Final Year Project Handoff Document

Project: Prompt-first AI Video Editor for Short-Form Creators
Owner: Lokesh
Date: 2026-02-15
Scope: Full technical handoff for continuation in future chats.

1. Problem Statement

Creators spend hours converting long videos into Shorts/Reels/TikToks.
This project aims to reduce that into minutes using:

1. Prompt-first editing controls.
2. Automatic timeline operations.
3. Faster preview/export rendering.
4. Auto-ingest and (next) auto-analysis/auto-clipping.

2. External References and Similar Tools

These are the benchmark products/repos provided and researched:

1. Paper: https://arxiv.org/abs/2509.16811
2. Paper HTML: https://arxiv.org/html/2509.16811v1
3. OpusClip: https://www.opus.pro/
4. OpusClip API: https://www.opus.pro/api
5. Magic Clips (Riverside): https://riverside.com/magic-clips
6. OpenShorts: https://github.com/mutonby/openshorts
7. ClippedAI: https://github.com/Shaarav4795/ClippedAI
8. AI-short-creator: https://github.com/shreesha345/AI-short-creator
9. Vinci Clips: https://github.com/tryvinci/vinci-clips

3. What We Are Building (Positioning)

Target product category:

1. Opus-style auto repurposing pipeline.
2. Prompt-driven editor for fast manual + AI hybrid control.
3. Research-aligned semantic/agentic layer (next phases) for narrative-aware short generation.

Differentiation target:

1. Faster iteration loop with transparent stage events.
2. Prompt + timeline dual control.
3. Open, modular architecture suitable for FYP demo and future extension.

4. Current Stack

Backend:

1. FastAPI + SQLModel + SQLite/Postgres compatible.
2. FFmpeg/FFprobe render pipeline.
3. yt-dlp URL ingestion pipeline.
4. Threaded worker queues for render + ingest jobs.

Frontend:

1. React + TypeScript + Vite.
2. Prompt console, media bin, timeline, inspector, render jobs panel.

Core binaries:

1. ffmpeg
2. ffprobe

3. yt-dlp

5. Implemented Features (Working)

Editor and timeline:

1. Trim, split, delete.
2. Merge/join and reorder.
3. Move/ripple operations.
4. Transitions.
5. Text overlays.
6. Clip speed, rotate, flip, crop.
7. Basic color adjustments and presets.
8. Audio controls: volume, mute, fade in/out, keyframes, track volume/mute/solo.
9. Undo/redo and history.

Media and render:

1. Media upload and list.
2. Render preview and export jobs.
3. Job queue and dedup window for active jobs.
4. Stage-wise job events endpoint.
5. URL ingestion endpoint (yt-dlp) into media library.

Prompt console:

1. Prompt parse/apply flow for supported commands.
2. Improved fallback for unsupported text by clickable suggestions.
3. Audio fade prompt parsing now targets audio track clip indexing.

6. Major Work Completed in This Build Cycle

6.1 Queue and orchestration hardening

1. Added render queue workers with configurable concurrency.
2. Added ingest queue workers with configurable concurrency.
3. Startup and shutdown now start/stop both worker pools.

6.2 Job observability

1. Added `JobEvent` model and event API.
2. Added stage transitions for render jobs:
   queued -> running -> build -> render -> complete/failed
3. Added stage transitions for ingest jobs:
   queued -> running -> download -> probe -> register -> complete/failed

6.3 Ingestion pipeline

1. Added `POST /api/v1/ingest/url`.
2. URL validation and yt-dlp download service.
3. Media probing and registration into project media assets.

6.4 Frontend updates

1. Render Jobs panel now shows latest stage events.
2. Prompt suggestions are clickable.
3. Media Bin now supports URL ingest input and queue button.

6.5 Audio fade bug fixes

1. Fixed render filter chain ordering for background audio so `afade` runs before `adelay`.
2. Fixed clip index lookup to support `track_kind` when resolving `clip: N`.
3. Updated prompt parser for `audio fade ...` to include `track_kind=audio`.

