# CASE STUDY: TRIỂN KHAI API THỰC TẾ CHO WEBSITE THƯƠNG MẠI ĐIỆN TỬ

**Phân tích bởi: HỒNG MINH (MINH HM)**
CEO LIGHT | Chuyên gia Marketing Online, lập trình với 12+ năm kinh nghiệm

---

## TỔNG QUAN DỰ ÁN

### Thông tin dự án

**Loại website:** Website bán hàng thương mại điện tử
**Lĩnh vực:** Thương mại điện tử - Phụ kiện thời trang
**Quy mô:** 5,000+ sản phẩm, 50,000+ khách hàng đăng ký
**Thời gian triển khai:** 3 tháng

### Thách thức ban đầu

Khách hàng đang gặp phải những vấn đề nghiêm trọng ảnh hưởng đến trải nghiệm người dùng và doanh số:

- **Đồng bộ dữ liệu thủ công:** Cập nhật giá, tồn kho từ 3 nhà cung cấp khác nhau mất 2-3 giờ mỗi ngày
- **Thanh toán đơn lẻ:** Chỉ hỗ trợ chuyển khoản ngân hàng, làm mất 40% khách hàng tiềm năng
- **Thiếu thông tin vận chuyển:** Khách hàng không thể theo dõi đơn hàng, gây 15% khiếu nại
- **Quản lý CRM thủ công:** Dữ liệu khách hàng rời rạc, khó phân tích và marketing

### Mục tiêu dự án

1. **Tự động hóa 90%** quy trình cập nhật dữ liệu sản phẩm
2. **Tích hợp 5+ cổng thanh toán** phổ biến tại Việt Nam
3. **Tracking đơn hàng realtime** từ 3 đơn vị vận chuyển
4. **Đồng bộ CRM** để phân tích hành vi khách hàng
5. **Tăng 30% conversion rate** trong 6 tháng đầu

---

## PHÂN TÍCH VÀ LỰA CHỌN API

### API được tích hợp

#### 1. Product Management APIs

**Supplier API - Đồng bộ sản phẩm từ nhà cung cấp**

```json
{
  "api_endpoint": "https://supplier-api.com/v2/products",
  "method": "GET",
  "authentication": "Bearer Token",
  "update_frequency": "Mỗi 30 phút",
  "data_fields": [
    "product_id", "name", "price", "stock_quantity",
    "images", "specifications", "warranty_info"
  ]
}
```

**Lý do lựa chọn:**

- Hỗ trợ webhook để cập nhật theo thời gian thực
- Tài liệu hướng dẫn đầy đủ, dễ triển khai
- Giới hạn tốc độ hợp lý (1000 yêu cầu/giờ)
- Đã được 500+ website thương mại điện tử sử dụng

**2. Payment Gateway APIs**

**VNPay API - Cổng thanh toán phổ biến**

```php
// Cấu hình VNPay API
$vnp_TmnCode = "MERCHANT_CODE";
$vnp_HashSecret = "SECRET_KEY";
$vnp_Url = "https://sandbox.vnpayment.vn/paymentv2/vpcpay.html";
$vnp_ReturnUrl = "https://watchpro.com/payment/callback";
```

**ZaloPay API - Thanh toán ví điện tử**

```javascript
const zalopayConfig = {
  app_id: "YOUR_APP_ID",
  key1: "YOUR_KEY1",
  key2: "YOUR_KEY2",
  endpoint: "https://sb-openapi.zalopay.vn/v2/create"
};
```

**Tiêu chí lựa chọn:**

- **Độ phổ biến:** VNPay (45% thị phần), ZaloPay (25% thị phần)

- **Phí giao dịch:** 1.5-2.9% tùy loại thẻ
- **Thời gian thanh toán:** T+1 đến T+3
- **Bảo mật:** Tuân thủ chuẩn PCI DSS Level 1

### 3. Shipping & Logistics APIs

#### Giao Hàng Nhanh (GHN) API

```python
import requests

ghn_headers = {
    'Token': 'YOUR_GHN_TOKEN',
    'ShopId': 'YOUR_SHOP_ID',
    'Content-Type': 'application/json'
}

def create_shipping_order(order_data):
    url = "https://dev-online-gateway.ghn.vn/shiip/public-api/v2/shipping-order/create"
    response = requests.post(url, headers=ghn_headers, json=order_data)
    return response.json()
```

