全体アーキテクチャ（概観）

・基本的なモジュール分割例（上流 → 下流）：

・フロントエンド（Web/モバイル/コールセンタ）

・API Gateway / 認可（Auth） — REST/GraphQL

・入力処理レイヤ（ASR / 音声前処理 / テキスト正規化）

・NLU（形態素解析・NLU・対話状態管理）

・法律知識ベース（条文 DB・判例 DB・メタ情報・KG）

・分析エンジン（過失割合推定、類似判例検索、和解案生成）

・NLG（応答生成）

・TTS（音声合成）

・管理・監査・学習基盤（ログ、フィードバック、モデル管理）

・モニタリング & セキュリティ & CI/CD

通信は基本 REST + JSON（内部の高性能部分は gRPC も検討）。モデル推論はコンテナ化して Kubernetes 上で運用するのが企業向け標準。

実現に必要な API

_____**機能要件**_____

1.*認証・ユーザー管理モジュール* __must__

役割：ユーザー認証、権限（一般／弁護士／管理者）、ログ保護。

主要 API 例

・POST /auth/login → { token, refresh_token, user }

・POST /auth/refresh → { token }

・GET /users/{id}（管理者向け）
・技術候補：OAuth2 / OpenID Connect、JWT、Keycloak、AWS Cognito、Auth0。
留意点：MFA、RBAC、管理者操作の二重承認。ログの WORM 保存。

Auth0 を使用予定（無料版を使用）

2.**音声処理（ASR）モジュール** __must__

役割：ユーザー音声をテキスト化。リアルタイム・バッチ両対応。雑音耐性・発話分割が重要。
主要 API 例

・POST /asr/stream (WebSocket or gRPC streaming) → partial transcripts, final transcript

・POST /asr/transcribe → { transcript, confidence, segments:[{start,end,text,confidence}] }
技術候補：

商用：Google Speech-to-Text, Amazon Transcribe, Azure Speech（企業導入で安定）

Gemini を使用予定

オープン：OpenAI/Whisper（ローカル化可）、Vosk（低レイテンシ）
留意点：発話分割、話者分離（diarization）、ASR のドメイン適応（法律用語辞書）を必須で導入。

3.**テキスト前処理 & 正規化モジュール** __must__

・役割：音声→テキスト後のノイズ除去、正規化（日付・金額・条文表記統一）、誤変換補正。
API 例

POST /text/normalize → { normalized_text, tokens, entities }
技術候補：ルールベース + ML 補正。正規表現、辞書置換、カスタム正規化ライブラリ。

できれば Gemini で完結、

4.**形態素解析／NLU（日本語)**__must__

役割：品詞分解、固有表現抽出（当事者、日付、金額、条文名）、依存解析、意図分類。
API 例

POST /nlu/analyze → { tokens, pos, entities:[{type,span,text}], dependency, intent, confidence }
技術候補：

形態素解析器：SudachiPy, MeCab, Kuromoji（日本語向け）

NER / 意図分類：spaCy（ja）、transformers（Hugging Face）で日本語 BERT 系（cl-tohoku 等）を利用

SRL／依存解析：GiNZA（spaCy ベース）など
留意点：法律ドメイン辞書（条文・専門語彙）を統合し、NER 精度を高める。F1 目標の設定と評価データが必要。

Ginza（spaCy ベース）で使用

5.**法律知識ベース（KB） & 検索／知識グラフ** __must__

役割：条文、判例、解説、過去事例、裁判例メタデータの保管・検索・参照。
API 例

GET /kb/law?query= → { results:[{id,title,excerpt,link}] }

GET /kb/case/{id} → { full_text, metadata, citations }

POST /kb/semantic_search → { query_embedding, top_k_results }
技術候補：

ドキュメント検索：Elasticsearch / OpenSearch

ベクトル検索（類似判例）：Milvus、Pinecone、Weaviate

知識グラフ：Neo4j、JanusGraph（将来的な因果関係分析用）
留意点：法改正反映パイプライン、ソースの信頼性管理、メタデータ（施行日など）を厳格に管理。

