Graduation Project

Next Gen Edu

Role-Based Smart LMS with IoT Integration

By

Ahmad El-sayed Abdullah El-sayed
Ahmed Ehab Gamil Ali
Abbas Ibrahim Abbas
Eslam Fawzi Metwally Abdelghany
Muhammad Essam Abd-El Hamid
Moaz Talaat Salem

Neveen Abbas Hassan
Hager Ahmed Mahmoud
Hager Tarek El-hassanein
Hala Reda Fawz
Hoda Osama Ibrahim

Supervised by

Dr. Elshaimaa Nabil Nada

Zagazig University
Faculty of Engineering
Computer and Systems Engineering

July 2025

Acknowledgment

It has been a great opportunity to get a lot of real-world project experience as well as knowledge about how to design and analyze real projects. We would like to express our gratitude to Allah for enabling us to complete our project and reach this learning point.

We would like to express our deepest appreciation to Dr. Elshaimaa Nabil Nada Ph.D. Professor, Computer and systems dep., for her supervision in harmony with us and guidance, continuous encouragement, and support during this project. We are deeply grateful to all who supported us in all aspects

Contents

1 Project Overview

Next Gen Edu is a modern, role-based Learning Management System designed to digitize and enhance the educational experience across universities and schools. It offers flexible customization, hardware integration (e.g., RFID attendance and smart cameras), and dashboards tailored to students, teachers, and administrators.

Unlike existing LMS platforms, it focuses on local needs, Arabic support, and integration with university structures like schedules, departments, and building data. It is developed as a full-stack solution supporting web and mobile access, with an intelligent assistant and rich analytics.

2 Problem Statement

The rapid advancement in educational technology and increasing dependence on digital platforms have emphasized the need for robust, flexible, and user-centric Learning Management Systems (LMS). While platforms like Google Classroom, Microsoft Teams, and Blackboard have made strides in digitizing education, they often fail to meet the specific needs of diverse educational institutions.

Key challenges include:

- Limited customization options.
- Missing built-in features requiring external integrations.
- Lack of hardware integration (e.g., attendance scanners, smart classrooms).
- Incompatibility with local policies and language preferences.

Hence, there's a demand for a smart, adaptable LMS that bridges the gap between traditional and modern education systems using advanced technology and flexible configuration.

3 System Objectives

Next Gen Edu aims to provide:

- **Role-Based Dashboards**: Customized dashboards for students, teachers, and administrators.
- **Hardware Integration**: Seamless support for attendance scanners and classroom cameras.
- **Content Management**: Easy upload and organization of materials.
- **Scheduling Tools**: Dynamic timetables and calendar features.
- **Communication Tools**: Announcements, messaging, and forums.
- **Full Platform Access**: Consistent user experience on web and mobile.
- **Comprehensive Tools**: Assignments, exams, progress tracking, intuitive interface.

- **Institution-Based Customization**: Management of buildings, departments, and academic groups.
- **Data Privacy and Security**: Compliance with modern standards.

4 Key Features

4.1 Role-Based Dashboards

- **Student**: View courses, assignments, grades, attendance, and notifications.
- **Teacher**: Manage courses, attendance, assessments, and communication.
- **Admin**: User control, system settings, and reports.

4.2 Attendance Tracking

- RFID and facial recognition device support.
- Real-time tracking and attendance history.

4.3 Material Management

- Support for videos, documents, and interactive content.
- Version control and revision tracking.

4.4 Scheduling and Performance

- Dynamic calendar for sessions and exams.
- Monitor participation and academic performance.

4.5 Communication Tools

- Announcements.
- Internal messaging.
- Academic discussion forums.

4.6 Evaluation

- Assignment and exam management.
- Automated and manual grading.
- Performance insights and feedback.

4.7 Customization

- Configure departments, student groups.
- Manage halls and room allocations.

4.8 Usability

- Interactive campus map.
- AI-powered assistant.

5 System Comparison

This section presents a comparison between popular learning management systems — **Blackboard**, **Google Classroom**, and **Microsoft Teams** — alongside our proposed system **Next Gen Edu**. The comparison includes features, integration, advantages, and limitations to illustrate the strengths and gaps of each system in the context of modern education needs.

