System Architecture Summary

School Management Application

Architecture Overview

The School Management System follows a **three-tier architecture** consisting of:

Frontend (Client Layer) – React / Angular / Vue

Backend (Application Layer) – Node.js / Django / Spring Boot

Database (Data Layer) – PostgreSQL / MySQL / MongoDB

This structure ensures the system remains scalable, modular, and easy to maintain.

Component Breakdown

Frontend (React / Angular / Vue)

The frontend is responsible for everything the user sees and interacts with.
It handles:

UI screens (Login, Dashboard, Student Profile, Attendance, Fees, Timetable)

Input validation (before sending data to backend)

Displaying data received from backend APIs

State management (user session, role-based UI changes)

The frontend communicates with the backend using **REST APIs** over HTTPS.

Backend (Node.js / Django / Spring Boot)

The backend acts as the core engine of the system.
It performs:

Business logic (attendance marking, exam results calculation, fee updates)

Role-based authorization (Admin vs Teacher vs Student)

Input validation and sanitization

API endpoints for CRUD operations

Communicating with the database

Ensuring security (JWT authentication, password encryption)

The backend exposes a structured API layer like:

GET /students
POST /attendance
GET /fees/status
POST /auth/login

Database (PostgreSQL / MySQL / MongoDB)

Stores all persistent data related to:

Students, Teachers, Parents

Classes, Subjects, Timetable

Attendance, Exam results, Fees

Login credentials (hashed passwords)

Activity logs

Relational databases (PostgreSQL / MySQL) are ideal for structured academic data.
MongoDB works well when flexibility or document-style storage is preferred.

How All Components Work Together (The Flow):

Step-by-step Workflow

User interacts with the Frontend
Opens dashboard, views classes, marks attendance, etc.

Frontend sends an **API request** → Backend
Example:
POST /attendance/mark with student_id, date, status

Backend receives the request

Verifies JWT token

Applies business rules

Validates data

Checks permissions (e.g., only teachers can mark attendance)

Backend communicates with Database
Reads/writes required records
Example: Insert attendance for a student on a given date.

Database returns data → Backend → Frontend
Frontend updates the UI accordingly.
