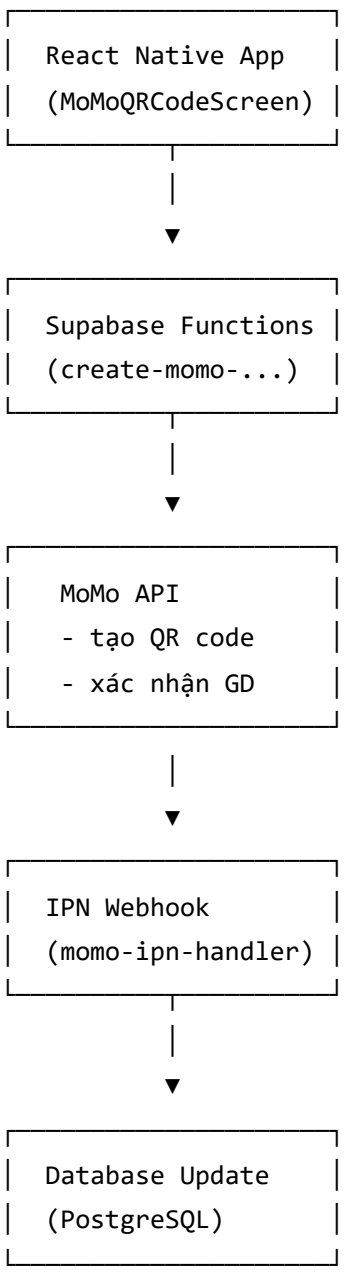
# 🧮 HƯỚNG DẪN TÍCH HỢP MOMO - TableFlow

Tài liệu này hướng dẫn chi tiết cách tích hợp thanh toán MoMo cho **React Native + Expo + Supabase** và **Flutter + Firebase**.

## 📑 MỤC LỤC

# PHẦN 1: React Native + Expo + Supabase

## 1.1 KIẾN TRÚC TÍCH HỢP MOMO



# 1.2 BƯỚC 1: CẤU HÌNH SUPABASE EDGE FUNCTIONS

## A. Tạo Function tạo MoMo QR Code

**Đường dẫn**: `supabase/functions/create-momo-payment/index.ts`

```ts
import { serve } from "https://deno.land/std@0.168.0/http/server.ts";
import { corsHeaders } from "../_shared/cors.ts";

interface MoMoRequest {
  amount: number;
  orderId: string;
  orderInfo: string;
  userAction?: string;
}

const MOMO_PARTNER_CODE = Deno.env.get("MOMO_PARTNER_CODE") || "";
const MOMO_ACCESS_KEY = Deno.env.get("MOMO_ACCESS_KEY") || "";
const MOMO_SECRET_KEY = Deno.env.get("MOMO_SECRET_KEY") || "";
const MOMO_API_ENDPOINT = "https://test-payment.momo.vn/v2/gateway/api/create";
const MOMO_IPN_URL = Deno.env.get("MOMO_IPN_URL") || "";

// Hàm tạo signature (HMAC SHA256)
function createSignature(rawSignature: string): string {
  const encoder = new TextEncoder();
  const data = encoder.encode(rawSignature);
  return crypto.subtle.digestSync("sha256", data).hex();
}

serve(async (req) => {
  // Xử lý CORS
  if (req.method === "OPTIONS") {
    return new Response("ok", { headers: corsHeaders });
  }

  try {
    const { amount, orderId, orderInfo, userAction }: MoMoRequest = await req.json();

    // ✅ Validate dữ liệu đầu vào
    if (!amount || !orderId) {
      return new Response(
        JSON.stringify({ success: false, message: "Missing amount or orderId" }),
        { status: 400, headers: corsHeaders }
      );
    }

    // 🔑 Tạo request cho MoMo API
    const requestId = `${Date.now()}`;
    const partnerName = "TableFlow";
```

```
const partnerCode = MOMO_PARTNER_CODE;
const accessKey = MOMO_ACCESS_KEY;
const secretKey = MOMO_SECRET_KEY;
const lang = "vi";
const requestType = "QR_PAY";
const redirectUrl = "tableflow://momo-callback";
const ipnUrl = MOMO_IPN_URL;
const extra = JSON.stringify({ orderId, userAction });

// 📝 Tạo raw signature (theo thứ tự chính xác từ MoMo docs)
const rawSignature = [
  `accessKey=${accessKey}`,
  `amount=${amount}`,
  `extraData=${extra}`,
  `ipnUrl=${ipnUrl}`,
  `orderId=${orderId}`,
  `orderInfo=${orderInfo}`,
  `partnerCode=${partnerCode}`,
  `partnerName=${partnerName}`,
  `redirectUrl=${redirectUrl}`,
  `requestId=${requestId}`,
  `requestType=${requestType}`,
].join("&");

// 🔒 Tạo signature bằng HMAC SHA256
const signature = await createHmacSha256(rawSignature, secretKey);

// 🌐 Gọi MoMo API
const momoResponse = await fetch(MOMO_API_ENDPOINT, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({
    partnerCode,
    partnerName,
    requestId,
    amount,
    orderId,
    orderInfo,
    redirectUrl,
    ipnUrl,
    lang,
    requestType,
    autoCapture: true,
```

```
        extraData: extra,
        signature,
        accessKey,
      }),
    });

    const momoData = await momoResponse.json();

    if (momoData.resultCode === 0 && momoData.qrCodeUrl) {
      return new Response(
        JSON.stringify({
          success: true,
          qrCodeUrl: momoData.qrCodeUrl,
          orderId: momoData.orderId,
          transactionId: momoData.transactionId,
          message: "QR Code created successfully",
        }),
        { status: 200, headers: corsHeaders }
      );
    } else {
      return new Response(
        JSON.stringify({
          success: false,
          message: momoData.message || "Failed to create QR code",
          resultCode: momoData.resultCode,
        }),
        { status: 400, headers: corsHeaders }
      );
    }
  } catch (error) {
    console.error("[create-momo-payment] Error:", error);
    return new Response(
      JSON.stringify({ success: false, message: error.message }),
      { status: 500, headers: corsHeaders }
    );
  }
});

// Hàm tạo HMAC SHA256
async function createHmacSha256(data: string, secret: string): Promise<string> {
  const encoder = new TextEncoder();
  const keyData = encoder.encode(secret);
  const messageData = encoder.encode(data);
```

```
  const key = await crypto.subtle.importKey(
    "raw",
    keyData,
    { name: "HMAC", hash: "SHA-256" },
    false,
    ["sign"]
  );

  const signature = await crypto.subtle.sign("HMAC", key, messageData);
  return Array.from(new Uint8Array(signature))
    .map((b) => b.toString(16).padStart(2, "0"))
    .join("");
}
```

