CoCounsel

Legal Research Agent - User's Guide

Complete Setup, Configuration & Troubleshooting

Version 2.0 - December 2025

Table of Contents

1. Introduction

CoCounsel is a multi-agent legal research system built in n8n that processes legal queries through a pipeline of five specialized AI agents. Inspired by Thomson Reuters' CoCounsel platform, it automates the identification of legal issues, analysis of case law, synthesis of findings, quality assessment, and generation of professional legal memoranda.

Key Features

- **Five specialized AI agents** powered by Claude Sonnet 4
- **Quality-based routing:** Score ≥80 auto-drafts memorandum, <80 flags for expert review
- **Session memory tracking** through decision trail
- **PostgreSQL persistence** with three tables: research_results, review_queue, audit_log
- **Structured JSON output** with JSONB storage for complex data
- **RESTful webhook API** for easy integration

Agent Pipeline

Agent	Purpose	Temperature
Issue Spotter	Identifies legal issues, generates search terms	0.3
Case Analysis	Analyzes case law, extracts holdings	0.2
Legal Synthesis	Creates executive summary, analysis, recommendations	0.3
Quality Assurance	Scores quality (0-100), flags gaps	0.2
Memorandum Drafter	Generates formal legal memorandum	0.4

2. Prerequisites

Before setting up CoCounsel, ensure you have the following:

Required Accounts & API Keys

1. **n8n Instance:** Self-hosted or n8n.cloud account
2. **Anthropic API Key:** For Claude Sonnet 4 (claude-sonnet-4-20250514)
3. **Supabase Account:** For PostgreSQL database (free tier available)
4. **CourtListener Account (Optional):** For live case law search API

System Requirements

- n8n version 1.0 or higher
- PostgreSQL 14+ (provided by Supabase)
- Network access to api.anthropic.com and courtlistener.com

3. Database Setup (Supabase PostgreSQL)

Step 1: Create Supabase Project

1. Go to https://supabase.com and sign in
2. Click "New Project"
3. Select your organization and enter project details
4. **IMPORTANT:** Save your database password securely
5. Select a region close to your n8n instance
6. Wait for project initialization (2-3 minutes)

Step 2: Get Connection Details

Navigate to **Project Settings** → **Database** and note these values:

Setting	Value
Host	aws-0-[region].pooler.supabase.com
Port	**6543** (Transaction Pooler - RECOMMENDED)
Database	postgres
User	postgres.[project-ref]
SSL	**Required** - Must be enabled

> **TIP:** Use port 6543 (transaction pooler) for n8n. Port 5432 is for direct connections and may cause pool exhaustion.

Step 3: Create Database Tables

Open the **SQL Editor** in Supabase and run the following SQL scripts. See **Appendix: Complete SQL Schemas** for full table definitions.

> **WARNING:** Never hardcode id values in INSERT statements. The id column uses SERIAL for auto-generation.

4. Anthropic API Configuration

Step 1: Create API Credential in n8n

1. In n8n, go to Settings → Credentials
2. Click "Add Credential" → Select "Anthropic"
3. Enter your Anthropic API key
4. Save the credential

Step 2: Configure Agent Nodes

Each AI agent node uses the same model but different temperatures:

Node Name	Model	Temperature
Anthropic Model - Issue Spotter	claude-sonnet-4-20250514	0.3
Anthropic Model - Case Analysis	claude-sonnet-4-20250514	0.2
Anthropic Model - Synthesis	claude-sonnet-4-20250514	0.3
Anthropic Model - QA	claude-sonnet-4-20250514	0.2
Anthropic Model - Memorandum	claude-sonnet-4-20250514	0.4

5. Case Law Search Node Setup

The Case Law Search node retrieves relevant case law based on search terms from the Issue Spotter. You have two options: use the Mock Case Law node for testing, or configure the CourtListener API for live data.

> **CRITICAL:** The most common workflow failure occurs when this node returns wrong data or empty results. Follow these instructions carefully.

