■ Day-12 Task: Complex Idea Explainer

■ Goal

Detect difficult or technical parts in study materials (summaries, uploaded text, flashcards) and generate simplified explanations for students.

■ Person-1 (Frontend – Explanation UI)

- New Page: ExplainIdeasPage.jsx - Button: 'Find & Explain Difficult Parts'. - Calls /explain API. - Display results as collapsible cards (difficult part, simplified explanation, example).
- Integration: - Add navigation link → 'Explain Ideas'. - Show spinner while LLM generates explanations. - Allow user to save explanations to their study library.
- Optional Enhancement: - Highlight difficult parts directly inside the summary view. - When clicked → pop-up with explanation.

■ Person-2 (Backend – Logic & API)

- Difficult Part Identification: - Add a function find_difficult_parts(text) in a new file complexity_analyzer.py. - Use heuristics (e.g., long sentences, uncommon academic terms, jargon frequency) OR ask LLM to identify difficult concepts. - Extract those parts (sentences/paragraphs).
- Explanation Generation: - Create explain_complex_parts(text) function. - For each detected difficult part, call LLM with a prompt: 'Explain this in simple terms with an example.' - Return JSON with difficult part, explanation, and example.
- API Route: - New file explain_route.py. - Endpoint: POST /explain with { fileId }. - Fetch text/summary → run analyzer → return simplified explanations.
- Store in MongoDB: - Create explanations_collection to store userId, fileId, difficult_part, explanation, date.

■ Deliverables (Day-12)

- Backend can automatically detect difficult text & return simple explanations.
- Frontend page where student can see & interact with explanations.
- Data saved in MongoDB for review later.