📘 LMS Database Schema Documentation (Finalized for MVP)

👨‍🎓 1. `users`

Stores all system users (students, teachers, admins).

Column	Type	Description
`id`	UUID (PK)	Unique user ID
`full_name`	VARCHAR(120)	Full name of the user
`email`	VARCHAR(120) UNIQUE	Login email
`password_hash`	TEXT	Hashed password
`role`	ENUM('student', 'teacher', 'admin')	User role
`status`	ENUM('active', 'inactive', 'suspended') DEFAULT 'active'	Account status
`created_at`	TIMESTAMP DEFAULT NOW()	Account creation date
`updated_at`	TIMESTAMP DEFAULT NOW()	Last update time

🏫 2. `courses`

Each record represents a **course** (e.g., "Physics 11", "Mathematics 10").

Column	Type	Description
`id`	UUID (PK)	Course ID
`title`	VARCHAR(150)	Course title
`description`	TEXT	Overview or syllabus
`thumbnail_url`	TEXT	Optional cover image
`created_by`	UUID (FK → users.id)	Creator (teacher/admin)
`status`	ENUM('draft', 'published', 'archived')	Publication state
`created_at`	TIMESTAMP DEFAULT NOW()	Created date
`updated_at`	TIMESTAMP DEFAULT NOW()	Updated date

📂 3. `modules`

Represents **hierarchical folders / topics / chapters** inside a course.
Each module can contain **submodules**, enabling multi-level structure.

Column	Type	Description
`id`	UUID (PK)	Module ID
`course_id`	UUID (FK → courses.id)	Belongs to course
`parent_id`	UUID (FK → modules.id, nullable)	Points to parent module (if subfolder)
`title`	VARCHAR(150)	Title of the module (e.g., "Chapter 1: Motion")

Column	Type	Description
`description`	TEXT	Optional description
`module_type`	ENUM('unit', 'chapter', 'topic')	Classification (optional, for UI)
`level`	INT DEFAULT 1	Nesting level (1 = top, 2 = child, etc.)
`order_index`	INT	Position order
`created_at`	TIMESTAMP DEFAULT NOW()	Created time

💡 Notes

- `parent_id = NULL` → top-level folder (Unit).
- Recursively nested structure possible for unlimited depth.
- Example hierarchy:

id	course_id	parent_id	title
1	PHYS11	NULL	Unit 1: Mechanics
2	PHYS11	1	Chapter 1: Laws of Motion
3	PHYS11	2	Topic 1: Newton's Laws

🎞️ 4. `contents`

Stores actual **learning content** (video, PDF, SCORM, quiz, etc.) under modules.

Column	Type	Description
`id`	UUID (PK)	Content ID
`module_id`	UUID (FK → modules.id)	Linked module/chapter
`type`	ENUM('pdf', 'video', 'scorm', 'link', 'document')	Type of content
`title`	VARCHAR(150)	Display title
`file_url`	TEXT	Cloud URL (PDF/video/etc.)
`launch_path`	TEXT	SCORM launch path (if applicable)
`duration`	VARCHAR(20)	Optional (for videos)
`order_index`	INT	Display order
`created_by`	UUID (FK → users.id)	Uploaded by
`created_at`	TIMESTAMP DEFAULT NOW()	Uploaded date

💡 Notes

- Each content belongs to one `module`, which itself may be nested under another.
- You can easily trace a content's **full path** through module hierarchy.
  Example:
  Physics 11 → Unit 1 → Chapter 2 → Topic 1 → Video: Motion Lecture.mp4

📦 5. `enrollments`

Tracks which student is enrolled in which course.

Column	Type	Description
`id`	UUID (PK)	Enrollment ID
`user_id`	UUID (FK → users.id)	Student
`course_id`	UUID (FK → courses.id)	Enrolled course
`status`	ENUM('active', 'completed', 'dropped')	Enrollment state
`progress`	DECIMAL(5,2) DEFAULT 0	Completion percentage
`enrolled_at`	TIMESTAMP DEFAULT NOW()	Enrollment date

📊 6. `progress_tracking`

Tracks a student's progress per content item.

Column	Type	Description
`id`	UUID (PK)	Record ID
`user_id`	UUID (FK → users.id)	Student
`content_id`	UUID (FK → contents.id)	Content being tracked
`status`	ENUM('not_started', 'in_progress', 'completed')	Progress state
`score`	DECIMAL(5,2)	Optional SCORM score
`last_accessed`	TIMESTAMP	Last viewed time

📜 7. `scorm_attempts` *(optional, for future use)*

Handles multiple SCORM attempt logs.

Column	Type	Description
`id`	UUID (PK)	Attempt ID
`user_id`	UUID (FK → users.id)	Student
`content_id`	UUID (FK → contents.id)	SCORM content
`attempt_no`	INT	Attempt number
`score`	DECIMAL(5,2)	Score achieved
`completion_status`	ENUM('passed', 'failed', 'incomplete', 'completed')	SCORM completion
`start_time`	TIMESTAMP	When attempt started
`end_time`	TIMESTAMP	When attempt ended

🛡️ 8. `roles_permissions` *(optional, for scalable roles later)*

You might add this later to customize teacher/admin access.

Column	Type	Description
`id`	UUID (PK)	Role ID
`role_name`	VARCHAR(50)	Role (admin, teacher, etc.)
`permissions`	JSONB	Permission set (future use)

🪜 Course Folder Hierarchy Example

Course: Physics 11
Structure:

```
Unit 1: Mechanics (module_id = 1)
  ├── Chapter 1: Laws of Motion (module_id = 2)
  │      ├── Topic 1: Newton's First Law (module_id = 3)
  │      │      ├── Video: Introduction.mp4 (content_id = 10)
  │      │      └── PDF: Notes.pdf (content_id = 11)
  │      └── SCORM Quiz: Laws of Motion (content_id = 12)
  └── Chapter 2: Work and Energy (module_id = 4)
         └── Video: Energy Concepts.mp4 (content_id = 13)
```

This allows **nested folders, unlimited levels**, and **any content type** at any depth.

🧮 Example Recursive SQL to Fetch Full Course Tree

```
WITH RECURSIVE module_tree AS (
  SELECT id, title, parent_id, course_id, 1 AS depth
  FROM modules
  WHERE course_id = 'COURSE_ID' AND parent_id IS NULL

  UNION ALL

  SELECT m.id, m.title, m.parent_id, m.course_id, t.depth + 1
  FROM modules m
  INNER JOIN module_tree t ON m.parent_id = t.id
)
SELECT * FROM module_tree ORDER BY depth, parent_id, title;
```

✅ Summary

Table	Purpose
`users`	Stores all platform users
`courses`	Main course metadata
`modules`	Hierarchical chapters/topics (multi-level folders)
`contents`	Videos, PDFs, SCORM files, etc.
`enrollments`	Tracks which user joined which course
`progress_tracking`	Tracks each user's completion
`scorm_attempts`	(Optional) Detailed SCORM tracking
`roles_permissions`	(Future) Fine-grained admin roles