### **25.3 Test Automation**

**Continuous Integration:** - **Automated Build**: Maven build automation - **Test Execution**: Automated test suite execution - **Code Coverage**: Test coverage reporting and monitoring - **Quality Gates**: Automated quality checks and validation

**Test Data Management:** - **Test Database**: Isolated test database environment - **Data Seeding**: Automated test data creation - **Data Cleanup**: Automatic test data cleanup - **Data Isolation**: Test data isolation and independence

# **PART VIII: DEPLOYMENT & MAINTENANCE**

## **26. Build & Deployment**

### **26.1 Build Configuration**

**Maven Build Configuration:** The Maven build configuration includes three essential plugins for the JavaFX application. The compiler plugin ensures Java 21 compatibility with proper source and target version settings. The JavaFX plugin provides JavaFX-specific build capabilities and runtime support. The shade plugin creates a self-contained executable JAR file that includes all dependencies, making the application portable and easy to distribute. This configuration ensures consistent builds across different environments and simplifies deployment.

**Build Profiles:** - **Development**: Debug mode with enhanced logging - **Production**: Optimized performance with minimal logging - **Testing**: Test-specific configurations and dependencies - **Distribution**: Self-contained executable JAR creation

### **26.2 Deployment Strategies**

**Desktop Application Deployment:** - **JAR Distribution**: Self-contained executable JAR files - **Installation Scripts**: Automated installation and setup - **Dependency Management**: Bundled dependencies and libraries - **Platform-Specific Packaging**: Windows, macOS, and Linux packages

**Database Deployment:** - **Schema Migration**: Automated database schema updates - **Data Migration**: Data transformation and migration scripts - **Backup Procedures**: Automated backup and recovery procedures - **Environment Setup**: Database environment configuration

**Deployment Automation:**

#!/bin/bash  
# Deployment script  
  
# Build application  
mvn clean package -P production  
  
# Deploy database  
mysql -u root -p < database/migration.sql  
  
# Copy application files  
cp target/cafe-management-system.jar /opt/cafe/  
cp config/application.properties /opt/cafe/  
  
# Set permissions  
chmod +x /opt/cafe/cafe-management-system.jar  
  
# Start application  
java -jar /opt/cafe/cafe-management-system.jar

### **26.3 Environment Management**

**Environment Configuration:** - **Development Environment**: Local development setup - **Testing Environment**: Isolated testing environment - **Staging Environment**: Pre-production validation - **Production Environment**: Live production deployment

**Configuration Management:** - **Environment Variables**: External configuration management - **Configuration Files**: Environment-specific configuration files - **Database Configuration**: Environment-specific database settings - **Logging Configuration**: Environment-specific logging levels

## **27. Performance Optimization**

### **27.1 Application Performance**

**JavaFX Performance Optimization:** - **UI Thread Management**: Proper thread handling for UI updates - **Event Handling**: Efficient event processing and handling - **Memory Management**: Proper object lifecycle management - **Rendering Optimization**: Efficient UI rendering and updates

**Database Performance:** - **Query Optimization**: Optimized SQL queries and indexing - **Connection Pooling**: Efficient database connection management - **Caching Strategy**: Application-level caching for frequently accessed data - **Batch Processing**: Efficient bulk data operations

**Memory Management:** - **Object Pooling**: Reuse of expensive objects - **Garbage Collection**: Optimized garbage collection settings - **Memory Leak Prevention**: Proper resource cleanup and disposal - **Memory Monitoring**: Real-time memory usage monitoring

### **27.2 Scalability Considerations**

**Horizontal Scaling:** - **Load Balancing**: Multiple application instances - **Database Clustering**: Database replication and clustering - **Caching Layer**: Distributed caching for improved performance - **Microservices Architecture**: Future migration to microservices

**Vertical Scaling:** - **Resource Allocation**: Optimal resource allocation and utilization - **Performance Tuning**: Application and database performance tuning - **Hardware Optimization**: Hardware resource optimization - **Capacity Planning**: Future capacity planning and scaling

