Third-Party Services & Dependencies Report

beprepared.ai Emergency Planner Platform

Generated: December 18, 2025 **Version:** 5.1.0

Executive Summary

This document provides a comprehensive inventory of all external third-party services integrated into the beprepared.ai platform. The application relies on 11 primary external services across AI, mapping, payments, email, authentication, scraping, and database infrastructure categories.

Total Services: 11 active services + 2 optional services **Monthly Cost Estimate:** $150-500 (varies by usage) **Critical Services:** Supabase, OpenRouter, Stripe, Google Maps APIs

Quick Reference: Services Summary

Service	Category	Criticality	Monthly Cost	Management Console
Supabase	Database & Auth	Critical	$25+	https://app.supabase.com
OpenRouter	AI/LLM	Critical	$50-200	https://openrouter.ai/dashboard
Google Cloud Platform	Maps & Geocoding	Critical	$50-150	https://console.cloud.google.com
Stripe	Payments	Critical	2.9% + $0.30/txn	https://dashboard.stripe.com
Resend	Email Delivery	High	$20/mo (10K emails)	https://resend.com/dashboard
Firecrawl	Web Scraping	Medium	$50-100	https://firecrawl.dev/dashboard
Decodo	Amazon Scraping	Medium	Usage-based	https://decodo.com/dashboard
SerpAPI	Search API	Low	$50/mo (5K searches)	https://serpapi.com/dashboard
WeatherAPI	Weather Data	Low	Free tier used	https://weatherapi.com/dashboard

Service	Category	Criticality	Monthly Cost	Management Console
Amazon PA-API	Product Data	Optional	Free (affiliate)	https://affiliate-program.amazon.com
Google Gemini	AI/LLM	Optional	Usage-based	https://makersuite.google.com
Zoom	Video API	Optional	Not active	https://marketplace.zoom.us
Google Custom Search	Search Engine	Optional	Not active	https://programmablesearchengine.google.com

Service Details

1. Supabase (Database & Authentication) CRITICAL

Management Console: https://app.supabase.com **Documentation:** https://supabase.com/docs

Environment Variables

```
# Database Connection
DATABASE_URL=postgres://postgres:[PASSWORD]@db.[PROJECT-
ID].supabase.co:6543/postgres

# Supabase Project Settings
NEXT_PUBLIC_SUPABASE_URL=https://[PROJECT-ID].supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJ...
SUPABASE_SERVICE_ROLE_KEY=eyJ...
```

Service Description

Supabase provides the complete backend infrastructure for the application:

- **PostgreSQL Database:** All application data storage (users, plans, products, billing)
- **Authentication:** Email-based OTP authentication system (passwordless by default)
- **Storage:** File storage for user uploads and generated content
- **Row Level Security (RLS):** Database-level access control

Usage Throughout Application

Authentication System:

- Email OTP verification flow: src/app/auth/verify-email/page.tsx
- Login/signup actions: src/app/actions/auth.ts
- Session management: src/utils/supabase/server.ts, src/utils/supabase/client.ts

Database Operations (via Drizzle ORM):

- User profiles and roles: src/db/schema/profiles.ts
- Mission plans: src/db/schema/plans.ts
- Billing history: src/db/queries/billing.ts
- Product catalog: src/db/schema/commerce.ts

Implementation Pattern:

```
// Server-side (async required)
const supabase = await createClient();
const { data: { user } } = await supabase.auth.getUser();

// Client-side
const supabase = createClient();
```

Cost Considerations

- **Free Tier:** 500MB database, 1GB bandwidth, 50,000 monthly active users
- **Pro Plan:** $25/month (8GB database, 250GB bandwidth, 100K MAU)
- **Current Usage:** Pro plan recommended for production

---

2. OpenRouter (AI Language Models) □ CRITICAL

Management Console: https://openrouter.ai/dashboard **Documentation:** https://openrouter.ai/docs **API Reference:** https://openrouter.ai/docs/api-reference

Environment Variables

```
OPENROUTER_API_KEY=sk-or-v1-...
```

Service Description

OpenRouter provides unified access to multiple AI language models. The application primarily uses Claude 3.5 Sonnet for generating comprehensive emergency preparedness plans and content.

