**Documentation and Training**

* Create user guides for the trainers on how to use the system.
* Offer training sessions for trainers to familiarize them with the platform.

Steps for the data mining :

1. Extract Text from PDF: The AI reads the PDF file and extracts text from each page.
2. Identify Headings: The AI identifies headings or section titles based on specific patterns, such as:

* The first line of each page.
* Lines that start with certain keywords (e.g., "Chapter", "Section", etc.).
* Lines that contain specific formatting (e.g., bold, larger font size).

1. Group Content: Once headings are identified, the AI groups the content that follows each heading until it encounters the next heading. This can be done by:

* Keeping track of the current section name.
* Appending lines of text to the current section until a new heading is found.

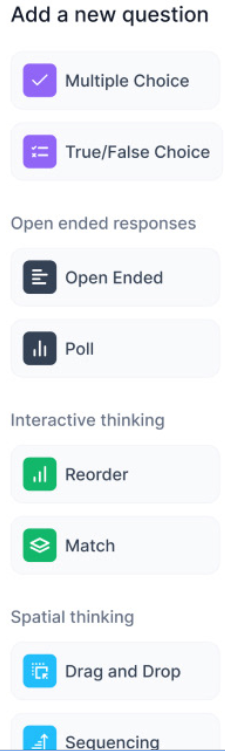
1. Store Sections: The AI stores each section in a structured format, such as a dictionary, where the keys are section names and the values are the corresponding text content.

Code steps:

1. upload\_file => Opens a file dialog for the user to select a PDF file.
2. extract\_sections => Extracts sections from the specified PDF file based on the first two words of each page + position of the img
3. extract\_shapes => Extracts shapes (like rectangles and ellipses) from a given page and saves their details (type, position, dimensions) to a JSON file.
4. save\_sections => Saves each section's content to a separate text file
5. read\_sections\_from\_files => Reads section files from a specified directory and returns their content.
6. preprocess\_text => Preprocesses the text by removing irrelevant content and applying lemmatization.
7. group\_similar\_sections => Groups similar sections based on their content using cosine similarity.
8. save\_grouped\_sections => Saves the grouped sections to text files
9. extract\_images => Extracts images from specified pages in the PDF.
10. extract\_text\_from\_image
11. generate\_description\_with\_openai
12. process\_images\_with\_openai
13. main

def extract\_elements(pdf\_path):  
 *"""Read the PDF and extract elements."""* elements = []  
 with fitz.open(pdf\_path) as pdf:  
 for page in pdf:  
 pix = page.get\_pixmap()  
 img = Image.frombytes("RGB", [pix.width, pix.height], pix.samples)  
 img\_path = "temp\_image.png"  
 img.save(img\_path) # Save the image temporarily  
 elements.append(img\_path)  
 return elements

def detect\_shapes(image\_path):  
 *"""Analyze shapes using OpenCV."""* img = cv2.imread(image\_path)  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 \_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH\_BINARY\_INV)  
 contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
 shapes = []  
 for contour in contours:  
 approx = cv2.approxPolyDP(contour, 0.02 \* cv2.arcLength(contour, True), True)  
 shapes.append(approx)  
 return shapes



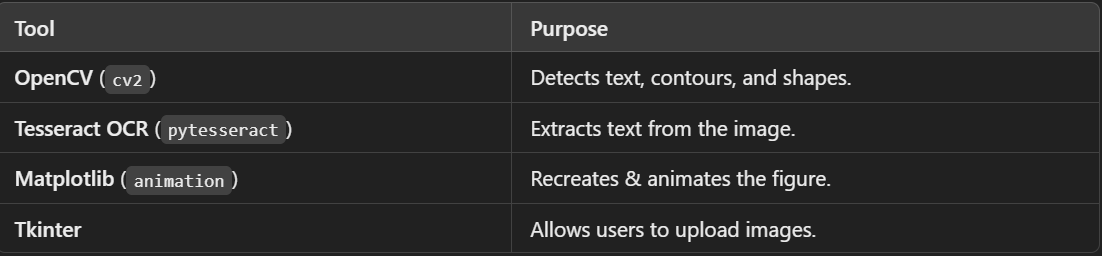
Create a GIF for the images??

#openai.api\_key = "sk-proj-bamTFRro8m\_MPVQC587vjpYOGHuEqgLVVe5nuNFBmSvE-3T\_jARh008h3XdM3jrxrjG-22CbhsT3BlbkFJ8hO69mFj-WK65mlVOf1M3JRPhzGmTUlP23OEymxr-Ud7FpHMcxa3xtl8Ki5S9iolAM6N2VkUYA"  
openai.api\_key = "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"

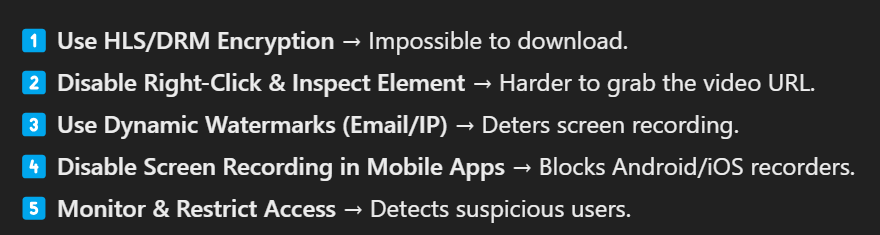
sk-proj-MPsc\_ZZ1IHD-8EzlEk6DchU2KdNLPNr0in3dQVxo4DPUWaSAihRp--gyvf6Hgzo4NtXrFQUSVLT3BlbkFJ7E1hNGZlAJLwdaiiF27JWa8x3AEGDD6G1cm\_eSU\_rhKH-iY\_KUnQm0wf2q1M0jYD240l7tLJIA

