Product Requirements Document

Product Requirements Document: API Integration Module

1. Overview

The API Integration Module provides a comprehensive RESTful API for third-party integrations, enabling external systems to interact with the application's core functionality. This module includes authentication, rate limiting, webhook support, comprehensive documentation, and versioning to ensure secure, scalable, and maintainable API integrations.

2. Business Objectives

- Enable third-party integrations with secure, well-documented APIÐ
- Support partner integrations and marketplace connectionsÐ
- Provide webhook capabilities for real-time event notificationsÐ
- Ensure API scalability to handle high-volume requestsÐ
- Maintain backward compatibility through API versioningÐ
- Facilitate developer onboarding with comprehensive documentationÐ

3. Functional Requirements

3.1 API Authentication

FR-001: API Key Authentication:

- Each API client must have unique API key and secretÐ
- API keys are generated in admin portalÐ
- API keys have expiration dates (default: 1 year, configurable)Ð
- API keys can be revoked at any timeÐ
- API requests must include API key in header: `X-API-Key: {api_key}`Ð
- API requests must include signature in header: `X-API-Signature: {signature}`Ð
- Signature is HMAC-SHA256 hash of request body + timestampÐ

FR-002: OAuth 2.0 Authentication:

- Support OAuth 2.0 client credentials flowÐ
- Clients register to obtain client_id and client_secretÐ
- Token endpoint: `POST /api/oauth/token`Ð
- Access tokens expire after 1 hourÐ
- Refresh tokens expire after 30 daysÐ
- Token endpoint returns: `{ "access_token": "...", "token_type": "Bearer", "expires_in": 3600, "refresh_token": "..." }`Ð

FR-003: JWT Token Authentication:

- Support JWT tokens for stateless authenticationÐ
- Tokens signed with RS256 algorithmÐ
- Token payload includes: user_id, permissions, expirationÐ
- Tokens included in Authorization header: `Authorization: Bearer {token}`Ð

- Token validation on every requestÐ

FR-004: API Key Management:

- Admin can create, view, update, and revoke API keysÐ
- API keys display: key name, created date, last used date, expiration date, status (active/revoked)Ð
- API keys can be regenerated (old key is immediately revoked)Ð
- API key usage statistics (requests per day, errors, etc.)Ð

3.2 API Endpoints

FR-005: Orders API:

- `GET /api/v1/orders` - List orders (with pagination, filtering, sorting)Ð
- `GET /api/v1/orders/{orderId}` - Get order detailsÐ
- `POST /api/v1/orders` - Create new orderÐ
- `PUT /api/v1/orders/{orderId}` - Update orderÐ
- `PATCH /api/v1/orders/{orderId}/status` - Update order statusÐ
- `DELETE /api/v1/orders/{orderId}` - Cancel order (soft delete)Ð

FR-006: Products API:

- `GET /api/v1/products` - List products (with pagination, search, filtering)Ð
- `GET /api/v1/products/{productId}` - Get product detailsÐ
- `POST /api/v1/products` - Create new productÐ
- `PUT /api/v1/products/{productId}` - Update productÐ
- `DELETE /api/v1/products/{productId}` - Delete productÐ

FR-007: Customers API:

- `GET /api/v1/customers` - List customers (with pagination, search)Ð
- `GET /api/v1/customers/{customerId}` - Get customer detailsÐ
- `POST /api/v1/customers` - Create new customerÐ
- `PUT /api/v1/customers/{customerId}` - Update customerÐ
- `DELETE /api/v1/customers/{customerId}` - Delete customer (GDPR compliant)Ð

FR-008: Inventory API:

- `GET /api/v1/inventory` - List inventory itemsÐ
- `GET /api/v1/inventory/{sku}` - Get inventory item by SKUÐ
- `POST /api/v1/inventory` - Add inventory itemÐ
- `PUT /api/v1/inventory/{sku}` - Update inventory quantityÐ
- `PATCH /api/v1/inventory/{sku}/reserve` - Reserve inventoryÐ
- `PATCH /api/v1/inventory/{sku}/release` - Release reserved inventoryÐ

FR-009: Webhooks API:

- `GET /api/v1/webhooks` - List registered webhooksÐ
- `POST /api/v1/webhooks` - Register new webhookÐ
- `PUT /api/v1/webhooks/{webhookId}` - Update webhookÐ
- `DELETE /api/v1/webhooks/{webhookId}` - Delete webhookÐ
- `POST /api/v1/webhooks/{webhookId}/test` - Test webhook deliveryÐ

3.3 Request/Response Format

FR-010: Request Format:

- All requests use JSON content type: `Content-Type: application/json`Đ
- Request body must be valid JSONĐ
- Query parameters for filtering, pagination, sortingĐ
- Headers: `X-API-Key`, `X-API-Signature`, `Content-Type`, `Accept`Đ

FR-011: Response Format:

- Success responses: HTTP 200 (OK), 201 (Created), 204 (No Content)Đ
- Error responses: HTTP 400 (Bad Request), 401 (Unauthorized), 403 (Forbidden), 404 (Not Found), 429 (Too Many Requests), 500 (Internal Server Error)Đ
- Standard error format: `{ "error": { "code": "ERROR_CODE", "message": "Human-readable message", "details": {...} } }`Đ
- Pagination format: `{ "data": [...], "pagination": { "page": 1, "perPage": 20, "total": 100, "totalPages": 5 } }`Đ

FR-012: Pagination:

- Default page size: 20 itemsĐ
- Maximum page size: 100 itemsĐ
- Query parameters: `?page=1&perPage=20`Đ
- Response includes pagination metadataĐ

FR-013: Filtering:

- Query parameters for filtering: `?status=active&category=electronics`Đ
- Support for multiple values: `?status=active,pending`Đ
- Support for range queries: `?priceMin=10&priceMax=100`Đ
- Support for date ranges: `?createdAfter=2024-01-01&createdBefore=2024-12-31`Đ

FR-014: Sorting:

- Query parameter: `?sort=createdAt&order=desc`Đ
- Support multiple sort fields: `?sort=name,price&order=asc,desc`Đ
- Default sort: `createdAt` descendingĐ

FR-015: Field Selection:

- Allow clients to specify which fields to return: `?fields=id,name,price`Đ
- Reduces response payload sizeĐ
- Improves API performanceĐ

3.4 Rate Limiting

FR-016: Rate Limit Rules:

- Default rate limit: 1000 requests per hour per API keyĐ
- Burst limit: 100 requests per minuteĐ
- Different limits for different API tiers (Free, Pro, Enterprise)Đ
- Rate limit headers in response: `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset`Đ

FR-017: Rate Limit Enforcement:

• When limit exceeded, return HTTP 429 (Too Many Requests)Đ

• Error response: `{ "error": { "code": "RATE_LIMIT_EXCEEDED", "message": "Rate limit exceeded", "retryAfter": 3600 } }`Đ

• Include `Retry-After` header with seconds until resetĐ

• Log rate limit violations for monitoringĐ

FR-018: Rate Limit Tiers:

• Free tier: 100 requests/hourĐ

• Pro tier: 1000 requests/hourĐ

• Enterprise tier: 10000 requests/hour (custom limits available)Đ

• Tier limits are configurable per API keyĐ

3.5 Webhooks

FR-019: Webhook Events:

• Order created: `order.created`Đ

• Order updated: `order.updated`Đ

• Order cancelled: `order.cancelled`Đ

• Payment received: `payment.received`Đ

• Payment failed: `payment.failed`Đ

• Inventory low: `inventory.low`Đ

• Customer created: `customer.created`Đ

FR-020: Webhook Delivery:

• Webhooks are delivered via HTTP POST to registered URLĐ

• Request body contains event data: `{ "event": "order.created", "data": {...}, "timestamp": "2024-01-01T00:00:00Z" }`Đ

• Include signature header: `X-Webhook-Signature` (HMAC-SHA256)Đ

• Retry failed deliveries (3 retries with exponential backoff: 1min, 5min, 30min)Đ

• Log all webhook deliveries (success and failure)Đ

FR-021: Webhook Security:

• Verify webhook signature on client sideĐ

• Support for webhook secret (shared secret for signature verification)Đ

• HTTPS required for webhook URLsĐ

• Validate webhook URL before registration (send test webhook)Đ

FR-022: Webhook Management:

• Clients can register multiple webhooks for same eventĐ

• Clients can update webhook URL, events, or statusĐ

• Clients can pause webhooks (temporarily disable)Đ

• Webhook delivery logs available in admin portalĐ

3.6 API Versioning

FR-023: Version Strategy:

• URL-based versioning: `/api/v1/`, `/api/v2/`Đ

• Current version: v1Đ

- Support multiple versions simultaneouslyĐ
- Deprecation policy: minimum 6 months notice before version removalĐ
- Version header: `X-API-Version: 1`Đ

FR-024: Version Migration:

- Provide migration guide for version upgradesĐ
- Maintain backward compatibility within major versionĐ
- Breaking changes require new major versionĐ
- Deprecation warnings in API responses for deprecated endpointsĐ

3.7 Error Handling

FR-025: Error Codes:

- `VALIDATION_ERROR` - Request validation failedĐ
- `AUTHENTICATION_ERROR` - Invalid or missing authenticationĐ
- `AUTHORIZATION_ERROR` - Insufficient permissionsĐ
- `NOT_FOUND` - Resource not foundĐ
- `RATE_LIMIT_EXCEEDED` - Rate limit exceededĐ
- `INTERNAL_ERROR` - Server errorĐ
- `SERVICE_UNAVAILABLE` - Service temporarily unavailableĐ

FR-026: Error Response Format:

- Standard format: `{ "error": { "code": "ERROR_CODE", "message": "Error message", "details": { "field": "error description" } } }`Đ
- Include request ID in error response: `X-Request-Id: {uuid}`Đ
- Include timestamp in error responseĐ
- Include documentation link for error resolutionĐ

3.8 API Documentation

FR-027: API Documentation:

- Interactive API documentation (Swagger/OpenAPI)Đ
- Endpoint descriptions with request/response examplesĐ
- Authentication guideĐ
- Rate limiting guideĐ
- Webhook setup guideĐ
- Code samples in multiple languages (JavaScript, Python, PHP, Ruby)Đ
- Postman collection for testingĐ

FR-028: Developer Portal:

- Self-service API key registrationĐ
- API usage dashboard (requests, errors, rate limits)Đ
- Webhook management interfaceĐ
- Documentation and guidesĐ
- Support contact formĐ
- Changelog and release notesĐ

4. Non-Functional Requirements

4.1 Performance

NFR-001: API Response Time:

- Average response time: < 200ms (95th percentile)Đ
- Maximum response time: < 1 second (99th percentile)Đ
- Database query time: < 100msĐ
- Support 10,000+ requests per secondĐ

NFR-002: Availability:

- API uptime: 99.9% (8.76 hours downtime per year)Đ
- Graceful degradation during high loadĐ
- Health check endpoint: `GET /api/health`Đ

4.2 Security

NFR-003: Security Standards:

- HTTPS only (TLS 1.2+)Đ
- Input validation and sanitizationĐ
- SQL injection prevention (parameterized queries)Đ
- XSS preventionĐ
- CORS configurationĐ
- IP whitelisting (optional, for Enterprise tier)Đ

NFR-004: Data Protection:

- Encrypt sensitive data in transit and at restĐ
- GDPR compliance (data deletion, export)Đ
- Audit logging for all API requestsĐ
- PII (Personally Identifiable Information) masking in logsĐ

4.3 Scalability

NFR-005: Scalability:

- Horizontal scaling supportĐ
- Load balancing across multiple serversĐ
- Database connection poolingĐ
- Caching for frequently accessed data (Redis)Đ
- Async processing for heavy operationsĐ

5. User Flows

5.1 API Key Registration

1. Developer navigates to developer portal
2. Developer creates account or logs in
3. Developer clicks "Create API Key"
4. Developer enters API key name and selects tier
5. System generates API key and secret
6. System displays API key (shown only once)

7. Developer copies and saves API key
8. Developer can view API key details in dashboard

5.2 Making API Request

1. Developer constructs API request with endpoint, headers, and body
2. Developer includes API key in header: `X-API-Key: {key}`
3. Developer generates signature (HMAC-SHA256 of body + timestamp)
4. Developer includes signature in header: `X-API-Signature: {signature}`
5. Developer sends HTTP request to API
6. API validates authentication
7. API processes request
8. API returns response with data or error

