Lecture Notes Generation Workflow Documentation

1. Text Cleaning (Lecture Notes Preprocessing)

Purpose: Prepare raw lecture text for LLM processing by removing noise while keeping sentences readable and context intact.

Steps: - **Remove duplicate sentences:** Use Python `set()` for exact duplicates or `fuzzywuzzy` library for near-duplicates. - **Remove filler words and interjections:** Eliminate words like “umm”, “ah”, “like”, “you know” using Python `regex` patterns or a predefined stopword list. - **Clean extra spaces and blank lines:** Use Python string methods (`strip()`, `replace()`) to remove unnecessary whitespace and empty lines.

Tools & Libraries: - Python standard library (`set()`, `re`) - `fuzzywuzzy` for fuzzy duplicate detection (`pip install fuzzywuzzy[speedup]`) - Optional: `pandas` if handling large text datasets

Output: - Clean, human-readable, LLM-friendly text ready for **important topic extraction**.

---

2. Important Topics Extraction

Purpose: Identify key concepts, definitions, formulas, examples, and main points from the clean lecture text.

Steps: - Provide the **clean text** to an LLM (OpenAI GPT, LLaMA, or similar). - Use prompts like: *"Extract the key points, definitions, examples, and main concepts from this lecture text."* - Optionally, combine **frequency-based keyword extraction** (TF-IDF or RAKE) with LLM to ensure all main topics are captured.

Tools & Libraries: - OpenAI GPT or any LLM API - Python NLP libraries for keyword extraction (`scikit-learn` for TF-IDF, `rake-nltk`)

Output: - List of **important topics and main points** ready for structured note generation.

---

3. Notes Creation (Structured Markdown/HTML)

Purpose: Generate high-quality, structured lecture notes combining **full cleaned text** and **extracted important topics**.

Steps: - Provide LLM with: - The **cleaned lecture text** - The **list of important topics** - Ask LLM to generate notes with: - Proper **headings/subheadings** - **Bolded main points** - Examples clearly formatted - Optional numbering or bullet lists for clarity - Output notes in **Markdown** or **HTML** format for easy rendering or export.

Tools & Libraries: - OpenAI GPT or any LLM API - Markdown / HTML generators (Python: `markdown`, `jinja2`) - Optional: Frontend renderer in Next.js (`react-markdown`)

Output: - Human-readable, **well-structured lecture notes** in Markdown/HTML ready for use or export.

---

4. Output & Export

Purpose: Deliver final notes in a usable format for students or team.

Steps: - Export generated notes as: - **Markdown**: For further processing or web display - **HTML**: For direct rendering in the app - **PDF** (optional): Use Python libraries like `WeasyPrint`, `ReportLab`, or frontend export tools (`react-pdf`)

Tools & Libraries: - Python: `WeasyPrint`, `ReportLab` - Next.js / React: `react-markdown`, `react-pdf`

Output: - Final lecture notes ready for study, sharing, or integration in the app.