



Software Project Documentation: ClauseAI Ultimate

Project: AI Tool to Analyze Legal Contracts

Author/Developer: ARULDASS R

Program: Infosys Springboard Internship

Version: Build v1.0

1. Project Overview

1.1 Objective

The primary objective of **ClauseAI Ultimate** is to democratize legal access and streamline contract analysis. By leveraging Large Language Models (LLMs), **LangGraph-powered Multi-Agent orchestration**, and multimodal interactions (Voice/Video), the platform automates the extraction, risk assessment, and summarization of complex legal documents.

1.2 Base Research

This project operationalizes the theoretical framework presented in:

"A Reusable Prompting Framework for Applying Large Language Models to Legal Tasks" (Sriram et al., IEEE Access 2026).

While the base paper validated the use of "Structured Prompting" and "Role-Based Prompts" for legal accuracy over standard RAG models, ClauseAI transforms this script-based research into a highly accessible, consumer-facing Software-as-a-Service (SaaS) application powered by stateful agentic workflows.

2. Milestone Breakdown & Development Lifecycle

The development of ClauseAI was structured into 6 strategic milestones, moving from basic research implementation to a fully deployed product.

🚩 Milestone 1: Core Framework & Document Parsing

- **Goal:** Establish the foundational capabilities to read and process legal text.
- **Accomplishments:**
 - Set up the base Python environment and Streamlit application structure.
 - Implemented PDF and DOCX parsers (`utils/docsloader.py`, `utils/pdf_inspector.py`) to extract raw text and layout structures from legal contracts.

- Integrated the universal_llm.py wrapper using **LangChain** to securely connect to the Gemini API and manage prompt templates.
- Successfully replicated the base paper's static "Structured Prompting" for basic contract summarization.

🚩 Milestone 2: LangGraph Multi-Agent Orchestration & RAG

- **Goal:** Improve reasoning capabilities by dividing legal analysis into specialized domains using stateful graphs.
- **Accomplishments:**
 - Integrated **Pinecone** for robust Retrieval-Augmented Generation (RAG) using LangChain vector store wrappers.
 - Developed a stateful execution graph using **LangGraph** (graph/doc_graph.py) to manage document processing states.
 - Created planner/planner.py to route user queries intelligently to specific sub-agents.
 - Created specialized LangChain Agents (multi_agents/):
 - **Legal Agent:** Identifies liabilities, indemnification, and breach risks.
 - **Finance Agent:** Extracts payment terms, penalties, and financial obligations.
 - **Compliance Agent:** Checks for regulatory adherence and governing laws.
 - **Operations Agent:** Tracks deliverables and timelines.

🚩 Milestone 3: User Interface (UI) & Core Modules

- **Goal:** Build an intuitive, cyberpunk-inspired UI for users to interact with the AI.
- **Accomplishments:**
 - Designed the **Main Console** for uploading and scanning contracts.
 - Built **The Oracle** for deep, chat-based legal Q&A.
 - Implemented **Data Analytics** (views/analytics.py) to visualize contract risks using charts.
 - Built the **PDF Export Engine** (utils/export_utils.py) allowing users to download formal audit reports.

🚩 Milestone 4: The Multimodal AI Consultant (Video & Voice)

- **Goal:** Break the "text-only" barrier by introducing face-to-face AI interaction.
- **Accomplishments:**
 - Integrated speech_recognition (Google STT) to allow users to speak their queries.
 - Integrated edge_tts to generate high-quality, neural text-to-speech responses.
 - Engineered a custom **HTML/CSS-in-JS Video Player** that seamlessly loops a 9:16 vertical AI Avatar (idle.mp4 and talking.mp4) inside a 16:9 cinematic frame.
 - **Innovation:** Programmed state management to switch the avatar from "Idle" to "Speaking" dynamically based on Python backend states.

🚩 Milestone 5: Vernacular Localization & Smart Duration Engine

- **Goal:** Make the tool accessible to non-English speakers across India.

