

Peace of Mind - Project Setup & Configuration Guide

This document outlines the complete infrastructure, API configuration, and environment setup required to run the Peace of Mind project.



Table of Contents

1. [Prerequisites](#)
 2. [Environment Variables](#)
 3. [API Services & Keys](#)
 4. [Database Setup \(Supabase\)](#)
 5. [Project Structure](#)
 6. [Installation Steps](#)
 7. [Verification](#)
-

Prerequisites

Required Software

- **Node.js**: Version 18 or higher
- **npm** or **yarn**: Package manager
- **Git**: Version control

Required Accounts/Services

1. **Supabase Account** (Database & Authentication)
 2. **Google Cloud Platform** (Google Maps API)
 3. **OpenAI Account** (GPT-4o-mini for AI features)
 4. **Resend Account** (Email service)
 5. **Drivania API** (Optional - for quote integration)
-

Environment Variables

Create a `.env.local` file in the project root with the following variables:

Required Variables

```
# Supabase Configuration (REQUIRED)
NEXT_PUBLIC_SUPABASE_URL=https://your-project.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key-here

# Google Maps API (REQUIRED)
NEXT_PUBLIC_GOOGLE_API_KEY=your-google-maps-api-key
# Alternative names also supported:
# GOOGLE_MAPS_API_KEY
# NEXT_PUBLIC_GOOGLE_MAPS_API_KEY

# OpenAI API (REQUIRED for AI features)
OPENAI_API_KEY=sk-your-openai-api-key

# Resend Email Service (REQUIRED for email notifications)
RESEND_API_KEY=re_your-resend-api-key
```

```
# Base URL (Optional – defaults to auto-detect)
NEXT_PUBLIC_BASE_URL=https://your-domain.com
```

Optional Variables

```
# Drivania API (Optional – for quote integration)
DRIVANIA_API_BASE_URL=https://publicapi.drivania.com
DRIVANIA_API_USERNAME=your-drivania-username
DRIVANIA_API_PASSWORD=your-drivania-password
```

Important Notes

- **NEXT_PUBLIC_ prefix:** Variables that need to be accessible in the browser MUST be prefixed with `NEXT_PUBLIC_`
- **Server-only variables:** Variables without `NEXT_PUBLIC_` are only available in API routes and server-side code
- **Restart required:** After updating `.env.local`, restart the dev server (`npm run dev`)

API Services & Keys

1. Supabase (Database & Auth)

Purpose: Database storage, authentication, real-time subscriptions

Setup Steps:

1. Go to supabase.com
2. Create a new project
3. Go to **Settings** → **API**
4. Copy:
 - **Project URL** → `NEXT_PUBLIC_SUPABASE_URL`
 - **anon/public key** → `NEXT_PUBLIC_SUPABASE_ANON_KEY`

Database Schema: See `lib/database.types.ts` for TypeScript types

Required Tables:

- `trips` - Trip data, locations, reports
- `quotes` - Driver quotes
- `trip_destinations` - Available trip destinations

Features Used:

- PostgreSQL database
- Row Level Security (RLS)
- Real-time subscriptions
- Authentication (optional)

2. Google Maps API

Purpose: Location geocoding, maps display, place search, traffic predictions

Setup Steps:

1. Go to [Google Cloud Console](https://console.cloud.google.com)
2. Create a new project or select existing
3. Enable APIs:
 - **Maps JavaScript API**

- **Geocoding API**
- **Places API**
- **Directions API**
- **Distance Matrix API**

4. Create API Key:

- Go to **APIs & Services** → **Credentials**
- Click **Create Credentials** → **API Key**
- Restrict the key to the APIs above (recommended)
- Add domain restrictions for production

Environment Variable: `NEXT_PUBLIC_GOOGLE_API_KEY`

Usage:

- Frontend: Location search, map display (`hooks/useGoogleMaps.ts`)
- Backend: Geocoding, place verification (`app/api/extract-trip/route.ts`)

Cost: Pay-as-you-go (free tier: \$200/month credit)

3. OpenAI API

Purpose: AI-powered trip extraction, executive report generation

Setup Steps:

1. Go to platform.openai.com
2. Create an account
3. Go to **API Keys**
4. Create a new secret key

Environment Variable: `OPENAI_API_KEY` (server-only, no NEXT_PUBLIC_)

Models Used:

- `gpt-4o-mini` - Trip extraction, report generation

Usage:

- Trip extraction from text (`app/api/extract-trip/route.ts`)
- Executive report generation (`lib/executive-report.ts`)

Cost: Pay-per-token (~\$0.15 per 1M input tokens, ~\$0.60 per 1M output tokens)

4. Resend (Email Service)

Purpose: Sending quote request emails, driver notifications

Setup Steps:

1. Go to resend.com
2. Sign up for an account
3. Go to **API Keys**
4. Create a new API key

Environment Variable: `RESEND_API_KEY` (server-only)

