LTC AI Learning & Knowledge Platform

Master Architecture & API Specification

---

1. System Overview

The **LTC AI Learning & Knowledge Platform** transforms raw learning materials (videos, PDFs, transcripts) into structured, searchable, AI-powered learning assets using **Retrieval-Augmented Generation (RAG)**.

User Roles

- **Student** – Consume courses, track progress, interact with AI assistant
- **Instructor** – Upload and manage course content, trigger AI processing
- **Admin** – Platform-level management (future extension)

---

2. Architecture Overview

High-Level Flow

1. Instructor uploads learning content (video/PDF)
2. Content is stored in Azure Blob Storage
3. AI Service processes content (transcription, chunking, embeddings)
4. Embeddings are stored in Pinecone (vector DB)
5. Structured assets (summary, quiz, recap) are generated
6. Students interact with content via AI chat and assessments

Core Technologies - Frontend: Web / Mobile (future) - Backend: FastAPI (Core + AI services) - Auth & DB: Firebase (Auth + Firestore) - Storage: Azure Blob Storage - Vector DB: Pinecone - LLMs: OpenAI / Azure OpenAI

---

3. API Contracts (FastAPI)

All endpoints require **JWT authentication** unless explicitly stated.

Service Separation

- **Core Service** – Users, courses, modules, enrollment, progress
- **AI Service** – Processing, RAG, summaries, quizzes, recaps

---

AUTH SERVICE

POST `/auth/login`

Purpose: Authenticate user and issue JWT

Roles: Student, Instructor, Admin

Request

```
{
  "email": "user@email.com",
  "password": "user_password"
}
```

Response

```
{
  "token": "jwt_token_here",
  "user": {
    "id": "user_id",
    "role": "student | instructor | admin"
  }
}
```

Internal Logic - Firebase validates credentials - JWT issued and returned

---

CORE SERVICE (Platform Layer)

POST `/core/upload`

Upload video or PDF to Blob storage

Role: Instructor

Response

```
{ "file_url": "blob_storage_url" }
```

POST `/core/courses`

Create a new course

Role: Instructor

```json
{
  "title": "AI Foundations",
  "description": "Introduction to AI"
}
```

GET `/core/courses`

List available courses

Role: All authenticated users

POST `/core/courses/{course_id}/modules`

Add module to course

Role: Instructor

```json
{
  "title": "Introduction to RAG",
  "video_url": "blob_url",
  "order_index": 1
}
```

Flags initialized: - `has_summary = false` - `has_quiz = false` - `is_vectorized = false`

POST `/core/courses/{course_id}/enroll`

Enroll student in course

Role: Student

POST `/core/courses/{course_id}/modules/{module_id}/complete`

Mark module as completed

Role: Student

AI SERVICE (Intelligence Layer)

POST `/ai/process`

Trigger AI processing pipeline

Role: Instructor

Pipeline: - Transcript extraction - Cleaning & chunking (500–800 tokens) - Embedding generation - Store vectors in Pinecone - Generate summary & quiz - Update Firestore flags

---

POST `/ai/chat`

Ask questions over course content (RAG-powered)

Role: Student

```
{
  "course_id": "...",
  "question": "Explain embeddings"
}
```

GET `/ai/assets/summary/{module_id}`

Fetch module summary (Markdown)

GET `/ai/assets/quiz/{module_id}`

Fetch AI-generated quiz

POST `/ai/submit-quiz`

Submit quiz and receive evaluation

```
{
  "module_id": "...",
  "answers": [{ "question_id": "q1", "selected_option": "A" }]
}
```

Returns score, correct answers, weak concepts

POST `/ai/generate-recap`

Generate audio/podcast recap

Role: Student

---

4. Firestore Schema Summary

users/{user_id}

- email
- role
- enrollments/{course_id}
- enrolled_at
- progress
- completed_modules

courses/{course_id}

- title
- instructor_id
- is_published
- modules/{module_id}
- title
- video_url
- order_index
- has_summary
- has_quiz
- is_vectorized

ai_assets/{asset_id}

- type (summary | quiz | audio)
- linked_course_id
- linked_module_id
- transcript_text
- summary_markdown
- pinecone_vector_id
- model_used

---

5. Standard Error Format

```
{
  "status": "error",
  "message": "Description of issue",
  "code": "ERROR_CODE"
}
```

6. Notes for Team Implementation

- All AI jobs must be async
- Core service must stay deterministic
- AI outputs must be traceable to source modules
- Version all prompts and models used

End of Document