V3.0

Approach to V3.0

*\*Enable Outline for the tab for to get section wise analysis of the documentation*

# Purpose and Workflow :

The code is designed to automate the process of predicting judicial verdicts for a batch of legal cases. It does this by simulating a courtroom trial for each case, using specialized roles (Judge, Prosecution, Defense, etc.), each powered by a large language model (LLM) accessed via the GroqCloud API. The system processes each case in the test set, runs it through a multi-phase trial, and outputs only the final verdict in a standardized format.

# Data Loading and Preparation :

* The script reads the input file `test\_cases.csv`, which contains a list of legal cases.
* Each row in the CSV has a unique case ID and the full text describing the case background, facts, and issues.

# Agent and Courtroom Design :

## CourtroomAgent Class :

Each agent (e.g., Judge, Prosecution, Defense, Plaintiff, Witness) is represented by an instance of this class.

* Each agent maintains its own conversation history.
* When prompted, the agent sends its message history to the LLM and receives a response, which is then appended to its history.
* This ensures that each agent’s responses are context-aware and role-specific.

## Courtroom Class :

This class manages the overall trial process for a single case.

* It initializes the core agents (Judge, Prosecution, Defense, Defendant, and, for civil cases, Plaintiff).
* It supports dynamic creation of additional agents, such as witnesses and expert witnesses, as needed during the trial.
* The class is responsible for orchestrating the sequence of trial phases and managing agent interactions.

# Trial Phases and Prompts :

For each case, the trial proceeds through several phases, each with carefully crafted prompts for the relevant agents:

## Opening Statements :

The Prosecution, Defense, Defendant, and Plaintiff (if applicable) introduce their positions and outline the key issues.

Witness Interrogation & Argumentation :

The Prosecution and Defense examine witnesses and experts, presenting and challenging evidence.

Closing Statements :

Each side summarizes their arguments and evidence, making their final appeal to the court.

Judge’s Ruling :

The Judge reviews the arguments, evidence, and testimonies, and delivers a verdict. The prompt instructs the Judge to clearly state the verdict at the end of the response using a standardized tag (`#VERDICT: GRANTED` or `#VERDICT: DENIED`).

# Dynamic agent creation & Freely evolving trial structure:

| **Feature** | **Static (v2.0)** | **Dynamic (v2.5 & v3.0)** |
| --- | --- | --- |
| **Agent Roles** | Fixed | Agents can be created/removed as needed |
| **Phase Sequence** | Rigid, pre-set | Can branch, repeat, or skip based on need |
| **Prompts** | Generic, unchanging | Tailored to case facts and agent responses |
| **Evidence/Witness Handling** | Predefined list | Added on-the-fly during trial |
| **Verdict Timing** | Always at end | Only after all necessary arguments/evidence |

The dynamic nature of the trial simulation comes from its ability to:

* Adapt the structure and flow of phases based on the unique facts and needs of each case,
* Generate and adjust prompts in real time,
* Create or remove agents as required,
* And respond to the evolving arguments and evidence as the trial progresses.

This ensures that each simulated trial is not just a replay of a fixed script, but a flexible, context-aware process that mirrors the complexity and unpredictability of real courtroom proceedings.

# Verdict Extraction :

After the Judge issues a ruling, the script parses the Judge’s response to extract the verdict.

The verdict is mapped to a binary label:

* `GRANTED` → 1
* `DENIED` → 0

# Batch Processing and Output :

* The script iterates through each case in the test set, running the full trial simulation for each.
* For each case, it prints the result in the format: id,label
* where `id` is the case identifier and `label` is the predicted verdict (1 or 0).

# Error Handling and API Management :

* The script is designed to handle API rate limits by rotating through available API keys or pausing when necessary.
* If a case fails due to an unexpected error, the script logs the error and outputs a default label (0) for that case, ensuring that the batch process continues without interruption.

# Design Rationale :

* Role Separation: Each agent is responsible for its own dialogue and context, mirroring real courtroom proceedings.
* Extensibility: The modular design allows for easy addition of new roles or trial phases as needed.
* Efficiency: Only the final verdict is output, which is suitable for evaluation and further analysis.
* Robustness: The system is resilient to errors and can handle variations in input data formats.