# G# Google Ads Automation Tool - Design Documentation

## VietnamVisaHelp.com

Version: 1.0
Date: January 19, 2026
Company: VietnamVisaHelp.com
Contact: support@vietnamvisahelp.com

## Table of Contents

## 1. Executive Summary

### 1.1 Purpose

The Google Ads Automation Tool is designed to automatically manage Google Ads campaigns for VietnamVisaHelp.com, a visa processing service for travelers to Vietnam. The tool ensures that advertisements are only displayed when the corresponding visa services are actually available, preventing customer frustration and wasted ad spend.

### 1.2 Key Features

- **Automated Campaign Scheduling**: Enables/disables campaigns based on Vietnam office hours (UTC+7)
- **Service-Based Ad Control**: Different visa services (urgent, standard, weekend) are advertised only when available
- **Multi-Language Support**: Ad templates in 6 languages (English, Spanish, Portuguese, French, Russian, Hindi)

~~~~
- **Geographic Targeting**: Airport proximity targeting and country-level
geo-targeting
- **Real-Time Sync**: Runs every 15 minutes via Vercel Cron to keep ads
synchronized

### 1.3 Business Problem Solved

Vietnam visa processing services have specific availability windows:
- **Urgent services (1-hour, 2-hour, 4-hour)**: Only available during Vietnam
office hours (8 AM - 4 PM, Monday-Friday)
- **Weekend/Holiday services**: Available but at premium pricing
- **Standard services**: Available 24/7

Without automation, ads for unavailable services could run, leading to:
- Customer complaints when services can't be delivered
- Wasted advertising budget
- Damaged brand reputation

---

## 2. System Overview

### 2.1 High-Level Architecture

```
┌─────────────────────────────────────────────────────────────────────┐
│                    Google Ads Automation System                     │
├─────────────────────────────────────────────────────────────────────┤
│                                                                     │
│  ┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐     │
│  │   Vercel Cron   │────▶│    Scheduler    │────▶│   Google Ads    │     │
│  │  (15 min sync)  │     │     Service     │     │       API       │     │
│  └─────────────────┘     └─────────────────┘     └─────────────────┘     │
│           │                     │                      │              │
│           ▼                     ▼                      ▼              │
│  ┌─────────────────────────────────────────────────────────────┐     │
│  │                    Business Rules Engine                     │     │
│  │  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐       │     │
│  │  │ Vietnam Time │  │   Service    │  │   Pricing    │       │     │
│  │  │    Logic     │  │ Availability │  │   Engine     │       │     │
│  │  └──────────────┘  └──────────────┘  └──────────────┘       │     │
│  └─────────────────────────────────────────────────────────────┘     │
│                                                                     │
│  ┌─────────────────────────────────────────────────────────────┐     │
│  │                    Configuration Layer                       │     │
~~~~

~~~
│  │  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐      │     │
│  │  │  Countries   │  │   Airports   │  │ Ad Templates │      │     │
│  │  └──────────────┘  └──────────────┘  └──────────────┘      │     │
│  └─────────────────────────────────────────────────────────────┘     │
│                                                                       │
└──────────────────────────────────────────────────────────────────────┘
```

### 2.2 Technology Stack

| Component | Technology |
|-----------|------------|
| Runtime | Node.js 18+ with TypeScript |
| Framework | Next.js 14 (App Router) |
| Hosting | Vercel (Serverless) |
| Scheduling | Vercel Cron |
| Google Ads API | google-ads-api npm package |
| Authentication | OAuth 2.0 with refresh tokens |

---

## 3. Architecture Design

### 3.1 File Structure

```
app/src/
├── lib/
│   ├── google-ads/
│   │   ├── client.ts          # Google Ads API client initialization
│   │   ├── campaigns.ts       # Campaign CRUD operations
│   │   └── scheduler.ts       # Automated enable/disable logic
│   │
│   ├── business-rules/
│   │   ├── vietnam-time.ts    # Vietnam timezone handling
│   │   ├── service-availability.ts  # Service availability rules
│   │   └── pricing-engine.ts  # Tier-based pricing logic
│   │
│   └── config/
│       ├── countries.ts       # Country configurations & geo targets
│       ├── airports.ts        # Airport data for proximity targeting
│       └── ad-templates.ts    # Multi-language ad copy
│
├── app/api/
│   └── ads/
~~~

```
│       └── sync/
│           └── route.ts       # Cron endpoint for syncing ads
│
└── vercel.json                # Cron job configuration
```

### 3.2 Data Flow

```
1. Vercel Cron triggers /api/ads/sync every 15 minutes
          │
          ▼
2. System calculates current Vietnam time (UTC+7)
          │
          ▼
3. Business rules determine which services are available
          │
          ▼
4. For each campaign, system checks:
  - What service type does this campaign advertise?
  - Is that service currently available?
          │
          ▼
5. If mismatch detected (e.g., URGENT campaign enabled but office closed):
  - Update campaign status via Google Ads API
          │
          ▼
6. Log all changes for monitoring and audit
```

## 4. Core Components

### 4.1 Vietnam Time Module (`vietnam-time.ts`)

Purpose: Handles all timezone calculations for Vietnam (UTC+7)

Key Functions:

```typescript
// Get current Vietnam time and office status
getVietnamOfficeStatus(): OfficeStatus {
 isOpen: boolean;         // Is office currently open?
 isWeekend: boolean;      // Is it Saturday or Sunday?
```

