EduTrack: An AI-Powered System for Classroom Monitoring and Content Automation

A Technical Report on the Architecture, Implementation, and Capabilities of a Dual-Function Educational Platform

P. Sri Sathwik
Piyush Prakash Wakpaijan
Nishant Sheoran

GitHub Repository

Demo Video

— Contents

— Abstract

This report explains how, **EduTrack** an AI-driven system forms a smart, real-time feedback loop between classroom interaction and content coursework. The project will solve two related issues: the inability to assess the student comprehension objectively in real-time, and the necessity of specific additional aids. EduTrack offers a combined solution: **(1)** a computer vision module in real time video in the classroom to detect what techniques produce the most disengagement or misunderstanding among the students. **(2)** The results are used to automatically initiate an explicit content pipeline producing a focused, cartoon educational video and a PDF transcript covering the identified difficult topic. Attributable to its dynamic connection between engagement analytics and content generation, EduTrack would not only furnish educators with practical information on how well their attendees are receiving them but would also resolve the generation of remedial content, thus creating a more competent and timely learning service both in online and offline contexts.

— Introduction

1.1 Problem Domain

Student engagement is very important in determining the effectiveness of classroom instructions. Nevertheless, in physical and virtual classrooms teachers are at a disadvantage because they usually do not have the tools to measure the level of attention in a whole classroom in real-time in an unbiased way. Customary observation is personalized and non-exponential. Since the beginning of the school year, the need in supplementary digital learning materials has grown sharply, which puts the burden on the teachers to create videos, notes, and other materials, which is time-consuming and involves some technical skills. This project won work in these interrelated problems by developing a technological support to the educator.

1.2 Project Vision and Objectives

The primary objectives of the EduTrack project are:

- It is to have a stable, real-time computer vision application that monitors and measures the engagement of students.
- In order to transcribe the lectures in real-time and align the timestamps with engagement data to **automatically determine topics** that are associated with low student engagement.
- To create and develop an auto-piloted pipeline that is **prompted by the indication of low engagement** to produce desired, multi-modal educational materials (video and PDF) on precise challenging topics.
- To draw a centralized dashboard that shows the educators the analytics of engaged topics and allow them to be able to see the automatically generated support materials.
- Develop the system out of service-oriented architecture and modularity to achieve scalability and maintainability of the feedback loop.

1.3 Scope and Limitations

The EduTrack project scope is clearly stated, as it has to offer two main functionality options: the engagement analysis and content automation. The module of engagement is architected to handle one video signal on one set camera direction only. It is not even multi-camera compatible yet and cannot track people who are out of the camera range. It relies only on visual appearance (by using the position of the head and facial expression) and does not involve any audio-analysis as a way of engagement (e.g. student speech patterns).

The content generation module is modeled in order to work with pure, single-speaker audio files. It has not been designed to work in noisy environments. It also does not perform well in a multi-speaker scenario. The resulting animations are more abstract, more conceptual, and aim to demonstrate a point rather than draw a realistic setting. The system is proof of concept only and has not been optimized to run big and in parallel in the production system.

— System Architecture

2.1 Architectural Overview

EduTrack is developed in a form of service-based architecture, including three different components communicating through APIs. The modular design guarantees clean separation of concerns where the various parts of the system can be developed, tested, and deployed independently.

- **Engagement Monitoring Backend:** A Python service that given a video stream, produces a stream of structured JSON data where each JSON object corresponds to one form of engagement that can be summarized.
- **Content Generation Backend:** A FastAPI service written in Python that provides endpoints to process any audio and create its content.
- **Frontend Dashboard:**A web application built on Next.js that will serve as the so-called user-facing dashboard, and which consumes data and makes requests to the backend services.