#### Viettel Post API

```json
{
  "ORDER_NUMBER": "VP_ORDER_001",
  "SENDER_FULLNAME": "WatchPro Store",
  "RECEIVER_FULLNAME": "Nguyen Van A",
  "PRODUCT_WEIGHT": 500,
  "PRODUCT_PRICE": 5000000,
  "MONEY_COLLECTION": 5000000
}
```

### 4. CRM Integration APIs

#### HubSpot CRM API

```javascript
```

```
const hubspot = require('@hubspot/api-client');
const hubspotClient = new hubspot.Client({
  accessToken: 'YOUR_ACCESS_TOKEN'
});

// Tạo contact mới
const contactObj = {
  properties: {
    email: 'customer@email.com',
    firstname: 'John',
    lastname: 'Doe',
    phone: '0901234567',
    total_purchase: '5000000'
  }
};
```

---

## QUY TRÌNH TRIỂN KHAI CHI TIẾT

### Phase 1: Planning & Setup (Tuần 1-2)

#### Bước 1: Phân tích kiến trúc hệ thống hiện tại

**Công nghệ stack:**

- **Frontend:** ReactJS 18.2, TailwindCSS
- **Backend:** Node.js 18 với Express.js
- **Database:** MongoDB Atlas, Redis Cache
- **Server:** AWS EC2 t3.medium, Nginx proxy
- **Monitoring:** New Relic, Sentry error tracking

#### Bước 2: Thiết kế kiến trúc API

mermaid

```
graph TD
    A[Giao diện website] --> B[API Gateway]
    B --> C[Dịch vụ xác thực]
    B --> D[Dịch vụ sản phẩm]
    B --> E[Dịch vụ thanh toán]
    B --> F[Dịch vụ vận chuyển]
    B --> G[Dịch vụ CRM]

    D --> H[API nhà cung cấp 1]
    D --> I[API nhà cung cấp 2]
    D --> J[API nhà cung cấp 3]

    E --> K[VNPay API]
    E --> L[ZaloPay API]
    E --> M[MoMo API]

    F --> N[API Giao hàng nhanh]
    F --> O[API Viettel Post]
    F --> P[API GHTK]

    G --> Q[HubSpot CRM]
```

### Bước 3: Thiết lập môi trường phát triển

```bash
# Môi trường phát triển
NODE_ENV=development
MONGODB_URI=mongodb://localhost:27017/ecommerce_dev
REDIS_URL=redis://localhost:6379

# Thông tin đăng nhập API
VNPAY_TMN_CODE=your_vnpay_code
VNPAY_HASH_SECRET=your_vnpay_secret
ZALOPAY_APP_ID=your_zalopay_app_id
GHN_TOKEN=your_ghn_token
HUBSPOT_ACCESS_TOKEN=your_hubspot_token

# Giới hạn tốc độ
API_RATE_LIMIT=100
API_RATE_WINDOW=900000
```

## Phase 2: Core API Integration (Tuần 3-6)

### Triển khai API đồng bộ sản phẩm

## 1. Thiết kế cấu trúc cơ sở dữ liệu:

javascript

```javascript
// Cấu trúc sản phẩm
const productSchema = new mongoose.Schema({
  product_id: { type: String, unique: true, required: true },
  supplier_id: { type: String, required: true },
  name: { type: String, required: true },
  price: {
    original: Number,
    sale: Number,
    currency: { type: String, default: 'VND' }
  },
  stock: {
    quantity: { type: Number, default: 0 },
    status: { type: String, enum: ['in_stock', 'out_of_stock', 'pre_order'] }
  },
  images: [String],
  specifications: mongoose.Schema.Types.Mixed,
  last_sync: { type: Date, default: Date.now },
  sync_status: { type: String, enum: ['pending', 'success', 'failed'] }
});
```