~~~
 isHoliday: boolean;      // Is it a Vietnamese public holiday?
 currentTimeVN: Date;     // Current time in Vietnam
 currentTimeVNFormatted: string;  // Formatted for display
 nextOpenTime: Date | null;       // When does office next open?
}

// Check if current time is in weekend processing window
isWeekendWindow(): boolean

// Get next office opening time
getNextOfficeOpen(from: Date): Date
```

**Vietnam Public Holidays Supported (2024-2026)**:
- New Year's Day (January 1)
- Tết (Vietnamese Lunar New Year) - variable dates
- Hung Kings Commemoration Day (variable)
- Reunification Day (April 30)
- International Workers' Day (May 1)
- National Day (September 2)

### 4.2 Service Availability Module (`service-availability.ts`)

**Purpose**: Determines which visa services can be offered at any given time

**Service Types**:

| Service | Code | Availability |
|---------|------|--------------|
| 1-Hour Urgent | `URGENT_1H` | Office hours only (Mon-Fri 8-16 VN) |
| 2-Hour Urgent | `URGENT_2H` | Office hours only |
| 4-Hour Urgent | `URGENT_4H` | Office hours only |
| 1-Day Service | `1DAY` | Weekdays, with next-day processing |
| 2-Day Service | `2DAY` | Weekdays, with processing buffer |
| Weekend/Holiday | `WEEKEND` | Always available (premium) |
| Standard | `STANDARD` | Always available (3-5 business days) |

**Decision Logic**:

```typescript
function getAllServicesAvailability(): ServiceAvailability[] {
 const officeStatus = getVietnamOfficeStatus();

 // Weekend/Holiday logic
 if (officeStatus.isWeekend || officeStatus.isHoliday) {
~~~

```
    return [
      { service: 'URGENT_1H', available: false, reason: 'Office closed' },
      { service: 'URGENT_2H', available: false, reason: 'Office closed' },
      { service: 'URGENT_4H', available: false, reason: 'Office closed' },
      { service: '1DAY', available: false, reason: 'Office closed' },
      { service: '2DAY', available: false, reason: 'Office closed' },
      { service: 'WEEKEND', available: true },
      { service: 'STANDARD', available: true },
    ];
  }

  // Weekday office hours logic
  if (officeStatus.isOpen) {
    // All services available during office hours
    return ALL_SERVICES.map(s => ({ service: s, available: true }));
  }

  // After hours on weekday
  return [
    { service: 'URGENT_1H', available: false, reason: 'After hours' },
    // ... etc
  ];
}
```

### 4.3 Pricing Engine (`pricing-engine.ts`)

Purpose: Calculates service prices based on country tier and service type

Country Tiers:

Tier	Countries	Price Multiplier
TIER1	US, CA, AU, NZ, GB, DE	1.0x (highest)
TIER2	FR, IT, ES, NL, BE, AT, CH, SE, NO, DK, FI	0.85x
TIER3	IN, PH, ID, TH, MY, VN, KR, JP, SG	0.70x

Base Pricing Matrix:

Service	TIER1	TIER2	TIER3
URGENT_1H	$249	$199	$149
URGENT_2H	$199	$169	$129
URGENT_4H	$179	$149	$99
1DAY	$129	$99	$79

~~~~markdown
| 2DAY | $99 | $79 | $59 |
| WEEKEND | $299 | $249 | $199 |
| STANDARD | $79 | $59 | $49 |

---

## 5. Business Logic

### 5.1 Campaign Sync Logic

The core sync algorithm runs every 15 minutes:

```typescript
async function syncCampaignStatus(dryRun: boolean = false): Promise<SyncResult> {
 // 1. Get current service availability
 const availableServices = getAdvertisableServices();

 // 2. Fetch all campaigns from Google Ads
 const campaigns = await getAllCampaigns();

 // 3. For each campaign, determine if it should be enabled
 for (const campaign of campaigns) {
   const serviceType = parseServiceFromName(campaign.name);
   const shouldBeEnabled = isServiceAvailable(serviceType, availableServices);

   // 4. If status mismatch, update campaign
   if (campaign.status === 'ENABLED' && !shouldBeEnabled) {
     await updateCampaignStatus(campaign.resourceName, 'PAUSED');
   } else if (campaign.status === 'PAUSED' && shouldBeEnabled) {
     await updateCampaignStatus(campaign.resourceName, 'ENABLED');
   }
 }

 return result;
}
```

### 5.2 Campaign Naming Convention

Campaigns follow a strict naming convention for parsing:

```
{COUNTRY_CODE} - {LANGUAGE} - {AIRPORT_CODE|COUNTRY} - VietnamVisa - {SERVICE_TYPE}
```
~~~~

Examples:
- `US - EN - JFK - VietnamVisa - URGENT_1H`
- `DE - DE - COUNTRY - VietnamVisa - STANDARD`
- `AU - EN - SYD - VietnamVisa - WEEKEND`

### 5.3 Service Parsing from Campaign Name