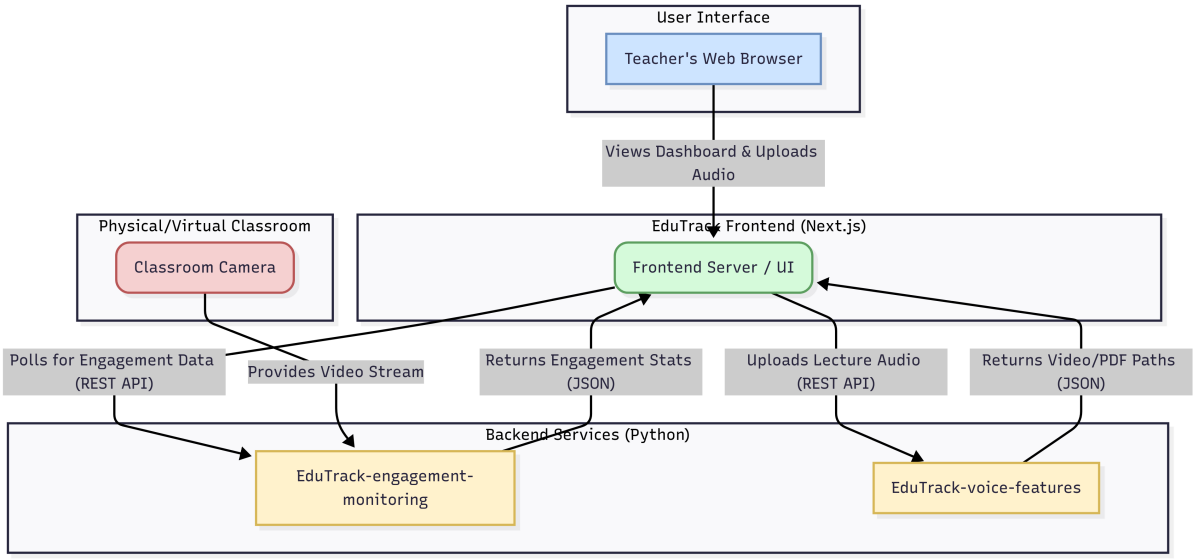

Figure 1: High Level System Architecture Diagram that shows the communication between the three main services.

2.2 Technology Stack

The technology stack, which is used in the project, is new and efficient to the requirements of each module.

Engagement Monitoring Python, OpenCV, OpenVINO, ONNX Runtime, Deep-Sort-Realtime.

Content Generation Python, FastAPI, OpenAI (Whisper, GPT), Manim, MoviePy, gTTS.

Frontend Dashboard Next.js, React, TypeScript, Tailwind CSS, Recharts.

2.3 Database Schema

A PostgreSQL database is used to handle the data persistence level of the system. The schema will help to effectively store the engagement events and associate them with transcribed material. Important tables are:

- **Sessions:** Stores metadata for each class session, e.g., `session_id`, `start_time`, `video_source`.
- **EngagementEvents:** A time-series table logging disengagement events, e.g., `event_id`, `session_id`, `timestamp`, `event_type` ('look_away', 'confused'), `duration_frames`.
- **TranscriptChunks:** Stores time-stamped text chunks from the audio transcription, e.g., `chunk_id`, `session_id`, `start_time`, `end_time`, `text`.
- **FlaggedTopics:** Links engagement events to transcripts, e.g., `topic_id`, `session_id`, `triggering_event_id`, `transcript_chunk_id`, `generated_content_url`.

2.4 Data Flow and Communication Protocols

Rest communication is done between the backend services and the frontend dashboard through RESTful APIs. Nevertheless, the main innovation of the system is an auto-regulatory feedback between the backend services themselves.

The Core Feedback Loop:

1. Engagement Monitoring service processes the video stream to produce engagement scores (e.g., % disengaged) and include date/time and seed it into the `EngagementEvents` table.
2. At the same time, the audio of the lecture is transcribed using the Content Generation service, again including timestamps, and fills the `TranscriptChunks` table.
3. A component of orchestrator queries the database periodically. It finds the episodes of high-level and a sustained disengagement and marries the timestamps within the transcribed text to help determine what current subject of discussion.
4. After flagging a topic a new row in the `FlaggedTopics` table is inserted and an generate request corresponding to the respective text fragment is inserted in the API call automatically with the respective text.

