EduMUSE

Soren Larsen snlarsen@ucsc.edu , Mudit Arora muarora@ucsc.edu ,
Yousuf Golding ygolding@ucsc.edu , Ishika Kulkarni ikulkar1@ucsc.edu
UC Santa Cruz, Silicon Valley Campus, Santa Clara, CA, USA

Abstract

EduMUSE is an interactive, multimodal educational assistant designed to enhance student learning through targeted, AI-powered study support. Built using a modular multi-agent architecture powered by CrewAI, EduMUSE allows users to upload academic PDFs, highlight specific text segments, and trigger customized AI workflows. These workflows include text summarization, quiz generation, contextual question answering, external knowledge retrieval, and podcast-style audio explanations. The system emphasizes user control, enabling learners to focus on areas they find most challenging or important. All modules are orchestrated via reusable agents, allowing flexible composition and easy expansion. EduMUSE showcases the potential of modular LLM agents to augment traditional study workflows and support personalized, multimodal learning experiences.

1 Introduction

We developed **EduMUSE**, an educational multimodal understanding and study engine that enables learners to interactively explore and review academic materials using large language models (LLMs). Unlike prior AI-based tools that offer static summarization or rigid learning paths, EduMUSE provides a dynamic, user-driven experience built around modular, agent-based workflows.

Our MVP prototype allows users to upload PDF documents and highlight specific portions of text to trigger targeted AI interactions. These include generating summaries, quizzes, question-answer pairs, relevant external knowledge retrieval, and podcast-style audio explanations via text-to-speech (TTS). This multimodal interaction model supports personalized and exploratory learning, helping users focus on content they find most challenging or conceptually dense.

Although we initially envisioned a more extensive pipeline with automated pacing control, grading of handwritten answers, and full session packaging, our current prototype provides a functional core experience that demonstrates the feasibility and utility of interactive, multimodal study augmentation. We also conducted preliminary evaluations of key components such as summarization quality and quiz generation accuracy, providing encouraging evidence for further development.

EduMUSE contributes to the growing space of AI-enhanced education by emphasizing modularity, user control, and multimodal support. It showcases how selective, user-initiated AI actions can augment traditional study practices and foster deeper, more flexible engagement with academic materials.

2 Related Work

EduMUSE is grounded in a growing body of research on intelligent tutoring systems, retrieval-augmented generation, and multimodal applications of large language models (LLMs) in education. While our prototype does not yet integrate every planned feature, several works informed our choices around summarization, content generation, and user interaction.

Interactive Human-AI Tutoring. Thomas et al. (1) demonstrated that hybrid human-AI tutoring systems can improve learning outcomes by balancing automation with user agency. Their findings support our design choice to build EduMUSE around user-initiated interactions—specifically, highlighting PDF segments to trigger modular AI workflows.

Retrieval-Augmented Generation (RAG). Dong et al. (2) present a flexible framework for building adaptive tutors using knowledge graph-augmented RAG pipelines. While our system does not use knowledge graphs, the modular RAG architecture inspired our implementation of a knowledge retrieval feature based on document context and keyword-driven external search.

Pedagogical Alignment in Tutors. Feng et al. (3) introduced *CourseAssist*, emphasizing difficulty adaptation and response appropriateness in CS education. Although our prototype did not fully implement adaptive pacing, we explored complexity control within generated questions and summaries.

Multimodal Learning Tools. Jiang and Jiang (4) emphasized the benefits of multimodal content—including diagrams and audio—to deepen conceptual understanding. Our podcast-style TTS summaries were directly motivated by this work, as a lightweight way to support auditory learning.

Keyphrase Extraction for Educational QA. Brun and Riedel (5) proposed using keyphrase extraction to drive question-answer generation. We employed a similar concept for generating quiz questions, situational-based questions, and higher-order questions based on user-selected text regions, aligning educational content with user-identified areas of focus.

Auditory Learning and TTS in Education. Prior work in educational accessibility has shown that auditory delivery of content (e.g., via podcasts or screen readers) supports deeper engagement for auditory learners and improves flexibility in study contexts. EduMUSE adopts a similar philosophy through its TTS agent, enabling users to consume AI-generated summaries in a spoken format that complements visual learning.

