Machine Learning Project Documentation

Deployment

1. Overview

The deployment phase transforms the multi-pipeline Lecture Companion system into a production-ready application by containerizing all components (Next.js frontend, FastAPI backend, Whisper ASR, Gemini LLM endpoints, and the planned CEFR classifier) and orchestrating them using Docker Compose. The deployed backend exposes the ASR, translation/simplification, RAG, and classifier services through stable REST endpoints, enabling real-time lecture processing across YouTube ingestion and user uploads. The architecture diagram shows a two-container workflow: the frontend, hosted on Vercel communicates via HTTPS with a backend hosted on Render. The backend coordinates transcription, file parsing, translation/summarization, RAG embedding/indexing (via ChromaDB/FAISS), caching, and CEFR inference. This deployment process ensures the ML pipelines operate reliably with scalable GPU support, consistent API interfaces, and persistent storage for transcript chunks and cached responses.

2. Model Serialization

Because the system integrates both pretrained foundation models (Whisper ASR, Gemini 2.5 Flash) and a traditional ML model (Pipeline 3: CEFR classifier), only the CEFR classifier requires explicit serialization. The classifier which is trained using TF-IDF character n-grams and an SGD Logistic Regression model, would be exported using joblib or pickle, producing two serialized artifacts:

- classifier.pkl - serialized sklearn model
- vectorizer.pkl - serialized TF-IDF character n-gram vectorizer

Serialization enables the backend to load the model once at startup and perform lightweight inference during runtime. The format was chosen for:

- Low storage overhead (MB-scale)
- Fast deserialization suitable for stateless API servers
- Compatibility with FastAPI + Docker environments

No serialization is needed for Whisper or Gemini: Whisper is invoked as a CTranslate2 runtime (loaded from weight folders), while Gemini is accessed through hosted APIs, as described in the Implementation Plan.

3. Model Serving

The deployed system serves three model families simultaneously:

(a) Whisper ASR (Faster-Whisper)

Running inside the backend container, Faster-Whisper serves transcription requests by streaming audio segments and performing batched inference. GPU-enabled nodes on Render can accelerate inference, reducing latency for long lectures.

(b) Gemini 2.5 Flash for Translation, Simplification & Summaries

These are served indirectly through FastAPI endpoints that wrap Gemini API calls. Because the LLM is not self-hosted, the backend handles batching, prompt construction, safety checks, caching, and fallback logic (e.g., if summaries or translations already exist).

(c) CEFR Vocabulary Classifier

The serialized classifier is loaded into memory on backend startup. It serves as a lightweight inference module that receives token lists and returns CEFR labels (A1–C2). The deployment of this feature mirrors the translation pipeline: predict → annotate → return structured metadata.

3.1 Deployment Platforms

Frontend: Vercel (static + serverless rendering)

Backend: Render (Docker-based service with persistent environment variables and GPU provisioning if needed)

Database/Vector Store: ChromaDB / local persistent storage; optional pgvector for production

This multi-platform approach aligns with the Implementation Plan's emphasis on modular, cloud-agnostic deployment.

4. API Integration

The FastAPI backend exposes a comprehensive set of endpoints to allow seamless communication between the models and the client interface. Endpoints follow structured JSON schemas to ensure reproducibility and predictable error handling.

Core transcription endpoints

Method	Endpoint	Description
POST	/api/transcribe/youtube	Ingest YouTube URL → fetch captions or fall back to Whisper ASR
POST	/api/transcribe/upload	Upload .pdf, .txt, .srt, .vtt → parse into transcript text

LLM endpoints (Translation & Summarization)

Method	Endpoint	Description
POST	/api/llm/translate	Translate English → Burmese
POST	/api/llm/summarize	Generate English & Burmese summaries

POST	/api/llm/transform/translate-and-summarise	Combined pipeline call for efficiency

RAG Endpoints

Method	Endpoint	Description
POST	/api/llm/rag/index	Build embeddings + vector index from transcript/summaries
POST	/api/llm/rag/query	Ask grounded questions in English or Burmese
GET	/api/llm/rag/stats	Retrieve index metadata
GET	/api/llm/rag/chunks	Return all indexed chunks with timestamps
DELETE	/api/llm/rag/index	Clear the vector index

CEFR Classifier Endpoints

Method	Endpoint	Description
POST	/api/cefr/predict	Classify words into CEFR levels using serialized sklearn model
POST	/api/cefr/annotate	Return transcript annotated with CEFR difficulty + glossary hints

5. Security Considerations

Security follows the principles outlined in the Implementation Plan:

Authentication & Authorization

- All Gemini API calls require secure server-side API keys stored in encrypted environment variables.
- Backend endpoints are isolated from the public except via HTTPS interfaces; no direct model access is exposed.

Data Handling & Privacy

- User-uploaded transcripts are processed in cached memory where possible and not stored long-term, aligning with the project's ethical commitment to privacy.
- RAG embeddings and caches store only lecture content, never personal data.

Network & Transport Security

- All communication between frontend ↔ backend ↔ external APIs occurs via HTTPS.
- Docker containers limit exposed ports and isolate services.

Injection & Prompt Safety

- The backend constructs prompts with strict formatting, enforcing rules that prevent the LLM from hallucinating external content.
- RAG queries automatically trigger refusal messages for off-topic questions, as validated in the refinement report.

6. Monitoring and Logging

A combination of application logs, performance metrics, and qualitative usage signals is used to monitor the deployed model in the backend.

System Logging

- FastAPI middleware logs request/response cycles, including latency for ASR, translation, and RAG calls.
- Whisper and embedding operations include timing logs for real-time performance monitoring.

Model Performance Monitoring

Key metrics tracked include:

- ASR latency & segment WPM proxy (from Implementation Plan)
- Translation expansion ratio (Burmese/English length)
- RAG grounding statistics (ROUGE-L recall, cosine similarity grounding scores, context diversity)
- CEFR classifier accuracy

Alerting Mechanisms

Automatic alerts shown in the backend log when:

- RAG index becomes corrupted or empty
- Whisper inference exceeds time thresholds
- Gemini API returns rate-limit or quota warnings (Metrics available in google AI studio)
- CEFR classifier fails to load serialized artifacts

Observability Tools (Metrics available on their respective hosting platforms)

- Render built-in logs for container-level monitoring
- Vercel analytics for frontend request volume