5. Afterward, Content Generation service will generate an additional video and PDF only about that problematic topic and update `FlaggedTopics` entry with the address of generated content.

— Module Deep Dive: Engagement Monitoring

This module acts as the backbone of the system that is the analytical part as it converts raw video into the structured and meaningful data. It works as a series of computer vision processes.

3.1 Face Detection

The pipeline commences by recognizing the entire faces of the students in a particular video frame. This was made possible by using a YOLOv8-face model, which was converted to the ONNX format, i.e. compatibility with frameworks. This model is additionally optimized on Intel hardware with the help of the OpenVINO™ toolkit to achieve maximum inference speed.

3.2 Face Tracking

After detecting faces the tracker `DeepSortFaceTracke` assigns a distinct nominal and constant `track_id` to each person. This would be important especially in examining the behavior of a student over a long period which qualifies the system to determine long term state of engagement in a given student.

3.3 Attention and Emotion Analysis

For each tracked face, two key analyses are performed:

1. **Head Pose Estimation:** The system estimates the yaw and the pitch of the students head using an OpenVINO model. Strong inattentiveness is a remark of great deviation.

2. **Emotion Recognition:** One more OpenVINO model identifies facial expressions to evaluate the emotional reactions, such as confusion, surprise, or happiness.

3.4 OpenVINO Optimization for Real-Time Performance

All computer vision models are intensively optimized with the Intel OpenVINO toolkit in order to support low-latency inference needed to process video in real-time on commodity GPU hardware. The process is how follows:

1. The first stage is to convert models out of their native framework (e.g. PyTorch) to a universal ONNX format.

2. Then, the ONNX models are compared to the OpenVINO Model Optimizer, which compiles them into its internal Intermediate Representation (IR) format, that is, the files with the file patterns `.xml` and `.bin`.

The process makes the models suitable to run as lightning quickly as possible on Intel Central Processing Unit (CPUs) and integrated Graphics (iGPUs) to ensure that the EduTrack system uses less specialized, inexpensive graphics cards.

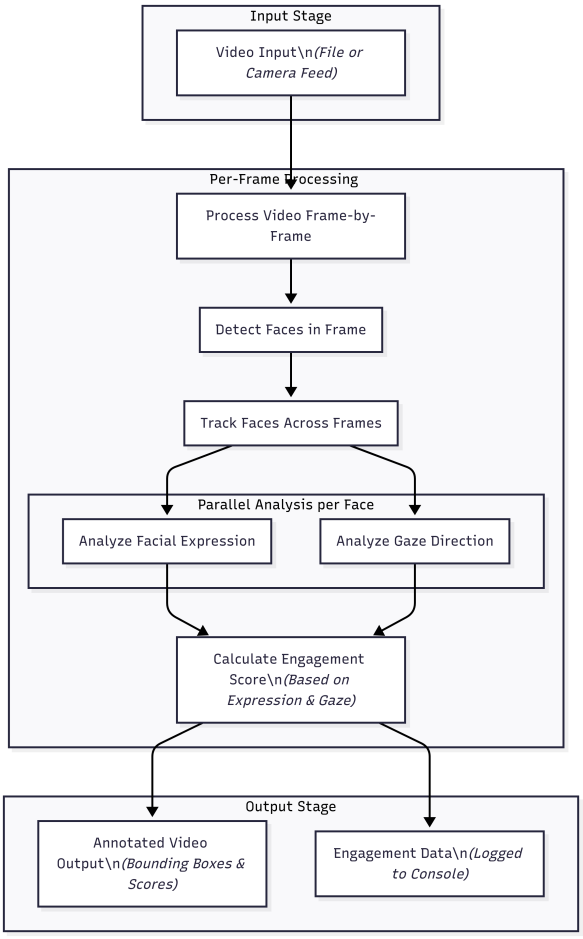

Figure 2: The Engagement Analysis Pipeline.

3.5 Configuration and Thresholds

The sensivity of the system is regulated upon several thresholds:

`YAW_THRESHOLD`: sets the threshold in degrees where the angle of left-to-right head swing is concerned above which the student is claimed to be looking away.

