2025

PROJECT

Class Sphere

Proposal Submission Guidelines

Group Members & Roles

Ahmed Usama (Team Leader)	Automated Attendance System (Face Recognition)
Mohamed Soltan	Student Behavior Detection(Violence Detection)
Mustafa Mahmoud	Creating short quizzes with multiple-choice questions
Salma Mohamed	Summarizing content by correcting transcription errors
Ammar Yasser	Developing and designing the user interface (Front-End Development)
Wessal Ayman	Server-side logic and database integration (Back-End Development)

Project Description

Core Functionality

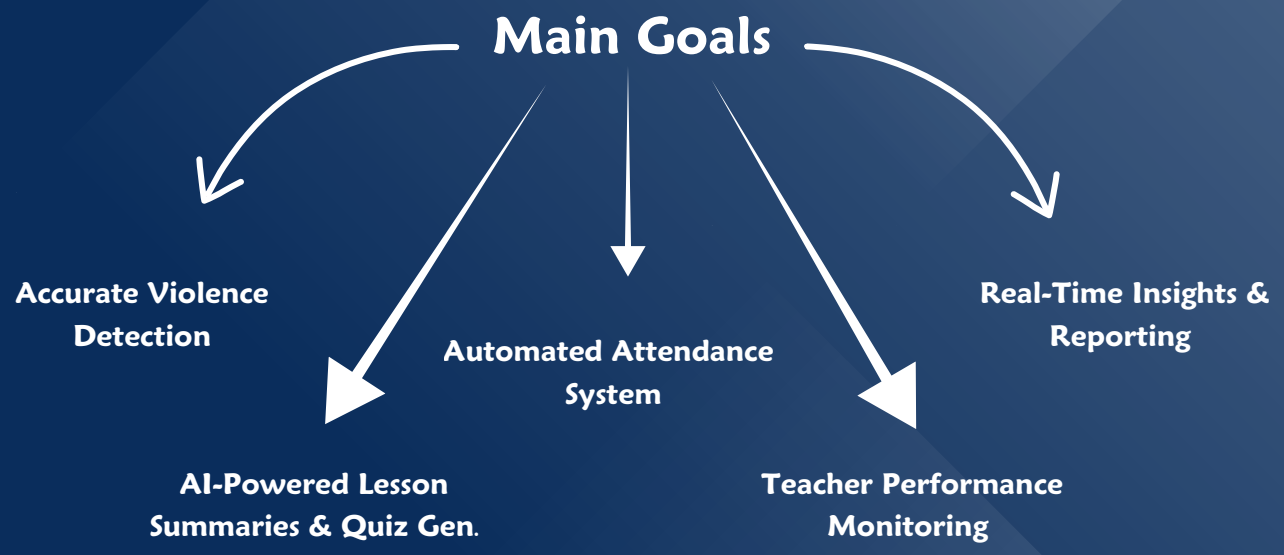
This AI-powered classroom management system uses cameras to automate attendance, analyze teacher performance, and summarize lectures

Behavioral Monitoring & Alerting

It also detects problematic student behavior (such as violence or disengagement) for real-time alerts to administration.

Its goal is to create a smarter, more transparent and efficient learning environment, connecting students, teachers, and administrators on a single platform

Objectives



- **Automated Attendance System** — Face recognition tracks student presence in real-time.
- **Teacher Performance Monitoring** — Captures teacher explanations and identifies lesson topics.
- **Real-Time Insights & Reporting** — Daily/weekly reports keep administrators and parents informed
- **AI-Powered Lesson Summaries** — Speech-to-text converts lectures into written form.
- **Accurate Violence Detection** — Alerts are sent immediately to school administrators for quick intervention.

About The Project

This project aims to transform traditional classrooms into intelligent learning environments powered by AI. By automating attendance, monitoring teaching quality, generating lesson summaries, and detecting student behavior the system creates a more transparent, efficient, and safe educational experience. It bridges the gap between students, teachers, administrators, and parents, ensuring everyone has access to real-time insights and actionable information.

Tools & Techologies

A Look at the Tools Used

1. Core (AI) Technologies

- Face Recognition for Automated Attendance
- Teacher Performance and Content Analysis
- Student Behavior Detection
- AI Lesson Summarization

2. Input Devices and Hardware

Cameras (Audio/Video Recording): The core sensor for the system. Cameras are used to capture the visual feed necessary for face recognition and behavior monitoring. They also serve to record the lectures through both video and audio.

