LearnLab: Enhancing Learning with Fun, Interactive Journeys for Better Comprehension, Revision, and Evaluation.

LearnLab: Enhancing Learning with Fun, Interactive Journeys for Better Comprehension, Revision, and Evaluation.

Sai Surya Madhav Rebbapragada

Uday Kiran Dasari

Venkat Akash Varun Pemmaraju

Introduction

Background

In today's digital learning landscape, there's a growing need for tools that can transform static PDF documents into diverse, interactive learning materials. Currently, learners often struggle to effectively extract, retain, and engage with information from PDF documents, leading to suboptimal learning outcomes.

Objective

To develop an intelligent platform that transforms PDF documents into multiple learning formats (audio podcasts, flashcards, quizzes) while automatically generating and distributing content, enabling comprehensive understanding and retention of material through various learning modalities.

Links

Github repository
Github projects

Project Proposal Video

Interaction with App

The user journey begins with uploading or selecting a PDF, which they can explore using querying and summarization features. Users can then generate an audio podcast to conveniently listen to the document's insights or create flashcards for revision. After engaging with the flashcards, users can take an interactive quiz to test their understanding of the material. Once satisfied, they can use the blog generator to produce a professional article summarizing the document's key points. If the generated blog meets their expectations, they can seamlessly post it to blog and social media post sharing their learning with a broader audience.

Core Features

Podcast Generation:

- Automated conversion of academic content to engaging podcast scripts

Interactive Flashcards:

- AI-powered extraction of key concepts
- Flashcards are generated with varying feedback from easy to hard

Content Transformation:

- Generation of platform-specific articles
- Citation and reference management

Interactive Quizzes:

- AI-driven question generation from academic content
- Instant feedback with detailed explanations

Project Overview

Scope

- PDF processing and storage system
- Multi-modal content generation (audio, textual)
- Interactive learning components
- Automated content distribution system
- User authentication and management

- Cloud-based deployment architecture

Stakeholders

- Students and learners
- Educators and content creators
- Researchers and academics
- Professional learners
- Content consumers

Problem Statement

Students and researchers struggle to effectively learn and retain knowledge from academic papers and textbooks, leading to reduced comprehension and inefficient study time. LearningLab solves this by automatically transforming dense academic content into engaging, multimodal learning experiences including podcasts, flashcards, and simplified articles, enabling personalized learning paths that accommodate different learning styles.

Initial Challenges

- Time-consuming Audio Generation which was later reduced by more than 50%
- Resource-intensive content creation process

What we Have Accomplished

- Enhanced learning experience through multiple modalities
- 2-person podcast generation time reduction by around 50% .
- Quiz and Flashcard Generation via Robust RAG Integration
- Time and resource optimization

Methodology

Podcast Generation:

- Our focus is on **audio generation from PDFs**, specifically creating podcasts that summarize and dramatize the document's content.
- The pipeline for podcast generation is as follows:
    1. **Content Cleaning:** Extract relevant text from the PDF using pre-trained large language models (LLMs) for cleaning.

2. **Context and Transcript Creation:** Generate a concise and coherent transcript based on the cleaned content (Step 2 of the process outlined in the provided images).
3. **Dramatization:** Use an LLM to dramatize the transcript for enhanced engagement.
4. **Audio Generation:** Convert the dramatized transcript into audio using tools like **bark/suno** , **eleven labs as such** or a TTS system for final podcast creation.

Flashcard Generation:

- Flashcards are generated using a similar multi-step LLM-driven pipeline:
    1. **Content Cleaning:** Identify key sections and concepts from the processed text.
    2. **Flashcard Formatting:** Organize the generated content into flashcard format, categorized by difficulty levels.
- This ensures a structured and interactive revision tool for users.

Question Generation (Interactive Quiz) :

This feature uses the PDF context to extract context and use LLMs for generating interactive questions in MCQ format.