Option A: Mock Case Law (Recommended for Testing)

The workflow includes a "Mock Code in JavaScript" node that returns realistic California non-compete case law. To use it:

1. Open the workflow in n8n editor
2. Disconnect "Split Search Queries" from "Case Law Database Search"
3. Connect "Split Search Queries" → "Mock Code in JavaScript"
4. Connect "Mock Code in JavaScript" → "Case Analysis Agent"

Option B: CourtListener API (Live Data)

To use the free CourtListener API for federal case law:

Step 1: Get API Token

1. Create account at https://www.courtlistener.com/sign-in/
2. Go to your profile and generate an API token
3. Note: Rate limit is 5,000 requests/hour

Step 2: Configure HTTP Request Node

> **CRITICAL:** The URL must include /search/ at the end. Missing this is the #1 cause of failures.

Setting	Value
Method	GET
URL	**https://www.courtlistener.com/api/rest/v4/search/**

Headers:

```
Authorization: Token YOUR_API_TOKEN
```

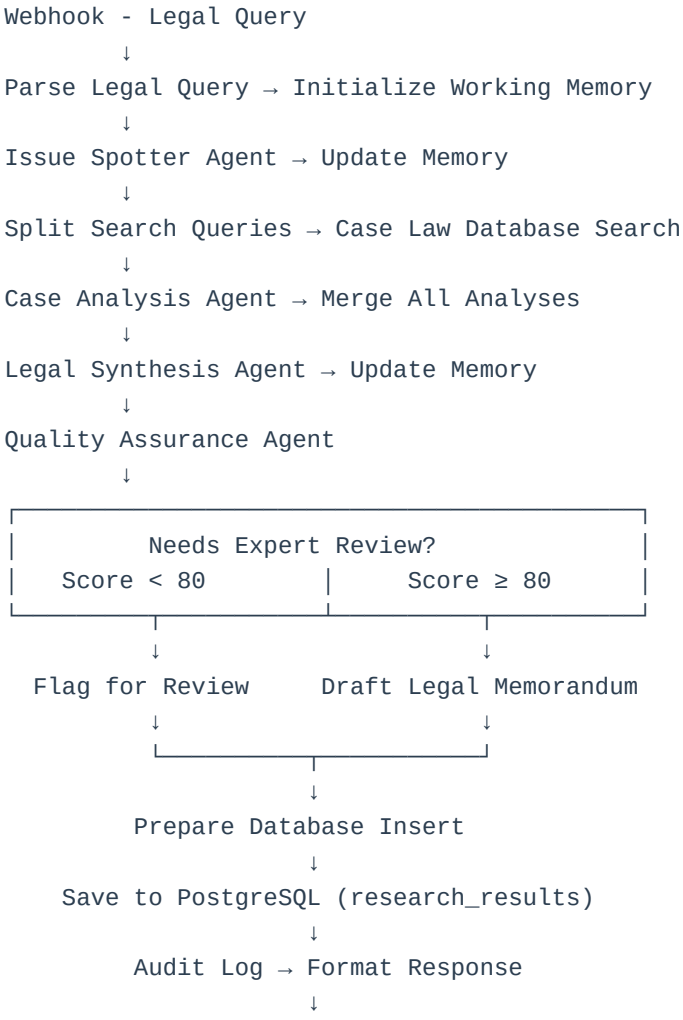
Query Parameters (NOT Headers!):

Parameter	Value
q	{{ $json.workingMemory.legalContext.searchTerms.join(' ') }}
type	o
order_by	score desc

> **COMMON MISTAKE:** Putting q, type, and order_by in the Headers section instead of Query Parameters. This will cause the API to return an index of endpoints instead of search results.

6. Workflow Architecture

The workflow follows this execution path:



7. Testing the Workflow

Webhook Request Format

Send a POST request to your webhook URL:

```
POST https://your-n8n-instance/webhook/legal-research
Content-Type: application/json

{
  "query": "Can an employer require a 2-year non-compete?",
  "userId": "user_001",
  "jurisdiction": "California",
  "practiceArea": "Employment Law"
}
```

cURL Test Command

```
curl -X POST https://your-n8n-instance/webhook/legal-research \
  -H "Content-Type: application/json" \
  -d '{
    "query": "Is a non-compete enforceable in California?",
    "userId": "test_001",
    "jurisdiction": "California",
    "practiceArea": "Employment Law"
  }'
```

Expected Success Response

```
{
  "sessionId": "session_1234567890_abc123",
  "status": "completed",
  "needsExpertReview": false,
  "qualityScore": 85,
  "confidenceRating": "high",
  "memorandum": "LEGAL MEMORANDUM...",
  "message": "Research completed successfully."
}
```

8. Troubleshooting Guide

Error: "Model output doesn't fit required format"

Symptom: Workflow fails at Case Analysis Agent, Quality Assurance Agent, or other AI nodes with output parser errors.