# B. Tạo Function xử lý IPN Webhook

**Đường dẫn**: `supabase/functions/momo-ipn-handler/index.ts`

```ts
import { serve } from "https://deno.land/std@0.168.0/http/server.ts";
import { createClient } from "https://esm.sh/@supabase/supabase-js@2.38.0";

const corsHeaders = {
  "Access-Control-Allow-Origin": "*",
  "Access-Control-Allow-Methods": "POST, OPTIONS",
  "Access-Control-Allow-Headers": "Content-Type",
};

const MOMO_SECRET_KEY = Deno.env.get("MOMO_SECRET_KEY") || "";
const SUPABASE_URL = Deno.env.get("SUPABASE_URL") || "";
const SUPABASE_SERVICE_ROLE_KEY = Deno.env.get("SUPABASE_SERVICE_ROLE_KEY") || "";

serve(async (req) => {
  if (req.method === "OPTIONS") {
    return new Response("ok", { headers: corsHeaders });
  }

  if (req.method !== "POST") {
    return new Response("Method not allowed", { status: 405, headers: corsHeaders });
  }

  try {
    const body = await req.json();
    console.log("[IPN] Nhận webhook từ MoMo:", body);

    const { resultCode, orderId, transactionId, amount, extraData } = body;

    // ✅ Xác thực signature (quan trọng để đảm bảo yêu cầu từ MoMo)
    // [BƯỚC BỎ QUA TRONG DEMO - triển khai trong production]

    // 🔄 Cập nhật database
    const supabase = createClient(SUPABASE_URL, SUPABASE_SERVICE_ROLE_KEY);

    if (resultCode === 0) {
      // ✅ Thanh toán thành công
      console.log(`[IPN] Giao dịch ${orderId} thành công`);

      const { error } = await supabase
        .from("orders")
        .update({
          status: "paid",
          payment_method: "momo",
```

```
        paid_at: new Date().toISOString(),
        total_price: amount,
      })
      .eq("id", orderId);

    if (error) {
      console.error("[IPN] Lỗi cập nhật database:", error);
      return new Response(
        JSON.stringify({ success: false, message: "Database update failed" }),
        { status: 500, headers: corsHeaders }
      );
    }

    // 📊 Ghi transaction log
    await supabase.from("transactions").insert([
      {
        order_id: orderId,
        amount,
        method: "momo",
        status: "completed",
        reference_id: transactionId,
        raw_response: JSON.stringify(body),
      },
    ]);

    return new Response(
      JSON.stringify({ success: true, message: "Payment recorded" }),
      { status: 200, headers: corsHeaders }
    );
  } else {
    // ❌ Thanh toán thất bại
    console.log(`[IPN] Giao dịch ${orderId} thất bại (code: ${resultCode})`);

    await supabase
      .from("orders")
      .update({ status: "pending", payment_status: "failed" })
      .eq("id", orderId);

    return new Response(
      JSON.stringify({ success: false, message: "Payment failed" }),
      { status: 400, headers: corsHeaders }
    );
  }
```

```
  } catch (error) {
    console.error("[IPN] Lỗi xử lý webhook:", error);
    return new Response(
      JSON.stringify({ success: false, message: error.message }),
      { status: 500, headers: corsHeaders }
    );
  }
});
```

