# **Table of Contents**

1. **Introduction**
2. **Project Structure Overview**
3. **Authentication API**
4. **CRUD API**
5. **Roles and Permissions**
6. **Database Models**
7. **How to Set Up and Run the Backend**
8. **Detailed API Endpoints**
9. **Example Data and Requests**

# **1. Introduction**

This document provides an in-depth explanation of the backend system, including the project structure, authentication system, CRUD operations, and database models. It also includes step-by-step instructions to set up and run the backend, as well as details on how to access each API endpoint.

# **2. Project Structure Overview**

application/

|-- database.py

|-- models.py

|-- auth\_apis.py

|-- crud\_apis.py

|-- seed.py

|-- config.py

|-- main.py

* **database.py**: Initializes and manages database connections.
* **models.py**: Defines database models for SQL and MongoDB.
* **auth\_apis.py**: Handles authentication and role-based access control.
* **crud\_apis.py**: Implements CRUD operations for courses and students.
* **seed.py**: Seeds initial data into the databases.
* **config.py**: Stores configuration settings.
* **main.py**: The entry point of the application.

# **3. Authentication API**

## **User Registration**

* **Endpoint:** POST /register
* **Request Body:**

{

"username": "john\_doe",

"email": "john@example.com",

"password": "securepassword",

"role": "admin"

}

* **Response:**

{

"msg": "User created successfully"

}

## **User Login**

* **Endpoint:** POST /login
* **Request Body:**

{

"email": "john@example.com",

"password": "securepassword"

}

* **Response:**

{

"msg": "Login successful",

"access\_token": "<JWT\_TOKEN>"

}

## **Protected Resource (Admin Only)**

* **Endpoint:** GET /protected
* **Authorization:** Bearer token required
* **Response:**

{

"msg": "This is a protected resource",

"user": {

"username": "john\_doe",

"email": "john@example.com"

}

}

# **4. CRUD API**

## **Get All Courses**

* **Endpoint:** GET /courses?page=1
* **Response:**

{

"res": [

{

"id": "123e4567-e89b-12d3-a456-426614174000",

"title": "Python Basics",

"description": "Learn Python from scratch",

"category": "Programming"

}

],

"page": 1,

"has\_more": true

}

## **Get Student By ID**

* **Endpoint:** GET /students/{student\_id}
* **Response:**

{

"id": "123e4567-e89b-12d3-a456-426614174001",

"name": "Alice",

"email": "alice@example.com",

"course\_progress": []

}

## **Update Student**

* **Endpoint:** PUT /students/{student\_id}
* **Request Body:**

{

"name": "Alice Updated",

"email": "alice\_new@example.com"

}

* **Response:**

{

"msg": "Student updated successfully"

}

# **5. Roles and Permissions**

## **Available Roles**

* **Admin**: Full access to all operations.
* **Instructor**: Can create and manage courses.
* **Student**: Can enroll and access courses.

## **Role-Based Access Control**

* **Only Admin** can assign roles.
* **Instructors** can add/edit courses but not assign roles.
* **Students** can view and complete courses.

# **6. Database Models**

## **SQLAlchemy Models (PostgreSQL)**

* **User**: Stores user credentials and roles.
* **Role**: Stores different user roles.
* **Lecture**: Stores lecture details.

## **MongoDB Models**

* **Course**: Stores course details.
* **Student**: Stores student progress.
* **WeeklyContent**: Stores weekly videos and assignments.

# **7. How to Set Up and Run the Backend**

### **Step 1: Create and Activate Virtual Environment**

python -m venv venv

source venv/bin/activate # On macOS/Linux

venv\Scripts\activate # On Windows

### **Step 2: Install Dependencies**

pip install -r requirements.txt

### **Step 3: Run the Application**

python main.py

### **Step 4: Access the API**

* API runs at http://127.0.0.1:5000/

# **8. Detailed API Endpoints**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /register | Register a new user |
| POST | /login | Login and get JWT token |
| GET | /protected | Access protected resource (admin only) |
| GET | /courses | Get all courses |
| GET | /students/{id} | Get student details |
| PUT | /students/{id} | Update student details |

# **9. Example Data and Requests**

## **Sample JWT Authorization**

Authorization: Bearer <JWT\_TOKEN>

## **Sample Request to Get Courses**

GET /courses?page=1

## **Sample Request to Update Student**

PUT /students/{id}

Content-Type: application/json

{

"name": "Updated Name"

}

This document serves as a complete guide to the backend architecture, APIs, and database interactions. Ensure proper authentication before accessing protected endpoints.