API Integration Strategy for Rasuto Project

Top 3 Recommended APIs for MVP

Based on availability, documentation quality, and feature completeness, here are the recommended APIs for your MVP phase:

1. Best Buy API

Integration Approach:

- **API Type:** REST API with direct developer access
- **Documentation:** Available via Best Buy Developer Portal
- **Authentication:** API key-based authentication
- **Data Coverage:** Products, stores, categories, prices, availability
- **Rate Limits:** Need to check current limits on developer portal

Implementation Strategy:

1. Register for a Best Buy developer account at developer.bestbuy.com
2. Create application credentials to receive API key
3. Implement the following endpoints:
    - `/products` - Search and retrieve product information
    - `/products/{productId}` - Get specific product details
    - `/categories` - Browse category hierarchy
    - `/stores` - Find store information for local inventory

Benefits:

- Comprehensive electronics and tech product data
- Well-documented API with support for filtering
- Provides real-time inventory availability
- Product reviews and ratings included

2. Walmart API

Integration Approach:

- **API Type:** REST API via Walmart Marketplace

- **Documentation:** Available via Walmart Developer Portal
- **Authentication:** OAuth 2.0 with Client ID/Secret
- **Data Coverage:** Products, prices, inventory, categories
- **Security Note:** TLS 1.2 or later required (1.3 recommended)

Implementation Strategy:

1. Register as a developer at developer.walmart.com
2. Generate Client ID and Client Secret for API access
3. Implement OAuth 2.0 token generation for all API calls
4. Integrate with the following endpoints:
    - Items API for product details
    - Inventory API for stock information
    - Search API for product discovery

Benefits:

- Wide range of product categories
- Competitive pricing information
- High volume of products across multiple categories
- Detailed product specifications

3. eBay API

Integration Approach:

- **API Type:** RESTful APIs with JSON payloads
- **Documentation:** Available via eBay Developer Program
- **Authentication:** OAuth 2.0 for application authentication
- **Data Coverage:** Product listings, pricing, details, images
- **Rate Limits:** Daily call limits apply (check current limits)

Implementation Strategy:

1. Join the eBay Developers Program (free)
2. Create application credentials
3. Implement OAuth 2.0 token generation
4. Integrate with these key APIs:

- Browse API for searching items by keyword, category, or product ID
- Inventory API for detailed product information
- Marketing API for promotional information

Benefits:

- Access to a marketplace with diverse product types
- Multiple price points (new, used, auction)
- Global inventory from various sellers
- Robust search capabilities

Implementation Recommendations

1. **Authentication Manager:**
    - Create a centralized authentication service to handle tokens, keys, and credentials for all APIs
    - Implement token refresh and expiration handling
    - Store credentials securely (use Keychain for iOS)
2. **API Client Structure:**
    - Implement a protocol-based approach for all API clients
    - Create a common `ProductAPIClient` protocol
    - Develop concrete implementations for each retailer API
    - Use a factory pattern to instantiate the appropriate client
3. **Data Normalization Layer:**
    - Create adapters to normalize responses from different APIs to your schema
    - Implement mapping functions to transform API-specific fields
    - Handle inconsistencies across retailers (naming, units, currencies)
4. **Caching Strategy:**
    - Implement local caching for product information
    - Use network response caching for images and static content
    - Define appropriate cache expiration policies
5. **Error Handling:**
    - Create a unified error handling approach across all APIs
    - Map API-specific errors to your application error domain
    - Implement retry logic for transient failures

6. **Rate Limiting & Throttling:**
   - Monitor API usage against rate limits
   - Implement request throttling to avoid exceeding limits
   - Queue and prioritize requests when approaching limits

Scaling Considerations

1. **API Expansion:**
   - After MVP success, consider adding Nike, Adidas, and Zappos APIs
   - Prioritize addition based on user feedback and product category demand
2. **Performance Optimization:**
   - Implement background fetching for frequently accessed data
   - Consider server-side proxy API to aggregate multiple API responses
   - Optimize payload size by requesting only needed fields
3. **Fallback Strategy:**
   - Develop contingency plan for API outages
   - Implement cached results as fallback
   - Consider multi-source product matching for critical items
4. **Analytics:**
   - Track API performance metrics
   - Monitor success rates and response times
   - Identify frequently requested products for preemptive caching