7. API Inventory (Current)

Projects:

1. `POST /api/v1/projects`
2. `GET /api/v1/projects`
3. `GET /api/v1/projects/{project_id}`
4. `POST /api/v1/projects/{project_id}/undo`
5. `POST /api/v1/projects/{project_id}/redo`

Media:

1. `POST /api/v1/media/upload`
2. `GET /api/v1/media?project_id=...`

Ingest:

1. `POST /api/v1/ingest/url?project_id=...`

Prompt:

1. `POST /api/v1/prompt/parse`
2. `POST /api/v1/prompt/apply?project_id=...`

Timeline:

1. `POST /api/v1/timeline/operations?project_id=...`
2. `GET /api/v1/timeline/history?project_id=...`

Render/jobs:

1. `POST /api/v1/render/preview?project_id=...`
2. `POST /api/v1/render/export?project_id=...`
3. `GET /api/v1/jobs/{job_id}`
4. `GET /api/v1/jobs/{job_id}/events`

Health:

1. `GET /health` returns `ffmpeg`, `ffprobe`, `yt_dlp` availability.

8. Data Models (Important)

Key tables:

1. `Project`
2. `Timeline`
3. `TimelineVersion`
4. `OperationRecord`
5. `MediaAsset`
6. `Job`
7. `JobEvent`

9. Environment Variables (Current)

From `backend/.env.example`:

1. `APP_NAME`
2. `DATABASE_URL`
3. `UPLOAD_DIR`
4. `RENDER_DIR`
5. `TMP_DIR`
6. `ALLOWED_ORIGINS`
7. `MAX_UPLOAD_MB`

8. `STORAGE_BACKEND`
9. `FFMPEG_BIN`
10. `FFPROBE_BIN`
11. `YT_DLP_BIN`
12. `MAX_CONCURRENT_RENDER_JOBS`
13. `MAX_CONCURRENT_INGEST_JOBS`

10. Validation Status (As of 2026-02-15)

Automated checks passed:

1. Backend tests: `20 passed`
2. Frontend build: success

Manual smoke checks passed:

1. Preview render queue lifecycle and event stream.
2. Ingest URL job queue/event flow through integration tests (mocked download in test).

11. Known Gaps / What Is Still Left

Core product gaps:

1. No transcription pipeline yet.
2. No clip candidate detection/ranking yet.
3. No auto 9:16 smart reframing pipeline yet.
4. No semantic index or planner/retrieval agent layer yet.
5. No batch clip generation endpoint yet.
6. No social publish integrations yet.

Engineering gaps:

1. No cancellation endpoint for queued/running jobs.
2. No retry endpoint for failed jobs.
3. No persistent workflow checkpoints beyond current job/event model.
4. Lifespan API refactor pending (current FastAPI `on_event` deprecation warnings).

12. Exact Next Execution Plan

Phase N1: Analysis foundation

1. Add `POST /api/v1/analyze?project_id=...`
2. Implement analysis job kind and stages:
   queued -> running -> asr -> segment -> index -> complete/failed
3. Add transcript and segment tables:
   `transcripts`, `segments`

Phase N2: Auto-clipping MVP

1. Build heuristic candidate generator from transcript + pauses + energy proxies.
2. Add `POST /api/v1/clips/suggest`.
3. Add batch render endpoint for top-N suggestions.

Phase N3: Prompt-to-clip planning

1. Add prompt intent schema for audience/tone/topic/length.
2. Add ranking and retrieval alignment layer.
3. Expose intermediate artifacts in UI.

13. How To Continue Quickly (Commands)

Backend:

```bash
cd backend
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
cp .env.example .env
uvicorn app.main:app --reload --port 8000
```

Backend tests:

```bash
cd backend
source .venv/bin/activate
pytest -q
```

Frontend:

```bash
cd frontend
npm install
npm run dev
```

14. New Chat Continuation Prompt (Copy-Paste)

Use this in a new chat to resume fast:

```text
Continue this repo from /home/lokesh/ai video editor.
Read docs/fyp-full-handoff-2026-02-15.md first, then execute Phase N1.
Implement /api/v1/analyze with queued job stages (asr, segment, index), add
transcript+segment models, add tests, and wire frontend status view.
Do code changes directly and run backend/frontend validations before final
summary.
```

15. File Map for Key Context

Core docs:

1. `docs/fyp-full-handoff-2026-02-15.md`
2. `docs/fyp-competitive-benchmark.md`
3. `docs/fyp-product-roadmap.md`

Backend core:

1. `backend/app/main.py`
2. `backend/app/jobs.py`
3. `backend/app/render_service.py`
4. `backend/app/ingest_service.py`
5. `backend/app/timeline_service.py`
6. `backend/app/prompt_parser.py`
7. `backend/app/models.py`
8. `backend/app/schemas.py`
9. `backend/app/routers/`

Frontend core:

1. `frontend/src/App.tsx`
2. `frontend/src/lib/api.ts`

3. `frontend/src/types.ts`

This handoff is intended to be the single source of truth for project continuation from future chats.