Feature	**Blackboard**	**Google Classroom**	**Microsoft Teams**	**Next Gen Edu**
User Interface	Complex but customizable	Simple and user-friendly	Modern, can be overwhelming	Role-based, may require training
Integration	Supports LTI, third-party tools	Full Google Workspace (Docs, Drive)	Microsoft 365 (Word, Excel)	Hardware + university systems + third-party apps like Zoom
Grading and Feedback	Advanced rubrics and auto-grading	Basic grading, limited rubrics	Custom grading with Microsoft Forms	Manual and auto-calculated grades, feedback on submissions
Communication	Announcements, messages, forums	Comment threads only	Chats, meetings, channels	Role-based chats, notifications, subject groups
Collaboration	Shared spaces, forums	Google Docs projects	Office apps, Teams channels	Student forum, uploads section, materials sharing
Content Management	SCORM support, all file types	Google Drive integration	File sharing via SharePoint	Instructor and student sections, lecture videos, schedules
Mobile App	Functional but limited	Very user-friendly	Similar to desktop	Full parity with web, college map, class access
Customization	Highly customizable	Limited	Moderate via third-party apps	Fully customizable (UI, dashboards, alerts)
Hardware Integration	Not supported	Not supported	Not supported	Fully integrated (attendance scanners, cameras)
Advantages	Rich tools, analytics, institutional fit	Free, easy to use, best for Google users	Communication-focused, O365-native	Unified hardware + software, progress analytics, dashboards
Disadvantages	Expensive, complex, steep learning	Limited grading, less robust overall	Overwhelming UI, Office-dependent	Requires initial training and setup effort

6 System Architecture

- **Frontend:** modern technologies such as React.js and Tailwind CSS for the web app and Flutter for the mobile app.
- **Backend:** PHP Laravel for base server-side technologies; Node.js services for multiple service providers.
- **Databases:** MySQL for structured data; MongoDB for flexible storage.
- **Hardware Layer:** Microservices architecture with IoT integrations.
- **Cloud Support:** For notifications, backups, and analytics.

7 User Flow

The NextGen-Edu system adopts a role-based architecture. Below is a breakdown of user flows based on system roles.

7.1 Authentication Flow

- **Login:** Users authenticate via email and password. The system validates credentials and returns a Bearer token with role-specific profile data.
- **Logout:** Access token is invalidated and the user is securely logged out.

7.2 Student Flow (Text Description)

Once a student logs in, the system verifies their role and grants access to the student dashboard. From there, they can:

- View and update personal profile.
- Access “My Courses” to view enrolled courses and detailed information.
- Take quizzes:
  - Start available quizzes.
  - Submit answers.
  - View results.
- View and submit assignments.
- Download course materials.
- Read announcements.
- Check schedule and class timetable.
- View campus information (buildings, halls).
- Join group chats and Q&A hubs.

- Access the integrated ChatBot for quick academic assistance.
- Logout securely.

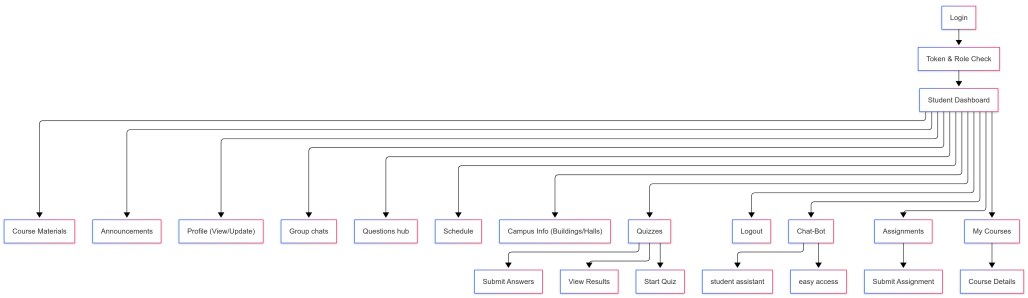


Figure 1: Student Role Flow

7.3 Teacher Flow (Text Description)

Once logged in, teachers land on their dashboard with access to:

- View/update profile information.
- View assigned courses and their enrolled students.
- Create/edit quizzes and view student submissions.
- Create/edit assignments and grade submissions.
- Upload and manage course materials.
- Create and manage course-specific announcements.
- View or postpone their class schedule.
- Logout securely.

