School SAS - Product Requirements Document (PRD)

1. Overview

School SAS is a full-featured School Automation System built using JavaScript-based web technologies. It provides dedicated portals for Admin, Accountant, Student, Teacher, and Parent. The system is divided into four modular subsystems: Notification System, Payment System, Study System, and AI Integration. Each subsystem is designed for fault isolation—if one fails, others continue to function.

2. Goals & Objectives

The primary goal is to digitize school operations and improve communication among stakeholders. Key objectives include: - Streamlined academic management - Secure digital payments - Real-time communication through notifications (Email & WhatsApp) - Parent visibility into student performance - Modular architecture supporting future AI integrations

3. System Architecture

The application follows a modular microservice-based architecture with fault tolerance. Each module—Study, Payment, Notification, and AI—is independently deployable. Frontend will use React.js/Next.js, Backend Node.js (Express.js), and MongoDB as the database. Message queues (e.g., RabbitMQ or Kafka) handle asynchronous notifications and inter-module communication.

4. User Roles & Key Features

a. Student Portal:

- View class schedule, progress, marks, and attendance.
- Access circulars, digital diary, academic syllabus, notes, and previous year questions.

b. Parent Portal:

- Monitor child’s progress, attendance, and diary.
- Receive notifications via email/WhatsApp.
- Make secure fee payments and download receipts.

c. Teacher Portal:

- Enter marks, upload assignments, and manage attendance.
- Send circulars, diary entries, and notifications to parents.

d. Accountant Portal:

- View payments, verify transactions, and manage financial reports.

e. Admin Portal:

- Monitor overall school statistics including student results and teacher performance.
- Manage user roles and permissions.
- View class-wise analytics dashboards.

5. Module Breakdown

5.1 Notification System
Teachers can send announcements, circulars, and diary updates. Messages are automatically sent to parents through email and WhatsApp using an integrated API (e.g., Twilio, WhatsApp Business API).
5.2 Payment System
Parents can make online payments via Razorpay or Stripe integration. The accountant validates and generates digital receipts. Payment history and pending dues are trackable in real-time.
5.3 Study System
Comprehensive academic management covering timetables, notes, assignments, progress reports, attendance, syllabus, and academic circulars. Teachers update data; students and parents view reports dynamically.
5.4 AI Integration (Future Stage)
Planned for predictive analytics, performance prediction, and smart recommendation systems using AI models integrated with TensorFlow.js or PyTorch backend.

6. Technical Stack

Frontend: React.js / Next.js Backend: Node.js (Express) Database: MongoDB / Firebase Notification Service: Twilio API, SMTP Payment Gateway: Razorpay / Stripe Hosting: AWS / Vercel / Render Authentication: JWT + OAuth 2.0 Version Control: GitHub / GitLab

7. Non-Functional Requirements

- System uptime ≥ 99.5%
- All modules operate independently (fault-tolerant).
- Data security: AES-256 encryption, HTTPS enforced.
- Scalable microservice deployment (Docker/Kubernetes).

8. Future Enhancements

Integration of AI models for auto-grading, smart attendance prediction, and personalized learning recommendations. Expand notification coverage to include Telegram and SMS.

9. Project Development Flow

The system development follows this structured flow based on the provided Mermaid diagram, ensuring clarity from ideation to deployment with checks at every phase.