```typescript
function parseServiceFromName(name: string): ServiceType | null {
 const lowerName = name.toLowerCase();

 const servicePatterns = {
   'URGENT_1H': ['urgent_1h', '1-hour', '1 hour', '30-min'],
   'URGENT_2H': ['urgent_2h', '2-hour', '2 hour'],
   'URGENT_4H': ['urgent_4h', '4-hour', '4 hour'],
   '1DAY': ['1day', '1-day', '1 day', 'next day'],
   '2DAY': ['2day', '2-day', '2 day'],
   'WEEKEND': ['weekend', 'holiday'],
   'STANDARD': ['standard', 'regular', 'normal'],
 };

 for (const [service, patterns] of Object.entries(servicePatterns)) {
   if (patterns.some(p => lowerName.includes(p))) {
     return service as ServiceType;
   }
 }

 return null;
}
```

## 6. API Integration

### 6.1 Google Ads API Client

Authentication Flow:

```typescript
// OAuth 2.0 credentials required
const credentials = {
 client_id: process.env.GOOGLE_ADS_CLIENT_ID,
 client_secret: process.env.GOOGLE_ADS_CLIENT_SECRET,
 developer_token: process.env.GOOGLE_ADS_DEVELOPER_TOKEN,
```

~~~
 refresh_token: process.env.GOOGLE_ADS_REFRESH_TOKEN,
 customer_id: process.env.GOOGLE_ADS_CUSTOMER_ID,
};

// Initialize client
const client = new GoogleAdsApi({
 client_id: credentials.client_id,
 client_secret: credentials.client_secret,
 developer_token: credentials.developer_token,
});

const customer = client.Customer({
 customer_id: credentials.customer_id,
 refresh_token: credentials.refresh_token,
});
```

### 6.2 API Operations

**Query Campaigns**:
```typescript
const campaigns = await customer.query(`
 SELECT
   campaign.id,
   campaign.name,
   campaign.status,
   campaign.resource_name,
   campaign_budget.amount_micros,
   metrics.impressions,
   metrics.clicks,
   metrics.conversions
 FROM campaign
 WHERE campaign.advertising_channel_type = 'SEARCH'
   AND campaign.status != 'REMOVED'
 ORDER BY campaign.name
`);
```

**Update Campaign Status**:
```typescript
await customer.campaigns.update({
 resource_name: campaignResourceName,
 status: 'ENABLED' | 'PAUSED',
});
```
~~~

### 6.3 Rate Limiting & Error Handling

```typescript
// Batch updates with error collection
async function batchUpdateCampaignStatus(
 updates: { resourceName: string; status: 'ENABLED' | 'PAUSED' }[]
): Promise<{ success: number; failed: number; errors: string[] }> {
 const errors: string[] = [];
 let success = 0;
 let failed = 0;

 for (const update of updates) {
   try {
     await customer.campaigns.update({
       resource_name: update.resourceName,
       status: update.status,
     });
     success++;
   } catch (error) {
     failed++;
     errors.push(`${update.resourceName}: ${error.message}`);
   }
 }

 return { success, failed, errors };
}
```

## 7. Campaign Management

### 7.1 Geographic Targeting

Country-Level Targeting:
```typescript
// Set country targeting for campaign
await customer.campaignCriteria.create({
 campaign: campaignResourceName,
 location: {
   geo_target_constant: `geoTargetConstants/${countryGeoId}`,
 },
});
```

~~~
**Airport Proximity Targeting** (5km radius):
```typescript
// Set airport proximity targeting
await customer.campaignCriteria.create({
 campaign: campaignResourceName,
 proximity: {
   geo_point: {
     latitude_in_micro_degrees: airport.lat * 1_000_000,
     longitude_in_micro_degrees: airport.lng * 1_000_000,
   },
   radius: 5000, // 5km
   radius_units: 'METERS',
 },
});
```

### 7.2 Supported Airports (50+)

Major international airports with high Vietnam travel demand:

**North America**: JFK, LAX, SFO, ORD, MIA, YYZ, YVR
**Europe**: LHR, CDG, FRA, AMS, MAD, FCO, MUC
**Asia-Pacific**: SYD, MEL, NRT, HND, ICN, SIN, HKG, BKK
**Middle East/India**: DXB, DEL, BOM, MAA

### 7.3 Language Targeting

Supported languages with full ad template coverage:

| Language | Code | Target Markets |
|----------|------|----------------|
| English | EN | US, CA, AU, NZ, UK, SG |
| Spanish | ES | ES, MX, AR, CO |
| Portuguese | PT | BR, PT |
| French | FR | FR, CA-QC, BE |
| Russian | RU | RU, UA, KZ |
| Hindi | HI | IN |

---

## 8. Security & Authentication

### 8.1 Environment Variables
~~~

~~~
All sensitive credentials are stored as environment variables:

```env
# Google Ads API Credentials
GOOGLE_ADS_CLIENT_ID=xxx.apps.googleusercontent.com
GOOGLE_ADS_CLIENT_SECRET=GOCSPX-xxx
GOOGLE_ADS_DEVELOPER_TOKEN=xxx
GOOGLE_ADS_REFRESH_TOKEN=1//xxx
GOOGLE_ADS_CUSTOMER_ID=1234567890

# Feature Flags
GOOGLE_ADS_ENABLED=true
GOOGLE_ADS_DRY_RUN=false

# Cron Security
CRON_SECRET=xxx
```

### 8.2 API Endpoint Security

```typescript
// Verify request is from Vercel Cron or has valid authorization
function isAuthorized(request: Request): boolean {
 const cronSecret = process.env.CRON_SECRET;

 // Check for Vercel Cron header (automatic)
 if (request.headers.get('x-vercel-cron')) {
   return true;
 }