These works collectively support the feasibility and relevance of EduMUSE's MVP design, even at a reduced scope. They validate our emphasis on personalization, modular AI tooling, and multimodal engagement as key components of AI-augmented study systems.

3 Proposed Solution

3.1 System Overview

EduMUSE is implemented as a modular AI-powered study assistant that enables users to explore their own learning materials through targeted AI actions. The current MVP focuses on supporting interactive study sessions based on user-selected text regions within uploaded PDF documents.

The system workflow is as follows: users upload a PDF file and can then highlight specific passages they wish to analyze or review. These selected passages are processed through a set of AI modules, each of which generates a different form of educational output. This design emphasizes flexibility and user control, allowing learners to focus on the content most relevant to their needs.

The prototype includes the following AI modules:

1. **Summarization:** Generates concise summaries of highlighted text using LLM prompts.
2. **Quiz Generation:** Produces multiple-choice and short-answer quiz questions based on the selected content.
3. **Question Answering:** Allows users to pose additional questions about the highlighted text and returns contextual answers. The questions are context-based as well as situational and higher-order questions to enhance understanding and apply reasoning skills.
4. **Knowledge Retrieval:** Retrieves relevant external information (articles, videos) based on key concepts extracted from the selected text.
5. **Podcast-style Summary:** Converts the generated text summary into an audio file using text-to-speech (TTS), supporting auditory learning.

Modules can be invoked independently for any selected text segment, enabling personalized, on-demand learning interactions. Although we originally planned additional features such as adaptive pacing, automated grading of handwritten answers, and complete session packaging, these were not included in the current MVP due to project scope constraints.

Overall, EduMUSE demonstrates the feasibility of interactive, modular AI tooling for educational content augmentation and provides a strong foundation for future development.

3.2 Models and Tools

The following models and tools were used to implement the current version of EduMUSE:

LLMs for Generation Tasks. We used CrewAI workflows, which abstract over large language models, to implement summarization, quiz generation, question answering, and knowledge retrieval features. Most flows leveraged either GPT-4 or Gemini 2.5 models, depending on task suitability and latency.

Knowledge Retrieval and Term Extraction. Key term extraction and external knowledge retrieval were implemented using CrewAI's retrieval flows, which combine LLM-driven extraction with external search APIs. No additional NLP libraries (e.g., spaCy, Sentence-BERT) were required beyond what was handled within CrewAI modules.

Text-to-Speech (TTS). Podcast-style summaries were generated using CrewAI flows integrated with TTS APIs, with OpenAI's TTS services along with Eleven Labs providing the best quality for our prototype.

Frontend and Backend Stack. The frontend was implemented as a lightweight web interface supporting PDF upload, text highlighting, and interaction with CrewAI workflows. A custom Flask server handles file uploads, file serving, and basic file management, acting as a bridge between the frontend and the CrewAI-based AI processing pipelines. The server exposes REST endpoints for PDF upload and retrieval, while AI workflows are triggered through separate CrewAI APIs.

This architecture enabled rapid prototyping and flexible experimentation, focusing our efforts on user experience design and evaluation of AI-assisted study features.

3.3 Knowledge Retrieval Implementation

Our Knowledge Retrieval module implements three distinct approaches using CrewAI agents with specialized prompting strategies:

Web Search Flow. Uses SerperDevTool integration with an Academic Searcher agent prompted to:

> *"Search for credible academic sources about [topic]. Focus on .edu domains, academic databases, and recent publications. For each source, provide title, URL, publication venue, and credibility assessment targeting 2023-2025 publications..."*

LLM Knowledge Flow. Employs a Knowledge Specialist agent that synthesizes information from training data:

> *"Using your comprehensive training data knowledge, provide academic sources about [topic]. Generate a curated list of 5-8 sources including key papers, textbooks, publication details, and relevance scores. Focus on foundational papers and seminal works..."*

Hybrid Retrieval Flow. Combines both approaches through a Hybrid Integration Specialist:

> *"Create a comprehensive academic source collection using BOTH web search for recent developments (2023-2025) AND training data for foundational works. Use web search component for current papers and LLM knowledge component for foundational/seminal works. Combine both sources into 8-10 sources total with clear categorization by retrieval method and relevance ranking..."*

CrewAI Agent Architecture. Each retrieval flow is implemented as a specialized CrewAI agent:

- **Web Search Agent**: Academic Searcher with SerperDevTool access
- **LLM Knowledge Agent**: Knowledge Specialist with pure LLM synthesis
- **Hybrid Agent**: Integration Specialist combining both approaches

All agents execute tasks through CrewAI's orchestration framework, with results processed through a standardized flow registry system enabling modular evaluation and comparison.

4 Experimental Design

4.1 Evaluation Goals

Our evaluation aimed to measure whether **EduMUSE** modules improve students' ability to review and understand educational materials. Given the modular design of the system, we performed separate evaluations on core components, starting with our Knowledge Retrieval agent.

4.2 Inputs

- User-uploaded study materials (PDFs)
- User-selected text spans (highlighted in frontend)
- User-generated follow-up questions (QA module)

4.3 Outputs

- Summaries of highlighted content
- Quiz questions based on selected text
- Answers to user questions (QA flow)
- External knowledge results (Knowledge Retrieval agent)
- Podcast-style audio summaries

4.4 Knowledge Retrieval Agent Evaluation

We conducted a systematic evaluation of the Knowledge Retrieval module, comparing three distinct retrieval flows:

- **Web Search Flow**: uses Web APIs to retrieve current sources
- **LLM Knowledge Flow**: synthesizes knowledge from model training data
- **Hybrid Retrieval Flow**: combines Web Search with LLM synthesis

Methodology. We tested each flow on two academic topics: *Machine Learning Transformers* and *Quantum Computing Algorithms*. Evaluation metrics included:

- **Relevance** (30% weight)
- **Credibility** (25%)
- **Coverage** (20%)
- **Recency** (15%)
- **Accessibility** (10%)

Each flow was scored by manual evaluation of sources retrieved (quality scores normalized to a 10-point scale). Processing time and content length were also recorded.

Prompting Design and Task Instructions. Each retrieval flow was implemented with carefully crafted prompts designed to maximize source quality and educational relevance. The Web Search Flow targets current academic publications through domain-specific search constraints (.edu sites, academic databases), while the LLM Knowledge Flow focuses on synthesizing foundational knowledge from training data. The Hybrid Flow explicitly combines both strategies through a two-stage retrieval and synthesis process.

All flows received identical input specifications: a topic description and context from user-selected text. The prompts were designed to return sources in a consistent format including title, authors, publication venue, credibility assessment, and relevance justification. Key design principles included explicit output format specifications, domain-specific search constraints, credibility assessment requirements, and educational context alignment.

Evaluation Protocol. For each topic (Machine Learning Transformers, Quantum Computing Algorithms), we executed all three flows and manually evaluated the returned sources using our weighted scoring system. Sources were assessed for academic rigor, citation potential, and educational value for the target learning context. Processing time and content length were recorded to assess computational efficiency alongside quality metrics.

Flow	Avg Time (s)	Total Score	Sources Count	Content Length
Web Search	11.64	7.5	4.0	1744.5
LLM Knowledge	17.16	8.05	4.0	3722.0
Hybrid Retrieval	19.83	9.15	5.5	3206.5

Table 1: Knowledge Retrieval module evaluation results:contentReference[oaicite:1]index=1

Results.

Discussion. The Hybrid Retrieval flow produced the highest overall score (9.15), with strong coverage and credibility. LLM Knowledge flow performed well for synthesizing foundational content but lacked recency. Web Search was fast and provided current sources but with lower coverage depth.

4.5 Quiz Generation Evaluation

The Quiz Generation module is designed as a multi-purpose CrewAI agent that receives a highlighted text segment and generates a structured quiz. This agent runs in the QA pipeline implemented in the ‘ishika/qa-pipeline‘ branch and is invoked via the ‘MultiAgentOrchestrator‘ class defined in our backend.

