Product Requirements Document

Product Requirements Document: Payment Gateway Integration

1. Overview

The Payment Gateway Integration module enables secure processing of online payments through integration with Stripe payment processing API. This module handles credit card payments, debit card payments, and provides comprehensive transaction management, error handling, and security features.

2. Business Objectives

- Enable secure online payment processing for e-commerce transactionsÐ
- Support multiple payment methods (credit cards, debit cards)Ð
- Ensure PCI DSS compliance for payment data handlingÐ
- Provide real-time transaction processing with < 3 second response timeÐ
- Minimize payment failures and improve conversion ratesÐ
- Support transaction amounts from $0.01 to $10,000 per transactionÐ

3. Functional Requirements

3.1 Payment Form

FR-001: Payment form must include:

- Credit card number field (16 digits, formatted with spaces)Ð
- Expiry date field (MM/YY format)Ð
- CVV field (3-4 digits, masked)Ð
- Cardholder name field (max 100 characters)Ð
- Billing address fields (street, city, state, zip, country)Ð
- "Pay Now" buttonÐ
- "Cancel" buttonÐ
- Payment method selection (Credit Card / Debit Card)Ð

FR-002: Credit card number validation:

- Must accept Visa, Mastercard, American Express, DiscoverÐ
- Must validate using Luhn algorithmÐ
- Must format as user types (e.g., "4111 1111 1111 1111")Ð
- Must display card type icon based on first digitsÐ
- Must display error for invalid card number: "Please enter a valid card number"Ð

FR-003: Expiry date validation:

- Must accept MM/YY formatÐ
- Must validate that date is not in the pastÐ
- Must display error: "Card has expired" for past datesÐ
- Must display error: "Please enter a valid expiry date" for invalid formatÐ

FR-004: CVV validation:

- Must accept 3 digits for Visa/Mastercard/DiscoverÐ
- Must accept 4 digits for American ExpressÐ
- Must be masked (display as dots)Ð
- Must display error: "Please enter a valid CVV" for invalid inputÐ

FR-005: Cardholder name validation:

- Must accept alphanumeric characters and spacesÐ
- Must accept special characters: hyphens, apostrophes (for international names)Ð
- Maximum length: 100 charactersÐ
- Must display error: "Please enter cardholder name" if emptyÐ

FR-006: Billing address validation:

- All fields requiredÐ
- Zip code must match country format (US: 5 digits or ZIP+4)Ð
- Must validate address using address verification service (AVS)Ð
- Must display validation errors for each invalid fieldÐ

3.2 Payment Processing

FR-007: Transaction processing:

- On "Pay Now" click, validate all form fieldsÐ
- Display loading indicator during processingÐ
- Send payment request to Stripe APIÐ
- Handle Stripe API response (success or error)Ð
- Update order status in databaseÐ
- Send confirmation email to customerÐ
- Redirect to order confirmation page on successÐ

FR-008: Transaction amount limits:

- Minimum transaction: $0.01Ð
- Maximum transaction: $10,000.00Ð
- Display error if amount exceeds limit: "Transaction amount exceeds maximum limit of $10,000"Ð
- Display error if amount below minimum: "Transaction amount must be at least $0.01"Ð

FR-009: Payment method support:

- Credit cards: Visa, Mastercard, American Express, DiscoverÐ
- Debit cards: Same as credit cards (processed as credit)Ð
- Display supported card icons on payment formÐ
- Display error for unsupported card type: "Card type not supported"Ð

FR-010: Transaction status handling:

- Success: Update order status to "Paid", send confirmation emailÐ
- Failure: Display error message, allow retry, keep items in cartÐ
- Pending: Display pending status, send notification emailÐ
- Refunded: Update order status, send refund confirmation emailÐ

3.3 Error Handling

FR-011: Stripe API error handling:

- **Card declined:** "Your card was declined. Please try a different payment method."Đ
- **Insufficient funds:** "Insufficient funds. Please use a different card or payment method."Đ
- **Expired card:** "Card has expired. Please use a different card."Đ
- **Invalid CVV:** "CVV is incorrect. Please check and try again."Đ
- **Network error:** "Payment processing error. Please try again in a moment."Đ
- **Timeout:** "Payment request timed out. Please try again."Đ