## 2. Triển khai dịch vụ đồng bộ:

javascript

```
class ProductSyncService {
  constructor() {
    this.suppliers = [
      { id: 'supplier_1', endpoint: 'https://api.supplier1.com', token: process.env.SUPPLIER1_TOKEN },
      { id: 'supplier_2', endpoint: 'https://api.supplier2.com', token: process.env.SUPPLIER2_TOKEN },
      { id: 'supplier_3', endpoint: 'https://api.supplier3.com', token: process.env.SUPPLIER3_TOKEN }
    ];
  }

  async syncAllProducts() {
    const results = [];

    for (const supplier of this.suppliers) {
      try {
        const products = await this.fetchProductsFromSupplier(supplier);
        const syncResult = await this.updateLocalProducts(products, supplier.id);
        results.push({
          supplier_id: supplier.id,
          status: 'success',
          products_updated: syncResult.length
        });
      } catch (error) {
        logger.error(`Sync failed for supplier ${supplier.id}:`, error);
        results.push({
          supplier_id: supplier.id,
          status: 'failed',
          error: error.message
        });
      }
    }

    return results;
  }

  async fetchProductsFromSupplier(supplier) {
    const response = await fetch(`${supplier.endpoint}/products`, {
      headers: {
        'Authorization': `Bearer ${supplier.token}`,
        'Content-Type': 'application/json'
      }
    });

    if (!response.ok) {
      throw new Error(`HTTP ${response.status}: ${response.statusText}`);
    }
```

```
    return await response.json();
  }
}

// Cron job để sync tự động mỗi 30 phút
const cron = require('node-cron');
const productSync = new ProductSyncService();

cron.schedule('*/30 * * * *', async () => {
  console.log('Bắt đầu đồng bộ sản phẩm...');
  const results = await productSync.syncAllProducts();
  console.log('Hoàn thành đồng bộ:', results);
});
```

**Triển khai tích hợp cổng thanh toán**

**1. Mẫu thiết kế Factory cho thanh toán:**

```
javascript
```

```
class PaymentFactory {
  static createPaymentProcessor(gateway) {
    switch (gateway) {
      case 'vnpay':
        return new VNPayProcessor();
      case 'zalopay':
        return new ZaloPayProcessor();
      case 'momo':
        return new MoMoProcessor();
      default:
        throw new Error(`Unsupported payment gateway: ${gateway}`);
    }
  }
}

class VNPayProcessor {
  constructor() {
    this.config = {
      tmnCode: process.env.VNPAY_TMN_CODE,
      hashSecret: process.env.VNPAY_HASH_SECRET,
      url: process.env.VNPAY_URL,
      returnUrl: process.env.VNPAY_RETURN_URL
    };
  }

  async createPaymentUrl(orderInfo) {
    const vnp_Params = {
      'vnp_Version': '2.1.0',
      'vnp_Command': 'pay',
      'vnp_TmnCode': this.config.tmnCode,
      'vnp_Locale': 'vn',
      'vnp_CurrCode': 'VND',
      'vnp_TxnRef': orderInfo.orderId,
      'vnp_OrderInfo': orderInfo.description,
      'vnp_OrderType': 'other',
      'vnp_Amount': orderInfo.amount * 100,
      'vnp_ReturnUrl': this.config.returnUrl,
      'vnp_IpAddr': orderInfo.ipAddr,
      'vnp_CreateDate': this.formatDate(new Date())
    };

    // Sắp xếp tham số và tạo chuỗi truy vấn
    const sortedParams = this.sortObject(vnp_Params);
    const signData = querystring.stringify(sortedParams);
    const secureHash = crypto
      .createHmac('sha512', this.config.hashSecret)
```

```
      .update(Buffer.from(signData, 'utf-8'))
      .digest('hex');

    sortedParams['vnp_SecureHash'] = secureHash;

    return this.config.url + '?' + querystring.stringify(sortedParams);
  }
}
```

**2. Xử lý Webhook thanh toán:**

javascript

```
app.post('/api/payment/webhook/:gateway', async (req, res) => {
  const { gateway } = req.params;
  const paymentData = req.body;

  try {
    const processor = PaymentFactory.createPaymentProcessor(gateway);
    const isValidSignature = await processor.verifyWebhook(paymentData);

    if (!isValidSignature) {
      return res.status(400).json({ error: 'Invalid signature' });
    }

    // Cập nhật trạng thái đơn hàng
    await OrderService.updatePaymentStatus(
      paymentData.orderId,
      paymentData.status,
      paymentData.transactionId
    );

    // Gửi email xác nhận
    if (paymentData.status === 'success') {
      await EmailService.sendPaymentConfirmation(paymentData.orderId);
    }

    res.json({ status: 'success' });
  } catch (error) {
    logger.error('Payment webhook error:', error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```



## Giai đoạn 3: Tính năng nâng cao (Tuần 7-10)

## Hệ thống theo dõi vận chuyển

javascript