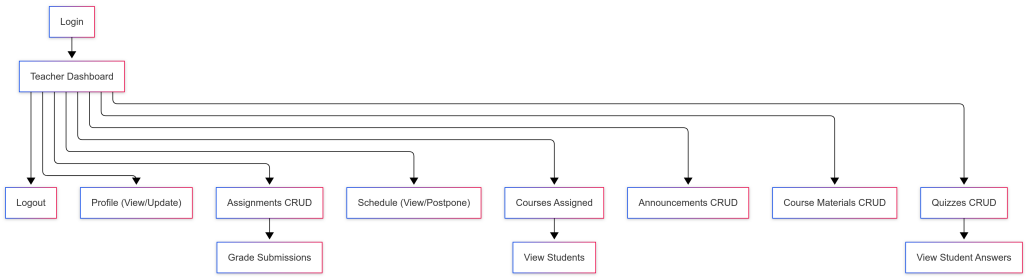


Figure 2: Teacher Role Flow

7.4 Admin & Super Admin Flow (Text Description)

Admins and Super Admins log in via the dashboard and get access to full administrative tools:

- **Student Management:** View, add, edit, delete students; import/export student data.
- **Teacher Management:** View, add, edit, delete teachers; import teacher data.
- **Course Management:** Create and manage courses with full detail.
- **Department and Building Management:** Manage departments, buildings, halls; import department data.
- **Schedule Management:** Create, edit, or delete timetables manually or dynamically.
- **System Announcements:** Post and manage system-wide updates.
- **Admin Management (Super Admin only):** Create, update, delete other admin accounts.
- **Semester Management (Super Admin only):** Create, edit, archive academic semesters.
- **System Settings (Super Admin only):** Access to platform-level configuration.
- **Logout securely.**