### **27.3 Monitoring and Metrics**

**Performance Monitoring:** - **Application Metrics**: Response time, throughput, and error rates - **Database Metrics**: Query performance and connection usage - **System Metrics**: CPU, memory, and disk usage - **User Experience Metrics**: User interaction and satisfaction metrics

**Performance Optimization Tools:** - **Profiling Tools**: Application profiling and analysis - **Monitoring Dashboards**: Real-time performance monitoring - **Alerting Systems**: Performance threshold alerting - **Performance Testing**: Load and stress testing tools

## **28. Maintenance & Monitoring**

### **28.1 System Maintenance**

**Regular Maintenance Tasks:** - **Database Maintenance**: Regular database optimization and cleanup - **Log Management**: Log rotation and archival - **Backup Verification**: Regular backup testing and verification - **Security Updates**: Regular security patches and updates

**Preventive Maintenance:** - **Health Checks**: Regular system health monitoring - **Performance Monitoring**: Continuous performance monitoring - **Capacity Planning**: Proactive capacity planning and scaling - **Disaster Recovery**: Regular disaster recovery testing

### **28.2 Monitoring and Alerting**

**System Monitoring:** - **Application Monitoring**: Real-time application health monitoring - **Database Monitoring**: Database performance and health monitoring - **Infrastructure Monitoring**: Server and network monitoring - **User Experience Monitoring**: User interaction and satisfaction monitoring

**Alerting Systems:** - **Critical Alerts**: Immediate notification for critical issues - **Warning Alerts**: Proactive notification for potential issues - **Performance Alerts**: Performance threshold alerting - **Security Alerts**: Security incident alerting

**Monitoring Tools:** - **Application Performance Monitoring (APM)**: Detailed application performance analysis - **Database Monitoring**: Database performance and health monitoring - **Infrastructure Monitoring**: Server and network monitoring - **Log Management**: Centralized log collection and analysis

### **28.3 Troubleshooting and Support**

**Troubleshooting Procedures:** - **Issue Identification**: Systematic issue identification and classification - **Root Cause Analysis**: Comprehensive root cause analysis - **Resolution Procedures**: Standardized resolution procedures - **Documentation**: Comprehensive troubleshooting documentation

**Support Procedures:** - **User Support**: User assistance and problem resolution - **Technical Support**: Technical issue resolution and support - **Escalation Procedures**: Issue escalation and management - **Knowledge Base**: Comprehensive knowledge base and documentation

# **PART IX: APPENDICES**

## **29. API Documentation**

### **29.1 Service Layer API**

**UserService API:** The UserService interface provides comprehensive user management capabilities including authentication, user creation and modification, account deactivation, and role-based user retrieval. The service includes security features such as password validation and role-based access control, ensuring proper user lifecycle management throughout the application.

**OrderService API:** The OrderService interface defines the complete order management workflow including order creation, modification, and status tracking. The service supports complex order operations such as adding and removing products from orders, managing order status transitions, and retrieving orders based on various criteria including status and table assignments.

**PaymentService API:** The PaymentService interface handles all payment-related operations including payment processing, refund management, payment history tracking, and payment method validation. The service supports multiple payment methods and provides comprehensive transaction management with proper error handling and audit trail capabilities.

### **29.2 Exception Handling API**

**ServiceException:** The ServiceException class extends the standard Exception class to provide enhanced error handling capabilities for business logic operations. The class includes additional fields for error codes and user-friendly messages, allowing for better error categorization and user communication. The exception supports multiple constructor patterns for different error scenarios and maintains backward compatibility with standard exception handling.

**ValidationException:** The ValidationException class specializes in handling input validation errors by extending ServiceException and adding field-specific error tracking. The class maintains a map of field names to error messages, enabling detailed validation feedback for form inputs and user data. This approach provides granular error reporting while maintaining the overall exception hierarchy structure.

## **30. Database Schema Reference**

### **30.1 Complete Database Schema**

**Users Table:** The users table serves as the central repository for all system user information including staff members and administrators. The table includes comprehensive user profile data with security features such as encrypted password storage, role-based access control, and account status management. The table implements proper data integrity constraints and includes audit trail fields for tracking user account lifecycle.