 // Check for manual Authorization header
 const authHeader = request.headers.get('authorization');
 if (authHeader === `Bearer ${cronSecret}`) {
   return true;
 }

 return false;
}
```

### 8.3 Dry Run Mode

For testing without affecting live campaigns:

```typescript
~~~

~~~
// Enable dry run mode via environment variable
const isDryRun = process.env.GOOGLE_ADS_DRY_RUN === 'true';

if (isDryRun) {
 console.log('DRY RUN: Would update campaign', campaignName, 'to', newStatus);
 // Don't actually call Google Ads API
} else {
 await updateCampaignStatus(campaignResourceName, newStatus);
}
```

---

## 9. Deployment & Infrastructure

### 9.1 Vercel Configuration

**vercel.json**:
```json
{
 "crons": [
   {
     "path": "/api/ads/sync",
     "schedule": "*/15 * * * *"
   }
 ]
}
```

This configures the sync endpoint to run every 15 minutes.

### 9.2 Deployment Pipeline

```
1. Developer pushes to main branch
          │
          ▼
2. GitHub webhook triggers Vercel build
          │
          ▼
3. Vercel builds Next.js application
          │
          ▼
4. Vercel deploys to edge network
          │
~~~

```
          ▼
5. Cron jobs automatically configured
          |
          ▼
6. /api/ads/sync runs every 15 minutes
```

### 9.3 Environment Configuration

Environment	GOOGLE_ADS_DRY_RUN	Purpose
Development	true	Test without affecting real ads
Preview	true	PR preview deployments
Production	false	Live campaign management

## 10. Monitoring & Logging

### 10.1 Sync Result Logging

Every sync operation produces a detailed log:

```
============================================================
Google Ads Sync Report
============================================================
Timestamp: 2026-01-19T04:30:00.000Z
Vietnam Time: 2026-01-19 11:30:00
Office Open: YES
Weekend Window: NO
Dry Run: NO
------------------------------------------------------------
Service Availability:
 URGENT_1H: AVAILABLE
 URGENT_2H: AVAILABLE
 URGENT_4H: AVAILABLE
 1DAY: AVAILABLE
 2DAY: AVAILABLE
 WEEKEND: AVAILABLE
 STANDARD: AVAILABLE
------------------------------------------------------------
Campaign Summary:
 Processed: 15
 Enabled: 2
```

~~~
 Paused: 0
 Unchanged: 13
------------------------------------------------------------
Changes Made:
 [URGENT_1H] US - EN - JFK - VietnamVisa: PAUSED -> ENABLED
   Reason: Service URGENT_1H is available
 [URGENT_2H] US - EN - LAX - VietnamVisa: PAUSED -> ENABLED
   Reason: Service URGENT_2H is available
============================================================
```

### 10.2 Error Handling

```typescript
try {
 const result = await syncCampaignStatus(dryRun);
 console.log(formatSyncResultForLog(result));
 return NextResponse.json({ status: 'success', result });
} catch (error) {
 console.error('Ads sync failed:', error);
 return NextResponse.json({
   status: 'error',
   error: error.message,
   schedulerStatus: getSchedulerStatus(),
 }, { status: 500 });
}
```

### 10.3 Monitoring Endpoints

**GET /api/ads/sync?preview=true**
Returns what changes would be made without executing them.

**POST /api/ads/sync** with body `{ "action": "status" }`
Returns current scheduler status:
```json
{
 "status": "success",
 "action": "status",
 "schedulerStatus": {
   "vietnamTime": "2026-01-19 11:30:00",
   "officeStatus": { "isOpen": true, "isWeekend": false },
   "availableServices": ["URGENT_1H", "URGENT_2H", ...],
   "unavailableServices": []
 },
~~~

~~~~
 "googleAdsEnabled": true,
 "dryRunMode": false
}
```


---


## Appendix A: API Reference


### Sync Endpoint


**GET /api/ads/sync**


Query Parameters:
- `preview=true` - Preview changes without executing


**POST /api/ads/sync**


Body:
```json
{
 "action": "sync" | "preview" | "status" | "forceEnable" | "forcePause",
 "dryRun": boolean (optional)
}
```


### Response Format


```json
{
 "status": "success" | "error" | "skipped",
 "result": {
   "timestamp": "ISO8601",
   "vietnamTime": "formatted string",
   "campaignsProcessed": number,
   "campaignsEnabled": number,
   "campaignsPaused": number,
   "campaignsUnchanged": number,
   "changes": [...],
   "errors": [...],
   "dryRun": boolean
 }
}
```
~~~~

```
---

## Appendix B: Glossary

| Term | Definition |
|------|------------|
| Campaign | A Google Ads campaign containing ad groups and ads |
| Ad Group | A collection of ads with shared targeting |
| CPC | Cost Per Click - amount paid when ad is clicked |
| CPA | Cost Per Acquisition - target cost per conversion |
| Geo Target | Geographic location for ad targeting |
| MCC | Manager Account (My Client Center) |
| GAQL | Google Ads Query Language |
| UTC+7 | Vietnam timezone (Indochina Time) |

---

## Document Control

| Version | Date | Author | Changes |
|---------|------|--------|---------|
| 1.0 | 2026-01-19 | VietnamVisaHelp Dev Team | Initial release |

---

*This document is confidential and intended for Google Ads API access review
purposes.*
```

