EMI School Fees Collection Web App Documentation

Overview

This project is a web application for managing school fee collections on a monthly EMI basis. Built using **React** and **TypeScript**, it offers a centralized platform for handling administrative tasks across different user roles like Super Admin, Institute Admin, Student, Franchise Admin, Sales, and KYC Support. The system includes monthly payment reminders and tracks total, pending, and completed payments.

Key Features

🔐 User Authentication

- Secure login using email and password.
- Redirects users to dashboards based on role.

👩‍🏫 Role-Based Dashboards

- **Super Admin:** Full administrative access.
- **Institute Admin:** Manages institute activities.
- **Student:** Views fees, due dates, and payment history.
- **Sales & Referral:** Tracks sales and referrals.
- **Support:** kyc done/or not.

💳 Fees Scheduler Dashboard

- Displays next due amount, due date, total paid, and pending amount.

- Highlights due fees in red and upcoming ones in gray.
- Payment reminder for each month displayed clearly in the schedule.
- Integrated **UPI AutoPay/E-NACH** option with status.
- Auto reminder & notification system for due payments.

📅 Payment Section

- Lists what is due this month.
- Summarizes total paid and pending balance.
- Tracks recent payments with status tags (Paid/Due).

🧾 Transaction History

- Detailed table view of all transactions:
    - **Date**, **Transaction ID**, **Fee Type**, **Amount**, **Payment Method**, **Status**, and **Receipt**
- Filters available:
    - By **date range**, **method**, **type**
- **Receipt Download** button for every transaction
- Status Labels:
    - ✅ `Paid` – Payment successfully processed
    - ❌ `Failed` – Payment failure (retry option)
    - 🕒 `Due` – Pending dues
- Transparent log of:
    - Manual payments

- Auto-deductions (UPI AutoPay)
- Special fee types like **"Monthly + Exam Fee"**

🏫 Institute Management

- Institutes can register and manage their profile, contacts, and address.
- Each institute dashboard includes payment data and student insights.
- Add/Edit/Delete institutes.
- Manage institute staff and assign roles.

🧑‍🎓 Student Profile Page

This section allows students to view their personal, academic, and guardian-related information in a structured and readable layout. It's part of the **Student Dashboard** and enhances transparency and personalization.

📋 Information Displayed:

🔹 Student Information

- **Full Name:** Arjun Sharma
- **Student ID:** ABC2024001
- **Class:** 10th A
- **Roll Number:** 15
- **Admission Date:** April 1, 2024

🔹 Contact Information

- **Email:** arjun.sharma@email.com
- **Phone:** 9876543210
- **Address:** 123 Main Street, Mumbai, Maharashtra - 400001

🔹 **Parent/Guardian Information**

- **Parent Name:** Raj Sharma
- **Parent Phone:** 9876543211
- **Parent Email:** raj.sharma@email.com

🔹 **Academic Status**

- **Attendance:** 95%
- **Current Grade:** A+
- **Subjects Enrolled:** 8
- **Status:** Active

✨ **Features**

- Responsive UI for easy access across devices.
- Editable profile data via "Edit" button (if allowed by permissions).
- Integrated with backend to fetch real-time student data.
- Useful for both students and institute admins to verify details.

🧰 Admin Panel (New)

- View all users across all roles with search and filtering.
- Enable/disable user accounts.
- Monitor failed transactions and manually verify payments.
- Analytics section: fees collected vs. pending graphically represented.

🔔 Notification System (New)

- In-app and email notifications.
- Alerts for upcoming dues, payment failures, and profile updates.
- Configurable frequency and reminder timings.

📦 Data Schema (MongoDB)

Student Schema

```
const StudentSchema = new mongoose.Schema({
  fullName: String,
  studentId: String,
  class: String,
  rollNumber: Number,
  admissionDate: Date,
  email: String,
  phone: String,
  address: String,
  parent: {
    name: String,
    phone: String,
    email: String,
  },
  academicStatus: {
    attendance: Number,
    currentGrade: String,
    subjects: Number,
    status: String,
  }
});
```

Institute Schema

```
const InstituteSchema = new mongoose.Schema({
  name: String,
  address: String,
  contactEmail: String,
  contactPhone: String,
  students: [StudentSchema],
});
```

🔗 API Endpoints (REST)

Student API

- `GET /api/students/:id` → Get student profile
- `PUT /api/students/:id` → Update student profile
- `GET /api/students/:id/payments` → Get payment history

Payment API

- `GET /api/payments/:studentId` → Fetch all transactions
- `POST /api/payments/initiate` → Start new payment (manual/UPI)
- `PUT /api/payments/:transactionId/status` → Update payment status

Institute API

- `GET /api/institutes/:id` → Get institute profile
- `PUT /api/institutes/:id` → Update institute data

Admin API (New)

- `GET /api/admin/users` → List all users
- `PUT /api/admin/users/:id/status` → Activate/deactivate user
- `GET /api/admin/analytics/fees` → Get fee collection analytics

Notification API (New)

- `POST /api/notifications/send` → Send notification to user
- `GET /api/notifications/:userId` → Get all notifications for user

Project Structure

Root Directory

- `package.json`, `tsconfig.json`, `vite.config.ts`: Configuration and setup files.

`src` Directory

- `App.tsx`: Routes and structure setup.
- `components/`: Modular UI components (admin, franchise, institute, student, ui).
- `pages/`: Contains main views like `Login.tsx`, `StudentDashboard.tsx`, `RegisterInstitute.tsx`.
- `hooks/`: Custom reusable React hooks.
- `lib/`: Shared utilities and helper functions.

Technologies Used

- **Frontend:** React, TypeScript, Vite
- **UI Components:** Tailwind CSS, custom components
- **Authentication:** Role-based routing
- **Backend:** Node.js, Express.js
- **Database:** MongoDB with Mongoose

- **Payment Tracking:** UPI & manual options
- **Notification System:** Email + in-app alerts
- **Admin Dashboard:** Full control and analytics

Conclusion

This EMI fee collection system simplifies fee tracking and reminders for schools and parents. With user-specific dashboards, automated reminders, analytics, and admin control, it ensures a smooth and transparent fee management process.

For any questions or further documentation on specific components or logic, feel free to ask!