Blog Generation:

- **Purpose:** The blog generation feature completes the learning cycle by enabling users to share their knowledge and insights gained from the PDF resource. User's can show and share what they are learning and collaborate with anyone with similar interests. (Learning in Public)
- **Share the learning:**
    1. Users can create a professional blog summarizing the document's key points.
    2. The generated blog can be directly shared on **Blogger** for blogging purposes.
    3. Currently we worked on sharing with Blogger & X-API, based on the access we'll go ahead with the blogging and social media platforms
- **Workflow:**
    1. The blog post is generated using the context from the processed PDF
    2. Users can publish their blogs and share it on social media.

By offering podcast generation, flashcards, quizzes, and blog/social media integration, our platform ensures a **complete cycle of learning** for any PDF resource:

1. **Audio Podcast:** Enables users to listen and learn on the go.
2. **Flashcards:** Facilitate quick revisions of key concepts.

3. **Quizzes:** Help test understanding and retention.
4. **Blog and Social Media Sharing:** Allow users to share their learnings and insights with a broader audience.

This holistic approach ensures that learning is not only effective but also engaging and shareable.

Data Sources

- User-uploaded PDF documents
- Pre-processed PDF repository
- Generated content database
- User authentication data

Technologies and Tools

- Backend: Python, FastAPI
- Frontend: Next / Streamlit
- Authentication: JWT
- CI/CD: GitHub Actions
- Cloud Services: GCP (VM, PostgreSQL), AWS (Storage)
- LLM Framework: LangChain,Langgraph
- Orchestration: Apache Airflow
- Containerization: Docker
- AI/ML: Large Language Models for content generation
- Vector Databases for document indexing

Pipeline Design

1. Input Layer:

- PDF upload/selection interface
- Document validation
- AWS S3 storage integration

2. Processing Layer:

- PDF parsing and text extraction
- Document vectorization
- Content chunking and indexing

3. Generation Layer:

- Podcast creation pipeline

- Flashcard generation system
- Quiz generation engine
- Blog post creation and distribution

4. Output Layer:

- Content delivery system
- User interaction interface
- Analytics collection

5. DevOps Pipeline:

- Automated build and testing
- Continuous integration workflow
- Continuous deployment to GCP
- Infrastructure as code management
- Monitoring and logging integration

Data Processing and Transformation

- PDF parsing
- Document chunking strategies for optimal content generation
- Vector embeddings for semantic search
- Multi-modal content transformation
- Automated content distribution workflows

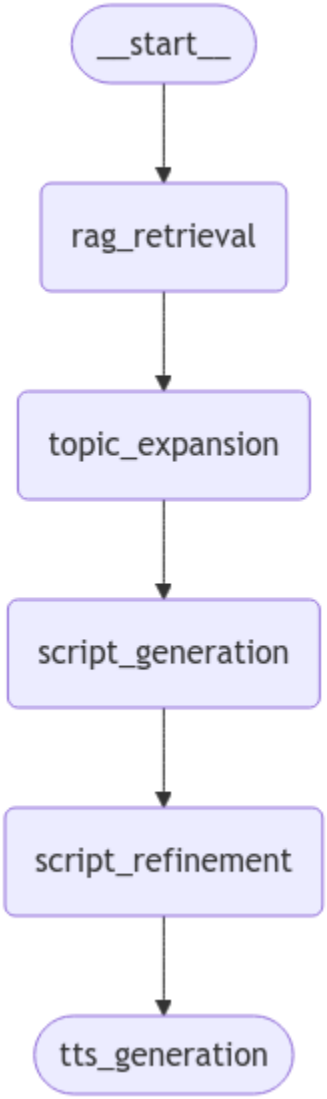
Proof of Concept

In our repository's **POC (Proof of Concept) folder**, you'll find multiple experimental implementations that showcase the evolution of this project. These POCs demonstrate different approaches we tested and iteratively improved upon to arrive at our current solution.