**Products Table:** The products table manages the complete product catalog with detailed product information including pricing, categorization, and inventory tracking. The table supports product lifecycle management with availability and active status flags, cost tracking for profit margin calculations, and comprehensive product descriptions. The table maintains referential integrity with the categories table for proper product organization.

**Orders Table:** The orders table represents the core business entity for customer orders with comprehensive order tracking and financial management. The table includes detailed order information such as customer and table associations, financial calculations including discounts and final amounts, payment processing status, and order lifecycle tracking. The table maintains multiple foreign key relationships to ensure data consistency across the system.

### **30.2 Database Views**

**Sales Report View:** The sales report view provides comprehensive business intelligence by aggregating order data into daily sales metrics. The view calculates key performance indicators including total orders, revenue, average order value, and unique customer counts for each day. The view filters for completed orders to ensure accurate financial reporting and includes proper date-based grouping for trend analysis.

**Menu Items View:** The menu items view creates a unified interface for product display by joining product and category information. The view filters for active products and categories to ensure only available items are displayed, includes comprehensive product details such as pricing and availability status, and maintains proper ordering based on category display preferences and product names.

### **30.3 Indexes and Performance**

**Primary Indexes:** The database implements strategic single-column indexes on frequently queried fields to optimize query performance. These indexes target critical business operations such as user role filtering, product availability checks, order date-based queries, and customer phone lookups. The indexing strategy ensures fast response times for common operations while maintaining efficient storage utilization.

**Composite Indexes:** The system employs multi-column indexes for complex queries that involve multiple search criteria. These composite indexes optimize performance for operations such as date and status-based order filtering, category and availability-based product searches, and order detail lookups. The composite indexing strategy supports efficient execution of business intelligence queries and reporting operations.

## **31. Configuration Reference**

### **31.1 Application Configuration**

**Database Configuration:** The database configuration section defines comprehensive connection parameters including the database URL with SSL and timezone settings, user credentials with environment variable support for security, and the MySQL JDBC driver specification. The connection pool configuration optimizes database performance with appropriate pool sizing, timeout settings, and prepared statement caching to ensure efficient resource utilization and connection management.

**Application Configuration:** The application configuration encompasses system-wide settings including application identification, localization preferences with Vietnamese timezone and currency settings, user interface customization options, and comprehensive security parameters. The configuration includes password policies, session management settings, and account protection mechanisms to ensure secure system operation.

### **31.2 Logging Configuration**

**Logback Configuration:** The Logback configuration implements a comprehensive logging strategy with multiple appenders for different output destinations. The console appender provides real-time logging during development, while file appenders create persistent logs with automatic rotation and size management. The error file appender specifically captures error-level messages for focused troubleshooting. The configuration includes application-specific logger settings for detailed debugging and SQL query monitoring.

## **32. Troubleshooting Guide**

### **32.1 Common Issues and Solutions**

**Database Connection Issues:** Database connection problems typically stem from server availability, credential configuration, or network connectivity issues. The troubleshooting process involves verifying the database server status, checking connection parameters in the configuration files, ensuring proper database existence and accessibility, and validating network connectivity and firewall settings that might block database connections.

**Database Health Check Implementation:**

public class DatabaseHealthCheck {  
 private static final Logger logger = LoggerFactory.getLogger(DatabaseHealthCheck.class);  
   
 public static boolean checkConnection() {  
 try (Connection conn = DatabaseConfig.getConnection()) {  
 return conn.isValid(5); // 5 second timeout  
 } catch (SQLException e) {  
 logger.error("Database connection failed", e);  
 return false;  
 }  
 }  
   
 public static void diagnoseConnectionIssues() {  
 // Check if MySQL service is running  
 if (!isMySQLServiceRunning()) {  
 logger.error("MySQL service is not running");  
 return;  
 }  
   
 // Check connection parameters  
 String url = ConfigurationManager.getProperty("db.url");  
 String username = ConfigurationManager.getProperty("db.username");  
   
 logger.info("Testing connection to: " + url);  
 logger.info("Username: " + username);  
   