oogle Ads Automation Tool - Design Documentation

```
## VietnamVisaHelp.com

**Version:** 1.0
**Date:** January 19, 2026
**Company:** VietnamVisaHelp.com
**Contact:** support@vietnamvisahelp.com

---

## Table of Contents

1. [Executive Summary](#1-executive-summary)
2. [System Overview](#2-system-overview)
3. [Architecture Design](#3-architecture-design)
4. [Core Components](#4-core-components)
5. [Business Logic](#5-business-logic)
```

```markdown
6. [API Integration](#6-api-integration)
7. [Campaign Management](#7-campaign-management)
8. [Security & Authentication](#8-security--authentication)
9. [Deployment & Infrastructure](#9-deployment--infrastructure)
10. [Monitoring & Logging](#10-monitoring--logging)

---

## 1. Executive Summary

### 1.1 Purpose

The Google Ads Automation Tool is designed to automatically manage Google Ads campaigns for VietnamVisaHelp.com, a visa processing service for travelers to Vietnam. The tool ensures that advertisements are only displayed when the corresponding visa services are actually available, preventing customer frustration and wasted ad spend.

### 1.2 Key Features

- **Automated Campaign Scheduling**: Enables/disables campaigns based on Vietnam office hours (UTC+7)
- **Service-Based Ad Control**: Different visa services (urgent, standard, weekend) are advertised only when available
- **Multi-Language Support**: Ad templates in 6 languages (English, Spanish, Portuguese, French, Russian, Hindi)
- **Geographic Targeting**: Airport proximity targeting and country-level geo-targeting
- **Real-Time Sync**: Runs every 15 minutes via Vercel Cron to keep ads synchronized

### 1.3 Business Problem Solved

Vietnam visa processing services have specific availability windows:
- **Urgent services (1-hour, 2-hour, 4-hour)**: Only available during Vietnam office hours (8 AM - 4 PM, Monday-Friday)
- **Weekend/Holiday services**: Available but at premium pricing
- **Standard services**: Available 24/7

Without automation, ads for unavailable services could run, leading to:
- Customer complaints when services can't be delivered
- Wasted advertising budget
- Damaged brand reputation

---
```

## 2. System Overview

### 2.1 High-Level Architecture

```
┌─────────────────────────────────────────────────────────────────────┐
│                    Google Ads Automation System                     │
├─────────────────────────────────────────────────────────────────────┤
│                                                                     │
│  ┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐  │
│  │   Vercel Cron   │───▶│    Scheduler    │───▶│   Google Ads    │  │
│  │  (15 min sync)  │    │     Service     │    │      API        │  │
│  └─────────────────┘    └─────────────────┘    └─────────────────┘  │
│           │                      │                       │          │
│           ▼                      ▼                       ▼          │
│  ┌─────────────────────────────────────────────────────────────┐    │
│  │                    Business Rules Engine                    │    │
│  │  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐       │    │
│  │  │ Vietnam Time │  │   Service    │  │   Pricing    │       │    │
│  │  │    Logic     │  │ Availability │  │   Engine     │       │    │
│  │  └──────────────┘  └──────────────┘  └──────────────┘       │    │
│  └─────────────────────────────────────────────────────────────┘    │
│                                                                     │
│  ┌─────────────────────────────────────────────────────────────┐    │
│  │                    Configuration Layer                      │    │
│  │  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐       │    │
│  │  │  Countries   │  │   Airports   │  │ Ad Templates │       │    │
│  │  └──────────────┘  └──────────────┘  └──────────────┘       │    │
│  └─────────────────────────────────────────────────────────────┘    │
│                                                                     │
└─────────────────────────────────────────────────────────────────────┘
```

### 2.2 Technology Stack

Component	Technology
Runtime	Node.js 18+ with TypeScript
Framework	Next.js 14 (App Router)
Hosting	Vercel (Serverless)
Scheduling	Vercel Cron
Google Ads API	google-ads-api npm package
Authentication	OAuth 2.0 with refresh tokens

## 3. Architecture Design

### 3.1 File Structure

```
app/src/
├── lib/
│   ├── google-ads/
│   │   ├── client.ts           # Google Ads API client initialization
│   │   ├── campaigns.ts        # Campaign CRUD operations
│   │   └── scheduler.ts        # Automated enable/disable logic
│   │
│   ├── business-rules/
│   │   ├── vietnam-time.ts     # Vietnam timezone handling
│   │   ├── service-availability.ts  # Service availability rules
│   │   └── pricing-engine.ts   # Tier-based pricing logic
│   │
│   └── config/
│       ├── countries.ts        # Country configurations & geo targets
│       ├── airports.ts         # Airport data for proximity targeting
│       └── ad-templates.ts     # Multi-language ad copy
│
├── app/api/
│   └── ads/
│       └── sync/
│           └── route.ts        # Cron endpoint for syncing ads
│
└── vercel.json                 # Cron job configuration
```

### 3.2 Data Flow

```
1. Vercel Cron triggers /api/ads/sync every 15 minutes
          │
          ▼
2. System calculates current Vietnam time (UTC+7)
          │
          ▼
3. Business rules determine which services are available
          │
          ▼
4. For each campaign, system checks:
```

~~~markdown
  - What service type does this campaign advertise?
  - Is that service currently available?
          │
          ▼
5. If mismatch detected (e.g., URGENT campaign enabled but office closed):
  - Update campaign status via Google Ads API
          │
          ▼
6. Log all changes for monitoring and audit
```