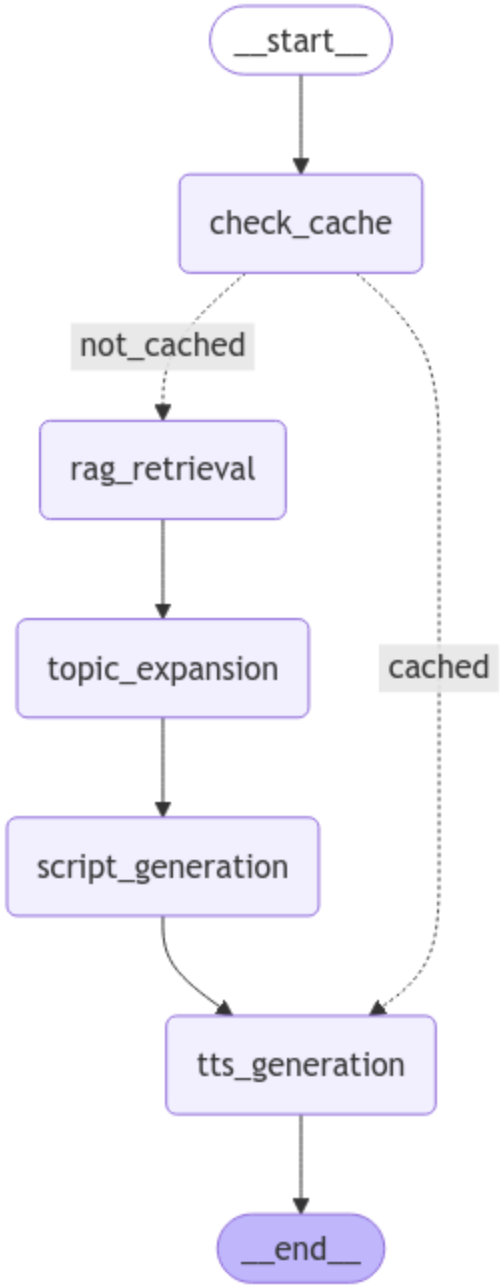
This **RAG-based (Retrieval-Augmented Generation)** system transforms PDF documents into AI-powered podcasts using **LangGraph, OpenAI, and ElevenLabs** technologies. At its core, it employs **OpenAI's text-embedding model for semantic understanding, Pinecone for vector storage, and ElevenLabs for text-to-speech synthesis**. The system has recently integrated the Gemini LearnLM Model, cutting podcast generation time nearly in half from 3 minutes to 1.5-2 minutes.