Inputs.

- User-highlighted text extracted from the uploaded PDF via the frontend.
- Optional user question or refinement input passed as part of the orchestrator query.

Outputs.

- Question-Answer (QA) pairs
- Multiple-Choice Questions (MCQs)
- Higher-order and situational-based questions

Implementation. The Quiz Agent is configured in ‘agents.py‘ using a CrewAI agent with a static system prompt template. The orchestrator passes the highlighted text to the agent, which formats its output into multiple structured quiz blocks. The entire process is managed by the ‘MultiAgentOrchestrator‘, which runs this agent either in isolation or as part of a larger flow involving summarization or QA.

Evaluation. We generated quizzes using the prompt “Explain photosynthesis in plants” and inspected the resulting content. The outputs were manually evaluated on:

- **Coverage of input text**
- **Clarity of questions and answers**
- **Diversity across question types**
- **Accuracy of answers**

Results. The agent consistently produced distinct question types with minimal prompt engineering. MCQs and QA pairs were highly accurate. Average runtime per quiz generation was 8.2 seconds using GPT-3.5 Turbo. The outputs required no post-processing, showing strong template consistency.

4.6 TTS Module Evaluation

The podcast-style summary feature is implemented as a CrewAI agent integrated with OpenAI and Eleven Labs API. This module converts AI-generated summaries into spoken audio streams to support multimodal learning.

Inputs.

- Text summaries generated by the Summarization Agent
- Optional metadata such as language or speaker preferences (not yet user-configurable)

Outputs.

- Byte stream of synthesized speech, returned as audio files (MP3/WAV)

Implementation. The TTS Agent is a CrewAI agent that receives summarized text (via the orchestrator). The input text like “Explain photosynthesis in plants” is passed and converted to audio chunks. The final audio is reconstructed from streaming responses and returned to the user.

Evaluation. Testing was performed using 5 sample summaries from different subjects (biology, mathematics, history). Evaluation focused on:

- **Audio quality and pronunciation**
- **Fidelity to the input summary**
- **Frontend compatibility for audio playback**

Results. Audio output from the TTS Agent was clear, natural-sounding, and required no post-processing. Integration with the frontend enabled successful download and playback. Runtime for audio generation averaged 10 seconds per summary. Minor failures occurred when input summaries exceeded API token limits, but these were resolved by limiting summary length. As showcased below, user feedback indicated strong engagement potential for auditory learning contexts.

4.7 Summary Agent Assessment

The summary assessment is designed to generate summaries of varying complexities based on topics introduced to EduMUSE. To evaluate the module, we utilized the ROUGE and Bertscore metrics to understand the grammatical quality of the responses.

Metric	Score
Engagement	4.1
Clarity	4.5
Naturalness	3.2
Relevance	4.4

Table 2: TTS Agent evaluation results.

Table 3: Summary flow evaluation results using ROUGE metrics and BERTScore.

Level	R-1	R-2	R-L	BERT	Time(s)
Beginner	0.463	0.163	0.257	0.826	76.3
Intermediate	0.213	0.025	0.107	0.812	81.9
Advanced	0.190	0.035	0.129	0.805	67.5
Beginner	**0.293**	0.079	0.164	0.823	55.7
Intermediate	0.245	0.035	0.140	0.779	68.7
Intermediate	0.316	0.061	0.165	0.807	61.5
Advanced	0.242	0.049	0.161	0.793	50.7
Average	0.280	0.064	0.160	0.806	64.7

The rouge-scores measured the lexical overlap while bert scores focused on the semantic quality of the summaries in comparison to te original documents. However to a lack of available metrics to properly evaluate the relevancy and clarity of the summaries, we utilized human evaluations to measure this as showcased above.

Metric	Score
Usefulness	4.0
Clarity	3.5
Coverage	3.0
Conciseness	4.0

Table 4: Note Summarization Agent evaluation results.

The summaries were evaluated based on four chosen metrics. As the table showcases, all of the results showcased average to above average responses, with the strongest metrics being the usefulness and conciseness of the summaries. However it is clear that the quality of the information can be improved given the lower results in clarity and coverage.