 // Test connection with timeout  
 try (Connection conn = DriverManager.getConnection(url, username,   
 ConfigurationManager.getProperty("db.password"))) {  
 logger.info("Database connection successful");  
 } catch (SQLException e) {  
 logger.error("Database connection failed: " + e.getMessage());  
 }  
 }  
   
 private static boolean isMySQLServiceRunning() {  
 try {  
 Process process = Runtime.getRuntime().exec("mysqladmin ping -h localhost");  
 return process.waitFor() == 0;  
 } catch (Exception e) {  
 logger.warn("Could not check MySQL service status", e);  
 return false;  
 }  
 }  
}

**JavaFX Runtime Issues:** JavaFX runtime problems often occur due to missing SDK components or incompatible Java versions. The resolution process includes verifying proper JavaFX SDK installation, checking Maven dependency configurations, validating module path settings, ensuring Java version compatibility, and reinstalling the JavaFX SDK if necessary to resolve component conflicts.

**JavaFX Module Path Configuration:**

# Windows  
set JAVA\_OPTS=--module-path "lib/javafx-sdk-24.0.2/lib" --add-modules javafx.controls,javafx.fxml  
  
# Linux/macOS  
export JAVA\_OPTS="--module-path lib/javafx-sdk-24.0.2/lib --add-modules javafx.controls,javafx.fxml"  
  
# Maven configuration  
mvn clean compile exec:java -Dexec.mainClass="com.cafe.CafeManagementApplication" \  
 -Dexec.args="--module-path lib/javafx-sdk-24.0.2/lib --add-modules javafx.controls,javafx.fxml"

**JavaFX Application Launcher:**

public class JavaFXLauncher {  
 public static void main(String[] args) {  
 try {  
 // Check JavaFX modules  
 ModuleLayer.boot().findModule("javafx.controls")  
 .orElseThrow(() -> new RuntimeException("JavaFX Controls module not found"));  
   
 // Launch application  
 CafeManagementApplication.main(args);  
 } catch (Exception e) {  
 System.err.println("JavaFX Runtime Error: " + e.getMessage());  
 System.err.println("Please ensure JavaFX SDK is properly installed and configured");  
 System.exit(1);  
 }  
 }  
}

**Memory Issues:** Memory-related problems manifest as OutOfMemoryError exceptions or degraded performance. The solution involves increasing JVM heap allocation, implementing memory monitoring and profiling tools, identifying and resolving memory leaks in the application code, optimizing database query performance, and implementing proper connection pooling to manage resource utilization.

**Memory Monitoring and Optimization:**

public class MemoryMonitor {  
 private static final Logger logger = LoggerFactory.getLogger(MemoryMonitor.class);  
   
 public static void logMemoryUsage() {  
 Runtime runtime = Runtime.getRuntime();  
 long totalMemory = runtime.totalMemory();  
 long freeMemory = runtime.freeMemory();  
 long usedMemory = totalMemory - freeMemory;  
 long maxMemory = runtime.maxMemory();  
   
 logger.info("Memory Usage:");  
 logger.info(" Used: " + formatBytes(usedMemory));  
 logger.info(" Free: " + formatBytes(freeMemory));  
 logger.info(" Total: " + formatBytes(totalMemory));  
 logger.info(" Max: " + formatBytes(maxMemory));  
 logger.info(" Usage: " + String.format("%.2f%%",   
 (double) usedMemory / maxMemory \* 100));  
 }  
   
 public static void checkMemoryLeaks() {  
 // Force garbage collection  
 System.gc();  
   
 // Check for memory leaks in connection pools  
 if (DatabaseConfig.getConnectionPool() != null) {  
 int activeConnections = DatabaseConfig.getConnectionPool().getActiveConnections();  
 logger.info("Active database connections: " + activeConnections);  
 }  
   
 // Check for memory leaks in cached data  
 if (ProductService.getCache() != null) {  
 int cachedProducts = ProductService.getCache().size();  
 logger.info("Cached products: " + cachedProducts);  
 }  
 }  
   
 public static void optimizeMemory() {  
 // Clear unnecessary caches  
 ProductService.clearCache();  
   