Our Initial Agents

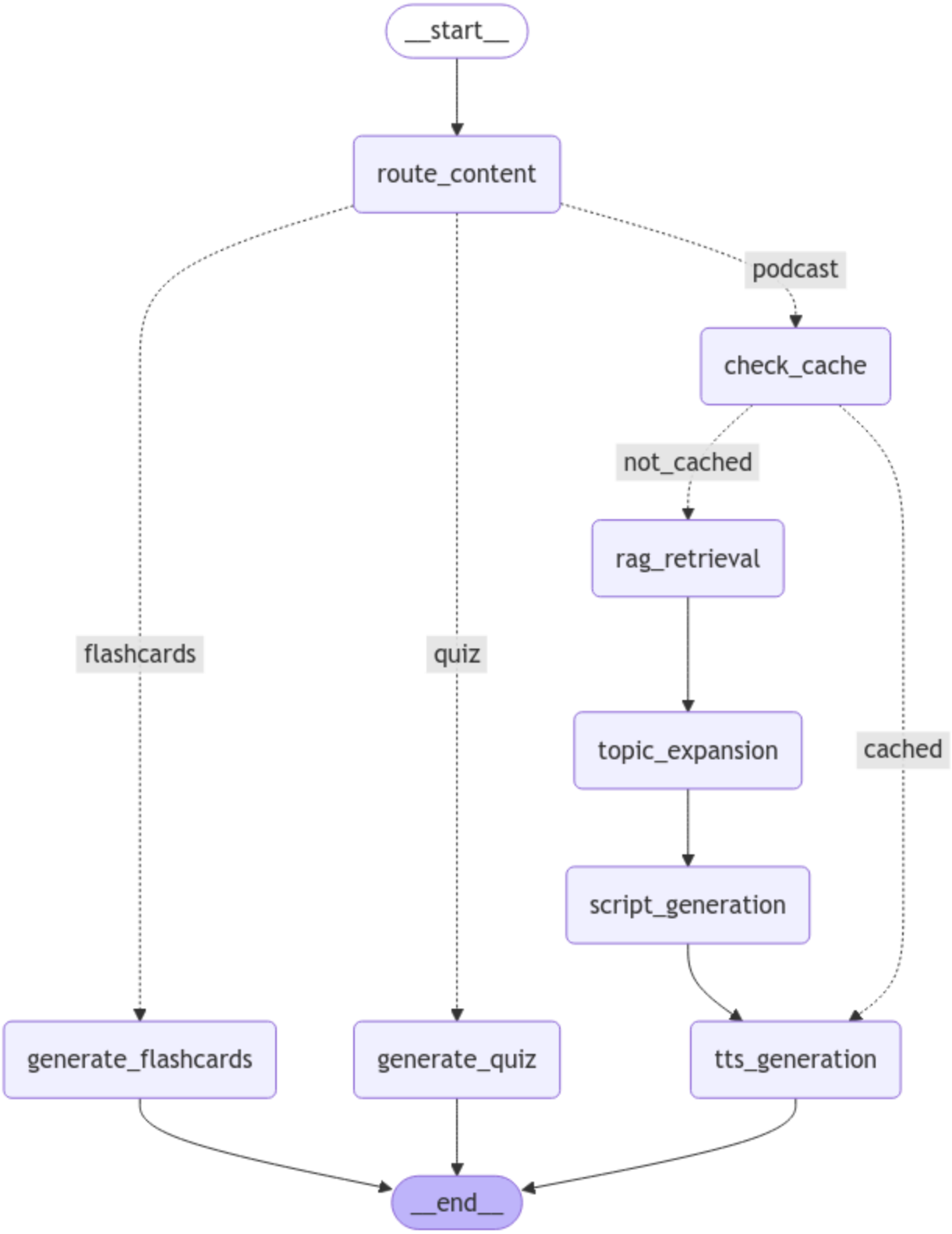
Podcast Agent V1



Podcast Agent v2



We Have then Started Adding our Agents one by one to agentic Architecture


__start__
route_content
podcast
check_cache
not_cached
rag_retrieval
flashcards
quiz
topic_expansion
cached
script_generation
generate_flashcards
generate_quiz
tts_generation
__end__

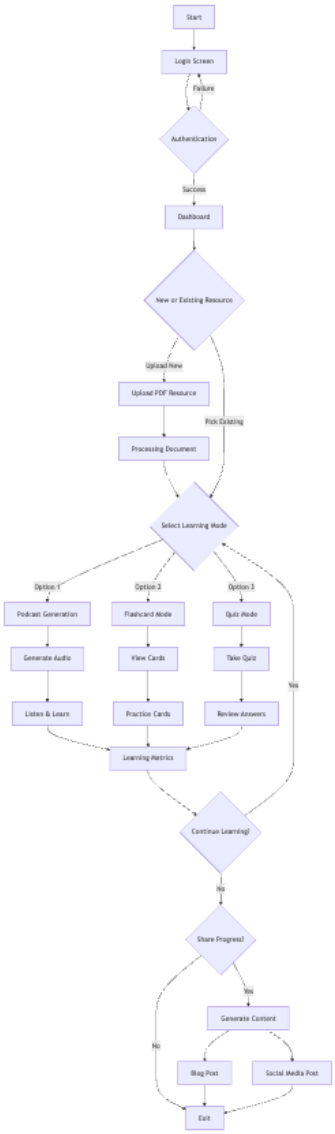
FlashCards: This RAG-based flashcard generation system creates intelligent study materials from PDF documents using OpenAI and LangChain technologies. The system leverages semantic understanding to generate relevant question-answer pairs, with difficulty levels ranging from basic to advanced. Recently integrated with the Gemini Learn LM Model, it significantly improves generation speed while maintaining high-quality educational content.