4.8 Quiz Generation Evaluations

Similar to the summary agent, we determined the best way to evaluate the quality of the generated quizzes was to perform human evaluations and provide similar metrics (Helpfulness, Correctness, Difficulty, Coverage) for the questions to be graded on. The results of this are showcased below:

Metric	Score
Helpfulness	4.0
Correctness	3.5
Difficulty	4.0
Coverage	4.0

Table 5: Practice Assessment Agent evaluation results.

Most of the metrics showcased above average results, the correctness metric being the only case with a score below 4. This is however a concern because one of the most important parts of the module is ensuring that the questions being generated make sense and are correct according to the information in the pdfs.

Another concern is the balance between difficulty and helpfulness, because we do not want the questions to be too challenging or too simple as this would negatively affect how useful the assessments actually are.

5 Generated Examples from Treeformer Content

To demonstrate the capabilities of EduMUSE's summarization and assessment agents, we include example outputs generated from a technical research paper: *Treeformers: Enhancing Hierarchical Understanding in NLP* (6).

5.1 Generated Summary (Intermediate Level)

Title: Treeformers: Enhancing Hierarchical Understanding in NLP

Traditional Transformers struggle with modeling hierarchical sentence structure. Treeformer, inspired by the CKY parsing algorithm, enhances compositional generalization and improves machine translation and summarization. It introduces tree-encoder layers that aggregate subphrases through attention-based pooling, capturing predicate-argument structures more effectively. With optimizations reducing complexity from cubic to linear, Treeformer outperforms baselines in several NLP tasks while remaining computationally feasible.

5.2 Generated Practice Assessment (Intermediate)

Multiple Choice:

Q: What is one benefit of Treeformer over traditional Transformer models?

A) It requires no training data.

B) It improves compositional generalization using hierarchical encodings. ()

C) It uses convolutional layers instead of attention.

D) It eliminates positional encodings entirely.

Short Answer:

Q: What algorithm inspired the Treeformer design, and why?

A: The CKY algorithm, because it structures sentences hierarchically using composition and pooling strategies to simulate parse trees.

Essay Prompt (8 pts):

Explain how Treeformer captures hierarchical phrase structure and why this matters for NLP tasks such as translation and summarization. Support your claims with concepts from the Treeformers paper.

6 Conclusion

EduMUSE demonstrates the potential of modular, multimodal AI agents to transform the way students engage with academic content. By empowering learners to initiate tailored workflows—ranging from summarization and quiz generation to knowledge retrieval and podcast-style explanations—our system promotes deeper comprehension, flexibility, and sustained engagement.

Although our MVP does not yet incorporate all envisioned features, it lays a strong foundation for future work in adaptive, user-centered educational tooling. Going forward, we plan to enhance session-level personalization, integrate learning analytics, and explore more robust evaluation strategies.

EduMUSE represents a step toward AI systems that not only inform but actively support how students learn.

7 Future Work

Here are several promising extensions remain for future development:

- **Grading and Feedback Modules:** Implement handwritten answer grading and automated feedback generation via a Grading Assistant Agent. This would enable more comprehensive assessment workflows within the same platform.
- **Difficulty Calibration and Adaptation:** Introduce difficulty-level selection for generated content (summaries, quizzes), allowing students to engage with materials at beginner, intermediate, or advanced levels depending on their background.
- **Multimodal Content Understanding:** Expand input support beyond text, including diagrams, tables, and figures from academic PDFs. This would involve visual content parsing and cross-modal reasoning capabilities.
- **Session Packaging and Study Progression:** Develop the ability to package a complete study session—including summaries, assessments, podcast audio, and retrieved sources—into a structured, exportable learning bundle.
- **User Interface Improvements and Accessibility:** Enhance the frontend to better support accessibility (e.g., screen reader compatibility, keyboard navigation) and enable richer, more intuitive interactions with the AI modules.
- **User Studies and Longitudinal Evaluation:** Perform user-centered studies to evaluate learning outcomes, engagement, and usability over time, particularly focusing on how students use different agents to reinforce understanding.