Primary Model: `anthropic/claude-3.5-sonnet` (default) **Alternative Models:**

- `anthropic/claude-opus-4` (highest quality)
- `anthropic/claude-3-haiku` (cost-effective)

Usage Throughout Application

Mission Plan Generation:

- Main mission generator: src/lib/ai/mission-generator.ts

- Streaming generation: src/app/actions/generate-mission-streaming.ts
- Model configuration: src/lib/openrouter.ts

Emergency Contacts Generation:

- Contact generator: src/lib/ai/emergency-contacts-generator.ts
- Regeneration action: src/app/actions/regenerate-emergency-contacts.ts

Bundle Recommendations:

- Bundle context building: src/lib/ai/bundle-context.ts
- Product recommendations integrated into plan generation

AI Usage Logging:

- All API calls tracked: src/lib/ai/usage-logger.ts
- Stored in `calls` table with tokens, duration, cost, and model used

Implementation Pattern:

```
import { getMissionGenerationModel } from '@/lib/openrouter';
import { generateText } from 'ai';

const model = getMissionGenerationModel();
const result = await generateText({
  model,
  prompt: megaPrompt,
  system: systemPrompt,
});
```

Cost Considerations

- **Claude 3.5 Sonnet:** ~$3 per million input tokens, ~$15 per million output tokens
- **Typical Plan Generation:** 15K-30K tokens (~$0.50-$1.00 per plan)
- **Estimated Monthly Cost:** $50-200 depending on usage volume
- **Cost Tracking:** Built-in via `calls` table in database

---

3. Google Cloud Platform APIs CRITICAL

Management Console: https://console.cloud.google.com **API Dashboard:** https://console.cloud.google.com/apis/dashboard **Documentation:** https://cloud.google.com/maps-platform/docs

Environment Variables

```
NEXT_PUBLIC_GOOGLE_SERVICES_API_KEY=AIza...
GOOGLE_CUSTOM_SEARCH_CX=... # Optional
```

Required APIs (Must be enabled in console)

1. **Maps JavaScript API** - Map rendering and display
2. **Geocoding API** - Convert addresses to coordinates
3. **Routes API** - Compute driving routes with waypoints
4. **Places API (New)** - Emergency services location search
5. **Directions API** - Route directions (complementary to Routes API)

Service Description

Google Cloud Platform provides comprehensive mapping, geocoding, and location services essential for the emergency planning features.

Usage Throughout Application

Map Display & Visualization:

- Map component: src/components/plans/map/MapComponent.tsx
- Map tab in plan details: src/components/plans/plan-details/MapTab.tsx
- Maps JavaScript API loaded via `@react-google-maps/api`

Location Autocomplete:

- Wizard location input: src/components/plans/wizard/LocationAutocomplete.tsx
- Uses Places API autocomplete for scenario location selection
- Detects climate zones from coordinates: src/lib/climate.ts

Geocoding (Address → Coordinates):

- Geocoding utility: src/lib/geocoding.ts
- Route waypoint geocoding: src/lib/google-routes.ts
- Converts user-entered waypoint names to lat/lng coordinates

Route Computation:

- Routes API integration: src/lib/google-routes.ts
- Evacuation route generation: src/lib/mission-generation/evacuation-routes.ts
- Computes drivable routes with multiple waypoints
- Includes polyline encoding/decoding for route visualization

Emergency Services Nearby Search:

- Places API integration: src/lib/google-places.ts
- Emergency contacts tab: src/components/plans/plan-details/EmergencyContactsTab.tsx
- Meeting locations tab: src/components/plans/plan-details/MeetingLocationsTab.tsx
- Searches for hospitals, police, fire stations, pharmacies within 10-mile radius