The architecture consists of a PDFProcessor for document handling, RAGApplication for context retrieval, and FlashcardGenerator for content creation. The system uses spaced repetition principles to organize cards and includes features like difficulty tracking, topic categorization, and progress monitoring. In our repository's POC folder, you can explore various experimental implementations showing the evolution from basic flashcard generation to our current sophisticated system.

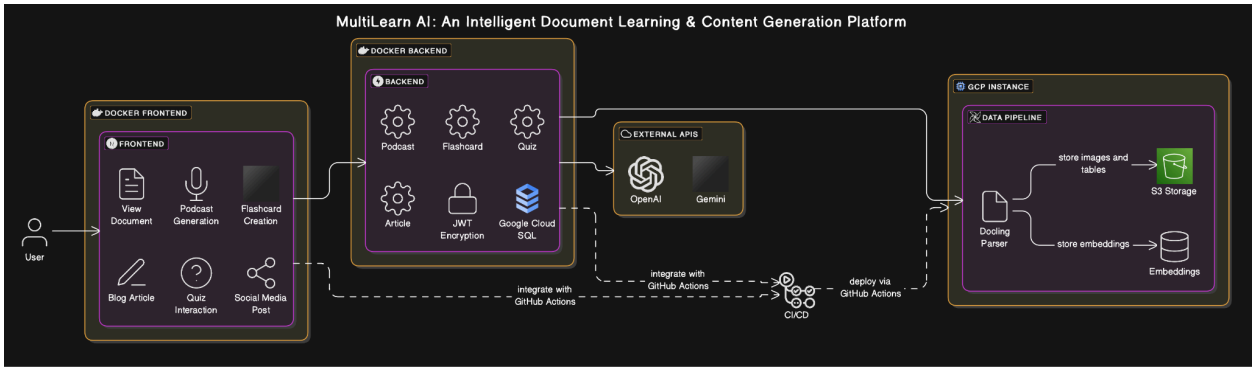
Quiz Generation: Our quiz generation system is a sophisticated RAG-based tool that creates dynamic, multi-format assessments from PDF content using OpenAI and LangChain. It supports various question types including multiple choice, true/false, and short answer, with automatic difficulty scaling. The system employs advanced semantic understanding to ensure questions test comprehension rather than mere recall.

The architecture includes components for document processing, context-aware question generation, answer validation, and automated scoring.