# 1.3 BƯỚC 2: CẤU HÌNH REACT NATIVE SCREEN

## A. MoMoQRCodeScreen.tsx (đã có trong project)

**Xem file**: `screens/Orders/MoMoQRCodeScreen.tsx`
**Các bước chính**:

1. **Gọi Supabase Function**:

```
const { data, error } = await supabase.functions.invoke('create-momo-payment', {
  body: {
    amount,
    orderId,
    orderInfo: `Thanh toan Don hang ${orderId.slice(-6)}`,
    userAction: pendingPaymentAction,
  },
});
```

2. **Hiển thị QR Code**:

```
<QRCode value={qrValue} size={250} />
```

3. **Lắng nghe Realtime Updates**:

```
const channel = supabase
  .channel(`momo_order_status_${orderId}`)
  .on('postgres_changes', {
    event: 'UPDATE',
    schema: 'public',
    table: 'orders',
    filter: `id=eq.${orderId}`,
  }, (payload) => {
    // Cập nhật UI khi thanh toán thành công
    if (payload.new?.status === 'paid') {
      navigateToPrintPreview();
    }
  })
  .subscribe();
```

# 1.4 BƯỚC 3: CẤU HÌNH ENVIRONMENT VARIABLES

**Tệp**: `.env.local` hoặc `.env.production`

```
# Supabase
SUPABASE_URL=https://your-project.supabase.co
SUPABASE_ANON_KEY=your-anon-key
SUPABASE_SERVICE_ROLE_KEY=your-service-role-key

# MoMo
MOMO_PARTNER_CODE=your-partner-code
MOMO_ACCESS_KEY=your-access-key
MOMO_SECRET_KEY=your-secret-key
MOMO_IPN_URL=https://your-domain.com/functions/v1/momo-ipn-handler
```

# 1.5 BƯỚC 4: DATABASE SCHEMA

**Tạo bảng transactions**:

```sql
CREATE TABLE transactions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  order_id UUID NOT NULL REFERENCES orders(id) ON DELETE CASCADE,
  amount INT NOT NULL,
  method VARCHAR(50) NOT NULL, -- 'momo', 'vietqr', 'cash', 'bank'
  status VARCHAR(50) NOT NULL DEFAULT 'pending', -- 'pending', 'completed', 'failed', 'refunded
  reference_id VARCHAR(255), -- transaction ID từ MoMo/VietQR
  raw_response JSONB, -- response từ payment gateway
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Cập nhật bảng orders
ALTER TABLE orders
ADD COLUMN payment_method VARCHAR(50),
ADD COLUMN paid_at TIMESTAMP WITH TIME ZONE,
ADD COLUMN payment_status VARCHAR(50) DEFAULT 'pending';
```

# 1.6 BƯỚC 5: KIỂM THỬ

## A. Test Flow Cơ Bản

```
# 1. Chạy ứng dụng
npm run android  # hoặc npm run ios
# 2. Vào màn hình thanh toán MoMo
# 3. Nhấn nút "(Test) Giả lập thanh toán thành công"
# 4. Kiểm tra xem:
#    - Database được cập nhật?
#    - Realtime listener trigger?
#    - Màn hình in hóa đơn hiện ra?
```