Implementation Example:

```
// Places API - Emergency Services
const services = await fetchEmergencyServices(lat, lng, radius, [
  'hospital', 'police', 'fire_station', 'pharmacy'
]);

// Geocoding API - Waypoint Conversion
const coords = await geocodeWaypoint('Central Park, New York, NY');

// Routes API - Route Computation
const route = await computeRouteWithWaypoints(geocodedWaypoints);
```

Cost Considerations

- **Maps JavaScript API:** $7 per 1,000 loads (first 28,000 free/month)
- **Geocoding API:** $5 per 1,000 requests (first 40,000 free/month)
- **Routes API:** $5-10 per 1,000 requests (SKU-dependent)
- **Places API:** $17-32 per 1,000 requests (field-dependent)
- **Free Monthly Credits:** $200 for eligible new customers
- **Estimated Monthly Cost:** $50-150 depending on plan generation volume

---

4. Stripe (Payment Processing) □ CRITICAL

Management Console: https://dashboard.stripe.com **Customer Portal:** Configured per account
Documentation: https://stripe.com/docs

Environment Variables

```
# Stripe Keys
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_...
STRIPE_SECRET_KEY=sk_test_...
STRIPE_WEBHOOK_SECRET=whsec_...

# Price IDs
STRIPE_BASIC_PRICE_ID=price_...
STRIPE_PRO_PRICE_ID=price_...

# Customer Portal URL
STRIPE_CUSTOMER_PORTAL_URL=https://billing.stripe.com/p/login/test_...
```

Service Description

Stripe handles all payment processing, subscription management, and billing operations for the platform's Basic and Pro subscription tiers.

Subscription Tiers:

- **Basic Plan:** Limited features (5 plans/month)
- **Pro Plan:** Full features (unlimited plans, priority support)

Usage Throughout Application

Subscription Management:

- Stripe client configuration: src/lib/stripe.ts
- Subscription actions: src/app/actions/subscriptions.ts
- Profile subscription card: src/components/profile/SubscriptionCard.tsx

Webhook Handling:

- Stripe webhook endpoint: src/app/api/webhooks/stripe/route.ts
- Handles subscription events: `customer.subscription.created`, `customer.subscription.updated`, `customer.subscription.deleted`, `invoice.payment_succeeded`, `invoice.payment_failed`

Billing History:

- Billing queries: src/db/queries/billing.ts
- Billing history tab: src/components/profile/BillingHistoryTab.tsx
- CSV export: src/app/api/export/billing-csv/route.ts

Admin User Management:

- Admin user details: src/components/admin/UserDetailModal.tsx
- Subscription sync cron: src/app/api/cron/sync-subscriptions/route.ts

Implementation Pattern:

```
import { stripe, STRIPE_CONFIG } from '@/lib/stripe';

// Create checkout session
const session = await stripe.checkout.sessions.create({
  customer: stripeCustomerId,
  line_items: [{ price: STRIPE_CONFIG.proPriceId, quantity: 1 }],
  mode: 'subscription',
  success_url: `${siteUrl}/profile?success=true`,
  cancel_url: `${siteUrl}/profile?canceled=true`,
});
```

Cost Considerations

- **Transaction Fee:** 2.9% + $0.30 per successful charge
- **Subscription Management:** Included (no additional fee)
- **Billing Portal:** Included
- **No Monthly Fee:** Pay only for successful transactions
- **Estimated Monthly Cost:** Variable based on subscription revenue

5. Resend (Transactional Email) 🟡 HIGH PRIORITY

Management Console: https://resend.com/dashboard **Documentation:** https://resend.com/docs

Environment Variables

```
RESEND_API_KEY=re_...
FROM_EMAIL=your-email@domain.com
ADMIN_EMAIL=admin@domain.com
```

Service Description

Resend provides transactional email delivery for authentication, notifications, and user communications.