These improvements will further position EduMUSE as a comprehensive, AI-enhanced study companion capable of supporting diverse learning preferences and educational contexts.

8 Codebase

The EduMUSE system was collaboratively developed using GitHub, and the full codebase is available at https://github.com/iamsorenl/EduMUSE.

A Prompts

This appendix contains all prompts used within EduMUSE.

A.1 Prompts from agents.py

A.1.1 Answering with Context

```
You are an expert assistant. Use the following context to answer:

{context_block}

Question: {question}
Answer:
```

A.1.2 Answering without Context

```
You are an expert assistant.
Question: {question}
Answer:
```

A.1.3 Answer Verification

```
You are a fact-check assistant.
Context:
{combined_context}

Answer to verify:
{answer}

Is the answer fully supported by the context? If yes, just respond 'YES'. If not, respond 'NO'
and briefly explain which part is not supported.
```

A.1.4 Quiz Generation

```
From the following content, generate a structured quiz in this format:

1. 10 Question-Answer Pairs
2. 20 Multiple Choice Questions (4 options each, correct answer marked)
3. 5 Higher-Order Thinking Questions with Answers
4. 5 Situational-Based Questions with Answers

Content:
{content}
```

A.2 Prompts from assessment_tasks.yaml

A.2.1 Assessment Question Design

```
Design {num_questions} high-quality assessment questions about '{topic}' based on the provided
academic sources.

**Question Distribution Requirements**:

1. **Question Types**:
    - Multiple Choice: {mc_count} questions (4 options each)
    - Short Answer: {sa_count} questions (2-3 sentence responses)
    - Essay/Long Answer: {essay_count} questions (paragraph+ responses)

2. **Cognitive Level Distribution** (Bloom's Taxonomy):
```

```
    - Remember (recall facts): 20% of questions
    - Understand (explain concepts): 20% of questions
    - Apply (use in new situations): 25% of questions
    - Analyze (break down, compare): 20% of questions
    - Evaluate (judge, critique): 10% of questions
    - Create (design, construct): 5% of questions

3. **Question Design Standards**:
    - Clear, unambiguous wording free of double negatives
    - Test understanding, not reading comprehension
    - Culturally neutral and inclusive language
    - No trick questions or gotchas
    - Progressive difficulty within each type
    - Cover key concepts proportionally

4. **For Each Question, Provide**:
    - Question number and text
    - Question type (MC/SA/Essay)
    - Cognitive level targeted
    - Concept(s) being tested
    - Estimated difficulty (Easy/Medium/Hard)
    - Time estimate for completion

Base questions on these sources:
{sources}

Ensure questions comprehensively assess understanding of the material.
```

A.2.2 Answer Validation and Rubric Creation

```
Create comprehensive answer keys and assessment materials for all {num_questions} questions.

**For Multiple Choice Questions**:
1. Clearly mark the correct answer
2. Explain why the correct answer is right
3. Explain why each distractor is wrong
4. Note common misconceptions addressed
5. Provide learning tips for the concept

**For Short Answer Questions**:
1. Provide ideal model answer (2-3 sentences)
2. List key points that must be included (partial credit)
3. Create scoring rubric:
    - 3 points: Complete, accurate answer with all key points
    - 2 points: Mostly correct with minor omissions
    - 1 point: Partially correct or major omissions
    - 0 points: Incorrect or no relevant content
4. Note acceptable answer variations
5. Common errors to watch for

**For Essay Questions**:
1. Create detailed scoring rubric with criteria:
    - Content accuracy and completeness (40%)
```

    - Analysis and critical thinking (30%)
    - Use of examples/evidence (20%)
    - Organization and clarity (10%)
2. Provide exemplar response outline
3. List required elements for full credit
4. Describe levels of performance (Excellent/Good/Satisfactory/Needs Improvement)
5. Sample strong response excerpts

Additional Requirements:
- Reference specific source material for each answer
- Include educational explanations that teach, not just evaluate
- Suggest follow-up learning for incorrect answers
- Ensure all answers are factually accurate and up-to-date

A.2.3 Difficulty Calibration and Final Assessment Creation

Finalize the assessment for {user_level} learners with appropriate difficulty calibration and enhancements.