---

## 4. Core Components

### 4.1 Vietnam Time Module (`vietnam-time.ts`)

**Purpose**: Handles all timezone calculations for Vietnam (UTC+7)

**Key Functions**:

```typescript
// Get current Vietnam time and office status
getVietnamOfficeStatus(): OfficeStatus {
 isOpen: boolean;         // Is office currently open?
 isWeekend: boolean;      // Is it Saturday or Sunday?
 isHoliday: boolean;      // Is it a Vietnamese public holiday?
 currentTimeVN: Date;     // Current time in Vietnam
 currentTimeVNFormatted: string;  // Formatted for display
 nextOpenTime: Date | null;       // When does office next open?
}

// Check if current time is in weekend processing window
isWeekendWindow(): boolean

// Get next office opening time
getNextOfficeOpen(from: Date): Date
```

**Vietnam Public Holidays Supported (2024-2026)**:
- New Year's Day (January 1)
- Tết (Vietnamese Lunar New Year) - variable dates
- Hung Kings Commemoration Day (variable)
- Reunification Day (April 30)
- International Workers' Day (May 1)
~~~

~~~
- National Day (September 2)

### 4.2 Service Availability Module (`service-availability.ts`)

**Purpose**: Determines which visa services can be offered at any given time

**Service Types**:

| Service | Code | Availability |
|---------|------|--------------|
| 1-Hour Urgent | `URGENT_1H` | Office hours only (Mon-Fri 8-16 VN) |
| 2-Hour Urgent | `URGENT_2H` | Office hours only |
| 4-Hour Urgent | `URGENT_4H` | Office hours only |
| 1-Day Service | `1DAY` | Weekdays, with next-day processing |
| 2-Day Service | `2DAY` | Weekdays, with processing buffer |
| Weekend/Holiday | `WEEKEND` | Always available (premium) |
| Standard | `STANDARD` | Always available (3-5 business days) |

**Decision Logic**:

```typescript
function getAllServicesAvailability(): ServiceAvailability[] {
  const officeStatus = getVietnamOfficeStatus();

  // Weekend/Holiday logic
  if (officeStatus.isWeekend || officeStatus.isHoliday) {
    return [
      { service: 'URGENT_1H', available: false, reason: 'Office closed' },
      { service: 'URGENT_2H', available: false, reason: 'Office closed' },
      { service: 'URGENT_4H', available: false, reason: 'Office closed' },
      { service: '1DAY', available: false, reason: 'Office closed' },
      { service: '2DAY', available: false, reason: 'Office closed' },
      { service: 'WEEKEND', available: true },
      { service: 'STANDARD', available: true },
    ];
  }

  // Weekday office hours logic
  if (officeStatus.isOpen) {
    // All services available during office hours
    return ALL_SERVICES.map(s => ({ service: s, available: true }));
  }

  // After hours on weekday
  return [
~~~

```typescript
   { service: 'URGENT_1H', available: false, reason: 'After hours' },
   // ... etc
 ];
}
```

### 4.3 Pricing Engine (`pricing-engine.ts`)

Purpose: Calculates service prices based on country tier and service type

Country Tiers:

Tier	Countries	Price Multiplier
TIER1	US, CA, AU, NZ, GB, DE	1.0x (highest)
TIER2	FR, IT, ES, NL, BE, AT, CH, SE, NO, DK, FI	0.85x
TIER3	IN, PH, ID, TH, MY, VN, KR, JP, SG	0.70x

Base Pricing Matrix:

Service	TIER1	TIER2	TIER3
URGENT_1H	$249	$199	$149
URGENT_2H	$199	$169	$129
URGENT_4H	$179	$149	$99
1DAY	$129	$99	$79
2DAY	$99	$79	$59
WEEKEND	$299	$249	$199
STANDARD	$79	$59	$49

## 5. Business Logic

### 5.1 Campaign Sync Logic

The core sync algorithm runs every 15 minutes:

```typescript
async function syncCampaignStatus(dryRun: boolean = false): Promise<SyncResult> {
 // 1. Get current service availability
 const availableServices = getAdvertisableServices();

 // 2. Fetch all campaigns from Google Ads
 const campaigns = await getAllCampaigns();
 // 3. For each campaign, determine if it should be enabled
 for (const campaign of campaigns) {
   const serviceType = parseServiceFromName(campaign.name);
   const shouldBeEnabled = isServiceAvailable(serviceType, availableServices);

   // 4. If status mismatch, update campaign
   if (campaign.status === 'ENABLED' && !shouldBeEnabled) {
     await updateCampaignStatus(campaign.resourceName, 'PAUSED');
   } else if (campaign.status === 'PAUSED' && shouldBeEnabled) {
     await updateCampaignStatus(campaign.resourceName, 'ENABLED');
   }
 }

 return result;
}
```

### 5.2 Campaign Naming Convention

Campaigns follow a strict naming convention for parsing:

```
{COUNTRY_CODE} - {LANGUAGE} - {AIRPORT_CODE|COUNTRY} - VietnamVisa - {SERVICE_TYPE}
```

Examples:
- `US - EN - JFK - VietnamVisa - URGENT_1H`
- `DE - DE - COUNTRY - VietnamVisa - STANDARD`
- `AU - EN - SYD - VietnamVisa - WEEKEND`