Usage Throughout Application

Authentication Emails:

- Email service: src/lib/email.ts
- OTP verification codes: `sendVerificationEmail()`
- Password reset links: `sendPasswordResetEmail()`
- Welcome emails: `sendWelcomeEmail()`
- Password change confirmations: `sendPasswordChangeConfirmationEmail()`

Plan Sharing:

- Share invitation emails: `sendPlanShareEmail()` - src/lib/email.ts
- Access notification emails: `sendShareAccessNotification()` - src/lib/email.ts
- Share modal: src/components/plans/modals/SharePlanModal.tsx

Admin Notifications:

- Manual verification requests: `sendManualVerificationNotification()` - src/lib/email.ts
- Admin notifications utility: src/lib/admin-notifications.ts

Email Flow Examples:

```
// 1. User signs up → OTP email sent
await sendVerificationEmail(email, code);

// 2. User shares plan → Recipient receives invitation
await sendPlanShareEmail(recipientEmail, {
  shareToken, planTitle, senderName, permissions: 'view'
});

// 3. Shared plan accessed → Owner notified
await sendShareAccessNotification(ownerEmail, {
  planTitle, accessedByEmail, accessedAt, permissions: 'view'
});
```

```
```

Cost Considerations

- **Free Tier:** 100 emails/day, 3,000 emails/month
- **Pro Plan:** $20/month for 50,000 emails
- **Overage:** $1 per 1,000 additional emails
- **Estimated Monthly Cost:** $20-40 depending on user growth

---

6. Firecrawl (Web Scraping) 🟡 MEDIUM PRIORITY

Management Console: https://firecrawl.dev/dashboard **Documentation:** https://docs.firecrawl.dev

Environment Variables

```
FIRECRAWL_API_KEY=fc-...
```

Service Description

Firecrawl is an AI-powered web scraping service that extracts structured product data from non-Amazon websites. Used for importing product information into the admin product catalog.

Extraction Schema:

- Product name, price, description
- Images, brand, SKU
- Color, size, dimensions, weight
- Model number, UPC

Usage Throughout Application

Product Data Extraction:

- Firecrawl utility: src/lib/firecrawl.ts
- Product edit dialog: src/app/(protected)/admin/products/components/ProductEditDialog.tsx
- Product admin actions: src/app/(protected)/admin/products/actions.ts
- API endpoint: src/app/api/firecrawl/extract/route.ts

Admin Workflow:

1. Admin pastes non-Amazon product URL
2. Firecrawl extracts product data using AI
3. Structured data pre-fills product form fields
4. Admin reviews and saves to catalog

Implementation Example:

```
```

```
const result = await firecrawl.extract({
  urls: [productUrl],
  prompt: 'Extract product information from this product page',
  schema: productSchema,
});
```

Cost Considerations

- **Free Tier:** 500 credits/month
- **Starter Plan:** $50/month for 5,000 credits
- **Pro Plan:** $100/month for 20,000 credits
- **1 Credit:** ~1 page extraction
- **Estimated Monthly Cost:** $50-100 depending on product import volume

---

7. Decodo (Amazon Web Scraping) 🟡 MEDIUM PRIORITY

Management Console: https://decodo.com/dashboard **API Documentation:** https://scraper-api.decodo.com/docs **Authentication:** Basic Auth with username/password

Environment Variables

```
DECODO_USER=U0000...
DECODO_PASS=PW_...
```

Service Description

Decodo Web Core Scraping API provides asynchronous web scraping services specifically for Amazon product pages. It bypasses Amazon's anti-bot measures to extract product details, pricing, and availability.