3. Data Processing Technologies

Speech-to-Text Conversion: This technology is explicitly mentioned as the process used to convert the spoken content of the lectures into a written format. This written script is essential input for the AI to then generate the lesson summaries

4. Output and Reporting Tools

- Real-Time Dashboard
- Smart Reporting and File Generation
- Instant Alert System

Milestones & Deadlines

Milestone 1

Deadline 16/9

- **Key Achievements:**

1. Automated Attendance System (Face Recognition):

This is a Smart Attendance System built with Flask that uses facial recognition to automatically track student attendance during scheduled sessions. The system combines computer vision, deep learning, and scheduling capabilities to provide an API-based attendance management solution.EncodeGenerator.py+4

- **Core Technology**

1. Computer Vision & Face Recognition

- OpenCV (cv2): Image processing and face detection using YuNet model (face_detection_yunet_2023mar.onnx)main.py
- DeepFace: Generates 512-dimensional face embeddings using the ArcFace model for recognitionEncodeGenerator.py+1
- scikit-learn: Computes cosine similarity between face embeddings to match detected faces with stored student profilesmain.py

2. Data Augmentation & Storage

- imgaug: Applies image augmentation techniques (horizontal flip, Gaussian blur, brightness/contrast adjustment, rotation, shear) to improve recognition accuracy with multiple embedding variations per studentEncodeGenerator.py
- pickle: Serializes and stores face embeddings in embeddings.pkl for efficient recognition lookupsmain.py+1
- pickle: Serializes and stores face embeddings in embeddings.pkl for efficient recognition lookupsmain.py+1CSV: Stores attendance records with presence percentage, session metadata, and timestampsmain.py

• Key Features

1. Facial Recognition System

The system uses a confidence threshold of 0.6 for matching faces and marks students present if they appear in at least 25% of session frames. Face embeddings are generated from student photos with optional augmentation (default 1-5 variations per image) to handle different angles and lighting conditions.EncodeGenerator.py+1

2.Session Management

Scheduled sessions run automatically at specified times, accepting real-time frame uploads via API during the active window. The scheduler tracks recognized faces throughout each session and finalizes attendance data automatically when the session ends.Run.py

3.Student Administration

The API provides endpoints to add students with multiple base64-encoded images from browser cameras, remove students and their associated data, and update the embeddings database incrementally without reprocessing existing students.Student_Manage.py+1

These achievements validate the core technologies for efficiency and transparency, allowing the project to proceed to the monitoring and behavior detection features in Milestone 2.

Milestone 2

Deadline 30/9

• Key Achievements

• Flask-based Violence Detection

This is a Flask-based Violence Detection API that uses deep learning to classify video content as violent or non-violent in real-time. The system processes sequences of video frames through a CNN-LSTM architecture to detect temporal patterns associated with violence.model_utils.py+2

• Core Technologies

1.Deep Learning Stack

- PyTorch: Neural network framework handling model training, inference, and GPU accelerationModelCreation.py+1
- MobileNetV2: Pretrained CNN backbone for spatial feature extraction (1280-dimensional feature vectors per frame)model_utils.py+1
- Bidirectional LSTM: Temporal modeling layer with 256 hidden units capturing sequential patterns across 16 framesModelCreation.py+1
- Binary Cross-Entropy Loss: Trains the model with sigmoid activation for violence/non-violence classificationModelCreation.py

2.Computer Vision & Preprocessing

- OpenCV (cv2): Video capture, frame extraction, image decoding, and color space conversionsmodel_utils.py+1
- Albumentations: Data augmentation library applying transforms including horizontal flip, brightness/contrast adjustment, and ImageNet normalizationmodel_utils.py+1
- NumPy: Numerical operations for frame sampling (evenly spaced frame selection via np.linspace)VideoTest.py+1

• Key Features

1. Real-Time Detection

The API accepts base64-encoded frames and buffers them in a sliding window (deque with maxlen=16). Once 16 frames are collected, the model predicts violence probability with a configurable threshold (default 0.5).app.py+1

2.Dual Inference Modes

Video Upload: Accepts complete video files, extracts evenly-spaced frames, and returns a single prediction. Frame Streaming: Processes individual frames incrementally, suitable for live camera feeds or continuous monitoring.app.py

3.Model Architecture

The CNN-LSTM hybrid processes sequences by flattening frames for parallel CNN feature extraction, reshaping them back to temporal sequences, passing through bidirectional LSTM, and using the final time step output for classification.ModelCreation.py+1

4.Training Pipeline

The system was trained on a Real Life Violence Dataset with 70/15/15 train/validation/test split, using Adam optimizer with learning rate 0.001, ReduceLROnPlateau scheduler, and early stopping with patience=5.ModelCreation.py

