AI-Powered Student LMS with Agent Tutor - Documentation

1. Introduction

This project is a **FastAPI-based Learning Management System (LMS)** integrated with an **AI Agent Tutor**. It provides authentication, role-based access control (RBAC), activity management, and AI-assisted learning.

2. Folder Structure

```
student_lms_ai/
│── app/
│   │── __init__.py
│   │── config.py             # Configuration file (DB, JWT, OAuth settings)
│   │── database.py                   # Database connection
│   │── models/               # ORM Models
│   │   │── __init__.py
│   │   │── user.py           # User Model (Students, Teachers, Admins)
│   │   │── role.py           # Role-based Access Control (RBAC)
│   │   │── activity.py       # Activity-related models
│   │── routes/               # API Routes (FastAPI Endpoints)
│   │   │── __init__.py
│   │   │── auth.py           # Login, Signup, JWT, OAuth
│   │   │── users.py          # CRUD APIs for Users
│   │   │── activities.py     # APIs to manage activities
│   │   │── agent_tutor.py            # AI Agent interaction
│   │── services/             # Business logic layer
│   │   │── __init__.py
│   │   │── auth_service.py           # Authentication & Authorization
│   │   │── user_service.py           # User management logic
│   │   │── activity_service.py       # Activity handling
│   │   │── agent_service.py          # AI Agent logic
│   │── middlewares/                  # Middleware for authentication
│   │   │── auth_middleware.py        # JWT/OAuth Middleware
│   │── utils/                # Utility functions
│   │   │── security.py       # Hashing, JWT Encoding/Decoding
│   │   │── file_upload.py              # Handle file uploads
│   │── main.py             # Entry point for FastAPI
│
│── assets/                 # Store AI-generated resources, prompts, PDFs
│── migrations/             # Alembic migrations for DB
│── tests/                  # Unit tests for APIs
│── requirements.txt                # Python dependencies
│── Dockerfile              # Dockerization
│── .env                    # Environment variables (DB, JWT Secret)
│── README.md                       # Documentation
```

3. API Endpoints & Functionalities

1) Authentication & User Management

User Authentication

- **POST /auth/signup** → Register a new user (Teacher, Student, Admin)
- **POST /auth/login** → User authentication with JWT token
- **GET /auth/me** → Get logged-in user details
- **POST /auth/logout** → Logout and invalidate token

Role & Permission Management

- **GET /roles** → Get all roles
- **POST /roles** → Create a new role
- **POST /permissions** → Assign permission to role

2) Activity Management

- **POST /activities** → Create a new activity
- **GET /activities** → List all activities
- **GET /activities/{activity_id}** → Get specific activity details
- **PUT /activities/{activity_id}** → Update activity
- **DELETE /activities/{activity_id}** → Delete activity

3) AI Agent Tutor (Automated Learning Support)

- **POST /agent/ask** → Student asks a question, AI prepares responses/resources
- **GET /agent/history/{student_id}** → Fetch AI-generated responses for a student
- **POST /agent/feedback** → Allow teacher to validate AI-generated content

4. Database Schema & Tables

Here are the updated database tables with additional fields and separate role-permission management:

1) Users Table

Field	Type	Description
id	UUID (PK)	Unique identifier
name	String	Full name
email	String (Unique)	Email ID
password	String	Hashed password
phone	String	Contact number
city	String	User city
country	String	User country
prime_member	Boolean	Premium membership status
is_active	Boolean	Account active status

is_available	Boolean	Availability status
created_at	Timestamp	Account creation date
updated_at	Timestamp	Last updated timestamp

2) Roles Table

Field	Type	Description
id	UUID (PK)	Unique Role ID
name	String	Role Name (Admin, Teacher, Student)
description	String	Role Description
is_active	Boolean	Role Active Status
created_at	Timestamp	Role Creation Date
updated_at	Timestamp	Last Updated Timestamp