API Endpoints:

- **Asynchronous:** `https://scraper-api.decodo.com/v2/task` (task creation)
- **Synchronous:** `https://scraper-api.decodo.com/v2/scrape` (real-time)

Usage Throughout Application

Amazon Product Scraping:

- Decodo utility: src/lib/decodo.ts
- Product edit dialog: src/app/(protected)/admin/products/components/ProductEditDialog.tsx
- Product admin actions: src/app/(protected)/admin/products/actions.ts

Scraping Workflow:

1. Admin enters Amazon product URL or ASIN
2. Decodo task created via async endpoint

3. System polls for results (up to 30 attempts)
4. Product data parsed from HTML response
5. Form fields auto-populated with extracted data

Fallback Chain:

1. **Primary:** Amazon PA-API (if configured and available)
2. **Secondary:** Decodo scraping API (current implementation)
3. **Tertiary:** Direct HTML scraping with rotating user agents

Implementation Example:

```
const task = await createDecodoTask({
  url: amazonUrl,
  render: true,
  wait_for: 'network_idle'
});
const result = await pollDecodoTask(task.id);
```

Cost Considerations

- **Pricing Model:** Usage-based (per successful scrape)
- **Typical Cost:** ~$0.01-0.05 per successful scrape
- **Estimated Monthly Cost:** $20-50 depending on product import frequency

---

8. SerpAPI (Search Engine Results) 🟢 LOW PRIORITY

Management Console: https://serpapi.com/dashboard **Documentation:** https://serpapi.com/docs

Environment Variables

```
SERPAPI_API_KEY=8282e3331b7769c1659a836f28de37878dca8c12d2df2057a8d872d6c05
9348d
```

Service Description

SerpAPI provides programmatic access to Google search results for content discovery and research features. Currently referenced in environment variables but implementation status unclear.

Potential Use Cases:

- Emergency preparedness article discovery
- YouTube video recommendations
- Product availability checking
- Location-specific resource searches

Usage Throughout Application

Status: Limited or experimental usage detected

- API endpoint: src/app/api/search/route.ts
- Scraper utility references: src/lib/scraper.ts

Note: This service appears to be configured but may not be actively used in core features. Review usage logs to determine if API key can be deactivated.

Cost Considerations

- **Free Tier:** 100 searches/month
- **Developer Plan:** $50/month for 5,000 searches
- **Production Plan:** $250/month for 30,000 searches
- **Estimated Monthly Cost:** $0-50 (if used minimally)

---

9. WeatherAPI (Weather Data) 🟢 LOW PRIORITY

Management Console: https://weatherapi.com/dashboard **Documentation:** https://www.weatherapi.com/docs

Environment Variables

```
NEXT_PUBLIC_WEATHERAPI_API_KEY=10358d99a84045e4b3943742251112
```

Service Description

WeatherAPI provides current weather conditions, forecasts, and historical weather data. Used in the mission planning wizard to display weather context for the scenario location.

Usage Throughout Application

Weather Display:

- Weather station component: src/components/plans/wizard/WeatherStation.tsx
- Displays current temperature, conditions, and forecast during plan creation

LLM Service Integration:

- Python service: LLM_service/app/workflows/services/weatherapi.py
- Emergency contacts workflow: LLM_service/workflows/definitions/emergency_contacts.json

Implementation Example:

```
const response = await fetch(
  `https://api.weatherapi.com/v1/current.json?
```

```
  key=${apiKey}&q=${lat},${lng}`
  );
  const weather = await response.json();
```

Cost Considerations

- **Free Tier:** 1,000,000 calls/month (sufficient for most use cases)
- **Paid Plans:** $4-10/month for additional features
- **Current Estimate:** Free tier adequate
- **Estimated Monthly Cost:** $0 (free tier)

---

10. Amazon Product Advertising API (PA-API) ○ OPTIONAL

Management Console: https://affiliate-program.amazon.com/assoc_credentials/home **Documentation:** https://webservices.amazon.com/paapi5/documentation

Environment Variables

```
AMAZON_ACCESS_KEY=not_set
AMAZON_SECRET_KEY=not_set
AMAZON_PARTNER_TAG=not_set
AMAZON_REGION=us-east-1
AMAZON_TLD=com
AMAZON_SCRAPING_MODE=paapi # or 'decodo' or 'direct'
```

Service Description

Amazon Product Advertising API (PA-API 5.0) provides official access to Amazon product data for affiliate partners. Requires Amazon Associates account approval.