Calibration Tasks:

1. **Difficulty Review and Adjustment**:
    - Verify each question matches intended difficulty
    - Adjust vocabulary and complexity for {user_level}
    - Ensure appropriate progression (easy → challenging)
    - Balance question types and cognitive demands
    - Check total time requirement is reasonable

2. **Assessment Organization**:
    - Sequence questions optimally:
        * Start with 1-2 confidence builders
        * Alternate question types if mixed format
        * Place most challenging items in middle third
        * End with moderate difficulty
    - Group by topic or format based on assessment type
    - Create clear section headers and transitions

3. **Instructions and Formatting**:
    - Write clear, student-friendly instructions
    - Include time recommendations per section
    - Format for easy reading and navigation
    - Add point values for each question
    - Create answer sheet template if needed

4. **Learning Enhancement Features**:
    - Pre-assessment preparation checklist
    - Test-taking strategy reminders
    - Self-check prompts before submission
    - Post-assessment reflection questions
    - Suggested study activities based on performance

5. **Create Final Versions**:
    - Student Version: Clean question-only format

```
    - Instructor Version: Complete with answers/rubrics
    - Quick Reference: Answer key summary
    - Digital Version: Formatted for online delivery

**Metadata to Include**:
- Total points possible
- Recommended time limit
- Passing score suggestion (e.g., 70%)
- Learning objectives alignment
- Prerequisite knowledge assumed
```

A.3 Prompts from podcast_flow.py

A.3.1 Podcast Script Generation

```
You are a podcast script writer. Given the following content, generate a podcast-style conversation between a Host and a Guest. The conversation should be informative, engaging, and cover the main points. The topic is: {topic}. Return a **valid JSON list** of dictionaries. Each dictionary must have keys: 'speaker', 'text', 'voice_id'. Use this voice_id for Host: JBFqnCBsd6RMkjVDRZzb, and this for Guest: EXAVITQu4vr4xnSDxMaL. Use only double quotes (") for all keys and values. Do not wrap the output in markdown or code blocks.

Content:
{content[:3000]}

Podcast Dialogue:
```

A.4 Prompts from web_search_flow.py

A.4.1 Academic Web Search

```
Search the web for high-quality academic sources about: {topic}

Use advanced search operators:
- site:edu OR site:arxiv.org OR site:scholar.google.com
- filetype:pdf for academic papers
- "peer reviewed" OR "journal article" for credible sources

Find 5-8 credible academic sources with:
- Title and URL
- Publication source/institution
- Brief description of relevance
- Credibility assessment (1-10 scale)

Focus on recent publications (2020+) and authoritative domains.
```

A.5 Prompts from hybrid_retrieval_flow.py

A.5.1 Hybrid Knowledge Retrieval

```
Create a comprehensive academic source collection about: {topic}

Use BOTH approaches:

1. WEB SEARCH COMPONENT:
   - Search for recent papers and current developments
```

```
   - Focus on 2023-2025 publications
   - Target academic databases and .edu sites

2. LLM KNOWLEDGE COMPONENT:
   - Identify foundational/seminal works from training data
   - Include highly-cited classic papers
   - Add authoritative textbooks and review articles

3. SYNTHESIS:
   - Combine both sources into a coherent collection
   - Remove duplicates and rank by relevance + credibility
   - Provide 8-10 sources total with clear categorization

For each source, specify:
- Title, authors, publication venue
- Source type (recent/foundational)
- Retrieval method (web/knowledge)
- Relevance and credibility scores
```

A.6 Prompts from llm_knowledge_flow.py

A.6.1 LLM Knowledge Retrieval

```
Using your comprehensive training data knowledge, provide academic sources about: {topic}

Generate a curated list of 5-8 academic sources including:
- Key academic papers, textbooks, and authoritative sources
- Publication details (authors, journals, approximate years)
- Brief description of each source's contribution
- Relevance to the topic (1-10 scale)

Focus on:
- Foundational papers and seminal works
- Highly cited research in the field
- Authoritative textbooks and review papers
- Recent significant developments (within training data cutoff)

Note: This is based on training data, not real-time web search.
```