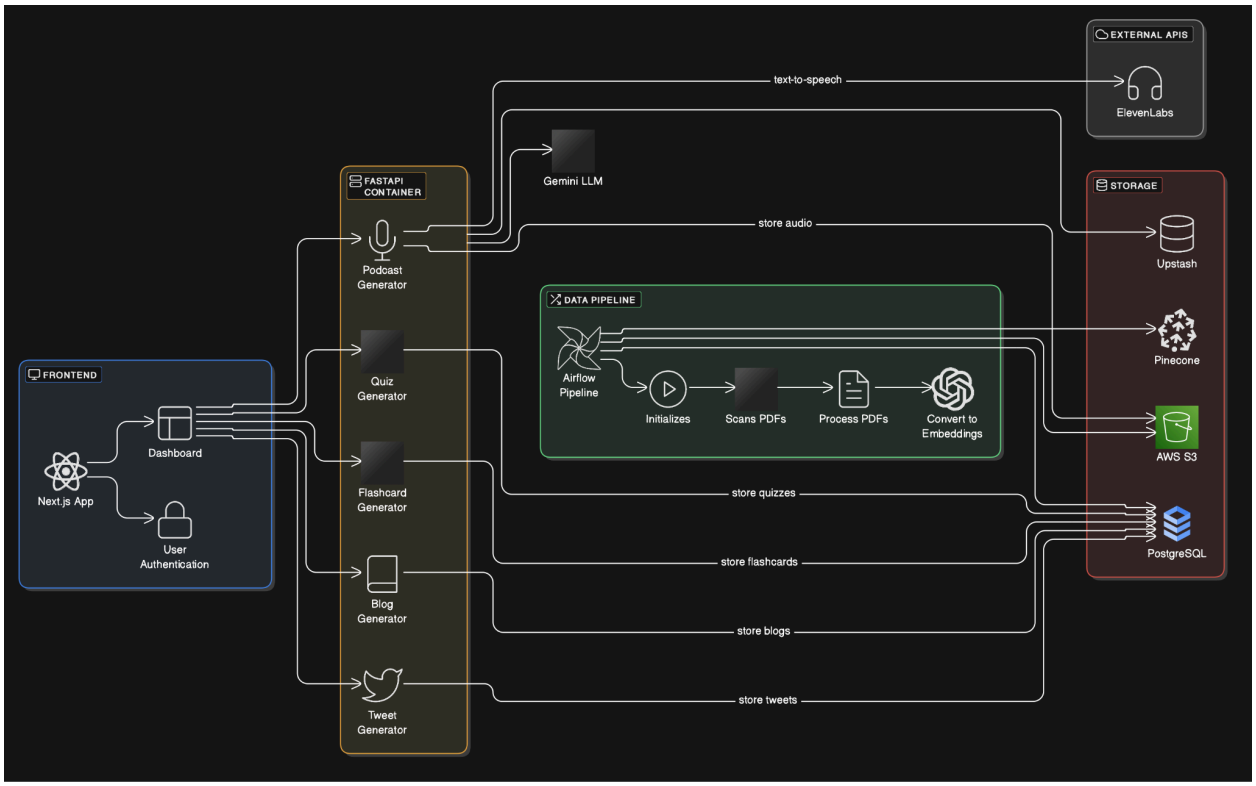
User Flow



Architecture Diagram



Our application is designed as a comprehensive tool for transforming PDF resources into various interactive formats, such as audio podcasts, Blogger articles, flashcards, and quizzes. It combines advanced processing pipelines, interactive interfaces, and generative AI capabilities to deliver an enriched user experience.



Backend and Data Pipeline

The backend, built using **FastAPI**, serves as the core orchestrator. It offers secure API endpoints encrypted with JWT for authentication, leveraging **Google Cloud SQL** for managing user credentials. The pipeline begins with an **Airflow-managed ETL process**, where PDF documents are parsed to extract data. This parsed content is stored in **Cloud Buckets** for durability. The textual embeddings are generated using **OpenAI** and indexed in **Pinecone VectorDB** for efficient querying. Different functionalities—such as podcast generation, summarization, flashcard creation, interactive quiz development, and Blogger article generation—are powered by specialized agents. A **RAG** ensures context-aware interactions for querying and summarizing documents. A separate agent, ContentCurator, handles blog generation and social media posting, providing seamless integration with X-API.