```
class ShippingTrackingService {
  constructor() {
    this.providers = {
      'ghn': new GHNProvider(),
      'viettelpost': new ViettelPostProvider(),
      'ghtk': new GHTKProvider()
    };
  }

  async trackOrder(orderId) {
    const order = await Order.findById(orderId);
    if (!order.shipping_info) {
      throw new Error('No shipping information found');
    }

    const provider = this.providers[order.shipping_info.carrier];
    const trackingInfo = await provider.getTrackingInfo(
      order.shipping_info.tracking_number
    );

    // Cập nhật tracking history
    await Order.findByIdAndUpdate(orderId, {
      $push: {
        tracking_history: {
          status: trackingInfo.status,
          location: trackingInfo.location,
          timestamp: new Date(),
          description: trackingInfo.description
        }
      }
    });

    return trackingInfo;
  }

  async setupWebhookForAllProviders() {
    for (const [providerName, provider] of Object.entries(this.providers)) {
      try {
        await provider.registerWebhook(`${process.env.BASE_URL}/api/shipping/webhook/${providerName}`);
        console.log(`Webhook registered for ${providerName}`);
      } catch (error) {
        console.error(`Failed to register webhook for ${providerName}:`, error);
      }
    }
```

```
  }
}
```

### Đồng bộ dữ liệu CRM

javascript

```
class CRMSyncService {
  constructor() {
    this.hubspot = new hubspot.Client({
      accessToken: process.env.HUBSPOT_ACCESS_TOKEN
    });
  }

  async syncCustomerData(customerId) {
    const customer = await Customer.findById(customerId);
    const orders = await Order.find({ customer_id: customerId });

    // Tính toán các chỉ số
    const totalOrders = orders.length;
    const totalValue = orders.reduce((sum, order) => sum + order.total_amount, 0);
    const avgOrderValue = totalValue / totalOrders || 0;
    const lastOrderDate = orders.length > 0 ?
      Math.max(...orders.map(o => new Date(o.created_at).getTime())) : null;

    const contactData = {
      email: customer.email,
      firstname: customer.first_name,
      lastname: customer.last_name,
      phone: customer.phone,
      total_orders: totalOrders,
      total_revenue: totalValue,
      average_order_value: avgOrderValue,
      last_order_date: lastOrderDate,
      customer_segment: this.calculateCustomerSegment(totalValue, totalOrders),
      registration_date: customer.created_at
    };

    try {
      // Kiểm tra contact đã tồn tại
      const existingContact = await this.hubspot.crm.contacts.basicApi.getById(
        customer.email,
        undefined,
        undefined,
        ['email']
      );

      // Cập nhật contact hiện có
      await this.hubspot.crm.contacts.basicApi.update(
        existingContact.id,
        { properties: contactData }
      );
    } catch (error) {
```

```
      if (error.code === 404) {
        // Tạo contact mới
        await this.hubspot.crm.contacts.basicApi.create({ properties: contactData });
      } else {
        throw error;
      }
    }
  }

  calculateCustomerSegment(totalValue, totalOrders) {
    if (totalValue >= 50000000) return 'VIP';
    if (totalValue >= 20000000) return 'Premium';
    if (totalOrders >= 5) return 'Loyal';
    if (totalOrders >= 2) return 'Regular';
    return 'New';
  }
}
```

## Giai đoạn 4: Kiểm thử và tối ưu (Tuần 11-12)

### Kiểm thử hiệu suất API

javascript

```
// Load testing với Artillery
const artilleryConfig = {
  config: {
    target: 'https://api.watchpro.com',
    phases: [
      { duration: 60, arrivalRate: 10 },
      { duration: 120, arrivalRate: 50 },
      { duration: 60, arrivalRate: 100 }
    ],
    defaults: {
      headers: {
        'Authorization': 'Bearer test-token',
        'Content-Type': 'application/json'
      }
    }
  },
  scenarios: [
    {
      name: 'Product API Load Test',
      requests: [
        { get: { url: '/api/products' } },
        { get: { url: '/api/products/{{ $randomInt(1, 1000) }}' } }
      ]
    },
    {
      name: 'Payment API Load Test',
      requests: [
        {
          post: {
            url: '/api/payment/create',
            json: {
              amount: 5000000,
              gateway: 'vnpay',
              orderId: '{{ $randomString(10) }}'
            }
          }
        }
      ]
    }
  ]
};