Usage:

- Quote request emails (`app/api/request-quote/route.ts`)
- Driver notifications (`app/api/notify-driver/route.ts`)

Cost: Free tier: 3,000 emails/month, then \$20/month for 50,000 emails

5. Drivania API (Optional)

Purpose: Integration with Drivania booking platform for quotes

Setup Steps:

1. Contact Drivania for API credentials
2. Obtain username and password

Environment Variables:

- `DRIVANIA_API_BASE_URL` (defaults to `https://publicapi.drivania.com`)
- `DRIVANIA_API_USERNAME`
- `DRIVANIA_API_PASSWORD`

Usage: Quote submission (`app/api/drivania/quote/route.ts`)

Documentation: See `DRIVANIA_API_SUMMARY.md` and `DRIVANIA API – Usage and integration manual V1.pdf`

6. Free Public APIs (No Keys Required)

These APIs are used without authentication:

UK Police API

- **Endpoint:** `https://data.police.uk/api/crimes-street/all-crime`
- **Purpose:** Crime statistics and safety scores
- **Rate Limit:** Unlimited
- **Cost:** 100% Free

Transport for London (TfL) API

- **Endpoint:** `https://api.tfl.gov.uk/Road/all/Disruption`
- **Purpose:** Road disruptions, traffic information
- **Rate Limit:** Generous (optional API key for higher limits)
- **Cost:** 100% Free

Open-Meteo Weather API

- **Endpoint:** `https://api.open-meteo.com/v1/forecast`
 - **Purpose:** Weather forecasts
 - **Rate Limit:** 10,000 requests/day
 - **Cost:** 100% Free
-

Database Setup (Supabase)

Required Tables

trips Table

Stores trip data, locations, reports, and metadata.

Key Columns:

- `id` (UUID, primary key)
- `locations` (JSONB) - Array of location objects
- `trip_results` (JSONB) - Analysis results (crime, weather, etc.)
- `executive_report` (JSONB) - AI-generated report
- `trip_notes` (TEXT) - Driver notes
- `lead_passenger_name` (TEXT)
- `vehicle` (TEXT)

- `passenger_count` (INTEGER)
- `trip_destination` (TEXT)
- `trip_date` (DATE)
- `status` (TEXT) - 'not confirmed', 'confirmed', 'pending', 'rejected', 'cancelled'
- `driver` (TEXT) - Driver email
- `password` (TEXT) - Optional password protection
- `version` (INTEGER) - For optimistic locking
- `created_at` , `updated_at` (TIMESTAMP)

`quotes` Table

Stores driver quotes for trips.

Key Columns:

- `id` (UUID, primary key)
- `trip_id` (UUID, foreign key to trips)
- `email` (TEXT) - Driver email
- `price` (DECIMAL)
- `currency` (TEXT)
- `created_at` (TIMESTAMP)

`trip_destinations` Table

Stores available trip destination cities.

Key Columns:

- `id` (UUID, primary key)
- `name` (TEXT) - City name
- `created_at` (TIMESTAMP)

Row Level Security (RLS)

RLS policies should be configured to:

- Allow trip owners to read/write their trips
- Allow drivers to read trips they're assigned to
- Allow public read access for password-protected trips (with password verification)

Project Structure

```
peaceofmind/
├── app/
│   ├── api/                                # API routes (Next.js API routes)
│   │   ├── extract-trip/                  # AI trip extraction
│   │   ├── executive-report/              # AI report generation
│   │   ├── get-quotes/                    # Fetch driver quotes
│   │   ├── request-quote/                 # Send quote request emails
│   │   ├── uk-crime/                      # UK Police API wrapper
│   │   ├── tfl-disruptions/               # TfL API wrapper
│   │   ├── weather/                      # Weather API wrapper
│   │   └── ...
│   ├── page.tsx                          # Homepage (trip creation)
│   ├── results/[id]/                     # Results page (trip viewing)
│   └── ...
├── lib/
│   └── supabase.ts                       # Supabase client
```

```
| |   ├── openai.ts           # OpenAI client
| |   ├── drivania-api.ts    # Drivania API client
| |   ├── google-*.ts       # Google Maps utilities
| |   ├── uk-police-api.ts   # UK Police API client
| |   ├── tfl-api.ts        # TfL API client
| |   ├── open-meteo-api.ts  # Weather API client
| |   └── ...
| ├── components/
| |   ├── GoogleLocationSearch.tsx
| |   ├── GoogleTripMap.tsx
| |   └── ui/                # shadcn/ui components
| ├── hooks/
| |   └── useGoogleMaps.ts   # Google Maps hook
| ├── .env.local            # Environment variables (not in git)
| ├── package.json
| ├── next.config.ts
| └── tsconfig.json
```

Installation Steps

1. Clone Repository

```
git clone <repository-url>
cd peaceofmind
```

2. Install Dependencies

```
npm install
```

3. Set Up Environment Variables

Create `.env.local` file:

```
cp .env.example .env.local # If example exists
# Or create manually
```

Add all required environment variables (see [Environment Variables](#) section above).

