要件定義書　画面設計書

A. 要件定義書 (`要件定義書`)

1. 機能要件（ユーザーストーリー形式）

エピック：プロンプト管理

- **ストーリー1.1:** 貢献者として、私はリッチテキストエディタまたはビジュアル・コンポーザーを使って新しいプロンプトを作成し、下書き（`draft`）として保存できる。
- **ストーリー1.2:** 貢献者として、私は下書きをピアレビューのために提出でき、そのステータスが「レビュー中（`pending_review`）」に変わることを確認できる。
- **ストーリー1.3:** ユーザーとして、私は既存の承認済みプロンプトを「フォーク」して、それをベースにした新しい下書きを作成できる。
- **ストーリー1.4:** ユーザーとして、私はプロンプト詳細ページで、プロンプトの本文、理論的根拠（`rationale`）、関連する技術、作成者、バージョン履歴を閲覧できる。

エピック：ピアレビューとモデレーション

- **ストーリー2.1:** 貢献者として、私は自分のダッシュボードで、提出したプロンプトの現在のステータス（レビュー中、承認済み、却下済み）とレビューコメントを確認できる。
- **ストーリー2.2:** レビュアー（レピュテーション > 500）として、私はモデレーションキューにアクセスし、レビュー待ちのプロンプトを一覧できる。
- **ストーリー2.3:** レビュアーとして、私はレビュー対象のプロンプトに対して「承認」または「却下」の投票を行い、任意でフィードバックコメントを追加できる。
- **ストーリー2.4:** モデレーター（レピュテーション > 2000）として、私は不適切なコメントを削除したり、スパムとして報告された投稿を処理したりできる。

エピック：ゲーミフィケーションと評価

- **ストーリー3.1:** ユーザーとして、私の投稿したプロンプトがアップボートされるか、ピアレビューで承認されると、私のレピュテーションスコアが上昇する。
- **ストーリー3.2:** ユーザーとして、私が実施したピアレビューが他のレビュアーの多数派意見と一致した場合、私のレピュテーションスコアが少量上昇する。
- **ストーリー3.3:** ユーザーとして、特定の条件（例：最初のChain-of-Thoughtプロンプトが承認される）を満たすと、自動的に「First CoT」バッジが付与され、プロフィールに表示される。
- **ストーリー3.4:** ユーザーとして、私は他のユーザーのプロフィールページを訪れ、その人のレピュテーション、獲得バッジ、貢献履歴を閲覧できる。

エピック：ビジュアル・プロンプト・コンポジション

- **ストーリー4.1:** ユーザーとして、私はキャンバス上に「システム指示」「Few-Shot事例」「ユーザー入力」などの定義済みノードをドラッグ＆ドロップできる。
- **ストーリー4.2:** ユーザーとして、私はノード間のハンドルを接続して、プロンプトの実行順序や論理的な流れを定義できる。
- **ストーリー4.3:** ユーザーとして、私は各ノード内のテキストフィールドに具体的な指示や事例を記述できる。
- **ストーリー4.4:** ユーザーとして、私は作成したビジュアルグラフを、実行可能な単一のテキストプロンプトにワンクリックで変換（コンパイル）できる。

2. 非機能要件

- **パフォーマンス:**
  - 主要ページの平均ロード時間は2秒未満とする。
  - ビジュアル・コンポーザーでのノード操作（ドラッグ、接続）は、100ms未満の遅延でリアルタイムに反応すること。
- **セキュリティ:**
  - 全ての通信はTLS 1.2以上で暗号化されること。
  - データベースに保存されるパスワードやAPIキーなどの機密情報は、AES-256などの強力なアルゴリズムで暗号化されて保存されること。
  - **プロンプトインジェクション対策:** OWASP LLM Top 10のガイドラインに従う[28]。
    - 全てのユーザー入力を厳格にサニタイズ処理すること。
    - バックエンドで最終的なプロンプトを組み立てる際、システム指示とユーザーからの信頼できない入力を明確なデリミタ（例：`###`）やXMLタグで分離し、モデルに両者が異なるコンテキストであることを明示的に指示すること。
- **スケーラビリティ:**
  - システムは、初期段階で10,000人の同時アクセスユーザーと、データベース内に100万件のプロンプトレコードを、パフォーマンスの顕著な低下なく処理できるように設計されること。
- **可用性:**
  - システムの稼働率は99.9%を目指す。

B. 画面設計書 (`画面設計書`)

1. 全体的なUI/UX哲学

GitHubやStack Overflowのような、技術者向けプラットフォームの設計思想に倣う。クリーンで情報密度が高く、ナビゲーションが直感的であることを最優先する。過度な装飾を排し、コンテンツの可読性と機能性を重視したプロフェッショナルなデザインとする。