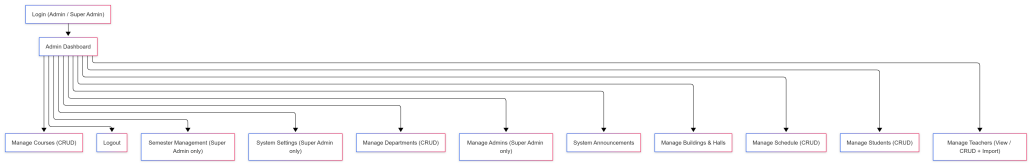


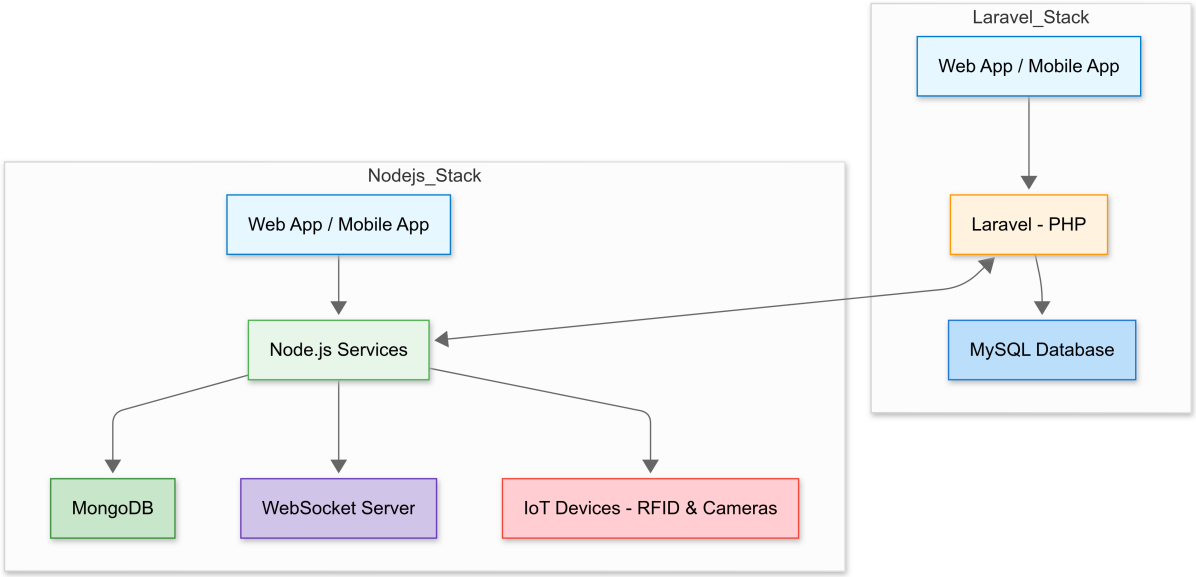
Figure 3: Admin & Super Admin Role Flow

7.5 Access Channels

- **Mobile App:** Used by Students and Teachers.
- **Web Dashboard:** Used by all users.
- **Public Chat:** Web-based AI ChatBot for university queries.

8 Technologies and Implementation

To support a scalable, modular, and responsive infrastructure, **Next Gen Edu** employs a hybrid architecture based on microservices. The system is divided across two main technology stacks to handle structured business data and real-time event-driven operations.



figureSystem architecture showing the separation of concerns between Laravel-PHP and Node.js stacks.

8.1 Database Architecture

The backend database design of **Next Gen Edu** is intentionally structured using a layered, modular approach. This design supports the diverse needs of the system such as structured academic content, role-based user interaction, dynamic scheduling, real-time notifications, and hardware-integrated events. The architecture relies on two complementary database technologies: **MySQL** and **MongoDB**, each responsible for handling different data types and workloads.

8.1.1 Layered Relational Model (MySQL)

MySQL is at the core of our structured data handling. It powers the Laravel-based backend and stores all data with high consistency requirements. The schema is designed based on a three-layer logic that mirrors the business model of academic institutions:

8.1.1.1 Layer 1 – Core Entities This layer represents the static and fundamental components of the university. It includes:

- **Users:** The main entry point representing students, instructors, and admins.
- **Departments, Courses, Semesters:** Represent the academic structure.
- **Buildings and Classrooms:** Represent the physical infrastructure of the campus.

8.1.1.2 Layer 2 – CourseDetails (Contextual Relationship) To address LMS limitations in modeling course context, we introduced `CourseDetail` which binds academic entities into context-aware representations. For example:

> ”CS101” in the Computer Science Department for second-year students taught by Dr. Smith is a unique `CourseDetail`.

All services (e.g., exams, materials, sessions) link to `CourseDetails`, enabling fine-grained customization.

8.1.1.3 Layer 3 – Service Layer This layer reflects dynamic features tightly linked to `CourseDetails`:

- Lecture management
- Exam and assignment distribution
- Attendance tracking
- Real-time scheduling