3) Permissions Table

Field	Type	Description
id	UUID (PK)	Unique Permission ID

name	String	Permission Name (READ, WRITE, DELETE, etc.)
description	String	Permission Description
created_at	Timestamp	Permission Creation Date
updated_at	Timestamp	Last Updated Timestamp

4) Role-Permission Table

Field	Type	Description
id	UUID (PK)	Unique Role-Permission ID
role_id	Foreign Key (Roles)	Associated Role
permission_id	Foreign Key (Permissions)	Assigned Permission
is_active	Boolean	Active Status
created_at	Timestamp	Creation Date
updated_at	Timestamp	Last Updated Timestamp

5) User-Roles Table

Field	Type	Description

id	UUID (PK)	Unique User-Role ID
user_id	Foreign Key (Users)	Associated User
role_id	Foreign Key (Roles)	Assigned Role
is_active	Boolean	Active Status
created_at	Timestamp	Creation Date
updated_at	Timestamp	Last Updated Timestamp

6) AI Response Table

Field	Type	Description
id	UUID (PK)	Unique Response ID
student_id	Foreign Key (Users)	Associated Student
question	Text	Student’s question
response	JSON	AI-generated response (Text/PDF)
status	String	Pending/Approved
created_at	Timestamp	Response generated time

🗂️ Activity Module - Database Schema

1) Activity Table

Stores all activities created by **teachers or students**.

Field	Type	Description
id	UUID (PK)	Unique Activity ID
name	String	Activity Name
description	Text	Detailed activity instructions
category_id	Foreign Key (activity_category)	Associated Category
sub_category_id	Foreign Key (activity_sub_category)	Associated Sub-Category
difficulty_level	Enum ('Beginner', 'Intermediate', 'Advanced')	Complexity of the activity
access_type	Enum ('private', 'global')	Private (User-Specific) or Global
created_by	Foreign Key (users)	**User (Teacher/Student) who created it**
ai_guide	Boolean	AI Assistance available
is_active	Boolean	Activity active status

created_at	Timestamp	Creation timestamp
updated_at	Timestamp	Last updated timestamp

2) activity_category Table

Defines the categories for organizing activities.

Field	Type	Description
id	UUID (PK)	Unique Category ID
name	String	Category Name
created_at	Timestamp	Creation timestamp

3) Activity_sub_category Table

Defines subcategories linked to a category.

Field	Type	Description
id	UUID (PK)	Unique Sub-Category ID
category_id	Foreign Key (activity_category)	Parent Category
name	String	Sub-Category Name

created_at	Timestamp	Creation timestamp

4) Activity_questions Table

Holds questions related to an activity.

Field	Type	Description
id	UUID (PK)	Unique Question ID
activity_id	Foreign Key (activity)	Associated Activity
question	Text	Question text
solution_steps	Text	Step-by-step solution
answer	String	Correct answer
ai_hint	Text	AI-generated hint (if AI Guide is enabled)
created_at	Timestamp	Creation timestamp

5) Activity_sessions Table

Tracks users attempting an activity.

Field	Type	Description

id	UUID (PK)	Unique Session ID
activity_id	Foreign Key (activity)	Associated Activity
user_id	Foreign Key (users)	**User (Teacher/Student)**
status	Enum ('In Progress', 'Completed', 'Graded')	Activity progress state
submitted_at	Timestamp	Timestamp when session was submitted
graded_by	Foreign Key (users, Nullable)	**Teacher who graded the session**
grade	Float	Score/grade assigned (if applicable)

6) Activity_documents Table

Handles activity-related **files** (CSV, PDF, Text, etc.).

Field	Type	Description
id	UUID (PK)	Unique Document ID
activity_id	Foreign Key (activity)	Associated Activity
file_name	String	File name
file_type	Enum ('csv', 'pdf', 'text', 'textbox')	Type of document

description	String	Document Description
metadata	JSON	Additional metadata (size, format, etc.)
created_at	Timestamp	Creation timestamp

7) Activity_ai_responses Table

Stores **AI-generated** responses for users' queries.