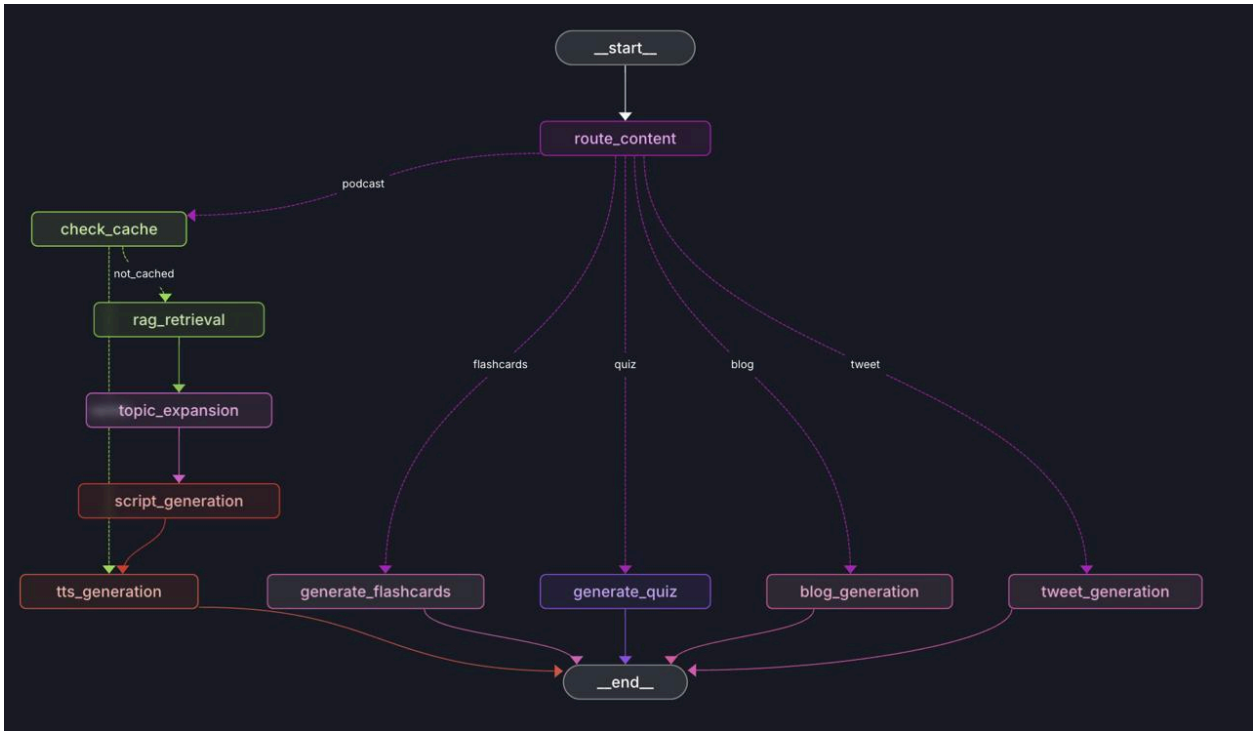
Frontend

The user interface offers a dynamic, intuitive experience. The frontend workflow is structured around the following modules:

1. **Document Interaction**: Users can upload, view, download, and explore the document's content through summaries or direct queries.
2. **Podcast Generation**: Converts PDFs into audio files, offering users a podcast-like experience.
3. **Flashcard and Quiz Creation**: Automatically generates flashcards covering key concepts from basic to advanced levels and interactive quizzes to test knowledge retention.
4. **Social Media Integration**: After Blogger article generation, users can instantly share the blog link on Twitter via the **Social Media Agent .**

By combining Airflow automation, FastAPI endpoints, and a user-friendly frontend, the application streamlines the conversion of PDF knowledge into accessible, interactive formats, helping users better understand and share their learnings.

LearnLab’s Assistant Agent Architecture



1. Podcast Generator Agent

A sophisticated **RAG (Retrieval-Augmented Generation) based podcast generation** system that creates AI-powered podcasts from PDF documents using **LangGraph, Gemini, and ElevenLabs**.

Features

- PDF document processing and indexing using Pinecone vector database
- Advanced semantic embedding using OpenAI's text-embedding-3-small model
- Dynamic text splitting with RollingWindowSplitter for optimal context preservation
- Context-aware podcast content generation using RAG architecture
- Natural conversational script generation with two distinct speakers
- Text-to-speech synthesis using ElevenLabs voices

Semantic Embeddings

Our system leverages OpenAI's text-embedding-3-small model for several key advantages:

1. Semantic Understanding:
   - Captures meaning beyond simple keyword matching
   - Understands context, synonyms, and related concepts
   - Handles technical terminology and domain-specific language effectively

2. Dynamic Text Splitting:
   - Maintains semantic coherence in document chunks
   - Prevents context fragmentation
   - Adapts split sizes based on content complexity

3. Improved Retrieval:
   - Higher accuracy in finding relevant content
   - Better handling of paraphrased concepts
   - Reduced false positives in context retrieval

4. Context Window Management:
   - Overlapping windows for continuous context
   - Optimal chunk sizing for GPT model input
   - Metadata linking for context reconstruction

Pipeline Optimization

The **Gemini Learn LM** model dramatically reduces the podcast generation pipeline from **4-5 minutes to 1.5-2 minutes**, delivering nearly a 50% speed enhancement across topic expansion, script generation, and refinement stages. This optimization maintains high-quality output while significantly accelerating the entire podcast creation process.