// Monitoring với custom metrics
const prometheusMetrics = {
  apiRequests: new prometheus.Counter({
    name: 'api_requests_total',
```

```
    help: 'Total number of API requests',
    labelNames: ['method', 'endpoint', 'status']
  }),
  apiDuration: new prometheus.Histogram({
    name: 'api_request_duration_seconds',
    help: 'API request duration',
    labelNames: ['method', 'endpoint']
  })
};
```

---

## KẾT QUẢ VÀ ĐÁNH GIÁ

### Chỉ số trước và sau triển khai

| Chỉ số | Trước triển khai | Sau triển khai | Cải thiện |
|---|---|---|---|
| **Thời gian cập nhật sản phẩm** | 2-3 giờ/ngày | 30 phút tự động | -83% |
| **Tỷ lệ thanh toán thành công** | 60% | 89% | +48% |
| **Thời gian xử lý đơn hàng** | 4-6 giờ | 15-30 phút | -87% |
| **Tỷ lệ khiếu nại giao hàng** | 15% | 3% | -80% |
| **Conversion rate** | 2.1% | 3.4% | +62% |
| **Average order value** | 3.2 triệu | 4.1 triệu | +28% |

### Chi phí và ROI

**Chi phí triển khai:**

- Phát triển: 45,000,000 VND
- Phí sử dụng API (6 tháng): 12,000,000 VND
- Nâng cấp hạ tầng: 8,000,000 VND
- **Tổng chi phí:** 65,000,000 VND

**Lợi ích kinh tế:**

- Tiết kiệm nhân sự: 15,000,000 VND/tháng
- Tăng doanh số: 50,000,000 VND/tháng
- Giảm chi phí vận hành: 8,000,000 VND/tháng
- **Tổng lợi ích:** 73,000,000 VND/tháng

**ROI:** 1,125% sau 12 tháng

### Phản hồi từ khách hàng

> *"Trước đây tôi phải chờ 2-3 ngày mới biết đơn hàng được xử lý. Giờ đây chỉ 30 phút sau khi đặt hàng, tôi đã nhận được thông tin chi tiết và có thể theo dõi realtime. Trải nghiệm tuyệt vời!"*
> **- Nguyễn Minh Anh, Khách hàng VIP**

> *"Payment gateway đa dạng giúp tôi có thể thanh toán bằng ZaloPay rất tiện lợi. Website load nhanh hơn và không bị lag như trước."*
> **- Trần Văn Nam, Khách hàng thường xuyên**

---

## BÀI HỌC VÀ KHUYẾN NGHỊ

### Những thách thức gặp phải

#### 1. Vấn đề giới hạn tốc độ

**Vấn đề:** Một số API nhà cung cấp có giới hạn tốc độ thấp (100 yêu cầu/giờ)
**Giải pháp:** Triển khai bộ nhớ đệm thông minh và xử lý theo lô

```javascript
class RateLimitedAPIClient {
  constructor(maxRequests, timeWindow) {
    this.maxRequests = maxRequests;
    this.timeWindow = timeWindow;
    this.requests = [];
  }

  async makeRequest(requestFn) {
    await this.waitIfNeeded();
    const result = await requestFn();
    this.requests.push(Date.now());
    return result;
  }

  async waitIfNeeded() {
    const now = Date.now();
    this.requests = this.requests.filter(time => now - time < this.timeWindow);

    if (this.requests.length >= this.maxRequests) {
      const oldestRequest = Math.min(...this.requests);
      const waitTime = this.timeWindow - (now - oldestRequest);
      await new Promise(resolve => setTimeout(resolve, waitTime));
    }
  }
}
```

#### 2. Vấn đề nhất quán dữ liệu

**Vấn đề:** Đồng bộ dữ liệu từ nhiều nguồn gây xung đột
**Giải pháp:** Triển khai event sourcing và mẫu CQRS

### 3. Phiên bản API

**Vấn đề:** Một số API thay đổi phiên bản đột ngột
**Giải pháp:** Luôn sử dụng endpoint có phiên bản và theo dõi changelog của API

## Thực hành tốt nhất được áp dụng

1. **Xử lý lỗi toàn diện**

javascript