## B. Test Production Flow

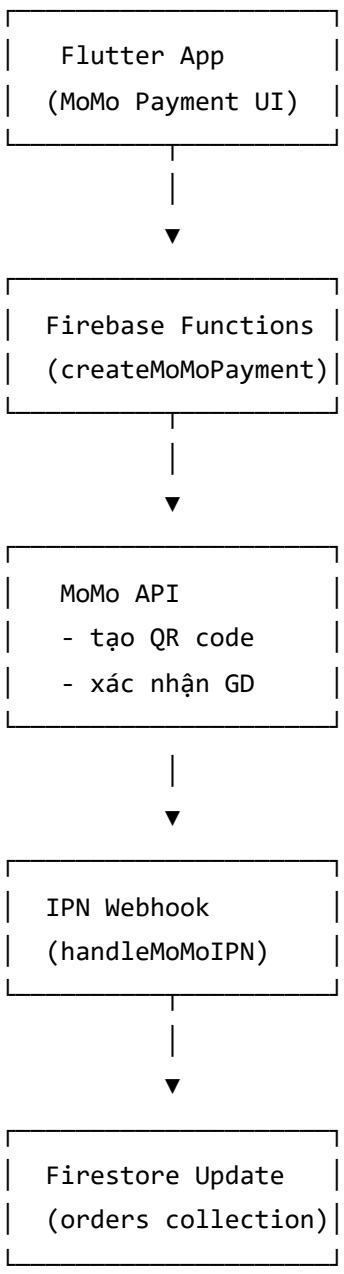
```
# 1. Lấy QR code từ MoMo API
# 2. Quét qua MoMo app (hoặc emulator)
# 3. Hoàn tất thanh toán
# 4. Webhook từ MoMo gọi tới IPN handler
# 5. Database cập nhật tự động
# 6. Frontend lắng nghe realtime và điều hướng
```

# 1.7 TROUBLESHOOTING - React Native

| Vấn Đề | Giải Pháp |
|---|---|
| QR Code không xuất hiện | Kiểm tra `MOMO_PARTNER_CODE` có chính xác? |
| IPN webhook không được gọi | Kiểm tra `MOMO_IPN_URL` đúng format https? |
| Realtime listener không trigger | Kiểm tra RLS policy cho bảng orders |
| Payment status không cập nhật | Kiểm tra Edge Function logs trong Supabase |

# PHẦN 2: Flutter + Firebase

## 2.1 KIẾN TRÚC TÍCH HỢP MOMO



# 2.2 BƯỚC 1: CẤU HÌNH FIREBASE FUNCTIONS

## A. Cài đặt Firebase CLI & Functions

```
# Cài Firebase CLI
npm install -g firebase-tools
# Login Firebase
firebase login
# Tạo project Firebase Functions
firebase init functions
cd functions
npm install --save axios
```