API Capabilities:

- Get product details by ASIN
- Search products by keywords
- Access pricing, images, features
- Affiliate link generation

Usage Throughout Application

Product Data Retrieval:

- PA-API utility: src/lib/amazon-paapi.ts
- Scraper fallback integration: src/lib/scraper.ts
- NPM package: `paapi5-nodejs-sdk`

Implementation Notes:

- **Current Status:** Credentials not configured (using Decodo instead)
- **Activation:** Requires Amazon Associates approval
- **Benefits:** Official API, no scraping issues, affiliate revenue
- **Limitations:** Strict rate limits, approval required

Implementation Example:

```
const { item, error } = await getAmazonItem(asin);
if (item) {
  // Use official Amazon data
  const product = {
    title: item.title,
    price: item.price,
    image: item.image,
    url: item.url // Contains affiliate tag
  };
}
```

Cost Considerations

- **Free:** No direct API costs
- **Requirement:** Must be Amazon Associate with approved account
- **Revenue:** Earn 1-10% commission on referred purchases
- **Rate Limits:** Varies by account performance
- **Estimated Monthly Cost:** $0 (free API, potential revenue generator)

---

11. Google Gemini (Alternative AI Provider) ○ OPTIONAL

Management Console: https://makersuite.google.com/app/apikey **Documentation:** https://ai.google.dev/docs

Environment Variables

```
GEMINI_API_KEY=<...>
```

Service Description

Google Gemini provides an alternative AI language model option. Currently configured but not actively used in production (OpenRouter/Claude is primary).

Potential Use Cases:

- AI provider redundancy/failover
- Cost optimization for specific tasks
- Multimodal capabilities (image + text analysis)

Usage Throughout Application

Status: Configured but not actively used

- NPM packages installed: `@google/genai`, `@ai-sdk/google`
- Model config file exists but references minimal

Implementation Note: This service is available as a backup AI provider but OpenRouter/Claude is the primary integration. Consider removing if not needed to reduce complexity.

Cost Considerations

- **Free Tier:** 60 requests per minute
- **Paid Plans:** Usage-based pricing
- **Estimated Monthly Cost:** $0 (not actively used)

---

12. Zoom API ○ OPTIONAL / INACTIVE

Management Console: https://marketplace.zoom.us/develop/apps **Documentation:** https://developers.zoom.com/docs/api

Environment Variables

```
ZOOM_API_KEY=gOB1bPYzQ6KAuLL07uMQTQ
ZOOM_API_SECRET=ajIh1457402cbTqInG8xl6ovKoAt35JB
```

Service Description

Zoom API credentials are configured but no active integration detected in the codebase.

Potential Use Cases:

- Emergency meeting coordination
- Virtual shelter check-ins
- Family communication planning

Usage Throughout Application

Status: Credentials configured but NOT IMPLEMENTED

- No Zoom SDK or API calls found in codebase
- Modal/dialog references found but no functional integration

Recommendation: Remove credentials if feature is not planned, or document intended use case.

Cost Considerations

- **Free Tier:** Limited features

- **Pro Plan:** $149.90/year per license
- **Estimated Monthly Cost:** $0 (not implemented)

---

13. Google Custom Search API ○ OPTIONAL / INACTIVE

Management Console: https://programmablesearchengine.google.com/controlpanel/all **Documentation:** https://developers.google.com/custom-search/v1/overview

Environment Variables

```
GOOGLE_CUSTOM_SEARCH_CX=7038737736f5a4a53
```

Service Description

Google Custom Search Engine ID configured for potential search features.