Field	Type	Description
id	UUID (PK)	Unique AI Response ID
session_id	Foreign Key (activity_sessions)	Related Session
user_id	Foreign Key (users)	**User receiving AI guidance**
question_id	Foreign Key (activity_questions)	Question reference
ai_response	Text	AI-generated explanation or hint
created_at	Timestamp	Timestamp

5. Sample Workflow

Use Case: Student Asks a Question

1. **Student sends a request** to /agent/ask with a question.
2. **AI generates a response** (fetches learning materials, prepares answers).
3. **Response is stored** in ai_response table.
4. **Teacher reviews the response** from /agent/history/{student_id}.
5. **Teacher can validate or modify the response** via /agent/feedback.

You're absolutely right! Let's refine the schema to differentiate between **Activity Types** (like "Assignments", "Quizzes") and **Activity Categories** (like "Maths > Algebra"). Here’s the improved breakdown:

Activity Tables and Their Relevance

1. activity_types Table

Purpose:

Defines the **type** of activity, such as whether it's an **Assignment, Quiz, Project, or Practice Test**. This helps in differentiating activities based on their function.

Columns:

- id (Primary Key): Unique identifier.
- name (Unique): Name of the activity type (e.g., "Assignment", "Quiz", "Project").

Relationships:

- An **activity belongs to one type** (one-to-many relationship with activities).

Relevance:

- Helps in organizing activities **based on function**.
- Enables the LMS to **filter and analyze activities** by type.

2. activity_categories Table

Purpose:

Defines **subject-based categories**, such as **Maths, Science, English**, to group activities **by subject or curriculum**.

Columns:

- id (Primary Key): Unique identifier.
- name (Unique): Name of the category (e.g., "Maths", "Science").

Relationships:

- A **category can have multiple subcategories** (one-to-many with activity_subcategories).
- A **category can have multiple activities** (one-to-many with activities).

Relevance:

- Organizes activities **by subject matter**.
- Helps students and teachers **browse relevant activities** easily.

3. activity_subcategories Table

Purpose:

Further divides a **category into specific subtopics**. For example, within **Maths**, there could be subcategories like **Algebra, Geometry, Trigonometry**.

Columns:

- id (Primary Key): Unique identifier.
- name (Unique): Subcategory name (e.g., "Algebra", "Geometry").
- category_id (Foreign Key → activity_categories.id): Links the subcategory to its category.

Relationships:

- A **subcategory belongs to one category**.
- A **subcategory can have multiple activities** (one-to-many with activities).

Relevance:

- Provides a **detailed breakdown of subjects**.
- Helps in structuring **curriculum-based activities**.

4. activities Table

Purpose:

Stores details about individual activities. Each activity belongs to an **Activity Type**, a **Category**, and a **Subcategory**.

Columns:

- id (Primary Key): Unique identifier.
- name: The name of the activity (e.g., "Solve Linear Equations").
- description: Brief details about the activity.
- type_id (Foreign Key → activity_types.id): Links an activity to an **Activity Type**.
- category_id (Foreign Key → activity_categories.id): Links an activity to a **Category**.
- subcategory_id (Foreign Key → activity_subcategories.id): Links an activity to a **Subcategory**.
- created_by (Foreign Key → users.id): The user (teacher/admin) who created the activity.

Relationships:

- An activity **has one type** (like Assignment, Quiz).
- An activity **belongs to one category** (like Maths).
- An activity **belongs to one subcategory** (like Algebra).
- An activity **is created by a user**.

Relevance:

- Allows activities to be grouped **both by type (Quiz, Assignment) and by subject (Maths > Algebra)**.
- Provides a **clear structure** for managing and retrieving activities.

Why This Structure?

✅ **Flexible:** Supports **multiple subjects and types** of activities.

✅ **Scalable:** New types, categories, and subcategories **can be added anytime**.

✅ **Efficient Searching & Filtering:** Activities can be filtered by **type, subject, and topic**.