PlanGenie Installation Manual

This guide provides step-by-step instructions for setting up the PlanGenie Travel Planner application on both **macOS** and **Windows** operating systems.

Table of Contents

Prerequisites

Required Software

For Both Platforms:

- **Python 3.11 or 3.12** (Python 3.14+ is not fully supported by LangChain/Pydantic)
- **Node.js 18.x or higher** (includes npm)
- **Git** (for cloning the repository)
- **A Supabase account** (free tier available at https://supabase.com)
- **API Keys** (see Environment Variables section)

macOS Specific:

- **Homebrew** (recommended for easy package management)

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/ins
```

Windows Specific:

- **Python Launcher for Windows** (usually included with Python)
- **PowerShell** or **Command Prompt**

Verify Prerequisites

Check Python Version:

```
# macOS/Linux
python3 --version

# Windows
python --version
```

Expected output: `Python 3.11.x` or `Python 3.12.x`

Check Node.js Version:

```
# Both platforms
node --version
npm --version
```

Expected output: Node.js `v18.x.x` or higher, npm `9.x.x` or higher

Backend Setup

Step 1: Navigate to Backend Directory

```
# macOS/Linux
cd Backend

# Windows
cd Backend
```

Step 2: Create Python Virtual Environment

macOS/Linux:

```
# Create virtual environment
python3.11 -m venv venv
# OR if you have Python 3.12:
# python3.12 -m venv venv

# Activate virtual environment
source venv/bin/activate
```

Windows:

```
# Create virtual environment
python -m venv venv
# OR if you have multiple Python versions:
# py -3.11 -m venv venv

# Activate virtual environment
# PowerShell:
venv\Scripts\Activate.ps1
# OR Command Prompt:
venv\Scripts\activate.bat
```

Note: If you encounter execution policy errors on Windows PowerShell, run:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

Step 3: Install Python Dependencies

```
# Both platforms (with virtual environment activated)
pip install --upgrade pip
pip install -r requirements.txt
```

Expected packages installed:

- fastapi
- uvicorn
- pydantic
- python-dotenv
- httpx
- sse-starlette
- langchain
- langchain-google-genai
- langchain-ollama
- google-generativeai
- supabase
- tenacity
- bcrypt

Step 4: Verify Backend Installation

```
# Check if all packages are installed
pip list

# Test Python import
python -c "import fastapi; import supabase; print('Backend dependencies OK')"
```

Frontend Setup

Step 1: Navigate to Frontend Directory

```
# From project root
cd Frontend
```

Step 2: Install Node.js Dependencies

```
# Both platforms
npm install
```

Note: This will install all dependencies listed in `package.json`, including:

- React 18.3.1
- Vite 5.4.19
- TypeScript 5.8.3
- Tailwind CSS
- Radix UI components
- And many more...

Step 3: Verify Frontend Installation

```
# Check if node_modules exists
# macOS/Linux
ls -la node_modules

# Windows
dir node_modules
```

Database Setup (Supabase)

PlanGenie uses **Supabase** (PostgreSQL) for data persistence. Follow these steps to set up your database:

Step 1: Create Supabase Account

1. Go to https://supabase.com
2. Sign up for a free account
3. Create a new project
4. Wait for the project to be provisioned (takes 1-2 minutes)

Step 2: Get Supabase Credentials

1. In your Supabase project dashboard, go to **Settings** → **API**
2. Copy the following:
   - **Project URL** (this is your `SUPABASE_URL`)
   - **anon/public key** (this is your `SUPABASE_KEY`)

Step 3: Create Database Tables

In your Supabase project, go to **SQL Editor** and create the following tables:

Table: `users`

This table stores user account information.

Required columns:

- `id` - BIGSERIAL (Primary Key, Auto-increment)
- `email` - VARCHAR(255) (Unique, Not Null)
- `password` - VARCHAR(255) (Not Null) - Stores bcrypt hashed passwords
- `full_name` - VARCHAR(255) (Nullable)
- `created_at` - TIMESTAMP WITH TIME ZONE (Default: NOW())
- `updated_at` - TIMESTAMP WITH TIME ZONE (Default: NOW())

Indexes to create:

- Index on `email` column for faster lookups

Table: `chats`

This table stores chat threads and conversation data.

Required columns:

- `id` - BIGSERIAL (Primary Key, Auto-increment)
- `chat_id` - UUID (Unique, Not Null) - Used as the primary identifier for chat threads
- `user_id` - BIGINT (Foreign Key to `users.id`, with CASCADE delete)
- `created_at` - TIMESTAMP WITH TIME ZONE (Default: NOW())
- `updated_at` - TIMESTAMP WITH TIME ZONE (Default: NOW())

- `chat_memory` - JSONB (Default: '{}'::jsonb) - Stores messages, plan data, and trip constraints

Indexes to create:

- Index on `user_id` column for faster user-based queries
- Index on `chat_id` column for faster chat lookups

Foreign Key Constraint:

- `user_id` references `users(id)` with `ON DELETE CASCADE`

Note: The `chat_memory` JSONB column stores:

- `title` - Chat thread title
- `messages` - Array of conversation messages
- `plan` - Complete trip plan data (flights, hotels, itinerary)
- `trip_constraints` - User's trip requirements and preferences

Step 4: Verify Database Connection

You can test the connection using the Supabase dashboard or by running the backend (it will log connection status).

Environment Variables

Backend Environment Variables

Create a `.env` file in the `Backend` directory by copying from the example file:

macOS/Linux:

```
cd Backend
cp .env.example .env
nano .env  # or use your preferred editor
```

Windows:

```
cd Backend
copy .env.example .env
notepad .env  # or use your preferred editor
```

Environment Variables in `.env.example`:

The `.env.example` file contains the following variables that need to be configured:

Required Variables:

- `SERPAPI_API_KEY` - API key for SerpAPI (used for flight and hotel searches)
- `TAVILY_API_KEY` - API key for Tavily (used for POI and restaurant searches)
- `SUPABASE_URL` - Your Supabase project URL
- `SUPABASE_KEY` - Your Supabase anon/public API key

Optional Variables (with defaults):

- `API_HOST` - Backend server host (default: `0.0.0.0`)
- `API_PORT` - Backend server port (default: `8000`)
- `CORS_ORIGINS` - Comma-separated list of allowed CORS origins (default: `http://localhost:5173,http://localhost:3000`)
- `OLLAMA_MODEL` - Ollama model name (default: `llama3.1:8b`)
- `OLLAMA_BASE_URL` - Ollama server URL (default: `http://localhost:11434`)
- `OLLAMA_TIMEOUT_S` - Ollama request timeout in seconds (default: `30`)
- `OLLAMA_TEMPERATURE` - Ollama temperature setting (default: `0.0`)

Note: Copy `.env.example` to `.env` and fill in the required API keys. Do not commit the `.env` file to version control.

Frontend Environment Variables

Create a `.env` file in the `Frontend` directory (optional):

```
# Optional: Backend API URL (defaults to /api which uses Vite proxy)
VITE_API_URL=http://localhost:8000/api
```

Note: The frontend uses Vite's proxy by default (configured in `vite.config.ts`), so you typically don't need to set `VITE_API_URL` unless running in production.

Getting API Keys

1. SerpAPI Key

- Go to https://serpapi.com/
- Sign up for a free account (100 searches/month free)
- Go to Dashboard → API Key
- Copy the key to `SERPAPI_API_KEY` in your `.env` file

2. Tavily API Key

- Go to https://tavily.com/
- Sign up for a free account
- Go to Dashboard → API Keys
- Copy the key to `TAVILY_API_KEY` in your `.env` file

3. Supabase Credentials

- Already obtained in Database Setup section

Running the Application

Start Backend Server

macOS/Linux:

```
cd Backend
source venv/bin/activate  # Activate virtual environment
python main.py
```

Windows:

```
cd Backend
venv\Scripts\activate  # Activate virtual environment
python main.py
```

Expected output:

```
INFO:     Started server process
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000
INFO:     API Documentation: http://localhost:8000/docs
```

The backend will be available at:

- **API Server:** http://localhost:8000
- **API Documentation:** http://localhost:8000/docs
- **Health Check:** http://localhost:8000/api/health

Start Frontend Development Server

Open a **new terminal window** (keep backend running):

Both Platforms:

```
cd Frontend
npm run dev
```

Expected output:

```
  VITE v5.x.x  ready in xxx ms

  ➜  Local:   http://localhost:8080/
  ➜  Network: use --host to expose
```

The frontend will be available at:

- **Frontend App:** http://localhost:8080

Note: The frontend is configured to proxy API requests to `http://localhost:8000` automatically.

Verify Installation

1. **Backend Health Check:**

   ```
   curl http://localhost:8000/api/health
   ```

   Should return: `{"status":"ok"}`
2. **Frontend:**
   - Open http://localhost:8080 in your browser
   - You should see the PlanGenie landing page
3. **API Documentation:**
   - Open http://localhost:8000/docs
   - You should see the Swagger UI with all available endpoints

---

Troubleshooting

Backend Issues

Issue: `ModuleNotFoundError` or `ImportError`

Solution:

```
# Make sure virtual environment is activated
# macOS/Linux
source venv/bin/activate

