**Online Cafe Ordering System - MCO 1**

**Product Requirements Document**

**Tech Stack**: Django + SQLite + Bootstrap 5  
**Focus**: Authentication Security & Data Encryption

**1. Project Overview**

A web-based cafe ordering system demonstrating real-world implementation of encryption and decryption techniques for securing user data. MCO 1 focuses on secure authentication and profile management, with complete UI design for the ordering system to be implemented in MCO 2.

**1.1 Project Objectives**

* Implement secure user authentication with password hashing
* Encrypt sensitive user data (email, phone) in database
* Protect against common web vulnerabilities
* Design complete system UI/UX for future implementation

**2. Scope Definition**

**2.1 Full Implementation (Backend + Frontend)**

| **Feature** | **What's Implemented** |
| --- | --- |
| **Landing Page** | Static homepage with hero section and features |
| **Sign Up** | User registration with encrypted data storage |
| **Sign In** | Authentication with rate limiting and session management |
| **Profile Management** | View/edit username, change password, delete account |
| **Menu Display** | Browse cafe menu items from database |

**2.2 Frontend Only (UI Design, No Backend Logic)**

| **Feature** | **What's Designed** |
| --- | --- |
| **Shopping Cart** | Cart UI with mock data and interactions |
| **Checkout** | Checkout flow UI with order type selection |
| **Payment** | Payment form UI with card input formatting |
| **Order Confirmation** | Success page with mock order details |
| **Order History** | Order list UI with mock historical data |

**Note**: Cart, checkout, payment, and order features will be fully implemented in MCO 2.

**3. Security Requirements (MCO 1 Focus)**

**3.1 Encryption Implementations**

**Password Security**

* **Algorithm**: Argon2 (Django default)
* **Implementation**: One-way hashing, cannot be reversed
* **Storage**: Hashed passwords in database
* **Validation**: Minimum 8 characters, uppercase, number, special character

**Symmetric Encryption (AES-256)**

* **Email addresses**: Encrypted at rest in database
* **Phone numbers**: Encrypted at rest in database
* **Algorithm**: AES-256-CBC
* **Key Management**: Derived from Django SECRET\_KEY

**Session Security**

* **Session Storage**: Encrypted session data
* **Cookies**: HttpOnly, Secure, SameSite=Strict flags
* **Timeout**: 30 minutes inactivity
* **Token Generation**: SHA-256 hashed session IDs

**3.2 Authentication Security**

**Features Implemented:**

* Password strength validation with real-time feedback
* Rate limiting: 5 login attempts per 15 minutes
* Account lockout after failed attempts
* CSRF protection on all forms
* Secure session management
* Login activity tracking (timestamp, IP)

**3.3 Web Application Security (OWASP Protection)**

| **Vulnerability** | **Protection Method** |
| --- | --- |
| **SQL Injection** | Django ORM parameterized queries |
| **XSS** | Django template auto-escaping |
| **CSRF** | Django CSRF tokens |
| **Broken Authentication** | Argon2 hashing, rate limiting, session security |
| **Sensitive Data Exposure** | AES-256 encryption, HTTPS enforcement |
| **Broken Access Control** | Django permissions, user ownership validation |

**Security Headers:**

* Strict-Transport-Security (HSTS)
* X-Content-Type-Options: nosniff
* X-Frame-Options: DENY
* Content-Security-Policy

**4. Features Specification**

**4.1 Landing Page**

**Design**: Bootstrap hero section with features overview

| **Section** | **Content** |
| --- | --- |
| Navigation Bar | Logo, Menu, Sign In, Sign Up buttons |
| Hero Section | Cafe image, tagline, "Order Now" CTA |
| Features Grid | 4 feature cards: Online Ordering, Secure Payments, Easy Tracking, Quality Coffee |
| How It Works | 3-step visual: Browse → Order → Enjoy |
| Footer | Contact info, privacy policy, terms |