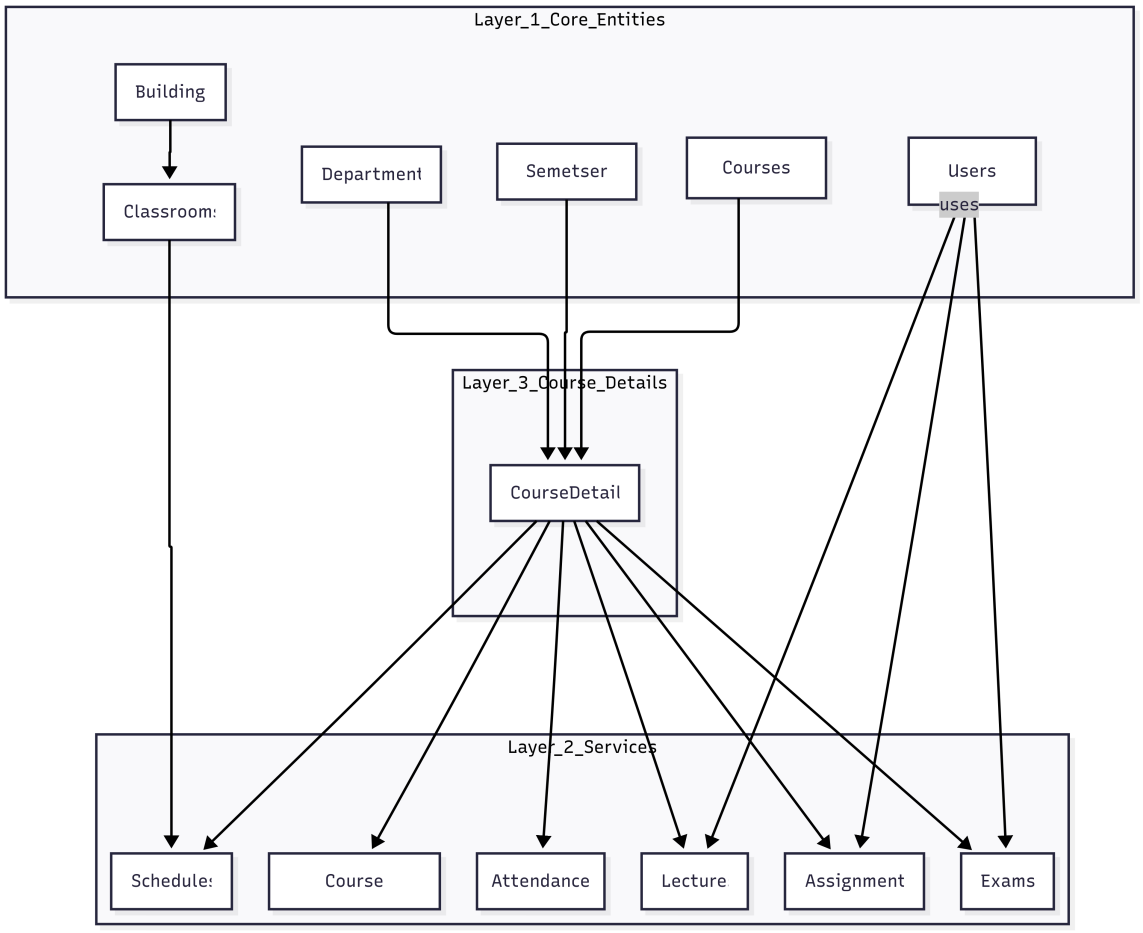


Figure 4: Layered database design architecture showing entity hierarchy and data flow

Benefits of the relational design:

- **Modularity**: Independent, scalable layers.

- **Data Integrity**: Normalized, consistent data operations.
- **Role-awareness**: Context-sensitive services.
- **Maintainability**: Easy to extend for new entities.

8.1.2 MongoDB (NoSQL)

MongoDB handles unstructured and real-time data, integrated with Node.js services. Responsibilities include:

- **Chats & Community**: Course discussions and Q&A hubs.
- **Notifications**: Real-time system alerts.
- **Hardware Logs**: RFID scans, face recognition, sensors.
- **Async Logs**: Background services, AI assistant logs.

This polyglot architecture ensures integrity for academic data and responsiveness for real-time services.

8.2 Business Logic and Microservices Communication

The application logic is distributed across two main stacks:

- **Laravel**: Structured backend logic.
- **Node.js**: Event-driven services.

8.2.1 Laravel (PHP)

NextGen-Edu uses Laravel to manage core backend workflows, including authentication, validation, academic content, and scheduling.

8.2.1.1 Overview

- Laravel Framework (v10+)
- MySQL for structured academic data
- REST APIs (e.g., `/api/v1/`)
- Modular MVC (Controllers, Services, Models, Facades)
- Centralized file handling and standardized API responses
- Seamless bridge with Node.js via events and queues

8.2.1.2 Authentication and Authorization Access control in **NextGen-Edu** is powered by Laravel Sanctum and extended through a robust Role-Based Access Control (RBAC) mechanism. Each authenticated user operates within a scoped academic context, defined by department, semester, and group mappings.

Laravel Sanctum Authentication

- Token- and cookie-based login for web and mobile clients
- CSRF-protected sessions for frontend SPAs
- Personal access tokens scoped to devices and expiration rules

Defined Roles and Responsibilities

- `student` – Interact with enrolled courses, assessments, and dashboards.
- `teacher` – Manage instructional content and evaluate student work.
- `admin` – Department-level management of users and structure.
- `super admin` – Full platform configuration and system oversight.