## B. Tạo Function tạo MoMo QR Code

**Đường dẫn**: `functions/src/momo.ts`

```
import * as functions from "firebase-functions";
import * as admin from "firebase-admin";
import axios from "axios";
import * as crypto from "crypto";

admin.initializeApp();

const MOMO_PARTNER_CODE = process.env.MOMO_PARTNER_CODE || "";
const MOMO_ACCESS_KEY = process.env.MOMO_ACCESS_KEY || "";
const MOMO_SECRET_KEY = process.env.MOMO_SECRET_KEY || "";
const MOMO_API_ENDPOINT = "https://test-payment.momo.vn/v2/gateway/api/create";
const MOMO_IPN_URL = process.env.MOMO_IPN_URL || "";

// Hàm tạo HMAC SHA256
function createSignature(rawSignature: string): string {
  return crypto
    .createHmac("sha256", MOMO_SECRET_KEY)
    .update(rawSignature)
    .digest("hex");
}

export const createMoMoPayment = functions.https.onCall(async (data, context) => {
  // ✅ Xác thực user đã đăng nhập
  if (!context.auth) {
    throw new functions.https.HttpsError(
      "unauthenticated",
      "User must be authenticated"
    );
  }

  const { amount, orderId, orderInfo, userAction } = data;

  // ✅ Validate dữ liệu
  if (!amount || !orderId) {
    throw new functions.https.HttpsError(
      "invalid-argument",
      "Missing amount or orderId"
    );
  }

  try {
    // 🔑 Tạo request cho MoMo API
    const requestId = `${Date.now()}`;
```

```
const partnerName = "TableFlow";
const lang = "vi";
const requestType = "QR_PAY";
const redirectUrl = "tableflow://momo-callback";
const extra = JSON.stringify({ orderId, userAction });

// 📝 Tạo raw signature
const rawSignature = [
  `accessKey=${MOMO_ACCESS_KEY}`,
  `amount=${amount}`,
  `extraData=${extra}`,
  `ipnUrl=${MOMO_IPN_URL}`,
  `orderId=${orderId}`,
  `orderInfo=${orderInfo}`,
  `partnerCode=${MOMO_PARTNER_CODE}`,
  `partnerName=${partnerName}`,
  `redirectUrl=${redirectUrl}`,
  `requestId=${requestId}`,
  `requestType=${requestType}`,
].join("&");

// 🔐 Tạo signature
const signature = createSignature(rawSignature);

// 🌐 Gọi MoMo API
const response = await axios.post(MOMO_API_ENDPOINT, {
  partnerCode: MOMO_PARTNER_CODE,
  partnerName,
  requestId,
  amount,
  orderId,
  orderInfo,
  redirectUrl,
  ipnUrl: MOMO_IPN_URL,
  lang,
  requestType,
  autoCapture: true,
  extraData: extra,
  signature,
  accessKey: MOMO_ACCESS_KEY,
});

if (response.data.resultCode === 0 && response.data.qrCodeUrl) {
```

```
      return {
        success: true,
        qrCodeUrl: response.data.qrCodeUrl,
        orderId: response.data.orderId,
        transactionId: response.data.transactionId,
      };
    } else {
      throw new functions.https.HttpsError(
        "internal",
        `MoMo API error: ${response.data.message}`
      );
    }
  } catch (error: any) {
    console.error("[createMoMoPayment] Error:", error);
    throw new functions.https.HttpsError(
      "internal",
      `Failed to create MoMo payment: ${error.message}`
    );
  }
});

// IPN Webhook Handler
export const handleMoMoIPN = functions.https.onRequest(
  async (req, res) => {
    if (req.method !== "POST") {
      return res.status(405).send("Method not allowed");
    }

    try {
      const body = req.body;
      console.log("[IPN] Webhook từ MoMo:", body);

      const { resultCode, orderId, transactionId, amount } = body;

      if (resultCode === 0) {
        // ✅ Thanh toán thành công
        console.log(`[IPN] Giao dịch ${orderId} thành công`);

        // 🔄 Cập nhật Firestore
        const ordersRef = admin.firestore().collection("orders");
        await ordersRef.doc(orderId).update({
          status: "paid",
          paymentMethod: "momo",
```

```
          paidAt: admin.firestore.FieldValue.serverTimestamp(),
          totalPrice: amount,
        });

        // 📊 Ghi transaction log
        await admin
          .firestore()
          .collection("transactions")
          .add({
            orderId,
            amount,
            method: "momo",
            status: "completed",
            referenceId: transactionId,
            rawResponse: body,
            createdAt: admin.firestore.FieldValue.serverTimestamp(),
          });

        return res.json({ success: true, message: "Payment recorded" });
      } else {
        // ❌ Thanh toán thất bại
        console.log(`[IPN] Giao dịch ${orderId} thất bại`);

        await ordersRef.doc(orderId).update({
          paymentStatus: "failed",
        });

        return res.json({ success: false, message: "Payment failed" });
      }
    } catch (error: any) {
      console.error("[IPN] Lỗi xử lý webhook:", error);
      return res.status(500).json({ success: false, message: error.message });
    }
  }
);
```

**Đường dẫn**: `functions/src/index.ts`

```
import * as momo from "./momo";
exports.createMoMoPayment = momo.createMoMoPayment;
exports.handleMoMoIPN = momo.handleMoMoIPN;
```

## C. Deploy Firebase Functions

```
# Deploy tất cả functions
firebase deploy --only functions

# Deploy riêng function
firebase deploy --only functions:createMoMoPayment

# Lấy URL của HTTP function
firebase functions:list
```

