MRT Enterprise ERP - Project Documentation

1. Project Overview

This project is a comprehensive School Management System / Enterprise Resource Planning (ERP) application developed for an educational trust (MRT). It manages multiple academic programs, student life cycles, financial operations, human resources, and quality assurance processes.

The system leverages a modern web architecture, utilizing Next.js App Router and Server Actions, to provide a fast, secure, and user-friendly experience across all administrative layers. It operates with a strict Role-Based Access Control (RBAC) system to ensure that operations remain secure and are only performed by authorized personnel.

2. Technology Stack

Core Technologies

- **Framework:** Next.js 16 (App Router) with React 19.
- **Language:** TypeScript for end-to-end type safety.
- **Backend Architecture:** "API-less" Architecture using Next.js Server Actions.

Database & ORM

- **Database Engine:** PostgreSQL
- **ORM (Object Relational Mapper):** Prisma ORM (v5), offering strong typings and simplifying database migrations, seeding, and transactions.

Styling & UI

- **Styling Framework:** Tailwind CSS v4
- **UI Component Library:** Shadcn UI (built on top of Radix UI primitives) to provide accessible, highly customizable components (Tabs, Dialogs, Dropdowns, Selects, etc.).
- **Icons:** Lucide React
- **Animations:** `tailwindcss-animate`

Forms & Validation

- **Form State Management:** React Hook Form
- **Schema Validation:** Zod (used extensively both on the client side for form validation and on the server side inside Server Actions for data integrity).
- **Resolvers:** `@hookform/resolvers` integration for smooth React Hook Form and Zod compatibility.

Authentication & Security

- **Auth Library:** Auth.js (NextAuth.js v5 beta).
- **Password Hashing:** `bcrypt` for encrypting user passwords.

Utilities

- **Charts and Data Visualization:** Recharts
- **Date Manipulation:** `date-fns` and `react-day-picker`
- **Toast Notifications:** Sonner

3. System Architecture & How It Works

Architectural Guidelines

Per the `mrt-architecture-guidelines` module, the main architecture follows these principles:

- **Server Components Paradigm:** Uses React Server Components (RSC) to reduce client-side bundle size, fetch data instantly on the server, and improve page load speeds.
- **API-less Structure:** Traditional REST or GraphQL API routes (e.g., `/api/rooms`) are bypassed. Instead, Next.js Server Actions securely encapsulate business logic. The client calls these server actions directly, treating them like RPCs (Remote Procedure Calls).
- **Optimistic UI:** Uses `revalidatePath` and `revalidateTag` inside Server Actions to instantly reflect data mutations onto the UI.
- **ACID Transactions:** To implement safe bulk operations (e.g., fee voucher generation), Prisma transactions are employed.

Execution Flow

1. **Authentication:** The user signs in via the `auth` module. The active session identifies their `Role` (e.g., `SUPER_ADMIN`, `TEACHER`).
2. **Routing:** Depending on the user's role, the Next.js middleware or layout components verify authorization. The `app/dashboard` defines module-specific routes.
3. **Data Fetching:** When accessing a route like `/dashboard/finance` or `/dashboard/student`, data retrieval logic strictly executes on the server.
4. **Data Mutation:** Submitting a form passes the payload to a Server Action situated in the `actions/` directory (like `actions/finance.ts`). The Action applies Zod schema validations then executes Prisma queries to finalize changes.

4. Roles and Responsibilities

The system implements a granular role structure via the `Role` enum in the database.

1. **SUPER_ADMIN**
   - **Functions:** Possesses unhindered access to all functional areas of the system. Usually handles the core configuration, system-level audits, database overrides, and creating high-level management accounts.
2. **TRUST_MANAGER**
   - **Functions:** An executive-level role focused on overseeing the enterprise. Can access reports, supervise Human Resources (HR), run Staff Evaluations, and analyze quality matrices.
3. **SECTION_HEAD**
   - **Functions:** In charge of supervising specific academic sectors. Manages classes, programs (RFL, MRHSS, MRA), and oversees teachers assignments over particular subjects.
4. **FEE_DEPT**
   - **Functions:** Governs finances entirely. Has the authority to determine class fee structures, issue monthly fee vouchers, log defaults, and track financial disbursements corresponding to students.

5. **ADMISSION_DEPT**
    - **Functions:** The gatekeepers of the system. Manages student registration workflows, handles inquiries, enacts program enrollment, registers guardian info, and specifies student accommodation needs (like hostels).
6. **EXAM_DEPT**
    - **Functions:** Concentrates heavily on academics, particularly testing. Registers tests/exams, uploads and processes student marks globally, and publishes results.
7. **TEACHER**
    - **Functions:** Primary academic facilitators assigned to subjects within a class. Can mark daily student attendance, leave performance remarks, submit feedback regarding students, and view their assigned coursework.
8. **STUDENT**
    - **Functions:** End-users accessing their personal metrics. They view their marksheets, download fee vouchers, scrutinize attendance logs, submit surveys, and provide feedback on teachers entirely anonymously.

5. Key Modules

1. Finance Module (`app/dashboard/finance` & `actions/finance.ts`)

Tracks monetary records comprehensively. Defines the baseline `FeeStructure` representing tuition and hostel costs. Operates the `FeeVoucher` and `Disbursement` systems to manage income operations and outgoing expenditures, separating student statuses into "PAID" and "UNPAID".

2. Admissions Module (`app/dashboard/admissions` & `actions/admissions.ts`)

Facilitates entry to the institution. Models the `StudentProfile`, establishing distinct linkage directly to a `User` account while retaining guardian/beneficiary backgrounds, and records their status through the `ProgramEnrollment`.

3. Academics Module (`app/dashboard/academics` & `actions/academics.ts`)

Puts the educational engine together. Manages core curriculum records comprising `Class`, `Subject`, and applies teaching staff through the `TeacherAssignment` junction.

4. HR Module (`app/dashboard/hr` & `actions/hr.ts`)

Focuses on employee health. Captures the `StaffEvaluation` dataset leveraging key 1-5 scalar metrics covering punctuality, quality of work, engagement, and communication to derive average scores automatically.

5. Quality Enhancement Cell / Exams (`app/dashboard/qec` & `actions/qec.ts`)

Guarantees systemic quality control. Uses models like `Survey`, `Feedback`, and tracked `SurveySubmissions` to allow anonymous feedback channels between participants to rate performance seamlessly. It also acts as the central hub for entering and processing `StudentMark`.

6. RFL (Roots for Life) Module (`app/dashboard/rfl` & `actions/rfl.ts`)

Alumni and higher education tracking system spanning long-term progress. Captures `RFLAcademicRecord` for students advancing to universities, noting details such as university name, dynamic course lengths (Semesters), and cumulative GPAs across their lifecycle.