PITCH_THRESHOLD: Determines the highest degree of up down movement of the head. This plays a significant part in realizing whether a student is down with a phone or sleeping.

DISSOCIATION_FRAME_THRESHOLD: To prevent treating brief looks away as a case of disengagement only declared to be such after a certain amount of time, this parameter determines how many frames in a row a student must be registered with a status of ”Looking away” or ”Confused” before his/her status will officially become ”Disengaged.” This makes the analysis to have temporal smoothing effect..

— Module Deep Dive: Content Generation

This module is used as an automated aid to production: a teacher lecture will be transformed into a full-scale digital education package.

4.1 API and Transcription

It starts through an appeal to the FastAPI backend. An audio recording is uploaded and transcribed into raw text with the help of OpenAI whisper model.

4.2 AI-Powered Scripting and Prompt Engineering

A GPT-4-based script generator takes the raw transcript. So that the output is trustworthy and organized, a definite prompt engineering approach is applied. The model is expected to perform as an actual task of becoming an `Educational Content Creator`: it receives a JSON object with the following keys: `title`, `summary`, `key_points` (an array), `visual_cues` and a cleaned `full_text`. This well-organized prompting makes the LLM less of a text generator but rather a predictable data processing element.

4.3 Narration and Animation

The processing sources of the structured script drive two parallel subsystems: a pure audio narration is produced through TTS engine (gTTS), and the `visual_cues` is fed to the `VideoAnimator` class applying the Manim library to create an animation in the style done by 3Blue1Brown.

4.4 Final Composition

The last step takes the output of the previous one, an audio file and a video one, and puts them together into a synchronous MP4 file with `VideoMerger`, a MoviePy library-controlled program. The script is also converted into a professional PDF.

Component	Performance (FPS)
Face Detection (YOLOv8)	∼35-40 FPS
Head Pose Estimation	∼60+ FPS
Emotion Recognition	∼55-60 FPS
Full Pipeline (Single Stream)	∼**28-32 FPS**

Figure 5: Engagement module performance on a test system with an Intel Core i7-12700H CPU.

Component	Processing Time
Transcription (Whisper)	∼0.1x audio length
Script Generation (GPT-4)	5-10 seconds
Narration (gTTS)	∼1.2x audio length
Animation & Render	15-30 seconds per minute

Figure 6: Content generation module performance benchmarks for a 1-minute audio segment.

— Module Deep Dive: Frontend Dashboard

The teacher command center is the frontend done on the Next.js.

5.1 UI/UX and Layout

The dashboard uses a contemporary, responsive grid design (`BentoGrid.tsx`) using the modules in a clean and structured grid. It is dark-mode first in design, with the emphasis on clarity and the ease on the eyes of the educator when spending long sessions in it. The tailwind CSS engine is applied to fast, utility-factor styling.

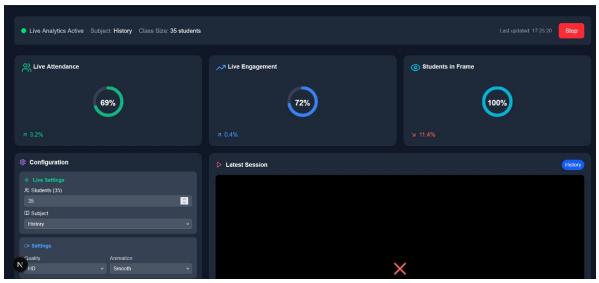

Figure 3: Main Dashboard View.

5.2 Data Visualization

Data is presented through several key components:

- **KPI Boxes (`KPIBox.tsx`):** Display at-a-glance metrics like 'Live Attendance' and 'Average Engagement %'.
- **Interactive Charts (`ChartBox.tsx`):** Use the Recharts library to visualize historical data, like an engagement timeline or a pie chart of emotion distribution.

