Overview of Mobile Money APIs for African Markets

Mobile money services like MTN MoMo and Orange Money are dominant in African markets, enabling seamless transactions via APIs for collections, disbursements, and payments. These APIs are designed for integration into apps and websites, supporting use cases like recording transactions in your Smart Informal Business Credit & Record App. They emphasize security, ease of use, and compliance with local regulations. Integrations typically involve a developer portal for API keys, sandbox testing, and production rollout. For your React Native app with Node.js backend, handle API calls server-side to protect credentials, using libraries like Axios for HTTP requests. Focus on offline-first by logging transactions locally and syncing when online.

MTN MoMo API

MTN's Mobile Money (MoMo) API allows integration for products like collections (e.g., receiving payments from users), disbursements (e.g., sending funds), and remittances. It's available in countries like Uganda, Ghana, Zambia, Cameroon, and others, making it suitable for pan-African apps. The API uses RESTful endpoints with OAuth 2.0 authentication.

Key Features:

- **Collections**: Enable businesses to collect payments remotely via request-to-pay or invoices.
- **Disbursements**: Bulk or individual fund transfers to mobile wallets.
- **Other Endpoints**: Account balance checks, refunds, and notifications.
- **Sandbox Environment**: Free testing with simulated users; supports up to 60 requests per minute in some setups.

Integration Steps:

1. **Sign Up**: Register at the MoMo Developer Portal and subscribe to products (e.g., Collection or Disbursement).
2. **Create Sandbox User**: Generate a provisioning user and API key via the portal or Postman collections.
3. **Authentication**: Use API User ID and API Key to get an access token (Bearer token) for requests.
4. **Core Endpoints** (Base URL: https://sandbox.momoapi.mtn.com for testing):
    - **Request to Pay**: POST /collection/v1_0/requesttopay – Initiate a payment request from a user's mobile wallet.

    - **Get Transaction Status**: GET /collection/v1_0/requesttopay/{referenceId}.
    - **Disburse Funds**: POST /disbursement/v1_0/transfer.
5. **Going Live**: After sandbox testing, apply for production access via the portal; involves compliance checks.
6. **Libraries**: Use npm packages like mtn-momo for simplified integration.

For detailed docs, refer to the API Documentation. Common challenges include handling callbacks (use ngrok for local testing) and error codes (e.g., 400 for invalid requests).

Orange Money API

Orange Money APIs focus on web and mobile payments, supporting cash-in, cash-out, merchant payments, and transfers. They're available in countries like Senegal, Cameroon, Ivory Coast, and others in West Africa. The API is REST-based with token authentication and is part of Orange's developer ecosystem.

Key Features:

- **Web Payment (WebPay)**: Integrates into apps/websites for online payments via Orange Money wallets.
- **Merchant Payments**: Direct payments to businesses.
- **Other Operations**: Transfers, refunds, and bill payments.
- **Rate Limiting**: 60 requests per minute in sandbox.

Integration Steps:

1. **Sign Up**: Create an account at Orange Developer Portal and apply for Orange Money access (may require partnership approval).
2. **API Credentials**: Get merchant key, authorization header, and tokens from your dashboard.
3. **Sandbox Testing**: Use dev endpoints to simulate transactions.
4. **Core Endpoints** (Base URL varies by country, e.g., https://api.orange.com for general; check regional like Orange-Sonatel):
    - **Initiate Payment**: POST /orange-money-webpay/v1/payment – Start a web payment session.
    - **Get Token**: POST /oauth/v3/token for access tokens.
    - **Check Status**: GET /orange-money-webpay/v1/transactionstatus.

5. **Going Live**: Switch to production URLs after testing; ensure PCI compliance for hosted payments.
6. **Libraries**: npm packages like @spreeloop/orange_money or custom implementations.

Detailed docs are in the Orange Money API Reference. For your app, use redirects or in-app webviews for payment confirmation to handle user PIN entry securely.

Recommendations for Your App

- **Backend Handling**: Implement API calls in Node.js to avoid exposing keys in React Native; use PostgreSQL to log transaction refs (e.g., mobile_money_ref in your data model).
- **Offline Support**: Record intents locally; sync and confirm via API when online.
- **Error Handling**: Manage timeouts, insufficient funds, and network issues with retries.
- **Compliance**: Ensure user consent for data sharing; adhere to local regs like those from central banks.
- **Testing**: Start with sandboxes; simulate African network conditions using tools like Charles Proxy.