Phase 1 : Preprocessing & Data Extraction

1. Video-to-Audio Extraction -

- **FFmpeg**: Video/audio extraction
- **Librosa**: Audio cleaning

```
 def extract_audio(video_path):
# FFmpeg wrapper
command = f"ffmpeg -i {video_path} -q:a 0 -map a audio.wav"
# Output: Clean WAV file for ASR
```

2. Automatic Speech Recognition (ASR) -

- **OpenAI Whisper**: Speech-to-text with word-level timestamps/ Facebook Wav2Vec2(HuggungFace Model)

```
 def transcribe_audio(audio_path):
model = whisper.load_model("base")  # Hugging Face alternative: Wav2Vec2
result = model.transcribe(audio_path, word_timestamps=True)
return result["segments"]  # [{start, end, text}]
```

3. Frame Sampling & Instructor Detection -

- **Ultralytics YOLOv8**: Real-time instructor detection and track instructor movement for engagement scoring.

```
 def detect_instructor_frames(video_path, fps=2):
model = YOLOv8('yolov8n-face.pt')  # Pretrained face detection

for frame in frames:
    if model.predict(frame).confidence > 0.8:
        instructor_frames.append(frame)
return instructor_frames
```

**Final Output of this phase - **

Preprocessed.json

```
 {
  "audio": "lecture_audio.wav",
  "transcript": [
    {"start": 0.0, "end": 5.3, "text": "Today we'll discuss quantum..."},
    {"start": 5.3, "end": 12.1, "text": "The Schrödinger equation ψ = ..."}
  ],
  "instructor_frames": [
    {"timestamp": 3.2, "path": "frame_032.jpg"},
    {"timestamp": 8.7, "path": "frame_087.jpg"}
  ]
}
```

Phase 2: Content Analysis & Classification

1. STEM Keyword Detection -

- **spaCy**: STEM term detection (`integral` , `eigenvalue` , `mitosis`) must be trained on STEM keywords.
- **Topic Segmentation: Using **BERTopic modeling technique that is HuggingFace transformers and based on c-TF-IDF.

2. Equation/Diagram Detection Pipeline -

- **LaTeX-OCR**: Use `pix2tex` model (pretrained) + fine-tune on handwritten equations
- **Detectron2**: Start with COCO, fine-tune on diagram dataset (AI2D, Textbook Figures)
- **Custom CNN**: Train ResNet50 on code screenshot dataset (GitHub + lecture slides)
- **MediaPipe**: Gesture recognition (pointing/writing)

Phase 3: Hybrid Summarization Engine

**1. Segment-Wise Summarization - **

- Hugging Face** BART-large-CNN **pretrained model effective when fine-tuned for text generation**.**
- Based on each different video transcript **custom post-processing** must done by **self training and improving accuracy**.

**2. Visual Asset Processing - **

- **MathPix API**: Equation OCR (fallback: LaTeX-OCR)
- **BLIP-2**: Diagram captioning
- **FFmpeg**: Video snippet creation

```
for segment in lecture_segments:
# Text summary
text_summary = bart_summarize(segment.text)

# Visual assets
visual_assets = []
for frame in segment.key_frames:
    if contains_formula(frame):
        latex = mathpix.extract(frame)
        visual_assets.append({"type": "formula", "content": latex})
    elif contains_diagram(frame):
        caption = blip2.caption(frame)
        visual_assets.append({"type": "diagram", "caption": caption})

# Generate hybrid segment
save_hybrid_segment(
    text_summary,
    visual_assets,
    start_time=segment.start,
    end_time=segment.end
)
```

Phase 4: Output Generation & Packaging

1. Multimodal Document Assembly -

Tech:

- **Pandoc**: Document conversion (MD → HTML/PDF)
- **MathJax**: Equation rendering
- **FastAPI**: JSON API endpoint **Output Formats**:
- **Markdown**: For general use
- **HTML + MathJax**: Web deployment
- **PDF**: With rendered equations

2. Video Chapter Generation -

```
def create_video_chapters(segments, video_path):
chapters = []
for segment in segments:
    # Create snippet
    snippet_path = f"snippets/{segment['title']}.mp4"
    ffmpeg_extract(video_path, snippet_path,
                  segment['start'], segment['end'])

    chapters.append({
        "title": segment['title'],
        "start": segment['start'],
        "end": segment['end'],
        "snippet": snippet_path
    })

return chapters
```

Final Output -

Markdown

```
 ## Quantum Entanglement (12:30-18:40)
**Summary:** Explains superposition principle...

**Key Visuals:**
▶ [Derivation Video](snippet_12_30.mp4) (0:15)
`\psi = \alpha|0\rangle + \beta|1\rangle`
![Wave Function](waveframe.png)
```