**Implementation**: Static HTML with Bootstrap styling

**4.2 Sign Up Page (Full Implementation)**

**Security Priority**: Demonstrate encryption at registration

| **Component** | **Implementation** |
| --- | --- |
| **Form Fields** | • Full Name  • Email (validated)  • Phone Number  • Password  • Confirm Password |
| **Password Validation** | Real-time strength indicator:   * Min 8 characters * 1 Uppercase letter * 1 Number * 1 Special character |
| **Security Features** | • CSRF token  • Server-side validation  • Email format check  • Password match verification  • Rate limiting (5 signups/hour per IP) |
| **Backend Process** | 1. Validate input  2. Hash password with Argon2  3. Encrypt email with AES-256  4. Encrypt phone with AES-256  5. Save to database  6. Create session  7. Redirect to menu |
| **Error Handling** | • "Email already exists"  • "Passwords don't match"  • Clear inline error messages |

**Database Storage:**

* Username: Plain text
* Password: Hashed with Argon2
* Email: Encrypted with AES-256
* Phone: Encrypted with AES-256

**4.3 Sign In Page (Full Implementation)**

**Security Priority**: Secure authentication with brute force protection

| **Component** | **Implementation** |
| --- | --- |
| **Form Fields** | • Email or Username  • Password  • Remember Me (optional) |
| **Security Features** | • Rate limiting (5 attempts/15 min per IP)  • Account lockout after 5 failed attempts  • CSRF protection  • Session creation with encryption  • Login event logging |
| **Session Management** | • HttpOnly cookies (prevent JavaScript access)  • Secure flag (HTTPS only)  • SameSite=Strict (CSRF protection)  • SHA-256 session ID  • 30-minute timeout |
| **Backend Process** | 1. Validate input  2. Check rate limit  3. Verify credentials  4. Check account not locked  5. Create encrypted session  6. Log login event  7. Redirect to menu |
| **Error Messages** | Generic "Invalid credentials" (prevents user enumeration) |
| **Forgot Password** | Link to password reset (UI only for MCO 1) |

**Login Event Logging:**

* User ID
* Event type (login success/failure)
* IP address
* Timestamp

**4.4 Profile Page (Full Implementation)**

**Security Priority**: Demonstrate data decryption and account management

| **Section** | **Implementation** |
| --- | --- |
| **Profile Information** | Display:  • Username  • Email (decrypted from database)  • Phone (decrypted from database)  • Member since date  • Last login timestamp |
| **Change Username** | • Input field with validation  • Username uniqueness check  • Update database  • Success message |
| **Change Password** | • Current password verification  • New password with strength validator  • Confirm new password  • Re-hash with Argon2  • Update database  • Invalidate old sessions |
| **Delete Account** | • Confirmation modal with warning  • Password verification required  • Permanent deletion from database  • Session invalidation  • GDPR compliance |
| **Security Info** | Display:  • Last login date/time  • Account created date  • Security tips |

**Access Control**: Users can only view/edit their own profile

**4.5 Menu Display (Full Implementation)**

**Design**: Bootstrap card grid layout

| **Component** | **Implementation** |
| --- | --- |
| **Menu Items** | Query from database:  • Item name  • Description  • Price  • Category  • Image  • Availability status |
| **Category Filter** | Tabs: All, Coffee, Tea, Pastries, Snacks |
| **Item Card** | Bootstrap card showing item details |
| **Add to Cart Button** | Button present (frontend click handler only for MCO 1) |
| **Security** | • SQL injection prevention via Django ORM  • XSS prevention via template escaping  • Public access (no login required) |

**4.6 Shopping Cart (Frontend Only)**

**Design**: Cart page with item list and summary

**UI Components:**

* Cart item cards (mock data)
* Quantity controls (+/- buttons)
* Remove item button
* Price calculations (client-side)
* Subtotal, tax, total display
* "Proceed to Checkout" button

**Frontend Interactions:**

* Update quantity (no backend save)
* Remove items (no backend save)
* Calculate totals (JavaScript only)
* Navigate to checkout