5.3 Webhook Registration

1. Developer navigates to webhooks section in developer portal
2. Developer clicks "Register Webhook"
3. Developer enters webhook URL (must be HTTPS)
4. Developer selects events to subscribe to
5. Developer enters webhook secret (optional, for signature verification)
6. System sends test webhook to verify URL
7. Developer confirms webhook is received
8. System activates webhook
9. System sends webhook events to registered URL

6. Technical Specifications

6.1 API Endpoints Summary

Base URL: `https://api.example.com/api/v1`

Authentication:

- `POST /oauth/token` - Get OAuth access tokenĐ
- `POST /oauth/refresh` - Refresh access tokenĐ

Orders:

- `GET /orders` - List ordersĐ
- `GET /orders/{id}` - Get orderĐ
- `POST /orders` - Create orderĐ
- `PUT /orders/{id}` - Update orderĐ
- `DELETE /orders/{id}` - Cancel orderĐ

Products:

- `GET /products` - List productsĐ
- `GET /products/{id}` - Get productĐ
- `POST /products` - Create productĐ
- `PUT /products/{id}` - Update productĐ
- `DELETE /products/{id}` - Delete productĐ

Webhooks:

- `GET /webhooks` - List webhooksÐ
- `POST /webhooks` - Register webhookÐ
- `PUT /webhooks/{id}` - Update webhookÐ
- `DELETE /webhooks/{id}` - Delete webhookÐ

6.2 Request Example

```
POST /api/v1/orders HTTP/1.1Ð
Host: api.example.comÐ
Content-Type: application/jsonÐ
X-API-Key: ak_live_1234567890abcdefÐ
X-API-Signature: sha256=abc123...Ð
X-Timestamp: 1640995200Ð
Ð
{Ð
  "customerId": "cust_123",Ð
  "items": [Ð
    {Ð
      "productId": "prod_456",Ð
      "quantity": 2,Ð
      "price": 29.99Ð
    }Ð
  ],Ð
  "shippingAddress": {Ð
    "street": "123 Main St",Ð
    "city": "New York",Ð
    "state": "NY",Ð
    "zip": "10001"Ð
  }Ð
}Ð
```

6.3 Response Example

```
{Ð
  "data": {Ð
    "id": "order_789",Ð
    "customerId": "cust_123",Ð
    "status": "pending",Ð
    "total": 59.98,Ð
    "createdAt": "2024-01-01T00:00:00Z"Ð
  },Ð
  "meta": {Ð
    "requestId": "req_abc123"Ð
  }Ð
}Ð
```

7. Acceptance Criteria

1. ' API authentication works (API key, OAuth, JWT)
2. ' All endpoints return correct responses
3. ' Rate limiting is enforced correctly
4. ' Webhooks are delivered successfully
5. ' Error handling returns proper error codes and messages
6. ' API documentation is comprehensive and accurate
7. ' Pagination, filtering, and sorting work correctly
8. ' API response time meets performance requirements
9. ' API versioning works correctly
10. ' Security requirements are implemented
11. ' Developer portal is functional

12. ' All API endpoints are tested and working

8. Dependencies

- API gateway (Kong or AWS API Gateway)Đ
- Authentication service (JWT, OAuth)Đ
- Database (PostgreSQL)Đ
- Redis (rate limiting, caching)Đ
- Message queue (webhook delivery)Đ
- Monitoring and logging (Datadog, New Relic)Đ

9. Out of Scope

- GraphQL API (future enhancement)Đ
- gRPC API (future enhancement)Đ
- WebSocket API (future enhancement)Đ
- API analytics dashboard (separate module)Đ
- API marketplace (future enhancement)Đ

10. Testing Requirements

- Unit tests for all API endpointsĐ
- Integration tests for authentication and authorizationĐ
- Load testing (10,000+ requests per second)Đ
- Security testing (penetration testing, OWASP Top 10)Đ
- Webhook delivery testingĐ
- API documentation accuracy testingĐ