FR-012: Retry mechanism:

- Allow user to retry failed payment up to 3 timesĐ
- Display retry count: "Attempt 2 of 3"Đ
- After 3 failures, suggest alternative payment methodĐ
- Clear form errors on retryĐ

FR-013: Cancellation:

- User can cancel payment before submissionĐ
- On cancel, return to cart pageĐ
- Preserve cart itemsĐ
- Display confirmation dialog: "Are you sure you want to cancel? Your items will remain in your cart."Đ

3.4 Security Requirements

FR-014: PCI DSS compliance:

- Never store full credit card numbers in databaseĐ
- Never transmit card data through application servers (use Stripe.js)Đ
- Use Stripe Elements for secure card input (PCI compliant)Đ
- Tokenize card data using Stripe Payment Intents APIĐ
- Store only payment method tokens (not actual card numbers)Đ

FR-015: Data encryption:

- All payment requests must use HTTPS (TLS 1.2+)Đ
- Encrypt sensitive data in transitĐ
- Use Stripe's secure tokenization (no card data touches our servers)Đ
- Implement request signing for API callsĐ

FR-016: Fraud prevention:

- Implement 3D Secure (3DS) for high-risk transactions (> $500)Đ
- Use Stripe Radar for fraud detectionĐ
- Block transactions flagged as fraudulentĐ
- Log all payment attempts for audit trailĐ

FR-017: Rate limiting:

- Limit payment attempts to 5 per hour per userĐ
- Limit payment attempts to 10 per hour per IP addressĐ
- Display error: "Too many payment attempts. Please try again later."Đ

3.5 Transaction Management

FR-018: Transaction logging:

- Log all payment attempts (success and failure)Đ
- Store transaction ID, amount, status, timestamp, user IDĐ
- Store Stripe payment intent ID for referenceĐ
- Maintain audit trail for 7 years (compliance requirement)Đ

FR-019: Transaction history:

- Display transaction history in user accountĐ
- Show transaction details: date, amount, status, order IDĐ
- Allow download of transaction receipt (PDF)Đ
- Filter transactions by date range, status, amountĐ

FR-020: Refund processing:

- Support full refunds and partial refundsĐ
- Process refunds through Stripe APIĐ
- Update order status to "Refunded"Đ
- Send refund confirmation emailĐ
- Display refund status in transaction historyĐ

4. Non-Functional Requirements

4.1 Performance

NFR-001: Payment processing time:

- Payment API response: < 3 seconds (95th percentile)Đ
- Page load time: < 2 secondsĐ
- Form validation: < 100msĐ

NFR-002: Availability:

- Payment service uptime: 99.9% (8.76 hours downtime per year)Đ
- Graceful degradation if Stripe API is unavailableĐ
- Display maintenance message during outagesĐ

4.2 Usability

NFR-003: User experience:

- Clear, step-by-step payment flowĐ
- Real-time form validation with helpful error messagesĐ
- Progress indicator during payment processingĐ
- Success confirmation with order detailsĐ
- Mobile-optimized payment formĐ

NFR-004: Error messages:

- User-friendly, actionable error messagesĐ
- No technical jargonĐ
- Suggest solutions for common errorsĐ

- Provide support contact for unresolved issuesÐ

4.3 Security

NFR-005: Security standards:

- PCI DSS Level 1 complianceÐ
- OWASP Top 10 security practicesÐ
- Regular security audits and penetration testingÐ
- Vulnerability scanning and patchingÐ

5. User Flows

5.1 Successful Payment Flow

1. User adds items to cart
2. User proceeds to checkout
3. User enters shipping information
4. User selects payment method (Credit Card)
5. User enters card details (number, expiry, CVV, name)
6. User enters billing address
7. User clicks "Pay Now"
8. System validates all fields
9. System displays loading indicator
10. System sends payment to Stripe API
11. Stripe processes payment (success)
12. System updates order status to "Paid"
13. System sends confirmation email
14. System redirects to order confirmation page
15. User sees order confirmation with order ID

5.2 Failed Payment Flow

1. User completes payment form
2. User clicks "Pay Now"
3. System validates fields (all valid)
4. System sends payment to Stripe API
5. Stripe declines payment (insufficient funds)
6. System receives error from Stripe
7. System displays error: "Insufficient funds. Please use a different card."
8. User can correct card details or select different payment method
9. User retries payment

5.3 Payment Cancellation Flow

1. User is on payment page
2. User has entered some payment details
3. User clicks "Cancel" button
4. System displays confirmation: "Are you sure you want to cancel?"