Potential Use Cases:

- Emergency preparedness article search
- PDF resource discovery
- Location-specific information retrieval

Usage Throughout Application

Status: Search engine ID configured but minimal usage

- Referenced in environment variables
- No active implementation in core features found

Cost Considerations

- **Free Tier:** 100 queries/day
- **Paid:** $5 per 1,000 queries (up to 10K/day)
- **Estimated Monthly Cost:** $0 (minimal usage)

Service Integration Architecture

Critical Path Dependencies

```
User Request
    ↓
Next.js App Router
    ↓
┌──────────────────────┬──────────────────────┬──────────────────────┐
│                      │                      │                      │
Supabase Auth    OpenRouter AI      Google Maps APIs         Stripe
```

```
(Authentication)  (Plan Generation)  (Routes & Places)    (Payments)
    ↓                    ↓                   ↓                ↓
PostgreSQL         Claude 3.5 Sonnet    Geocoding          Subscription
Database           Mission Plans        Route Computation   Management
    ↓                    ↓                   ↓                ↓
User Profile       AI Usage Logging    Emergency Services   Billing
Plans Storage      Cost Tracking       Meeting Locations    History
```

Supporting Services Layer

```
┌──────────────┬──────────────┬──────────────┬──────────────┐
│              │              │              │              │
Resend        Firecrawl     Decodo       WeatherAPI      SerpAPI
(Email)       (Scraping)    (Amazon)     (Weather)       (Search)
    ↓             ↓             ↓             ↓              ↓
OTP Codes    Product Data  Product Data  Plan Context   Content
Share Emails Auto-fill     Auto-fill     Display        Discovery
Notifications Admin Tools  Admin Tools   Widget         Research
```

Cost Management Recommendations

Monthly Budget Breakdown

Category	Services	Estimated Monthly Cost
Infrastructure	Supabase	$25-50
AI Processing	OpenRouter (Claude)	$50-200
Maps & Location	Google Cloud Platform	$50-150
Payments	Stripe	Variable (2.9% + $0.30)
Email	Resend	$20-40
Web Scraping	Firecrawl + Decodo	$70-150
Search & Data	SerpAPI + WeatherAPI	$0-50
TOTAL ESTIMATE		**$215-640/month**

Cost Optimization Strategies

1. **AI Token Management:**
    - Monitor `calls` table for high-cost operations
    - Implement token usage limits per user tier
    - Use streaming to provide faster UX with same cost
    - Consider Haiku model for simple tasks (10x cheaper)

2. **Google Maps Optimization:**
    - Cache geocoding results to avoid repeat API calls
    - Implement route caching for common waypoint combinations
    - Use session tokens for autocomplete to reduce costs
    - Monitor Places API field usage (each field adds cost)
3. **Scraping Efficiency:**
    - Prioritize Amazon PA-API (free) over Decodo (paid)
    - Cache product data to reduce repeated scrapes
    - Batch import operations during off-peak hours
    - Consider product data refresh schedules (weekly vs. real-time)
4. **Email Optimization:**
    - Batch notification emails where appropriate
    - Use email templates to reduce processing
    - Implement email preferences (reduce unwanted sends)
5. **Infrastructure Scaling:**
    - Monitor Supabase database size and bandwidth
    - Implement data archival strategy for old plans
    - Use Supabase storage for large files vs. database BLOBs

---

Security & Access Management

API Key Rotation Schedule

Service	Rotation Frequency	Last Rotation	Next Due
Supabase Service Role	Quarterly	-	Required
OpenRouter	Quarterly	-	Required
Google Cloud API Key	Quarterly	-	Required
Stripe Webhook Secret	On compromise	-	As needed
Resend API Key	Annually	-	Required
Firecrawl	Annually	-	Required
Decodo	Annually	-	Required

Access Control

Environment Variable Security:

- **Production:** Use deployment platform's secret management (Vercel/Render)
- **Development:** `.env.local` (gitignored)
- **Never commit:** API keys, secrets, passwords to version control