```
class APIError extends Error {
  constructor(message, status, provider) {
    super(message);
    this.status = status;
    this.provider = provider;
    this.timestamp = new Date();
  }
}

const handleAPICall = async (apiCall, retries = 3) => {
  for (let i = 0; i < retries; i++) {
    try {
      return await apiCall();
    } catch (error) {
      if (i === retries - 1) throw error;
      await new Promise(resolve => setTimeout(resolve, Math.pow(2, i) * 1000));
    }
  }
};
```

2. **Mẫu Circuit Breaker**

javascript

```
class CircuitBreaker {
  constructor(threshold = 5, timeout = 60000) {
    this.threshold = threshold;
    this.timeout = timeout;
    this.failureCount = 0;
    this.state = 'CLOSED'; // CLOSED, OPEN, HALF_OPEN
    this.nextAttempt = Date.now();
  }

  async call(fn) {
    if (this.state === 'OPEN') {
      if (Date.now() < this.nextAttempt) {
        throw new Error('Circuit breaker is OPEN');
      }
      this.state = 'HALF_OPEN';
    }

    try {
      const result = await fn();
      this.onSuccess();
      return result;
    } catch (error) {
      this.onFailure();
      throw error;
    }
  }
}
```

3. **Theo dõi toàn diện**

- Theo dõi thời gian phản hồi API
- Theo dõi tỷ lệ lỗi
- Bảng điều khiển chỉ số kinh doanh
- Hệ thống cảnh báo thời gian thực

## Khuyến nghị cho dự án tương lai

1. **Giai đoạn lập kế hoạch:**
   - Luôn có kế hoạch dự phòng cho mỗi API quan trọng
   - Đàm phán SLA rõ ràng với nhà cung cấp API
   - Thiết kế cơ chế dự phòng từ đầu
2. **Giai đoạn phát triển:**
   - Triển khai kiểm thử toàn diện (đơn vị, tích hợp, end-to-end)

- Sử dụng công cụ tài liệu API (Swagger/OpenAPI)
- Code review tập trung vào xử lý lỗi

3. **Giai đoạn triển khai:**
   - Triển khai blue-green để giảm thiểu downtime
   - Triển khai từ từ với feature flags
   - Theo dõi thời gian thực từ ngày đầu
4. **Giai đoạn bảo trì:**
   - Kiểm tra hiệu suất định kỳ
   - Theo dõi tình trạng API chủ động
   - Tối ưu liên tục dựa trên phản hồi người dùng

---

## KẾT LUẬN

Dự án tích hợp API cho website thương mại điện tử đã thành công vượt xa kỳ vọng ban đầu. Với việc tự động hóa 90% quy trình vận hành, website không chỉ cải thiện trải nghiệm người dùng mà còn tăng đáng kể hiệu quả kinh doanh.

**Những yếu tố thành công chính:**

- Phân tích kỹ lưỡng yêu cầu và lựa chọn API phù hợp
- Thiết kế kiến trúc linh hoạt, có thể mở rộng
- Triển khai xử lý lỗi toàn diện và theo dõi
- Kiểm thử kỹ lưỡng trước khi triển khai
- Tối ưu liên tục sau khi đưa vào sử dụng

**Tác động dài hạn:** Hệ thống API này đã trở thành nền tảng cho các tính năng mới như gợi ý sản phẩm sử dụng AI, phân tích nâng cao, và trải nghiệm khách hàng đa kênh. Chi phí bảo trì thấp và ROI cao đã thuyết phục ban lãnh đạo đầu tư thêm vào chuyển đổi số.

Với 12+ năm kinh nghiệm trong lĩnh vực lập trình và API integration, tôi tin rằng case study này sẽ cung cấp insights giá trị cho các doanh nghiệp đang cân nhắc triển khai API để tối ưu hóa vận hành và tăng trưởng kinh doanh.

---

## VỀ CHUYÊN GIA HỒNG MINH


### CHUYÊN GIA HỒNG MINH

CEO LIGHT

*Hơn 12 năm kinh nghiệm trong ngành Marketing Online, lập trình, SEO, thiết kế đồ họa, chạy quảng cáo, vv...*

- Trainning chuyên sâu về SEO, Google Ads, Quảng Cáo cho hơn 3000+ doanh nghiệp
- 20+ Khóa tư vấn đào tạo cho doanh nghiệp về Marketing Online
- 100+ Dự án API integration và website development
- Facebook: https://www.facebook.com/hm.phm
- Chi tiết: https://light.com.vn/minh-hm
- Youtube: https://www.youtube.com/@minhhmchanel2340

---

*Light - Nhanh - Chuẩn - Đẹp*