Context-Aware Access Policies

- Authorization is scoped through the `CourseDetail` academic wrapper.
- Instructors access only their assigned courses, students, and materials.

- Admins are bounded to their department; Super Admins are global.

Middleware and Guards

- `auth:sanctum` – Validates authenticated session.
- `role:{role}` – Filters endpoints based on role.
- `can:{action, model}` – Laravel's authorization policies.
- `academic.context` – Injects runtime academic scope into requests.

Token Lifecycle

- On login: system generates a unique access token tied to the user/device.
- Token is sent in Authorization header (`Bearer <token>`) or secured cookies.
- Token revocation occurs on logout or expiration.

8.2.1.3 Laravel Policies Authorization logic is encapsulated in Laravel's policy classes, ensuring that sensitive actions like material upload or quiz publishing are protected by context-sensitive permission checks. Policies map directly to models, and are auto-resolved using:

```
$this->authorize('update', $courseMaterial);
```

8.2.1.4 Validation Layer and Error Handling Laravel's validation engine is extended with custom form request classes and centralized strategies to ensure consistent input validation, maintainable rules, and unified error responses.

Form Request Classes

- Encapsulate validation logic in dedicated `FormRequest` subclasses.
- Define authorization rules via `authorize()`.
- Provide validation via `rules()`.

```
class CreateQuizRequest extends FormRequest {
    public function authorize() {
        return auth()->user()->can('create', Quiz::class);
    }
    public function rules() {
        return [
            'title' => 'required|string|max:255',
            'course_detail_id' => 'required|exists:course_details,id',
            'questions' => 'required|array|min:1',
        ];
    }
}
```

Centralized Validation Strategy

- Shared rule sets via base `BaseRequest` class.

- Unified error response format using `ApiResponse` helper.
- Reusable rule helpers for common checks.

Error Management and Exception Handling

- All validation, domain, and unexpected errors are handled in Laravel's global handler.
- Custom exceptions like `CourseAccessDeniedException` or `SessionConflictException` are thrown on domain rule violation.
- Logged via stack channels (file, Slack, Sentry).

Standardized API Responses

```
return ApiResponse::success('Operation succeeded', ['id' => $quiz->id]);
return ApiResponse::error('Validation failed', 422, $validator->errors());
```

Benefits

- Predictable response contracts for frontend clients.
- Decoupled rule logic from controllers.
- Support for multilingual validation errors.
- Easily unit-tested and extended.

8.2.1.5 Request Lifecycle (Visual Flow)

`HTTP Request` → `Middleware` → `FormRequest Validation` → `Controller` → `Service` → `Response`

8.2.1.6 Core Systems Architecture The core systems architecture of **NextGen-Edu** is built upon a modular, domain-driven Laravel application. It orchestrates key educational workflows across departments, courses, teachers, and students. Each module encapsulates its business logic and communicates through a shared academic context powered by the `CourseDetail` model.

The system follows Domain-Driven Design (DDD) principles, with every module acting as a bounded context—such as courses, sessions, materials, and users—while remaining interconnected via standardized APIs and policies.

Central Academic Entities

- **Courses** – Base academic subjects (e.g., Data Structures, AI).
- **Departments** – Faculty units (e.g., Computer Engineering).
- **Semesters** – Academic periods (e.g., Fall 2024, الفرقة الثانية).
- **Teachers** – Assigned instructors for each contextual course offering.

Each `CourseDetail` instance links the above into a unique offering with its own sessions, materials, assignments, and announcements.

Modular Laravel Services

- **Controllers:** API logic and request routing.

- **Services:** Core business logic (e.g., `TeacherService`, `TableService`).
- **Models:** Eloquent ORM classes for database mapping.
- **Resources:** Response formatting for APIs.
- **Requests:** Input validation logic using `FormRequest`.

This modular approach allows for easier testing, versioning, and independent scaling of system features.