2. コンポーネントアーキテクチャ

Reactのベストプラクティスに基づき、保守性と再利用性の高いコンポーネント構造を採用する。

- **Container/Presentational パターン**: データ取得や状態管理ロジックを持つ「コンテナコンポーネント」と、propsを受け取ってUIを描画することに専念する「プレゼンテーショナルコンポーネント」を明確に分離する [31]。
  - **例**: `PromptPageContainer.tsx` がAPIからプロンプトデータを取得し、そのデータを `PromptDetailView.tsx` コンポーネントにpropsとして渡す。
- **カスタムフック**: 状態を持つロジックをコンポーネントから抽出し、再利用可能なカスタムフックとしてカプセル化する [34]。
  - **例**: `usePromptData(promptId)`、`useUserProfile(userId)`、`useReviews(promptId)` などを作成し、データフェッチ、ローディング状態、エラーハンドリングを内部で管理する。
- **Compound Components パターン**: 密接に関連する複数のコンポーネントを一つのグループとして扱い、暗黙的な状態共有を可能にする。プロンプト投稿フォームのような複雑なUIに適している [31]。

3. 主要画面仕様

画面：プロンプト詳細ページ (`/prompts/{id}`)

- **レイアウト**: 2カラム構成。
- **左カラム（メインコンテンツ）**:
  - `PromptHeader` コンポーネント：プロンプトタイトル、作成者情報（アバター、ユーザー名、レピュテーション)、投稿日時、タグリストを表示。
  - `VoteControl` コンポーネント：アップ/ダウン投票ボタンと現在のスコアを表示。
  - `PromptBody` コンポーネント：
    - `VisualView` タブ：`prompt_body_json` を読み取り、React Flowを用いてビジュアル・コンポーザーの読み取り専用ビューを描画。
    - `TextView` タブ：`prompt_body_text` を、シンタックスハイライト付きのコードブロックとして表示。
- **右カラム（メタ情報）**:
  - `TechniqueCard` コンポーネント：このプロンプトに紐づく `PromptTechniques` をリスト表示。各技術は「技術ツリー」ページへリンク。
  - `RationaleBox` コンポーネント：`rationale` （理論的根拠）のテキストを表示。
  - `ABTestPanel` コンポーネント（フェーズ3）：このプロンプトの別バージョンとのパフォーマンス比較テストを開始・表示するUI。
  - `CommentsThread` コンポーネント：プロンプトに関するディスカッションを行うコメント欄。

画面：ビジュアル・プロンプト・コンポーザー (`/compose`)

- **コアコンポーネント**: `<ReactFlowCanvas>` コンポーネント。React Flowライブラリをラップし、Issuepedia独自の機能を追加する。
- **状態管理**: ノードとエッジの状態は、パフォーマンス上のボトルネックを避けるため、Zustandのような外部状態管理ライブラリで管理する [36]。これにより、コンポーネントツリーの深部にあるノードからでも、状態を効率的に更新できる。
- **サイドバーパネル**:
  - **ノードパレット**:
    - ドラッグ可能なカスタムノードのリスト。「役割設定（System Role）」「基本指示（Instruction Block）」「少数事例（Few-Shot Example）」「ユーザー入力（User Input）」などの種類を用意。
  - **プロパティインスペクタ**:
    - キャンバス上で選択されたノードの詳細（テキスト内容、設定など）を編集するためのフォームを表示。
- **カスタムノード**:
  - 各ノードタイプ（例：`InstructionNode`）は、独自のReactコンポーネントとして定義される。入力用のテキストエリア、接続用のハンドル（`Handle` コンポーネント）を持つ [3]。
- **ツールバー**:
  - 「テキストに変換」「保存」「A/Bテストに送信」などのアクションを実行するボタンを配置。

画面：ユーザープロフィール＆レピュテーションダッシュボード (`/users/{username}`)

- **レイアウト**: 上部にユーザー情報、下部にタブ付きのコンテンツエリア。
- `UserProfileHeader` **コンポーネント**: ユーザーのアバター、ユーザー名、レピュテーションスコア、所属、自己紹介文を表示。
- `TabbedContent` **コンポーネント**:
  - **「貢献」タブ**: そのユーザーが投稿したプロンプトのリストを表示。各項目にはタイトル、ステータス、スコアが含まれる。
  - **「レビュー」タブ**: そのユーザーが過去に行ったレビューの履歴を表示。
  - **「バッジ」タブ**: 獲得した全てのバッジをアイコンと共に表示。
  - **「アクティビティ」タブ**: レピュテーションの変動履歴を時系列で表示するタイムライン。Rechartsのようなグラフ描画ライブラリを使用し、レピュテーションの推移を視覚的に表示する [38]。