6.**分析エンジン（過失割合・類似ケース・和解案推定）** __must__

役割：争点抽出→証拠重み付け→過失割合の確率的推定→複数の和解案（中立的）生成。
API 例

POST /analysis/estimate_fault → { parties:[...], facts:[...],
estimates:[{party,ratio,confidence}], rationale:[...] }

POST /analysis/generate_settlement_options →
{ options:[{type,terms,probability, rationale}] }
技術候補：

モデル：統計的モデル + 機械学習（回帰、ベイズ推定）＋ルールベース補正

データ：判例特徴量、過去和解データ、専門家ラベル付けデータ

ライブラリ：scikit-learn, PyMC3/Pyro（ベイズ）, XGBoost, LightGBM, PyTorch/TensorFlow
留意点：結果は確率的である旨を UI/説明で明示。Explainable AI（SHAP, LIME）を併用して根拠を可視化する。

7.**NLG（応答生成）モジュール**__must__

役割：分析結果を自然で法的に中立な日本語で出力。複数トーン・テンプレをサポート。
API 例

POST /nlg/generate → { input:{analysis_id, tone, profile}, output:{text,
citations:[{type,id}], confidence } }
技術候補：

LLM(商用 or 専用):Anthropic Claude, OpenAI GPT 系、あるいは企業向けカスタム LLM

ロジック:テンプレート＋LLM ハイブリッド(法的用語はテンプレ強制)
留意点:「法的助言ではない」旨の免責テンプレート、自動で出典を添付。応答の検閲・ガードレールを必須で実装。

8.**TTS(音声合成)モジュール**__must__(音声対応がある場合)

役割:生成テキストを自然な音声で返す。トーンや性別選択を可能に。
API 例

POST /tts/synthesize → { text, voice, speed } → audio (stream or URL)
技術候補:

商用:Google TTS, Amazon Polly, Azure Speech

オープン:Coqui TTS, Tacotron2 / FastSpeech2 + ワークフロー
留意点:法的注意喚起は必ず音声でも明示。音声版でもログ(テキスト版)を保存。

9.**管理・監査・学習基盤**__must__

役割:ログ蓄積、フィードバック収集、モデル管理、データ匿名化パイプライン。
API 例

GET /admin/audit/logs

POST /admin/feedback → { session_id, user_rating, comment }
技術候補:

ログ保存:Elastic Stack(ELK)、S3(長期保存)

モデル管理:MLflow, Seldon, Kubeflow

データパイプライン:Airflow、dbt
留意点:ログは改竄防止、匿名化ルールを自動化。学習データは法令遵守で保管。

10.**モニタリング & アラート**__must__

役割：パフォーマンス／品質／不正検知を監視。
技術候補：Prometheus + Grafana、Alertmanager、Sentry（エラー追跡）
留意点：低信頼回答率・急増する同一ケースのアラートなど、ドリフト検知ルールを組み込む。

11.**UI/UX（フロント）**__must__

役割：ユーザー入力画面、チャット UI、管理画面、弁護士向け詳細表示。
技術候補：React / Vue、TypeScript、WebSocket（音声ストリーミング）、アクセシビリティ対応。
留意点：応答に必ず根拠リンクを表示。緊急・高リスク時は弁護士相談への誘導を明確に。