Root Cause: The AI model received empty or malformed input data, so it responded with a clarifying question instead of JSON output.

Solutions:

1. **Check Case Law Search output:** Run the workflow and inspect what the Case Law Database Search node returns. If it returns an API index (list of endpoints), the URL is wrong.
2. **Fix the URL:** Ensure URL is https://www.courtlistener.com/api/rest/v4/search/ (with /search/ at the end)
3. **Check Query Parameters:** Move q, type, order_by from Headers to Query Parameters
4. **Use Mock Data:** Switch to the Mock Case Law node to isolate the issue

Error: "duplicate key value violates unique constraint"

Symptom: Audit Log or Save Results node fails with primary key violation.

Root Cause: The PostgreSQL node has a hardcoded id: 0 in the column mappings.

Solution:

- Open the PostgreSQL node (Audit Log or Save Results)
- In Columns section, find the "id" field
- Either remove it entirely or mark it as "removed" - let PostgreSQL auto-generate

Error: "Connection refused" or "SSL required"

Root Cause: Incorrect PostgreSQL connection settings.

Solutions:

- Use port 6543 (transaction pooler), not 5432
- Enable SSL in the n8n PostgreSQL credential
- Verify host format: aws-0-[region].pooler.supabase.com

Error: "Invalid input syntax for type json"

Root Cause: JSONB columns receiving non-stringified data.

Solution:

- In "Prepare Database Insert" code node, ensure all JSONB fields use JSON.stringify():

```
legal_issues: JSON.stringify(issueSpotterOutput?.legalIssues || [])
```

Error: Empty response or "Query: " with no data

Symptom: Case Analysis Agent prompt shows "Query: [empty]" or "Search Results: [empty]"

Root Cause: The Case Analysis Agent's text field uses incorrect expression paths.

Solution:

Update the Case Analysis Agent's text field:

```
Query: {{ $('Parse Legal Query').item.json.originalQuery }}
Jurisdiction: {{ $('Parse Legal Query').item.json.jurisdiction }}
Search Results: {{ JSON.stringify($json) }}
```

9. Appendix: Complete SQL Schemas

Table: research_results

```
CREATE TABLE research_results (
    id SERIAL PRIMARY KEY,
    session_id TEXT NOT NULL,
    user_id TEXT NOT NULL,
    original_query TEXT,
    jurisdiction TEXT,
    practice_area TEXT,
    legal_issues JSONB,
    search_terms JSONB,
    practice_areas JSONB,
    case_analyses JSONB,
    relevant_cases JSONB,
    key_findings JSONB,
    executive_summary TEXT,
    legal_analysis TEXT,
    precedents JSONB,
    recommendations JSONB,
    quality_score INTEGER,
    confidence_rating TEXT,
    expert_review_needed BOOLEAN,
    quality_gaps JSONB,
    quality_strengths JSONB,
    memorandum TEXT,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

Table: audit_log

```
CREATE TABLE audit_log (
    id SERIAL PRIMARY KEY,
    session_id TEXT NOT NULL,
    user_id TEXT NOT NULL,
    workflow_path TEXT NOT NULL,
    final_action TEXT NOT NULL,
    status TEXT,
    started_at TIMESTAMPTZ NOT NULL,
    completed_at TIMESTAMPTZ NOT NULL,
    execution_time_seconds INTEGER,
    quality_score INTEGER,
    confidence_rating TEXT,
    expert_review_needed BOOLEAN,
    original_query TEXT,
    jurisdiction TEXT,
    practice_area TEXT,
    cases_analyzed INTEGER,
    issues_identified INTEGER,
    recommendations_generated INTEGER,
```

```
    metadata JSONB,
    created_at TIMESTAMPTZ DEFAULT NOW()
);
```

Indexes

```
CREATE INDEX idx_research_results_session ON research_results(session_id);
CREATE INDEX idx_research_results_user ON research_results(user_id);
CREATE INDEX idx_audit_log_session ON audit_log(session_id);
CREATE INDEX idx_audit_log_user ON audit_log(user_id);
CREATE INDEX idx_audit_log_status ON audit_log(status);
```

— End of User's Guide —