### 5.3 Service Parsing from Campaign Name

```typescript
function parseServiceFromName(name: string): ServiceType | null {
 const lowerName = name.toLowerCase();

 const servicePatterns = {
   'URGENT_1H': ['urgent_1h', '1-hour', '1 hour', '30-min'],
   'URGENT_2H': ['urgent_2h', '2-hour', '2 hour'],
   'URGENT_4H': ['urgent_4h', '4-hour', '4 hour'],
   '1DAY': ['1day', '1-day', '1 day', 'next day'],
   '2DAY': ['2day', '2-day', '2 day'],
   'WEEKEND': ['weekend', 'holiday'],
   'STANDARD': ['standard', 'regular', 'normal'],
 };

 for (const [service, patterns] of Object.entries(servicePatterns)) {
   if (patterns.some(p => lowerName.includes(p))) {
     return service as ServiceType;
   }
 }

 return null;
}
```

## 6. API Integration

### 6.1 Google Ads API Client

Authentication Flow:

```typescript
// OAuth 2.0 credentials required
const credentials = {
 client_id: process.env.GOOGLE_ADS_CLIENT_ID,
 client_secret: process.env.GOOGLE_ADS_CLIENT_SECRET,
 developer_token: process.env.GOOGLE_ADS_DEVELOPER_TOKEN,
 refresh_token: process.env.GOOGLE_ADS_REFRESH_TOKEN,
 customer_id: process.env.GOOGLE_ADS_CUSTOMER_ID,
};

// Initialize client
const client = new GoogleAdsApi({
 client_id: credentials.client_id,
 client_secret: credentials.client_secret,
 developer_token: credentials.developer_token,
});

const customer = client.Customer({
 customer_id: credentials.customer_id,
 refresh_token: credentials.refresh_token,
});
```

### 6.2 API Operations

~~~markdown
**Query Campaigns**:
```typescript
const campaigns = await customer.query(`
 SELECT
   campaign.id,
   campaign.name,
   campaign.status,
   campaign.resource_name,
   campaign_budget.amount_micros,
   metrics.impressions,
   metrics.clicks,
   metrics.conversions
 FROM campaign
 WHERE campaign.advertising_channel_type = 'SEARCH'
   AND campaign.status != 'REMOVED'
 ORDER BY campaign.name
`);
```

**Update Campaign Status**:
```typescript
await customer.campaigns.update({
 resource_name: campaignResourceName,
 status: 'ENABLED' | 'PAUSED',
});
```

### 6.3 Rate Limiting & Error Handling

```typescript
// Batch updates with error collection
async function batchUpdateCampaignStatus(
 updates: { resourceName: string; status: 'ENABLED' | 'PAUSED' }[]
): Promise<{ success: number; failed: number; errors: string[] }> {
 const errors: string[] = [];
 let success = 0;
 let failed = 0;

 for (const update of updates) {
   try {
     await customer.campaigns.update({
       resource_name: update.resourceName,
       status: update.status,
     });
     success++;
~~~

```
    } catch (error) {
      failed++;
      errors.push(`${update.resourceName}: ${error.message}`);
    }
  }

  return { success, failed, errors };
}
```

## 7. Campaign Management

### 7.1 Geographic Targeting

Country-Level Targeting:
```typescript
// Set country targeting for campaign
await customer.campaignCriteria.create({
  campaign: campaignResourceName,
  location: {
    geo_target_constant: `geoTargetConstants/${countryGeoId}`,
  },
});
```

Airport Proximity Targeting (5km radius):
```typescript
// Set airport proximity targeting
await customer.campaignCriteria.create({
  campaign: campaignResourceName,
  proximity: {
    geo_point: {
      latitude_in_micro_degrees: airport.lat * 1_000_000,
      longitude_in_micro_degrees: airport.lng * 1_000_000,
    },
    radius: 5000, // 5km
    radius_units: 'METERS',
  },
});
```

### 7.2 Supported Airports (50+)

Major international airports with high Vietnam travel demand:

North America: JFK, LAX, SFO, ORD, MIA, YYZ, YVR
Europe: LHR, CDG, FRA, AMS, MAD, FCO, MUC
Asia-Pacific: SYD, MEL, NRT, HND, ICN, SIN, HKG, BKK
Middle East/India: DXB, DEL, BOM, MAA

### 7.3 Language Targeting

Supported languages with full ad template coverage:

Language	Code	Target Markets
English	EN	US, CA, AU, NZ, UK, SG
Spanish	ES	ES, MX, AR, CO
Portuguese	PT	BR, PT
French	FR	FR, CA-QC, BE
Russian	RU	RU, UA, KZ
Hindi	HI	IN

## 8. Security & Authentication

### 8.1 Environment Variables

All sensitive credentials are stored as environment variables:

```env
# Google Ads API Credentials
GOOGLE_ADS_CLIENT_ID=xxx.apps.googleusercontent.com
GOOGLE_ADS_CLIENT_SECRET=GOCSPX-xxx
GOOGLE_ADS_DEVELOPER_TOKEN=xxx
GOOGLE_ADS_REFRESH_TOKEN=1//xxx
GOOGLE_ADS_CUSTOMER_ID=1234567890

# Feature Flags
GOOGLE_ADS_ENABLED=true
GOOGLE_ADS_DRY_RUN=false

# Cron Security
CRON_SECRET=xxx
```

### 8.2 API Endpoint Security

```typescript
// Verify request is from Vercel Cron or has valid authorization
function isAuthorized(request: Request): boolean {
 const cronSecret = process.env.CRON_SECRET;