# 2.3 BƯỚC 2: CẤU HÌNH FLUTTER UI

## A. Tạo MoMo Payment Screen

**Đường dẫn**: `lib/screens/orders/momo_payment_screen.dart`

```dart
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_functions/firebase_functions.dart';
import 'package:qr_flutter/qr_flutter.dart';
import 'package:fluttertoast/fluttertoast.dart';

class MoMoPaymentScreen extends StatefulWidget {
  final String orderId;
  final int amount;
  final String? userAction;

  const MoMoPaymentScreen({
    required this.orderId,
    required this.amount,
    this.userAction,
  });

  @override
  State<MoMoPaymentScreen> createState() => _MoMoPaymentScreenState();
}

class _MoMoPaymentScreenState extends State<MoMoPaymentScreen> {
  late FirebaseFunctions _functions;
  late FirebaseFirestore _firestore;

  String? _qrValue;
  bool _isLoading = true;
  bool _isNavigating = false;
  StreamSubscription? _orderSubscription;

  @override
  void initState() {
    super.initState();
    _functions = FirebaseFunctions.instance;
    _firestore = FirebaseFirestore.instance;

    _createMoMoQRCode();
    _listenToOrderUpdates();
  }

  // ✅ Tạo QR Code MoMo
  Future<void> _createMoMoQRCode() async {
    try {
```

```
    setState(() => _isLoading = true);

    final callable = _functions.httpsCallable('createMoMoPayment');
    final response = await callable.call({
      'amount': widget.amount,
      'orderId': widget.orderId,
      'orderInfo': 'Thanh toan Don hang ${widget.orderId.substring(widget.orderId.length - 6)}',
      'userAction': widget.userAction,
    });

    if (response.data['success'] == true) {
      setState(() {
        _qrValue = response.data['qrCodeUrl'];
        _isLoading = false;
      });
      print('✅ QR Code created: ${_qrValue!.substring(0, 50)}...');
    } else {
      throw Exception(response.data['message'] ?? 'Failed to create QR');
    }
  } catch (error) {
    print('❌ Error creating MoMo QR: $error');
    Fluttertoast.showToast(
      msg: 'Lỗi tạo mã QR: $error',
      backgroundColor: Colors.red,
    );
    setState(() => _isLoading = false);
    if (mounted) Navigator.pop(context);
  }
}

// 🔄 Lắng nghe cập nhật đơn hàng (Realtime)
void _listenToOrderUpdates() {
  _orderSubscription = _firestore
      .collection('orders')
      .doc(widget.orderId)
      .snapshots()
      .listen((snapshot) {
    if (snapshot.exists) {
      final status = snapshot['status'];
      print('[Realtime] Trạng thái đơn hàng: $status');

      if ((status == 'paid' || status == 'closed') && !_isNavigating) {
        _navigateToPrintPreview();
```

```
      }
    }
  });
}

// 🖨️ Điều hướng sang trang in hóa đơn
Future<void> _navigateToPrintPreview() async {
  if (_isNavigating) return;
  _isNavigating = true;

  try {
    setState(() => _isLoading = true);

    // Lấy dữ liệu đơn hàng
    final orderDoc = await _firestore
        .collection('orders')
        .doc(widget.orderId)
        .get();

    if (!orderDoc.exists) throw Exception('Order not found');

    // Lấy chi tiết items
    final itemsSnapshot = await _firestore
        .collection('orders')
        .doc(widget.orderId)
        .collection('items')
        .get();

    final billItems = itemsSnapshot.docs
        .map((doc) => {
              'name': doc['name'],
              'quantity': doc['quantity'],
              'unitPrice': doc['unitPrice'],
              'totalPrice': doc['quantity'] * doc['unitPrice'],
            })
        .toList();

    Fluttertoast.showToast(
      msg: 'Thanh toán thành công!',
      backgroundColor: Colors.green,
    );

    if (mounted) {
```

```
      Navigator.pushReplacementNamed(
        context,
        '/print-preview',
        arguments: {
          'orderId': widget.orderId,
          'items': billItems,
          'paymentMethod': 'momo',
        },
      );
    }
  } catch (error) {
    print('❌ Error navigating to print: $error');
    Fluttertoast.showToast(msg: 'Lỗi: $error', backgroundColor: Colors.red);
    _isNavigating = false;
    setState(() => _isLoading = false);
  }
}

// 🧪 Test: Giả lập thanh toán thành công (chỉ dùng cho development)
Future<void> _simulatePaymentSuccess() async {
  try {
    await _firestore.collection('orders').doc(widget.orderId).update({
      'status': 'paid',
      'paymentMethod': 'momo',
      'paidAt': FieldValue.serverTimestamp(),
      'totalPrice': widget.amount,
    });

    print('✅ Simulated payment success');
    Fluttertoast.showToast(msg: 'Giả lập thành công!');
  } catch (error) {
    print('❌ Simulation error: $error');
    Fluttertoast.showToast(msg: 'Lỗi: $error', backgroundColor: Colors.red);
  }
}

@override
void dispose() {
  _orderSubscription?.cancel();
  super.dispose();
}

@override
```