_____**非機能要件**_____
非機能要件向けの技術・実装項目（要約）

スケーラビリティ：Kubernetes、Horizontal Pod Autoscaler、GPU ノードプール（推論）

可用性：多 AZ 配置、冗長 DB（Postgres + Patroni）、CDN（静的）

セキュリティ：TLS、WAF、KMS（暗号化鍵管理）、SIEM、脆弱性スキャン、RBAC

データ管理：Pseudonymization パイプライン、自動削除（Right-to-Delete）API

ガバナンス：モデルバージョン、AB テスト、承認フロー、監査ログ

品質評価：自動回帰テスト、ユースケースベースのテストスイート、評価指標（WER, F1, BLEU/ROUGE for NLG, calibration）

コスト管理：推論キャッシュ、軽量モデルのエッジ活用、スポットインスタンス利用

_____**優先度（MVP）**_____

初期 MVP ではまず次を優先します（早期検証のため）：

MUST（MVP）

テキスト入力 → NLU → 法律 KB 参照 → 分析（簡易過失割合） → NLG（テキスト応答）

管理者ログ・監査・匿名化パイプライン

認証（基礎）＋利用規約 / 免責表示

モニタリング（基本メトリクス）

後段フェーズ（SHOULD）

フル音声（ASR/TTS）対応

高度な Explainability（可視化ツール）

知識グラフ・高度な類似判例検索

感情システムの本格導入（ユーザートーン適応）

_____**修正点**_____
新たに追加が必要な要件・モジュール
1.**感情対話調整モジュール(Emotional Mediation Layer)**

目的：夫婦・家族などの「法的グレーゾーン＋感情的要素」を扱う際に、攻撃的言動を緩和し、建設的な折衷案を生成する。

機能要件

発話ごとの感情ラベル付け（怒り・悲しみ・苛立ちなど）

会話全体のトーンを評価（敵対／中立／和解傾向）

過激表現を中和し、非攻撃的言語に変換する再構成機能

折衷案生成の際に「感情バランス」を考慮（例：「両者の努力点を認めつつ…」）

技術候補

日本語感情分類モデル（BERT-based、Livedoor ニュースコーパスなどで再学習）

言語中和生成：LLM（GPT/Claude）＋「非攻撃言語テンプレート」

対話状態追跡：Rasa、Dialogflow CX、あるいは自作 FSM

2.**価値観・倫理バイアス調整モジュール（Ethical Neutralizer）**

目的：国や文化、性別による倫理的偏りを排除し、中立な判断を保証する。

必要性：夫婦・家庭問題では「性別バイアス」「伝統観」などが強く影響するため、モデルが偏見を持たないようにする必要がある。

技術候補：

Bias detection & mitigation layer（LLM 出力のバイアスチェック）

文化的パラメータ設定（国・宗教・家族観）をプロファイルで切替

実装例：

POST /nlg/neutralize に "cultural_context": "JP", "gender_balance": true などのパラメータ追加。

3.**金銭評価・賠償提案モジュール（Compensation Estimator）**

目的：過失割合の結果に基づき、金銭換算を提案する。

機能要件

損害／慰謝料の類型別データベース（民法 709 条系、離婚関連など）

過失割合に基づく金額シミュレーション

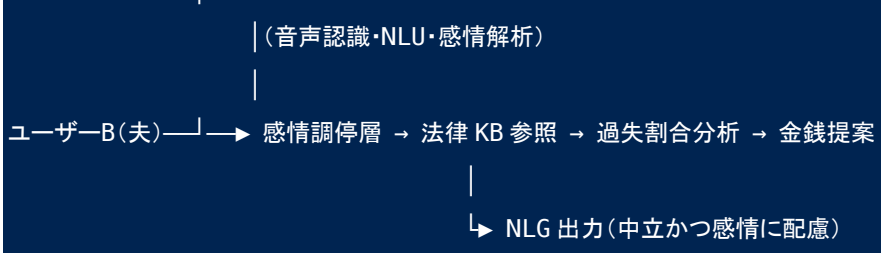
提案内容に「法的拘束力がない」旨を明記

技術候補：

統計データ＋判例ベースの相場表

数値推定：回帰モデル（XGBoost, LightGBM）

_____**アクティビティ、ユースケース**_____



最終出力（NLG）：

「お二人の言い分を整理すると、家事分担に関して夫側の負担割合は 30%、妻側 70%が妥当と思われます。
この場合、妻の精神的負担を考慮し、3 万円の謝罪的金銭補償を提案します。
ただし、これは法的拘束力を持たず、あくまで和解案の一つです。」