# Windows
venv\Scripts\activate

# Reinstall dependencies
pip install -r requirements.txt
```

Issue: `SERPAPI_API_KEY not set` or `TAVILY_API_KEY not set`

Solution:

- Verify `.env` file exists in `Backend` directory
- Check that `SERPAPI_API_KEY` and `TAVILY_API_KEY` are set
- Ensure no extra spaces or quotes around the key values
- Restart the backend server after changing `.env`

Issue: `Failed to initialize Supabase client`

Solution:

- Verify `SUPABASE_URL` and `SUPABASE_KEY` are correct
- Check that your Supabase project is active
- Ensure database tables are created (see Database Setup)

Issue: `Port 8000 already in use`

Solution:

```
# macOS/Linux - Find and kill process
lsof -ti:8000 | xargs kill -9

# Windows - Find and kill process
netstat -ano | findstr :8000
taskkill /PID <PID> /F

# OR change port in .env
API_PORT=8001
```

Issue: Python version incompatibility

Solution:

- Ensure you're using Python 3.11 or 3.12
- Check version: `python --version` or `python3 --version`
- If using wrong version, recreate virtual environment:

```
# Remove old venv
rm -rf venv  # macOS/Linux
rmdir /s venv  # Windows

# Create new venv with correct Python version
python3.11 -m venv venv  # or python3.12
```

Frontend Issues

Issue: `vite: command not found`

Solution:

```
cd Frontend
npm install
```

Issue: `Port 8080 already in use`

Solution:

- Change port in `vite.config.ts`:

```
server: {
  port: 8081,  // Change to available port
}
```

Issue: `Cannot connect to backend API`

Solution:

- Verify backend is running on http://localhost:8000
- Check `vite.config.ts` proxy configuration
- Verify `VITE_API_URL` in `.env` if set
- Check browser console for CORS errors

Issue: `npm install` fails

Solution:

```
# Clear npm cache
npm cache clean --force

# Delete node_modules and package-lock.json
rm -rf node_modules package-lock.json  # macOS/Linux
rmdir /s node_modules & del package-lock.json  # Windows

# Reinstall
npm install
```

Database Issues

Issue: `relation "users" does not exist` or `relation "chats" does not exist`

Solution:

- Create the required tables in Supabase SQL Editor following the table specifications in Database Setup
- Verify table names match exactly (case-sensitive)

- Ensure all required columns are created with correct data types

Issue: `permission denied for table`

Solution:

- Check that you're using the `anon` key (not `service_role` key)
- Verify Row Level Security (RLS) policies if enabled
- Check Supabase project settings

Issue: `connection timeout`

Solution:

- Verify `SUPABASE_URL` is correct
- Check internet connection
- Verify Supabase project is not paused (free tier projects pause after inactivity)

General Issues

Issue: Environment variables not loading

Solution:

- Ensure `.env` file is in the correct directory (`Backend/` for backend, `Frontend/` for frontend)
- Verify file is named exactly `.env` (not `.env.txt` or `env`)
- Restart the server after changing `.env` files
- Check for syntax errors (no spaces around `=` sign)

Issue: CORS errors in browser

Solution:

- Verify `CORS_ORIGINS` in backend `.env` includes your frontend URL
- Default: `http://localhost:5173,http://localhost:3000`
- If using port 8080, add: `http://localhost:8080`

Issue: Date showing wrong day (e.g., Nov 23 showing as Nov 22)

Solution:

- This is a timezone issue that should be fixed in the codebase
- Verify you're using the latest version of `ItineraryCard.tsx`
- The fix uses `date-fns` `parse()` function instead of `new Date()`

Production Deployment

Backend Production

1. **Set environment variables** on your hosting platform (Heroku, Railway, Render, etc.)
2. **Use production-grade ASGI server:**

```
uvicorn main:app --host 0.0.0.0 --port 8000 --workers 4
```

3. **Enable HTTPS** (required for Supabase in production)
4. **Set proper CORS origins** for your production domain

Frontend Production

1. **Build the application:**

```
cd Frontend
npm run build
```

2. **Deploy `dist/` folder** to your hosting platform (Vercel, Netlify, etc.)
3. **Set `VITE_API_URL`** to your production backend URL
4. **Configure environment variables** on your hosting platform

---

Additional Resources

- **FastAPI Documentation:** https://fastapi.tiangolo.com/
- **React Documentation:** https://react.dev/
- **Supabase Documentation:** https://supabase.com/docs
- **Vite Documentation:** https://vitejs.dev/

Support

If you encounter issues not covered in this guide:

1. Check the Troubleshooting section
2. Review error logs in the terminal
3. Check browser console for frontend errors
4. Verify all API keys are valid and have sufficient quota
5. Ensure all prerequisites are installed correctly

Last Updated: 2024
Version: 1.0.0