- **Accomplishments:**
 - Added native support for **Tamil, Telugu, Hindi**, Spanish, and French.
 - Configured specific neural voices for cultural accuracy (e.g., ValluvarNeural for Tamil, MohanNeural for Telugu).
 - **Engineering Feat:** Developed the calculate_duration() algorithm. Since Indian languages are syllable-dense, the algorithm mathematically slows down the video lip-sync timeout (1.5 words/sec for Tamil vs. 2.4 words/sec for English) to prevent the video from cutting off before the audio finishes.

🚩 Milestone 6: SaaS Gatekeeper & Productization

- **Goal:** Transform the prototype into a secure, monetizable web product.
- **Accomplishments:**
 - Built utils/db.py to initialize an **SQLite Database** for user management.
 - Created a robust **Authentication System** (Login & Registration pages).
 - Developed a **Landing Page** to showcase features before login.
 - Implemented a **Tiered Subscription Model** (Free vs. Pro) with simulated UPI payment gateways.
 - Locked premium features (like the AI Consultant and The Vault) behind the "Pro" subscription wall.

3. System Architecture

ClauseAI utilizes a modular, decoupled architecture to ensure scalability and maintainability.

3.1 Architecture Diagram Flow

1. **User Interface:** Streamlit Frontend (Renders Custom CSS & HTML Components).
2. **Auth Layer:** SQLite Database verifies user credentials and checks active Subscription Plan.
3. **Input Layer:** User uploads PDF/DOCX or speaks into the microphone (STT).
4. **Processing Layer (LangChain & LangGraph):**
 - Text is chunked and embedded into Pinecone via **LangChain**.
 - **LangGraph** initializes the state (doc_graph.py) and passes it to the Planner.
 - Planner assigns tasks to the relevant Multi-Agent (Legal, Finance, etc.).
 - Universal LLM processes the query using Structured Role-Based Prompts.
5. **Output Layer:**
 - Textual data is rendered as Markdown or PDF Reports.
 - Conversational data is passed to EdgeTTS for audio generation.
 - Streamlit state updates trigger the base64 Video Player to play talking.mp4.

4. Key Innovations (Beyond Academic Scope)

While the base academic paper focused purely on evaluating text prompts, this project introduced several industry-grade innovations:

- **Stateful Agentic Workflows:** Shifted from linear API calls to a **LangGraph** architecture, allowing agents to iteratively pass states, reflect, and correct their legal analyses before presenting them to the user.
- **Anti-Hallucination via Persona Injection:** Implemented system instructions directly into the audio loop to ensure the AI behaves strictly as a legal consultant and refuses illegal requests.
- **Dynamic Lip-Sync Math:** Solved the mismatch between TTS generation speed and video playback duration using custom buffer and word-count algorithms.
- **Stateful Micro-Interactions:** Overcame Streamlit's synchronous limitations by using `st.rerun()` and custom HTML payloads to create a real-time, uninterrupted video calling experience.

5. Software Requirements Specification (SRS)

5.1 Hardware Requirements

- **Processor:** Minimum Intel Core i3 / AMD Ryzen 3 (or equivalent).
- **RAM:** 4GB Minimum (8GB Recommended).
- **Peripherals:** Working Microphone (for AI Consultant feature).

5.2 Software Requirements

- **Operating System:** Windows 10/11, macOS, or Linux.
- **Language:** Python 3.9 or higher.
- **Core AI Libraries:** langchain, langchain-google-genai, langgraph, pinecone-client.
- **Core Utility Libraries:** streamlit, sqlite3, SpeechRecognition, edge-tts, asyncio.
- **External APIs:** Google Gemini API, active Internet connection for STT/TTS modules.

6. Future Enhancements

1. **Agentic Drafting:** Upgrading the LangGraph Multi-Agents from "Analysis" to "Drafting" (e.g., auto-generating counter-clauses or legal notices).
2. **On-Premise LLMs:** Integrating models like Llama-3 locally using Ollama and LangChain to ensure zero-data-leakage for highly confidential enterprise contracts.
3. **Mobile Application:** Porting the Multimodal Avatar to a React Native or Flutter environment for native mobile deployment.

End of Document