5. User confirms cancellation
6. System returns user to cart page
7. Cart items are preserved
8. User can continue shopping or retry checkout

6. Technical Specifications

6.1 API Integration

Stripe Payment Intents API:

- Endpoint: `POST https://api.stripe.com/v1/payment_intents`Đ
- Authentication: Secret key in Authorization headerĐ
- Request: `{ "amount": 5000, "currency": "usd", "payment_method": "pm_xxx" }`Đ
- Response: `{ "id": "pi_xxx", "status": "succeeded", "client_secret": "pi_xxx_secret_xxx" }`Đ

Stripe Elements (Frontend):

- Use Stripe.js library for secure card inputĐ
- Create payment method token client-sideĐ
- Send token to backend for processingĐ
- Handle 3D Secure authentication if requiredĐ

6.2 Database Schema

transactions table:

- id (UUID, primary key)Đ
- user_id (UUID, foreign key)Đ
- order_id (UUID, foreign key)Đ
- stripe_payment_intent_id (VARCHAR(255), unique)Đ
- amount (DECIMAL(10,2))Đ
- currency (VARCHAR(3), default 'USD')Đ
- status (ENUM: 'pending', 'succeeded', 'failed', 'refunded')Đ
- payment_method_type (VARCHAR(50))Đ
- billing_address (JSON)Đ
- created_at (TIMESTAMP)Đ
- updated_at (TIMESTAMP)Đ

payment_methods table:

- id (UUID, primary key)Đ
- user_id (UUID, foreign key)Đ
- stripe_payment_method_id (VARCHAR(255), unique)Đ
- card_type (VARCHAR(50))Đ
- last4 (VARCHAR(4))Đ
- expiry_month (INTEGER)Đ
- expiry_year (INTEGER)Đ
- is_default (BOOLEAN)Đ
- created_at (TIMESTAMP)Đ

6.3 Error Codes

- **card_declined:** Card was declined by issuerÐ
- **insufficient_funds:** Insufficient funds in accountÐ
- **expired_card:** Card has expiredÐ
- **incorrect_cvc:** CVV is incorrectÐ
- **processing_error:** Payment processing errorÐ
- **rate_limit:** Too many requestsÐ

7. Acceptance Criteria

1. ' User can successfully process payment with valid credit card
2. ' Payment form validates all fields correctly
3. ' Error messages display for invalid card details
4. ' Payment fails gracefully with appropriate error messages
5. ' Transaction is logged in database
6. ' Confirmation email is sent on successful payment
7. ' Payment processing completes in < 3 seconds
8. ' System handles Stripe API errors correctly
9. ' 3D Secure authentication works for high-risk transactions
10. ' Refund processing works correctly
11. ' Payment form is mobile-responsive
12. ' All security requirements are implemented

8. Dependencies

- Stripe API account and API keysÐ
- Stripe.js library (frontend)Ð
- Stripe Node.js SDK (backend)Ð
- HTTPS certificate (SSL/TLS)Ð
- Email service (transactional emails)Ð
- Database (transaction storage)Ð
- Redis (rate limiting)Ð

9. Out of Scope

- PayPal integration (future phase)Ð
- Cryptocurrency payments (future phase)Ð
- Buy now, pay later options (future phase)Ð
- Recurring payments/subscriptions (separate module)Ð
- International payment methods (Phase 2)Ð
- Payment analytics dashboard (separate feature)Ð

10. Testing Requirements

- Unit tests for validation logicÐ
- Integration tests with Stripe test APIÐ
- End-to-end tests for payment flowsÐ
- Security testing (penetration testing)Ð
- Performance testing (load testing)Ð

• Usability testing (user acceptance testing)Đ