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Thanh toán qua MoMo'),
      backgroundColor: Colors.white,
      foregroundColor: Colors.black,
      elevation: 1,
    ),
    body: _isLoading && _qrValue == null
        ? Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                CircularProgressIndicator(),
                SizedBox(height: 16),
                Text('Đang tạo mã QR...'),
              ],
            ),
          )
        : _qrValue != null
            ? SingleChildScrollView(
                child: Padding(
                  padding: EdgeInsets.all(16),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.center,
                    children: [
                      // QR Code
                      Container(
                        padding: EdgeInsets.all(16),
                        decoration: BoxDecoration(
                          color: Colors.white,
                          borderRadius: BorderRadius.circular(12),
                          boxShadow: [
                            BoxShadow(
                              color: Colors.grey.withOpacity(0.1),
                              blurRadius: 10,
                            )
                          ],
                        ),
                        child: QrImage(
                          data: _qrValue!,
                          version: QrVersions.auto,
                          size: 250,
```

```
    backgroundColor: Colors.white,
  ),
),
SizedBox(height: 24),

// Số tiền
Container(
  padding: EdgeInsets.all(16),
  decoration: BoxDecoration(
    color: Color(0xFFFFF0FB),
    borderRadius: BorderRadius.circular(12),
    border: Border.all(color: Color(0xFFFBCFE8)),
  ),
  child: Column(
    children: [
      Text(
        'Số tiền cần thanh toán',
        style: TextStyle(
          fontSize: 14,
          color: Colors.grey[600],
        ),
      ),
      SizedBox(height: 8),
      Text(
        '${widget.amount.toString()}đ',
        style: TextStyle(
          fontSize: 28,
          fontWeight: FontWeight.bold,
          color: Color(0xFFA60067),
        ),
      ),
    ],
  ),
),
SizedBox(height: 24),

// Hướng dẫn
Container(
  padding: EdgeInsets.all(16),
  decoration: BoxDecoration(
    color: Color(0xFFFFF0FB),
    borderRadius: BorderRadius.circular(12),
    border: Border.all(color: Color(0xFFFBCFE8)),
```

```
  ),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        'Hướng dẫn thanh toán',
        style: TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.bold,
          color: Color(0xFFA60067),
        ),
      ),
      SizedBox(height: 12),
      Text(
        '1. Mở ứng dụng MoMo',
        style: TextStyle(fontSize: 14),
      ),
      SizedBox(height: 8),
      Text(
        '2. Chọn "Quét mã QR"',
        style: TextStyle(fontSize: 14),
      ),
      SizedBox(height: 8),
      Text(
        '3. Quét mã ở trên màn hình này',
        style: TextStyle(fontSize: 14),
      ),
      SizedBox(height: 8),
      Text(
        '4. Hoàn tất thanh toán',
        style: TextStyle(fontSize: 14),
      ),
    ],
  ),
),
SizedBox(height: 32),

// Test Button (chỉ dùng development)
ElevatedButton.icon(
  onPressed: _simulatePaymentSuccess,
  icon: Icon(Icons.check_circle),
  label: Text('(Test) Giả lập thành công'),
  style: ElevatedButton.styleFrom(
```

```
                        backgroundColor: Color(0xFF16A34A),
                        foregroundColor: Colors.white,
                        padding: EdgeInsets.symmetric(
                          vertical: 14,
                          horizontal: 32,
                        ),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                        ),
                      ),
                    ),
                  ),
                ],
              ),
            ),
          )
        : Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Icon(Icons.error_outline, size: 48, color: Colors.red),
                SizedBox(height: 12),
                Text('Không thể tạo mã QR'),
                SizedBox(height: 16),
                ElevatedButton(
                  onPressed: _createMoMoQRCode,
                  child: Text('Thử lại'),
                ),
              ],
            ),
          ),
    );
  }
}
```

# 2.4 BƯỚC 3: CẤU HÌNH FIRESTORE

## A. Firestore Rules (Security)

**Đường dẫn**: `firestore.rules`

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Orders - Staff chỉ thấy order của họ
    match /orders/{orderId} {
      allow read, write: if request.auth != null &&
        (request.auth.token.role == 'staff' ||
         request.auth.token.role == 'admin');
    }

    // Transactions - Only admin can read
    match /transactions/{transactionId} {
      allow read: if request.auth != null &&
        request.auth.token.role == 'admin';
      allow write: if false; // Backend writes via service account
    }
  }
}
```

## B. Firestore Collections

```
// Collection: orders
{
  id: "order123",
  status: "pending", // "pending" -> "paid" -> "closed"
  paymentMethod: "momo",
  totalPrice: 250000,
  paidAt: timestamp,
  createdAt: timestamp,
  updatedAt: timestamp,
}

// Collection: transactions
{
  id: "trans123",
  orderId: "order123",
  amount: 250000,
  method: "momo",
  status: "completed", // "pending", "completed", "failed"
  referenceId: "momo-tx-123",
  rawResponse: { ... },
  createdAt: timestamp,
}
```