A.7 Prompts from summary_tasks.yaml

A.7.1 Key Concept Extraction

```
Analyze the provided academic sources about '{topic}' and extract:

1. **Core Concepts**: Identify 5-10 key concepts that are essential for understanding
    the topic
2. **Learning Objectives**: Determine what students should know/be able to do after
    studying this material
3. **Concept Relationships**: Map how concepts connect and build upon each other
4. **Prerequisite Knowledge**: Identify what learners need to know beforehand
5. **Difficulty Indicators**: Note which concepts are typically challenging for learners

Sources to analyze:
{sources}
```

```
Provide a structured analysis that will guide summary creation.
```

A.7.2 Multi-level Summary Creation

```
Based on the concept analysis, create educational summaries for topic '{topic}':

Create THREE versions targeting different levels:

1.  **Beginner Level**:
    * Use simple language and everyday examples
    * Focus on fundamental concepts only
    * Include helpful analogies and visual descriptions
    * Avoid technical jargon
    * Length: 300-400 words

2.  **Intermediate Level**:
    * Include more technical terminology with clear explanations
    * Cover main concepts and some applications
    * Provide real-world examples from the field
    * Show connections between concepts
    * Length: 500-700 words

3.  **Advanced Level**:
    * Use field-appropriate technical language
    * Include nuanced details and edge cases
    * Discuss current research and open questions
    * Connect to broader theoretical frameworks
    * Length: 800-1000 words

Summary format requested: {summary_format}
User's primary level: {user_level}
```

A.7.3 Learner-Adapted Summary Refinement

```
Refine and optimize the summaries for the specific learner:

User Level: {user_level}
Learning Objectives: {learning_objectives if learning_objectives else 'General understanding'}

Enhance the summaries by:
1.  **Highlighting** the most relevant summary level for the user
2.  **Adding transitions** between levels to show learning progression
3.  **Including study tips** specific to the user's level
4.  **Suggesting next steps** for continued learning
5.  **Creating a glossary** of important terms

Also provide:
- Key takeaways (5-7 bullet points)
- Recommended study sequence
- Self-check questions for understanding
```

References

[1] Danielle R. Thomas, Jionghao Lin, Erin Gatz, Ashish Gurung, Shivang Gupta, Kole Norberg, Stephen E. Fancsali, Vincent Aleven, Lee Branstetter, Emma Brunskill, and Kenneth R. Koedinger. 2023. Improving Student Learning with Hybrid Human-AI Tutoring: A Three-Study Quasi-Experimental Investigation. *arXiv preprint arXiv:2312.11274*.

[2] Chenxi Dong, Yimin Yuan, Kan Chen, Shupei Cheng, and Chujie Wen. 2023. How to Build an Adaptive AI Tutor for Any Course Using Knowledge Graph-Enhanced Retrieval-Augmented Generation (KG-RAG). *arXiv preprint arXiv:2311.17696*.

[3] Ty Feng, Sa Liu, and Dipak Ghosal. 2024. CourseAssist: Pedagogically Appropriate AI Tutor for Computer Science Education. *arXiv preprint arXiv:2407.10246*.

[4] Zhoumingju Jiang and Mengjun Jiang. 2024. Beyond Answers: Large Language Model-Powered Tutoring System in Physics Education for Deep Learning and Precise Understanding. *arXiv preprint arXiv:2406.10934*.

[5] Ethan Brun and Sebastian Riedel. 2019. Key Phrase Extraction for Generating Educational Question-Answer Pairs. *Stanford CS Paper*. Available at: https://cs.stanford.edu/people/ebrun/papers/Keyphrase_for_education.pdf

[6] Nilay Patel and Jeffrey Flanigan. 2024. Treeformers: Enhancing Hierarchical Understanding in Natural Language Processing. *Unpublished Manuscript, UC Santa Cruz*.