 // Check for Vercel Cron header (automatic)
 if (request.headers.get('x-vercel-cron')) {
   return true;
 }

 // Check for manual Authorization header
 const authHeader = request.headers.get('authorization');
 if (authHeader === `Bearer ${cronSecret}`) {
   return true;
 }

 return false;
}
```

### 8.3 Dry Run Mode

For testing without affecting live campaigns:

```typescript
// Enable dry run mode via environment variable
const isDryRun = process.env.GOOGLE_ADS_DRY_RUN === 'true';

if (isDryRun) {
 console.log('DRY RUN: Would update campaign', campaignName, 'to', newStatus);
 // Don't actually call Google Ads API
} else {
 await updateCampaignStatus(campaignResourceName, newStatus);
}
```

## 9. Deployment & Infrastructure

### 9.1 Vercel Configuration

vercel.json:
```json
```

```
{
 "crons": [
   {
     "path": "/api/ads/sync",
     "schedule": "*/15 * * * *"
   }
 ]
}
```

This configures the sync endpoint to run every 15 minutes.

### 9.2 Deployment Pipeline

```
1. Developer pushes to main branch
          │
          ▼
2. GitHub webhook triggers Vercel build
          │
          ▼
3. Vercel builds Next.js application
          │
          ▼
4. Vercel deploys to edge network
          │
          ▼
5. Cron jobs automatically configured
          │
          ▼
6. /api/ads/sync runs every 15 minutes
```

### 9.3 Environment Configuration

Environment	GOOGLE_ADS_DRY_RUN	Purpose
Development	true	Test without affecting real ads
Preview	true	PR preview deployments
Production	false	Live campaign management

## 10. Monitoring & Logging

### 10.1 Sync Result Logging

Every sync operation produces a detailed log:

```
============================================================
Google Ads Sync Report
============================================================
Timestamp: 2026-01-19T04:30:00.000Z
Vietnam Time: 2026-01-19 11:30:00
Office Open: YES
Weekend Window: NO
Dry Run: NO
------------------------------------------------------------
Service Availability:
 URGENT_1H: AVAILABLE
 URGENT_2H: AVAILABLE
 URGENT_4H: AVAILABLE
 1DAY: AVAILABLE
 2DAY: AVAILABLE
 WEEKEND: AVAILABLE
 STANDARD: AVAILABLE
------------------------------------------------------------
Campaign Summary:
 Processed: 15
 Enabled: 2
 Paused: 0
 Unchanged: 13
------------------------------------------------------------
Changes Made:
 [URGENT_1H] US - EN - JFK - VietnamVisa: PAUSED -> ENABLED
   Reason: Service URGENT_1H is available
 [URGENT_2H] US - EN - LAX - VietnamVisa: PAUSED -> ENABLED
   Reason: Service URGENT_2H is available
============================================================
```

### 10.2 Error Handling

```typescript
try {
 const result = await syncCampaignStatus(dryRun);
 console.log(formatSyncResultForLog(result));
 return NextResponse.json({ status: 'success', result });
} catch (error) {
 console.error('Ads sync failed:', error);
 return NextResponse.json({
   status: 'error',
   error: error.message,
   schedulerStatus: getSchedulerStatus(),
 }, { status: 500 });
}
```

### 10.3 Monitoring Endpoints

GET /api/ads/sync?preview=true
Returns what changes would be made without executing them.

POST /api/ads/sync with body `{ "action": "status" }`
Returns current scheduler status:
```json
{
 "status": "success",
 "action": "status",
 "schedulerStatus": {
   "vietnamTime": "2026-01-19 11:30:00",
   "officeStatus": { "isOpen": true, "isWeekend": false },
   "availableServices": ["URGENT_1H", "URGENT_2H", ...],
   "unavailableServices": []
 },
 "googleAdsEnabled": true,
 "dryRunMode": false
}
```

## Appendix A: API Reference

### Sync Endpoint

GET /api/ads/sync

Query Parameters:
- `preview=true` - Preview changes without executing

POST /api/ads/sync

Body:

~~~
```json
{
 "action": "sync" | "preview" | "status" | "forceEnable" | "forcePause",
 "dryRun": boolean (optional)
}
```

### Response Format

```json
{
 "status": "success" | "error" | "skipped",
 "result": {
   "timestamp": "ISO8601",
   "vietnamTime": "formatted string",
   "campaignsProcessed": number,
   "campaignsEnabled": number,
   "campaignsPaused": number,
   "campaignsUnchanged": number,
   "changes": [...],
   "errors": [...],
   "dryRun": boolean
 }
}
```

---

## Appendix B: Glossary

| Term | Definition |
|------|------------|
| Campaign | A Google Ads campaign containing ad groups and ads |
| Ad Group | A collection of ads with shared targeting |
| CPC | Cost Per Click - amount paid when ad is clicked |
| CPA | Cost Per Acquisition - target cost per conversion |
| Geo Target | Geographic location for ad targeting |
| MCC | Manager Account (My Client Center) |
| GAQL | Google Ads Query Language |
| UTC+7 | Vietnam timezone (Indochina Time) |

---

## Document Control
~~~

```
| Version | Date | Author | Changes |
|---------|------|--------|---------|
| 1.0 | 2026-01-19 | VietnamVisaHelp Dev Team | Initial release |

---

*This document is confidential and intended for Google Ads API access review
purposes.*
```