2. QuizGenerator Agent:

- Purpose: **Creates interactive quizzes based on document content**
- **Key features:**
  - Generates quiz questions with multiple choice options
  - Each question includes:
    - Question text
    - 4 answer options
    - Correct answer
    - Explanation
    - Difficulty level (easy, medium, hard)
  - Provides quiz metadata like total points and recommended time
  - Includes grading functionality to score user answers
  - Gives detailed feedback on incorrect answers

3. Flashcard Agent:

- Purpose**: Create study flashcards from document content**
- **Key features:**
  - Generates structured flashcard sets
  - Each flashcard has:
    - Front (question/concept)
    - Back (answer/definition)
    - Optional explanation
  - Supports custom instructions and prompt templates
  - Can display flashcards in a readable format
  - Handles error cases gracefully

4. TweetAgent:

- Purpose: **Generates concise tweets from document content**
- **Key features:**
  - Creates technically accurate tweets based on RAG context
  - Ensures tweets are engaging while maintaining factual accuracy
  - Uses a template-based approach for consistent output
  - Returns structured tweet content via the TweetContent model

5. BlogAgent:

- Purpose: **Generates blog posts from document content**
- **Key features:**

- Creates complete blog posts with titles and bodies
- Uses RAG context to ensure accuracy of content
- Maintains technical tone while being engaging
- Structure content with a clear introduction, main content, and conclusion
- Returns structured blog content via the BlogContent model

Common Features Across Agents:
1. All use **LangGraph, LangChain, and ChatOpenAI** for content generation
2. Implement error handling and validation
3. Use Pydantic models for structured data
4. Support **RAG-based content generation**
5. **Can be integrated into a larger workflow**
6. **Use templated prompts for consistent output**
7. Follow similar initialization patterns with API key validation

The agents work together in an integrated system where:
- **The main PodcastGenerator class orchestrates the overall workflow**
- **Each specialized agent (Quiz, Flashcard, Tweet, Blog) can be called independently**
- **All agents share common infrastructure (RAG, caching, storage)**
- The system supports multiple output formats from the same source content

This architecture allows for flexible content generation while maintaining consistency and quality across different output formats.

Risks and Mitigation Strategies

Technical Risks

Blog and Social Media Integration Challenges Risk:

API limitations and platform restrictions may prevent direct automated posting to blogging platforms and social media sites

- **Primary Mitigation**: Implement a draft generation system that creates formatted content ready for manual posting
    - Generate platform-specific content formats (Medium article format, social media post format)
    - Provide copy-paste functionality with proper formatting maintained
    - Include all necessary tags, links, and media elements in the draft

Content Generation Limitations

Risk: Generating 5 minute 2 way person podcasts.

- **Primary Mitigation**: Ensure content is preprocessed to extract **key points and main ideas** to fit within a 5-7 minute timeframe.

Document Processing Constraints

Risk: Processing large PDFs (100+ pages) may Rate Limit Issues

- **Primary Mitigation**: Implemented checks to limit PDF size

Generation Time and Response Caching

Risk: High latency in content generation and repeated processing of the same documents may impact development and testing

- **Primary Mitigation:** Implemented Semantic Cacheing For Podcast Retrieval

Conclusion

LearnLab represents a significant advancement in automated learning content generation and distribution. The platform will transform static PDFs into engaging, multi-modal learning experiences by leveraging cutting-edge AI technologies and cloud infrastructure. The project's success will demonstrate the potential for AI-driven educational tools to enhance learning outcomes while significantly reducing the time and effort required for content transformation and distribution.

References

https://github.com/gabrielchua/open-notebooklm?tab=readme-ov-file
https://github.com/suno-ai/bark
https://github.com/meta-llama/llama-recipes/blob/main/recipes/quickstart/NotebookLlama/README.md
https://github.com/satvik314/educhain
https://www.consillium.app/
https://github.com/5uru/Median?tab=readme-ov-file