8.2.1.7 Course and Content Services Logic The educational services in **NextGen-Edu** are structured as independent modules that operate under a shared academic context and provide well-defined workflows for managing teaching resources, assessments, and in-course communication.

Each module encapsulates a complete lifecycle, from creation and publishing to student interaction and feedback collection. Services are exposed via RESTful APIs and operate within the boundaries of the `CourseDetail` context to ensure alignment with department, semester, and assigned staff.

Quiz Management Logic

- Instructors create and schedule quizzes for a specific course instance.
- Quizzes include metadata such as title, duration, total degree, and publishing date.
- Students attempt quizzes within the scheduled window.
- Submissions are auto-graded or reviewed manually by the instructor.
- Tokens are issued to prevent multiple attempts.
- Grading and result visibility are controlled based on instructor configuration.

Assignment Lifecycle Logic

- Instructors create assignments linked to `CourseDetail`.
- Submissions are allowed until a specified deadline.
- Each file is validated and stored securely with timestamps.
- Instructors review submissions and optionally reopen assignments.
- Late submissions are flagged in reports.

Announcement Broadcasting Logic

- Instructors or admins compose announcements for courses.
- Optional media files (e.g., banners) can be attached.
- Notifications are pushed to student dashboards and mobile apps.
- Expiry date or visibility limits can be applied.
- Markdown formatting and analytics are supported.

Course Material Distribution Logic

- Instructors upload categorized resources (PDFs, videos, links).
- Materials are tagged by week or chapter and can be scheduled.
- Students interact with materials via dashboard and mobile.
- Engagement metrics such as views and completion are tracked.
- Versioning is supported for updated content.

Integration Notes

- All services operate within the academic context defined by `CourseDetail`.
- Internal events trigger notifications and analytics.
- Each service adheres to standardized API response contracts.

8.2.1.8 File Management & Storage

Scope and Responsibilities The file storage system in **NextGen-Edu** plays a critical infrastructural role in managing all static academic content. The subsystem is responsible for organizing, securing, and maintaining digital files across instructional and assessment domains. It ensures abstraction from user-specific flows and focuses on consistent backend logic for managing file lifecycle and context-bound associations.

Architectural Foundation The file management layer is encapsulated behind a centralized service facade, `FileHandler`, which is responsible for mediating access to Laravel's disk storage API. Files are structured under a predictable directory hierarchy to ensure clarity, traceability, and modular separation.

File Storage Structure:

```
storage/app/public/
  assignments/
      [assignment specification files]
  assignments/answers/
      [submission responses from students]
  course_materials/
      [instructional documents per course]
```

Each subdirectory is allocated based on domain module ownership and lifecycle boundaries.

File-Entity Binding Files are not stored independently; instead, they are mapped to domain models via dedicated metadata fields in the relational schema. These include:

- `assignments.file`: Stores teacher-uploaded task specifications
- `assignment_answers.file`: Holds student-uploaded solutions
- `course_materials.material`: Links to distributed learning resources

These bindings are transactional and updated synchronously with the associated model operation (create/update).

Module-Oriented Use Cases Each functional service integrates with the file storage layer without embedding file logic internally. Business logic modules such as `AssignmentService`, `MaterialService`, and `AnswerSubmissionService` delegate to `FileHandler` for physical file management.

Examples of decoupled responsibilities:

- Task creation pipelines initialize an assignment and commit the attached file using `storeFile()` under the relevant directory.
- Material provisioning services batch-import instructional PDFs or slides, organized by academic week and type.
- Evaluation subsystems attach student submissions via the answers subfolder and track timestamps in parallel.

Validation and Policy Enforcement All file transactions are validated at the application boundary via request validators. This validation includes:

- MIME type enforcement
- Maximum file size restriction (configured per module)
- Conditional access enforcement based on academic schedule or user privilege

The following validators are deployed:

- `AssignmentStoreRequest`, `AnswerStoreRequest`, `MaterialStoreRequest`

Role-based guards prevent unauthorized access to file upload endpoints. Only context-approved roles with scoped permissions are permitted to initiate file interactions.