# 2.5 BƯỚC 4: CẤU HÌNH ENVIRONMENT

**Đường dẫn**: `.env` hoặc `firebase/.env`

```
MOMO_PARTNER_CODE=your-partner-code
MOMO_ACCESS_KEY=your-access-key
MOMO_SECRET_KEY=your-secret-key
MOMO_IPN_URL=https://region-projectid.cloudfunctions.net/handleMoMoIPN
```

# 2.6 BƯỚC 5: DEPLOY

```
# Deploy Firebase Functions
firebase deploy --only functions

# Deploy Firestore Rules
firebase deploy --only firestore:rules

# Kiểm tra logs
firebase functions:log

# Test IPN endpoint
curl -X POST https://region-projectid.cloudfunctions.net/handleMoMoIPN \
  -H "Content-Type: application/json" \
  -d '{
    "resultCode": 0,
    "orderId": "order123",
    "amount": 250000,
    "transactionId": "tx123"
  }'
```

# 2.7 TROUBLESHOOTING - Flutter

| Vấn Đề | Giải Pháp |
|---|---|
| `PlatformException: Permission denied` | Kiểm tra Firestore Rules |
| QR Code URL bị null | Kiểm tra `MOMO_PARTNER_CODE` trong environment |
| Realtime listener không trigger | Kiểm tra Firestore đã cập nhật? |
| Firebase Function timeout | Kiểm tra `MOMO_API_ENDPOINT` có chạy? |

# PHẦN 3: So Sánh & Lựa Chọn

## 3.1 BẢNG SO SÁNH

| Tiêu Chí | React Native + Supabase | Flutter + Firebase |
|---|---|---|
| **Độ phức tạp** | ⭐ ⭐ ⭐ | ⭐ ⭐ ⭐ ⭐ |
| **Realtime** | ✅ Built-in Realtime | ✅ Firestore Realtime |
| **Cost** | Rẻ hơn | Tuỳ vào usage |
| **Scaling** | Tốt | Rất tốt |
| **Learning Curve** | Dễ | Khó hơn |
| **Community** | Lớn | Rất lớn |
| **Setup Time** | 1-2 ngày | 2-3 ngày |
| **Performance** | Ngang nhau | Ngang nhau |

## 3.2 LỰA CHỌN CÔNG NGHỆ

### ✅ Chọn React Native + Supabase nếu:

- Bạn chỉ cần mobile app
- Budget hạn chế
- Muốn tập trung vào coding chứ không infrastructure
- Team nhỏ, chỉ 1-2 developer

### ✅ Chọn Flutter + Firebase nếu:

- Cần scalability cao
- Có native iOS requirements
- Muốn ecosystem lớn từ Google
- Team lớn, support nhiều platforms

## 3.3 HYBRID APPROACH

Có thể **kết hợp cả hai**:

```
Backend: Firebase (scalability)
Frontend: React Native (quicker dev)
Payment: MoMo (both)
Database: Firestore + PostgreSQL (sync via middleware)
```

# TỔNG KẾT

| Bước | React Native | Flutter |
|---|---|---|
| **1** | Tạo Supabase Functions | Tạo Firebase Functions |
| **2** | Build React Native Screen | Build Flutter Screen |
| **3** | Setup Database Schema (PostgreSQL) | Setup Firestore Collections |
| **4** | Config Environment | Config Environment |
| **5** | Deploy Supabase Functions | Deploy Firebase Functions |
| **6** | Test with Simulator | Test with Emulator |

## 📚 TÀI LIỆU THAM KHẢO

- **Supabase Docs**: https://supabase.com/docs
- **Firebase Functions**: https://firebase.google.com/docs/functions
- **MoMo API**: https://developers.momo.vn
- **React Native**: https://reactnative.dev/docs
- **Flutter**: https://flutter.dev/docs

# ✉️ HỖ TRỢ

Nếu gặp vấn đề:

1. **Check logs**: `firebase functions:log` hoặc Supabase logs
2. **Test endpoint**: Dùng Postman/Thunder để test API
3. **Verify credentials**: Kiểm tra MOMO_PARTNER_CODE, secret key
4. **Database**: Kiểm tra RLS policies & Firestore rules

**Cập nhật lần cuối**: 2025-11-14
**Version**: 1.0