project

- ## 1. Functional Requirements
  - The proposed system should have at least five distinct features:
  - **Intuitive Course Resource Management**
    - Provides a structured interface where users can organize course materials.Allows users to take digital notes, highlight key concepts, and link resources to coding exercises for easy reference.
  - **Code-Linked Lecture Execution**
    - Embeds lecture slides directly within the IDE.Enables students to modify and execute code snippets inline.Supports real-time experimentation with lecture examples and provides instant feedback on code correctness.
  - **Collaborative Coding and Resource Sharing**
    - Supports real-time collaborative coding.Includes discussion threads linked to lecture content.Facilitates student-to-student resource sharing and offers built-in team project chat functionality.
  - **Course Progress Tracking**
    - Tracks student progress across course materials.Displays completed tasks, upcoming assignments, and areas needing attention.
  - **AI Learning Assistant**
    - Summarizes lecture content and highlights key points. Generates visual mind maps or flowcharts. Creates self-assessment quizzes with automated grading and explanations.
- ## 2. Non-Functional Requirements
  - **Usability**: IDE integrates coding and learning resources with curriculum awareness. And allow students to manage course materials, take notes, execute code snippets from slides, do exercises, learn from examples, and track progress in a unified environment.
  - **Security**: Implements role-based access control (RBAC) to ensure proper authorization for different users.
  - **Safety**: Data integrity mechanisms should be in place to prevent loss of user progress and learning resources.
  - **Performance**: The system should provide real-time feedback on code execution and maintain low latency for collaborative features.
  - **Scalability**: The system should support an increasing number of users without significant performance degradation.
- ## 3. Data Requirements
  - **Required Data**:

    - Course materials (lecture slides, code examples, assignments, etc.).
    - User-generated content (digital notes, code modifications, discussion posts, shared resources, etc.).
    - Student progress data (completed exercises, quiz scores, collaboration activity, etc.).
  - **Data Acquisition**:
    - Instructors upload lecture materials and assignments.
    - Students contribute notes, discussions, and shared resources.
    - AI assistant processes lecture content and generates summaries, quizzes, and visual aids.
- ## 4. Technical Requirements
  - **Operating Environment**:
    - **Runtime Environment:** Python 3.11
    - **Framework:** Django (Backend), Django Template Engine (Frontend)
    - **Database:** SQLite (depending on deployment)
    - **Dependencies:**
      - Django
      - Django REST Framework (if using API)
      - TensorFlow / PyTorch (for AI/ML tasks)
      - WebSockets (for real-time communication)
    - **Server & Deployment:**
      - Web Server: Nginx / Apache (for serving static files and reverse proxying)
      - Application Server: Gunicorn / uWSGI (for running Django applictaions)
      - Containerization: Docker (for creating consistent environments)
    - **Operating System:** Windows
    - **Version Control:** Git + GitHub/GitLab
    - **Package Management:** pip / conda (for AI-related dependencies)
  - **Tech Stack**:
    - **Frontend:** HTML, CSS, JavaScript, Django Template Engine (DTL)
    - **Backend:** Django (a web framework for building backend logic)
    - **AI/ML:** Python-based AI engine using TensorFlow/PyTorch for NLP and quiz generation, Optical plugin like OCR_inspection for code extraction, Call the Large Language Model (LLM) API(DeepSeek/文心一言/chatGPT/...)
    - **Real-time Features:** WebSockets for collaborative coding and chat