Based on the product blueprint for the School Management System, here is a complete backend folder structure and an explanation of the purpose of each component. This structure is designed to be scalable and maintain all the functionalities outlined in the document.

**Core Folder Structure**

The core of the backend is built around a clear, modular structure to manage different parts of the application.

* src/: This is the main source directory for all the backend code.
  + controllers/: Contains the business logic and request handling for each route. Each file here will handle a specific resource (e.g., students, teachers, exams).
    - admin.controller.js
    - attendance.controller.js
    - auth.controller.js
    - events.controller.js
    - exams.controller.js
    - notifications.controller.js
    - reports.controller.js
    - students.controller.js
    - teachers.controller.js
    - users.controller.js
  + middlewares/: Holds middleware functions for tasks like authentication, authorization, and validation.
    - adminAuth.middleware.js: Ensures only admin users can access certain routes.
    - auth.middleware.js: Verifies user tokens and authenticates requests.
    - teacherAuth.middleware.js: Ensures only teacher users can access certain routes.
    - studentAuth.middleware.js: Ensures only student/parent users can access certain routes.
  + models/: Defines the database schemas for each data type.
    - Admin.model.js
    - Attendance.model.js
    - Event.model.js
    - Exam.model.js
    - Notification.model.js
    - PreviousPaper.model.js
    - Report.model.js
    - Student.model.js
    - Teacher.model.js
    - User.model.js: A base model to handle shared user properties (name, email, role, etc.).
  + routes/: Defines the API endpoints and connects them to the appropriate controllers.
    - admin.routes.js
    - attendance.routes.js
    - auth.routes.js
    - events.routes.js
    - exams.routes.js
    - notifications.routes.js
    - reports.routes.js
    - students.routes.js
    - teachers.routes.js
    - analytics.routes.js
  + services/: Contains reusable business logic that is separate from the controllers. This can include services for interacting with AI, sending notifications, or handling file uploads.
    - AIService.js: Manages interactions with the AI content provider for live updates, study tips, and motivational quotes.
    - FileService.js: Handles file uploads for previous exam papers.
    - NotificationService.js: Manages the sending of notifications and announcements.
    - PaymentService.js: A service for integrating with external payment gateways for fee management.
  + config/: Stores configuration files, such as database connection settings, API keys, and server port.
    - db.config.js
    - keys.config.js
  + utils/: Utility functions that can be used across the application.
    - helpers.js: General helper functions.
    - validator.js: Functions for data validation.
  + app.js: The main entry point of the application, where the server is initialized and routes are configured.

**Functionality and Data Flow**

This structure supports all the features described in the blueprint by ensuring a clear separation of concerns.

* **Authentication & Authorization**: The auth.controller.js and auth.routes.js handle user login, while the auth.middleware.js and role-specific middleware (adminAuth.middleware.js, teacherAuth.middleware.js, studentAuth.middleware.js) ensure that each user can only access the features relevant to their role. For instance, a teacher can't edit events , and a student can only view events.
* **Student/Parent Features**:
  + **Data Retrieval**: The students.controller.js and students.model.js handle fetching and displaying student information like name, roll number, class, and attendance percentage. The

Reports.model.js provides academic performance and remarks.

* + **Upcoming Events & Exams**: The events.controller.js and exams.controller.js provide read-only access to event and exam timetables. The

PreviousPaper.model.js and associated controller manage the repository of past papers.

* **Teacher Features**:
  + **Attendance & Reports**: The attendance.controller.js handles marking and editing student attendance. The

reports.controller.js allows teachers to add academic and behavioral remarks.

* + **Exams & Schedules**: Teachers can upload exam results and manage previous papers via the exams.controller.js and PreviousPaper.model.js. They can also view and modify their teaching schedule using the

teachers.controller.js.

* **Admin/Management Features**:
  + **Full Control**: The admin.controller.js and admin.routes.js provide full control over all data. This includes managing events, reports, schedules, and exams.
  + **Notifications & Analytics**: The notifications.controller.js and analytics.routes.js handle sending announcements to all users and displaying dashboards for attendance trends, academic performance, and teacher activity logs.
* **AI Integration & Scalability**:
  + **AI Content**: The AIService.js pulls live AI content such as study tips, motivational quotes, and educational news. For teachers, it provides insights like "Class X has low average in Math".
  + **Future Scalability**: The structure has dedicated service files (PaymentService.js, AIService.js) to allow for easy integration with future features like payment gateways, chat support, or GPS tracking without overhauling the core structure.