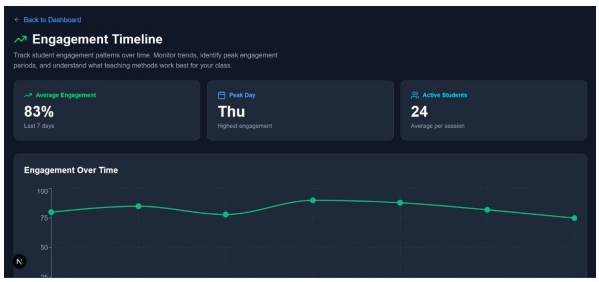

Figure 4: Emotion timeline, providing insights into the overall classroom mood.

5.3 State Management

The frontend's state is managed using React Hooks and Context. A custom hook, 'useDashboardData.ts', handles fetching and caching of analytics data from the backend. A 'ConfigContext.tsx' provides global state for system settings (like thresholds), which is persisted in the browser's local storage for a consistent user experience across sessions.

— System Evaluation and Performance

Measurements to gauge performance of the two core modules were set out to prove the viability of the system as a real-time system. It was tested on a development machine, having the Intel Core i7-12700H chip with 16GB RAM.

6.1 Engagement Module Performance

The major success indicator of this module will be the fact that it manages to process a video stream at a higher than default 24/30 FPS rate and avoids frame drops. The OpenVINO-optimized models are very efficient as revealed in Table 5 which is comfortable within real-time constraints and this enables the entire pipeline to perform its task comfortably within the real-time guidelines.

6.2 Content Generation Performance

The most important indicator in the content module relates to total processing time. Though it is not an activity done in real time, its aim is to bring a fast turnaround to the educators. As Table 6 indicates, even a video of a minute duration may be produced in less than two minutes, with transcribing and artificial intelligence scripting being the quickest part of the process.

— Deployment Strategy

A cloud-native deployment strategy will be employed to scale the EduTrack application to production level in this migration.

- **Containerization:** All the three major services (Engagement Backend, Content Backend, Frontend) will be containerized with Docker. This summarizes their dependencies and also makes them behave the same way in various environments.
- **Orchestration:** Kubernetes will control the containerized services. This makes automated scaling and load balancing as well as high availability possible. As an example, in case Content Generation service has a high-load, then Kubernetes is able to create new instances to process the traffic.
- **Cloud Platform:** The whole system will be deployed on the large cloud provider, e.g., AWS. This would imply the adoption of such services as Amazon EKS to run Kubernetes, Amazon S3 to store video/audio assets and Amazon RDS to run PostgreSQL database.

— Ethical Considerations and Data Privacy

EduTrack being a system through which students have to be watchdogged had its design given much thought in ethics and data confidentiality. The system is aimed at delivering aggregate, anonymized data to give power to educators and is not meant to spy on and punish individual students.

8.1 Data Anonymization and Aggregation

The main product of the Engagement Monitoring module consists of combined statistics. Although the system uses `track_id` to track people to perform temporal analysis, the given ID is session-based, and not related to any personally identifiable information (PII) such as a student name or ID number. The dashboard is made in such a way that it will reflect a trend in the whole class (e.g 25% of the class seems confused) instead of giving tendencies towards individuals.

8.2 Potential for Bias

Performance gaps can be observed in AI models (especially facial analysis model) across populations of various demographics. The models used in this project have been selected because they have high performance on various datasets. A well known downside is that biases could still exist. Additional research should involve extensive experimenting and validation of the models on a randomized sample of the student population in order to discover and eliminate possible sources of bias.

8.3 Intended Use and Data Handling

The video information can be processed real-time or through a temporary accomplishment. The system does not aim at permanent surveillance archive. Strict policies of data retention should be implanted so that when video feeds are processed they are deleted even without delay. The generated engagement data is anonymous and may be stored to support a longitudinal examination of the teaching procedures but needs securing and access-management.

— Conclusion and Future Work

The objectives of the EduTrack project have been met with success, and a prototype has been developed that proved the strength of using AI-driven analysis and automation in terms of an educational environment. The modular architecture gives a good basis of future developments.