FileHandler API Specification `FileHandler` abstracts the underlying file system operations. Its contract includes:

Method	Functionality
`storeFile()`	Saves the incoming file under the specified module path with a generated unique filename
`updateFile()`	Replaces an existing path reference with a new file, and deletes the obsolete version
`deleteFile()`	Removes the physical file from disk and clears associated metadata fields

Operational Guarantees

- **Atomicity:** File operations are transactionally bound to model lifecycle events.
- **Traceability:** All filenames and paths are deterministically generated.
- **Context Isolation:** File structure mirrors domain segmentation, allowing per-module security policies.
- **Extensibility:** Easily adaptable to cloud-based storage (e.g., S3) using Laravel's Storage API.

8.2.1.9 Session and Scheduling Engine – Business Logic Specification The scheduling subsystem in **NextGen-Edu** is a domain-driven service responsible for generating, structuring, and maintaining academic timetables across departments, semesters, and course groups. It abstracts sessions as first-class entities and supports time-based orchestration, spatial allocations, and session-level overrides for rescheduling scenarios.

1. Session Entity `Session` defines atomic class events—lectures, labs, or tutorials—bounded in time, location, and academic scope.

Core Attributes:

- `type`: Instructional mode (e.g., lecture, lab, section)
- `date`, `day`, `from`, `to`: Temporal boundaries
- `hall_id`: Physical space where the session occurs
- `course_id`, `department_id`, `semester_id`: Academic linkage
- `attendance`: Physical or online mode flag
- `week`: Week reference in academic calendar

Responsibilities:

- Canonical reference for all session operations
- Time-slot resolution and conflict detection
- Linkage for attendance, postponement, and calendar rendering

2. Postponement Handling `PostponedSession` enables dynamic calendar adjustments while preserving history and audit.

- One-to-one mapping with `Session`
- Stores alternate date/time/location
- Read-only for historical reference
- Used in transformation and visual rendering logic

3. Scheduling Engine Behavior

- Validates uniqueness of `(hall, time)` combinations
- Indexes sessions by department $\rightarrow$ semester $\rightarrow$ group
- Detects spatial conflicts
- Supports batch creation from admin input
- Distributes week indices across semester span

4. **Resource Binding** Each `Session` is linked to key academic and spatial resources:

- `Hall`: Includes geo-tag, capacity, and building
- `Course, Semester, Department`
- `CourseDetail`: Contextual wrapper for instructors and student groups

5. **Data Integrity and Transformation**

- Transformed via `TableResource` abstraction
- Merges base and postponed state where applicable
- Flags: `is_postponed`, `new_date`, `original_time`
- Contracts remain stable despite schema evolution

6. **Exception Handling and Validation Layer**

- Detects session overlaps and hall collisions
- Validates timing and resource access before saving
- Throws domain-specific exceptions on violation
- Triggers downstream events on change (notifications/logs)

Component	Responsibility
`Session`	Canonical container for academic events
`PostponedSession`	Isolated override for rescheduling
Scheduling Engine	Validates and orchestrates calendar logic
`TableResource`	API transformer for UI rendering
Validation Layer	Ensures data integrity, triggers alerts

8.2.1.10 Testing Strategy (Optional)

- Feature tests using Laravel's `test()` methods.
- Policy and permission unit tests.
- FormRequest validation testing.
- Role-specific scenario tests (student, teacher, admin).

8.2.1.11 Conclusion The Laravel-based backend in **NextGen-Edu** forms the foundation of the platform's structured data processing and academic business logic. Through modular service design, context-aware policies, robust authentication, and scalable file and session management, the system ensures maintainability, security, and flexibility across roles and departments.

This architecture allows seamless integration with frontend clients and real-time services, enabling NextGen-Edu to deliver a cohesive digital learning experience that aligns with institutional requirements and user expectations.

8.2.2 Node.js (Event-Driven Services)

- Real-time chat, notifications via WebSockets
- Integration with RFID/camera hardware
- Background task processing

8.2.3 Communication Between Services

- REST APIs for decoupled operations
- WebSockets for real-time sync
- Token-based security

8.2.4 Benefits of the Architecture

- **Scalability**: Independent service scaling
- **Maintainability**: Modular and testable code
- **Performance**: Best-fit tools for each use case