SANZEN Legal AI: Code Documentation

Author: Manus AI **Date**: December 18, 2025

1. Introduction

This document provides a detailed overview of the codebase for the SANZEN Legal AI platform. It covers the folder structure, key files, and coding patterns used throughout the project. This documentation is intended to help developers understand the codebase and contribute to the project effectively.

2. Folder Structure

The project is organized into a monorepo structure, with the frontend and backend code located in separate directories. The top-level directory structure is as follows:

```
/home/ubuntu/paris_group_legal_ai
├── client/          # Frontend code (React, Vite)
├── server/          # Backend code (Node.js, tRPC)
├── drizzle/         # Drizzle ORM schema and migrations
├── scrapers/        # Python scripts for data scraping
├── scripts/         # Utility scripts for database operations
├── docs/            # Project documentation
├── public/          # Static assets
├── ...              # Configuration files
```

2.1. Frontend (`client/`)

The frontend is a React application built with Vite. The folder structure follows a standard React project layout:

- `client/src/pages/`: Contains the main page components for each route.
- `client/src/components/`: Contains reusable UI components.

- `client/src/lib/`: Contains utility functions and libraries, including the tRPC client.
- `client/src/hooks/`: Contains custom React hooks.
- `client/src/contexts/`: Contains React context providers.

2.2. Backend (`server/`)

The backend is a Node.js application built with Express and tRPC. The folder structure is organized by feature:

- `server/_core/`: Contains the core application logic, including the tRPC server, authentication, and LLM integration.
- `server/routers.ts`: Defines the tRPC routers and procedures.
- `server/db.ts`: Contains database query functions.
- `server/legalDocumentTemplates.ts`: Contains templates for generating legal documents.

3. Key Files and Components

This section highlights some of the most important files and components in the codebase.

3.1. `server/routers.ts`

This file is the heart of the backend API. It defines all the tRPC routers and procedures that the frontend can call. Each procedure is defined with an input schema (using Zod) and a resolver function that implements the business logic.

3.2. `server/_core/unifiedLLM.ts`

This file implements the unified LLM system, which provides a single interface for interacting with different LLM providers (Gemini and Manus). It handles the logic for selecting the appropriate provider based on environment variables and falling back to a secondary provider if the primary one fails.

3.3. `client/src/lib/trpc.ts`

This file configures the tRPC client for the frontend. It creates a tRPC proxy that allows the frontend to call backend procedures as if they were local functions, with full type safety.

3.4. `client/src/pages/Consultation.tsx`

This component implements the main chat interface for legal consultations. It uses tRPC queries to fetch consultation data and messages, and tRPC mutations to send new messages.