**Note**: Backend implementation in MCO 2 will include session storage with encrypted cart data

**4.7 Checkout (Frontend Only)**

**Design**: Multi-step checkout form

**UI Components:**

* Order type selection (Pick Up / Dine In)
* Contact information (pre-filled if logged in)
* Delivery address field
* Special instructions textarea
* Order summary review
* "Continue to Payment" button

**Frontend Interactions:**

* Radio button selection
* Form validation (client-side)
* Navigate to payment page

**Note**: Backend implementation in MCO 2 will include order creation with encrypted customer data

**4.8 Payment (Frontend Only)**

**Design**: Payment form with card input

**UI Components:**

* Card number input (formatted with spaces)
* Expiry date (MM/YY format)
* CVV input
* Cardholder name
* Payment method tabs (Credit Card, GCash, Cash)
* Order total display
* "Place Order" button
* Security note: "This is a UI demonstration"

**Frontend Interactions:**

* Card number formatting (spaces every 4 digits)
* Form validation (client-side)
* Payment method switching
* Show success alert

**Note**: Backend implementation in MCO 2 will include payment data encryption with AES-256-GCM

**4.9 Order Confirmation (Frontend Only)**

**Design**: Success page with mock order details

**UI Components:**

* Success checkmark icon
* "Order Confirmed!" heading
* Mock order number
* Order details (type, time, total)
* Items list
* Email/SMS confirmation message
* "View Order History" button
* "Order Again" button

**Note**: Backend implementation in MCO 2 will include real order creation and confirmation emails

**4.10 Order History (Frontend Only)**

**Design**: List of past orders

**UI Components:**

* Order cards with mock data
* Order number, date, status badge
* Item summary and total
* "View Details" / "Reorder" buttons
* Empty state: "No orders yet"

**Status Badges**: Pending (orange), Completed (green)

**Note**: Backend implementation in MCO 2 will query real orders from encrypted database

**5. Technology Stack**

**5.1 Backend**

* **Framework**: Django 5.0+
* **Database**: SQLite 3
* **Python Version**: 3.10+

**5.2 Frontend**

* **CSS Framework**: Bootstrap 5.3
* **JavaScript**: Vanilla JS (no frameworks)
* **Template Engine**: Django Templates

**5.3 Security Libraries**

* **cryptography**: AES encryption implementation
* **django-encrypted-model-fields**: Encrypted database fields
* **django-ratelimit**: Rate limiting for authentication
* **argon2-cffi**: Password hashing
* **python-dotenv**: Environment variables management

**5.4 Development Tools**

* **django-debug-toolbar**: Development debugging
* **pytest-django**: Testing framework
* **coverage**: Code coverage analysis

**6. Security Implementation Checklist**

**6.1 Encryption**

* Password hashing with Argon2
* Email encryption with AES-256
* Phone encryption with AES-256
* Session data encryption
* SHA-256 session IDs

**6.2 Authentication**

* Secure user registration
* Password strength validation
* Rate limiting on login (5 attempts/15 min)
* Account lockout mechanism
* Secure session management
* HttpOnly, Secure, SameSite cookies
* Session timeout (30 minutes)
* Login event logging

**6.3 Web Security**

* CSRF protection (Django built-in)
* SQL injection prevention (Django ORM)
* XSS prevention (template auto-escaping)
* Security headers configured
* HTTPS enforcement
* Input validation (server-side)
* Error message sanitization

**6.4 Data Privacy**

* Encrypt PII (email, phone)
* User can view their data (decrypted)
* User can change username
* User can change password
* User can delete account (GDPR)
* No sensitive data in logs

**Conclusion**

This PRD defines a focused MCO 1 project that demonstrates strong security implementations while maintaining a complete system design. The separation of fully-implemented features (authentication, profile) from frontend-only features (ordering) allows for deep focus on web security while showing the complete user experience.