**✅ 1. システム全体設計に必要な項目**

**1.1 アーキテクチャ設計**

* クライアントサイド：SPA構成（Vue 3 + Vuetify + Pinia）
* API通信方式（REST + JWT認証）
* サーバー構成（Express + PostgreSQL）
* デプロイ構成（Docker化するか否か、CI/CD構成など）
* ロギング方式（例：Winston + Cloud Logging）

**1.2 非機能要件への対応策**

* レスポンス1秒以内のためのチューニング方針
* 負荷分散・同時接続200名に対するスケーラビリティ戦略
* バックアップの運用設計（日次の自動化手法、保存先、復元検証方法）

**✅ 2. フロントエンド設計**

**2.1 画面一覧と遷移設計**

* ログイン画面
* 勤怠打刻画面（従業員用）
* 勤怠閲覧画面（日別／月別／自分用）
* 店舗別ダッシュボード（責任者用）
* 管理者用管理画面（従業員・店舗マスタ）
* 共通UI部品（ナビゲーションバー、通知など）

**2.2 コンポーネント構成**

* ボタンや入力フォームなど、UI再利用設計
* Pinia Store設計（AuthStore, AttendanceStore, EmployeeStore等）

**2.3 レスポンシブ対応仕様**

* 画面幅によるコンポーネント表示条件
* タッチ操作優先のUI最適化（タブレット用）

**✅ 3. バックエンド設計**

**3.1 API設計詳細**

* 各エンドポイントのRequest/Response仕様書
* バリデーション設計（例：Joi or Zod）
* エラーハンドリング設計（エラーコード一覧、HTTPステータス対応表）

**3.2 認証・認可設計**

* JWTの有効期限、リフレッシュトークン有無
* RBAC（Role Based Access Control）の詳細ロール定義

**3.3 GPS・IP制限仕様（不正検知機能向け）**

* 緯度経度の取得・許容範囲
* IPホワイトリスト or ブロックリスト方式の可否

**✅ 4. データベース設計**

**4.1 論理データモデル（ER図）**

* users, roles, employees, stores, attendances, schedules など
* 外部キー・正規化・インデックス戦略

**4.2 マスタ設計**

* 勤務パターン（早番、遅番、フレックス等）
* 休日・祝日定義の柔軟な管理方法

**4.3 初期データ設計**

* 店舗マスタ・ロールマスタ・デモ従業員などのシードデータ

**✅ 5. テスト設計**

**5.1 単体テスト**

* バックエンド：ユニットテスト（Jestなど）
* フロントエンド：コンポーネント単位テスト（Vitest + Vue Test Utils）

**5.2 結合テスト**

* フロント ⇔ API間の一連の動作確認
* 打刻 → 閲覧 → 集計 → ロール別制御の流れ確認

**5.3 E2Eテスト**

* Playwright または Cypress を使った自動テスト設計

**✅ 6. 開発運用・セキュリティ設計**

**6.1 環境構成**

* .env 管理の設計（開発 / ステージング / 本番）

**6.2 ロールバックとエラートラッキング**

* Sentryなどでのエラー通知
* DBバージョン管理（例：Prisma Migrate または Flyway）

**6.3 セキュリティ対策**

* CSRF/XSS/SQL Injection対応
* JWTの漏洩・期限切れ対応
* HTTPS・CORS設定のポリシー

**✅ 7. ドキュメント／運用に必要なもの**

* ER図、API仕様書（OpenAPI/Swagger推奨）
* フロントエンドの画面遷移図・画面仕様書
* テスト仕様書（項目と目的）
* 運用マニュアル（バックアップ手順、アカウントロック対応など）

**✅ 8. 将来拡張を見越した検討ポイント（備考）**

* シフト作成：従業員ごとの週単位・月単位スケジューラ設計
* モバイルアプリ連携：API設計の共通化と認証方式の整合性
* 外部連携：CSV出力/インポート、給与システム連携API

ご希望があれば、**ER図のドラフト作成**や**Swagger形式のAPI設計サンプル**なども提供可能です。設計の次フェーズに入る際に、それぞれ具体化していきましょう。