4. Set Up Supabase

1. Create a Supabase project
2. Run database migrations (if any)
3. Set up RLS policies
4. Add environment variables:
 - `NEXT_PUBLIC_SUPABASE_URL`
 - `NEXT_PUBLIC_SUPABASE_ANON_KEY`

5. Set Up Google Maps

1. Create Google Cloud project
2. Enable required APIs
3. Create API key
4. Add to `.env.local` : `NEXT_PUBLIC_GOOGLE_API_KEY`

6. Set Up OpenAI

1. Create OpenAI account
2. Generate API key
3. Add to `.env.local` : `OPENAI_API_KEY`

7. Set Up Resend

1. Create Resend account
2. Generate API key
3. Add to `.env.local` : `RESEND_API_KEY`

8. Run Development Server

```
npm run dev
```

The app will be available at `http://localhost:3000`

Verification

Check Environment Variables

The app logs environment variable status on startup:

Supabase:

- Check browser console for: `Supabase Client Initialization`
- Should show: `Present` for both URL and Key

Google Maps:

- Check API route logs for: `[API] Google Maps API Key status: LOADED`
- Visit `/test-google-maps` page to test

OpenAI:

- Test via `/api/test-openai` endpoint (if exists)

Test APIs

1. **Google Maps:** Visit `/test-google-maps` page
 2. **Supabase:** Visit `/test-supabase` page (if exists)
 3. **Trip Creation:** Create a test trip on homepage
 4. **AI Extraction:** Try updating a trip with text input
-

Technology Stack

Frontend

- **Next.js 15.5.4** - React framework
- **React 19.1.0** - UI library
- **TypeScript 5** - Type safety
- **Tailwind CSS 4** - Styling
- **shadcn/ui** - UI components
- **@dnd-kit** - Drag and drop
- **@react-google-maps/api** - Google Maps integration

Backend

- **Next.js API Routes** - Serverless API endpoints

- **Supabase** - Database & Auth
- **OpenAI GPT-4o-mini** - AI features
- **Resend** - Email service

APIs Integrated

- Google Maps (Geocoding, Places, Directions)
 - UK Police API (Crime data)
 - Transport for London API (Traffic disruptions)
 - Open-Meteo API (Weather)
 - Drivania API (Optional - quotes)
-

Development Workflow

Running Locally

```
# Development server with Turbopack
npm run dev

# Production build
npm run build

# Production server
npm start
```

Key Development Features

- **Turbopack**: Fast bundler (enabled by default)
 - **Hot Reload**: Automatic page refresh on changes
 - **TypeScript**: Full type checking
 - **ESLint**: Code linting (if configured)
-

Troubleshooting

Common Issues

1. Google Maps not loading

- Check `NEXT_PUBLIC_GOOGLE_API_KEY` is set
- Verify API key has correct APIs enabled
- Check browser console for errors

2. Supabase connection errors

- Verify `NEXT_PUBLIC_SUPABASE_URL` and `NEXT_PUBLIC_SUPABASE_ANON_KEY`
- Check Supabase project is active
- Verify RLS policies allow access

3. OpenAI API errors

- Check `OPENAI_API_KEY` is set (no `NEXT_PUBLIC_` prefix)
- Verify API key has credits/quota
- Check API route logs for specific errors

4. Email not sending

- Verify `RESEND_API_KEY` is set
- Check Resend account has credits

- Verify sender domain is verified (if required)

Security Best Practices

1. **Never commit** `.env.local` - Already in `.gitignore`
2. **Restrict API keys** - Add domain/IP restrictions in service dashboards
3. **Use RLS** - Implement Row Level Security in Supabase
4. **Validate inputs** - All API routes should validate input
5. **Rate limiting** - Consider adding rate limits for production
6. **HTTPS only** - Use HTTPS in production

Production Deployment

Environment Variables

Ensure all environment variables are set in your hosting platform:

- **Vercel:** Project Settings → Environment Variables
- **Netlify:** Site Settings → Environment Variables
- **Other:** Follow platform-specific instructions

Required for Production

1. Set `NEXT_PUBLIC_BASE_URL` to your production domain
2. Restrict Google Maps API key to your domain
3. Enable Supabase RLS policies
4. Set up email domain verification in Resend (for custom sender)

Additional Resources

- **Supabase Docs:** <https://supabase.com/docs>
- **Google Maps API Docs:** <https://developers.google.com/maps/documentation>
- **OpenAI API Docs:** <https://platform.openai.com/docs>
- **Resend Docs:** <https://resend.com/docs>
- **Next.js Docs:** <https://nextjs.org/docs>
- **Drivania API:** See `DRIVANIA_API_SUMMARY.md`

Last Updated: Based on current codebase analysis **Project Version:** 0.1.0