 // Force garbage collection  
 System.gc();  
   
 // Log memory usage after optimization  
 logMemoryUsage();  
 }  
   
 private static String formatBytes(long bytes) {  
 if (bytes < 1024) return bytes + " B";  
 int exp = (int) (Math.log(bytes) / Math.log(1024));  
 String pre = "KMGTPE".charAt(exp-1) + "";  
 return String.format("%.1f %sB", bytes / Math.pow(1024, exp), pre);  
 }  
}

### **32.2 Performance Troubleshooting**

**Slow Application Startup:** - **Database Connection**: Check database connection timeouts - **Resource Loading**: Optimize resource loading and caching - **Dependency Resolution**: Verify Maven dependency resolution - **JVM Startup**: Optimize JVM startup parameters

**Slow Database Queries:** - **Query Optimization**: Analyze and optimize slow queries - **Index Usage**: Verify proper index usage - **Connection Pooling**: Check connection pool configuration - **Database Statistics**: Update database statistics

**UI Performance Issues:** - **Event Handling**: Optimize event handling and processing - **UI Thread**: Ensure proper UI thread management - **Memory Usage**: Monitor and optimize memory usage - **Rendering**: Optimize UI rendering and updates

### **32.3 Debugging Techniques**

**Logging and Debugging:** The logging and debugging approach involves enabling detailed debug logging for specific operations, implementing performance timing measurements to identify bottlenecks, and providing comprehensive exception handling with contextual information. The debugging strategy includes structured logging with parameter substitution, performance monitoring with millisecond precision, and detailed error reporting with stack trace information for effective problem resolution.

**Database Debugging:** Database debugging techniques include enabling comprehensive query logging to capture all database operations, analyzing slow query logs to identify performance bottlenecks, and monitoring active database connections to detect connection pool issues. The debugging process involves examining query execution patterns, connection utilization, and performance metrics to optimize database operations and resolve connectivity problems.

**Performance Monitoring:** - **JVM Monitoring**: Use JVM monitoring tools (JConsole, VisualVM) - **Database Monitoring**: Monitor database performance and connections - **Application Monitoring**: Use application performance monitoring tools - **System Monitoring**: Monitor system resources and performance

### **32.4 Recovery Procedures**

**Database Recovery:** Database recovery procedures involve restoring data from backup files using MySQL import commands and implementing point-in-time recovery using binary log analysis. The recovery process includes comprehensive backup restoration with proper credential authentication and time-based recovery procedures that allow restoration to specific timestamps for data consistency and integrity.

**Application Recovery:** Application recovery procedures include restarting the application service using system control commands, verifying application status and health, and monitoring application logs for error detection and resolution. The recovery process involves service management, status verification, and continuous log monitoring to ensure proper application operation and identify potential issues.

**Data Recovery:** - **Backup Restoration**: Restore from latest backup - **Transaction Recovery**: Recover incomplete transactions - **Data Validation**: Validate data integrity after recovery - **Service Restoration**: Restore affected services

## **CONCLUSION**

This comprehensive technical documentation provides a complete guide to the Cafe Management System, covering all aspects from system architecture and design to implementation details and maintenance procedures. The system demonstrates modern software development practices with robust architecture, comprehensive testing, and scalable design patterns.

**Key Achievements:** - **Comprehensive Functionality**: Complete cafe management capabilities - **Modern Architecture**: Clean separation of concerns with MVC pattern - **Security Implementation**: Robust authentication and authorization - **Performance Optimization**: Efficient database and application performance - **Maintainable Code**: Well-structured and documented codebase - **Scalable Design**: Architecture supporting future growth and enhancements

**Future Enhancements:** - **Cloud Integration**: Migration to cloud-based deployment - **Mobile Application**: Companion mobile app for customers - **Advanced Analytics**: Machine learning and predictive analytics - **Multi-location Support**: Support for multiple cafe locations - **API Development**: RESTful API for external integrations

The Cafe Management System represents a modern, enterprise-grade solution for cafe operations, providing the foundation for efficient business management and customer service excellence.