The future developments will concern the transformation of EduTrack into a production-level system. Some of the essential development areas are:

- **Full-Stack Integration and API Hardening:** Migration out of mock data to real, strong API connection between the frontend and backend services. Authentication, authorization and error handling should be enabled in every endpoint.
- **Advanced Engagement Metrics:** Add more advanced behavioural triggers to the engagement algorithm including hand-raising detection, student-student communication and audio analysis to detect question frequency.
- **Interactive Content Generation:** Supply the teachers with the possibility to revise and correct

the AI-generated text before the narration and animation stages. Offer a variety of various animation themes and voice styles to use.

- **Real-time Teacher Feedback Loop:** Add an alarm feature to the dashboard that would display discreet real-time messages to the teacher when the overall student engagement in the classroom has dipped below a particular threshold over a span of time so that she can intervene at that time.
- **Cloud Deployment and Scalability:** Wholeheartedly enact the above-mentioned deployment strategy, where each service is containerized and deployed to a cloud provider with the help of Kubernetes.
- **Bias Auditing and Mitigation:** Formally audit the computer vision models on a different data set to find and actively counter any demographic bias in the performance.

— Code Snippet: Engagement Monitoring Main Loop

The following code from `EduTrack-engagement-monitoring/main.py` demonstrates the core processing loop where each frame is analyzed.

```python
# In main.py
while cap.isOpened():
    ret, frame = cap.read()
    if not ret: break
    frame_num += 1

    detections = detector.detect(frame)
    tracked_faces = tracker.update_tracks(detections, frame)

    for track_id, bbox in tracked_faces:
        x1, y1, x2, y2 = map(int, bbox)
        face_crop = frame[y1:y2, x1:x2]

        if face_crop.size == 0: continue

        emotion, _ = emotion_recognizer.infer(face_crop)
        yaw, pitch, _ = pose_estimator.predict_angles(face_crop)

        is_looking_away = (abs(yaw) > YAW_THRESHOLD) or \
                          (abs(pitch) > PITCH_THRESHOLD)
        is_confused = emotion in ['surprise', 'fear']

        current_tracker = dissociation_tracker[track_id]
        if is_looking_away or is_confused:
            current_tracker['count'] += 1
        else:
            current_tracker['count'] = 0
            current_tracker['status'] = 'Engaged'

        if current_tracker['count'] > DISSOCIATION_FRAME_THRESHOLD:
            current_tracker['status'] = 'Disengaged'
```

Listing 1: Core processing loop for video analysis.

— Code Snippet: Content Generation API Endpoint

This snippet from `EduTrack-voice-features/api.py` shows the main FastAPI endpoint that orchestrates the content generation pipeline.

```python
# In api.py
@app.post("/generate")
async def generate_educational_video(
    audio: UploadFile = File(...),
    topic_hint: str = Form(""),
):
    with tempfile.NamedTemporaryFile(...) as tmp_audio:
        # ... file writing logic ...
        audio_path = tmp_audio.name

    try:
        # Step 1: Transcribe audio
        transcriber = AudioTranscriber()
        transcript = await transcriber.transcribe(audio_path)

        # Step 2: Generate educational script from transcript
        script_gen = ScriptGenerator()
        script_data = await script_gen.generate_script(transcript, topic_hint)

        # Step 3: Create narration audio from script
        narrator = Narrator()
        narration_path = await narrator.create_narration(script_data['full_text'])

        # Step 4: Create animations from script
```

```
        animator = VideoAnimator()
        animation_path = await animator.create_animation(script_data)

        # Step 5: Merge video and audio
        merger = VideoMerger()
        final_video_path = await merger.merge_audio_video(animation_path, narration_path)

        # Step 6: Generate PDF transcript
        pdf_gen = PDFGenerator()
        pdf_path = await pdf_gen.create_pdf(script_data)

        return JSONResponse({
            "video_path": os.path.abspath(final_video_path),
            "pdf_path": os.path.abspath(pdf_path),
            "title": script_data.get('title', 'Educational Video')
        })
    # ... error handling and cleanup ...
```

Listing 2: FastAPI endpoint for generating educational content.