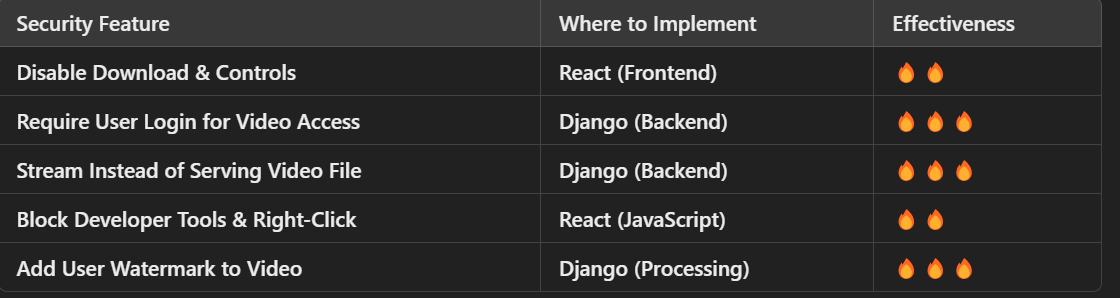
Requirements.txt :

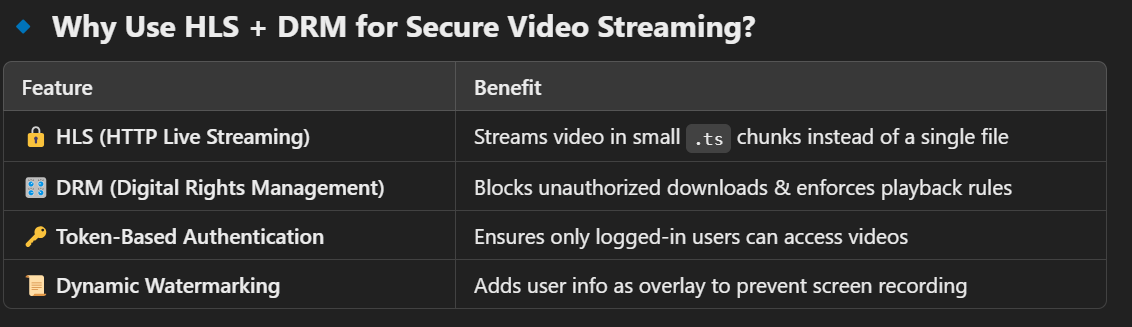
pdfplumber  
spacy  
transformers  
scikit-learn  
gTTS  
pyttsx3  
matplotlib  
Pillow  
opencv-python  
django  
tensorflow  
PyPDF2  
nltk  
PyMuPDF  
ollama  
IPython  
openai

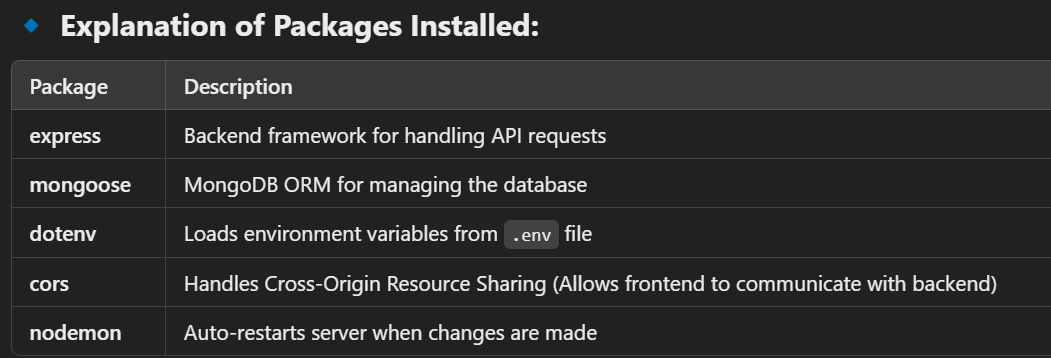


Best practice for blocking the controls of the generated video:









**Summary of Tools/Technologies for Your MERN E-learning Project:**

1. **Backend:**
   * **Node.js** (runtime)
   * **Express.js** (server framework)
   * **PostgreSQL** (database)
   * **Sequelize.js** (for ORM with PostgreSQL)
   * **JWT** (for authentication)
   * **Passport.js** or **OAuth** (for Google/Facebook login)
   * **Bcrypt.js** (for password hashing)
   * **Socket.io** (for real-time features)
2. **Frontend:**
   * **React.js** (library for building UI)
   * **Redux or Context API** (state management)
   * **Axios** (API requests)
   * **React Router** (routing)
   * **TailwindCSS** or **Material-UI** (styling)
3. **Additional Tools:**
   * **Jest** (testing)
   * **Supertest** (API testing)
   * **Docker** (containerization)
   * **Heroku or AWS** (deployment)