API Key Restrictions:

- **Google Cloud:** Restrict by domain and IP in GCP Console
- **Stripe:** Use webhook signing for secure webhook processing
- **OpenRouter:** Monitor usage dashboard for anomalies
- **Supabase:** Enable RLS policies, use service role key only server-side

---

Monitoring & Alerts

Recommended Monitoring Setup

1. **Cost Tracking Dashboard**
   - OpenRouter usage via admin debug tools
   - Google Cloud billing alerts
   - Stripe MRR tracking
   - Database storage usage (Supabase)
2. **Service Health Checks**
   - API endpoint uptime monitoring
   - Error rate tracking for external APIs
   - Response time monitoring
   - Webhook delivery success rates
3. **Usage Analytics**
   - Plans generated per day/week/month
   - API calls per service
   - User tier distribution
   - Feature adoption metrics
4. **Alert Thresholds**
   - AI token usage > $100/day
   - Google Maps API calls > 5K/day
   - Email bounce rate > 5%
   - Payment failure rate > 2%
   - Database size > 75% of plan limit

Service Status & Health

Health Check Endpoints

```
// src/app/api/health/route.ts
export async function GET() {
  const checks = {
    supabase: await checkSupabaseConnection(),
```

```
    openrouter: await checkOpenRouterAPI(),
    google_maps: await checkGoogleMapsAPI(),
    stripe: await checkStripeConnection(),
    resend: await checkResendAPI(),
  };

  return Response.json(checks);
}
```

Incident Response Plan

Service Outage Priority:

1. **P0 - Critical:** Supabase, OpenRouter, Stripe (affects core functionality)
2. **P1 - High:** Google Maps, Resend (degrades user experience)
3. **P2 - Medium:** Firecrawl, Decodo (affects admin workflows)
4. **P3 - Low:** WeatherAPI, SerpAPI (supplementary features)

Failover Strategy:

- **OpenRouter:** Fallback to Google Gemini if available
- **Decodo:** Fallback to direct scraping or Amazon PA-API
- **Google Maps:** Cache last known good data, degrade gracefully
- **Resend:** Queue emails for retry, log failures

Appendix: Configuration Checklist

New Deployment Setup

- [ ] Create Supabase project and configure environment variables
- [ ] Set up OpenRouter account and obtain API key
- [ ] Enable required Google Cloud Platform APIs
- [ ] Configure Stripe products and webhook endpoint
- [ ] Set up Resend domain and verify sending domain
- [ ] Optional: Configure Firecrawl account for web scraping
- [ ] Optional: Configure Decodo account for Amazon scraping
- [ ] Optional: Set up WeatherAPI account
- [ ] Configure all environment variables in deployment platform
- [ ] Test all integrations in staging environment
- [ ] Set up monitoring and alerts
- [ ] Document API key locations and rotation schedule

Environment Variables Quick Reference

All environment variables should be set in:

- **Development:** `.env.local` (not committed to git)
- **Production:** Deployment platform secrets (Vercel/Render)

See .env.example for complete list with descriptions.

Support & Documentation Links

Primary Services

- **Supabase:** https://supabase.com/docs
- **OpenRouter:** https://openrouter.ai/docs
- **Google Cloud:** https://cloud.google.com/docs
- **Stripe:** https://stripe.com/docs
- **Resend:** https://resend.com/docs

Secondary Services

- **Firecrawl:** https://docs.firecrawl.dev
- **Decodo:** https://scraper-api.decodo.com/docs
- **WeatherAPI:** https://www.weatherapi.com/docs
- **Amazon PA-API:** https://webservices.amazon.com/paapi5/documentation

Internal Documentation

- CLAUDE.md - Project overview and development guide
- DEPLOYMENT.md - Deployment procedures
- RENDER_ENV_SETUP.md - Environment configuration

Report End *For questions or updates to this document, contact: [admin@beprepared.ai]*