Milestone 3

Deadline 11/10

• Key Achievements

• ClassSphere Audio Summarizer & Quiz API

This is the ClassSphere Audio Summarizer & Quiz API, a Flask-based system that processes educational session audio to automatically generate transcripts, summaries, and quizzes integrated with Firebase Firestore. The system connects with an existing attendance system to provide comprehensive session documentation and assessment capabilities.INTEGRATION_GUIDE.md+1

• Core Technologies

1.AI & Speech Processing

- **OpenAI Whisper: Speech-to-text transcription using the "base" model loaded at application startuptranscription.py**
- **Google Gemini: Generates 6-10 sentence summaries and creates 5-question multiple-choice quizzes from transcriptsmain.py+1**
- **Custom Quiz Parser: Extracts structured question data (question text, options A-D, correct answers) from AI-generated textapp.py**

2.File Processing

- **Temporary File Handling: Audio files uploaded as MP3/WAV are saved temporarily during processing then automatically deletedapp.py**
- **Processing Status Tracking: Sessions track their state through "pending" → "processing" → "completed" or "failed" stagesINTEGRATION_GUIDE.md+1**

• Key Features

1. Audio Processing Pipeline

The /process-session-audio endpoint orchestrates a complete workflow: transcribing uploaded audio, saving the transcript with processing time metrics, generating a summary using Gemini, creating a quiz with parsed structured data, and updating the session status. All components are linked via sessionId for easy retrieval.app.py+1

2.Student Data Access

Students can retrieve session data through dedicated endpoints: /get-session-transcript/<session_id> for full transcripts with language detection, /get-session-summary/<session_id> for condensed summaries, and /get-session-quiz/<session_id> for both raw quiz text and structured question objects.INTEGRATION_GUIDE.md+1

3.Interactive Quiz System

The /submit-quiz endpoint accepts student answers, automatically calculates scores by comparing against correct answers, provides detailed feedback showing which questions were answered correctly, and saves attempts to Firestore with timestamps and time-spent metrics. Teachers and students can query quiz attempts through /get-student-attempts/<student_id> with optional session filtering.app.py+1

4.Integration Architecture

The system extends an existing attendance system by adding new fields (classId, processingStatus, audioProcessed) to session documents and creating relational links through document IDs (transcriptId, summaryId, quizId). The /get-session-data/<session_id> endpoint consolidates all related information in a single response.QUICK_START.md+2

Key Performance Indicators

Data Quality

- Percentage of missing values handled: 100% of corrupted or missing files removed (Since image/video datasets don't usually have "missing values" like text or tables.)
- Data accuracy after preprocessing:
  98% of files successfully processed after cleaning, resizing, and normalization.
- Dataset diversity (representation of different categories): Approximately 95% balanced representation across all classes.

Model Performance

- Model accuracy (Accuracy/F1-Score): F1-Score: 92% (overall model accuracy on test set).
- Model prediction speed (Latency): Average latency: 50 milliseconds per frame.
- Error rate (False Positive/False Negative Rate): Overall error rate: 8%.

Deployment & Scalability

- Applied in the Coming Milestone

DEPLOYMENT

Milestone 4 — Backend Deployment

Deadline 29/10

• Key Achievements

Milestone 4 focuses on deploying all backend microservices (Attendance API, Audio Summarizer & Quiz API, and Violence Detection API) into a production-ready environment. The deployment ensures secure access, stable performance, and integration with the frontend.

• Core Technologies

- Containerize backend services (optional).
- Configure environment variables for all modules.
- Deploy Flask services on production ports (5000, 5001, 5002).
- Enable HTTPS and CORS.
- Connect all services to Firebase using production credentials.
- Perform backend API testing and attach visual evidence.

• Deployment Steps

1. Prepare Production Builds

- Clean and refactor file structure.
- Move API keys and Firebase credentials into environment variables.
- Verify all models load correctly in production

2. Smart Attendance Service Deployment (Port 5000)

- Install dependencies.
- Configure Gunicorn service file.
- Configure Nginx reverse proxy.
- Test API endpoints using Postman.

3. Audio Summarizer & Quiz API Deployment (Port 5001)

- Deploy Whisper model.
- Configure Gemini API key.
- Enable endpoint for /process-session-audio.
- Confirm transcript, summary, and quiz retrieval.

4.Violence Detection API Deployment (Port 5002)

- Deploy CNN-LSTM model with PyTorch.
- Enable streaming frame endpoint.
- Test video upload endpoint.
- Validate probability outputs.

5. Deployment Validation

- All APIs accessible via HTTPS
- CORS enabled for frontend domain.
- Logs and error tracking active.

Milestone 5 — Frontend Deployment

Deadline 15/11

• Key Achievements

Milestone 5 deploys the ClassSphere Frontend built with Next.js and Bit.dev.
It integrates all backend microservices into a unified web platform for students, teachers, and administrators.

• Core Technologies

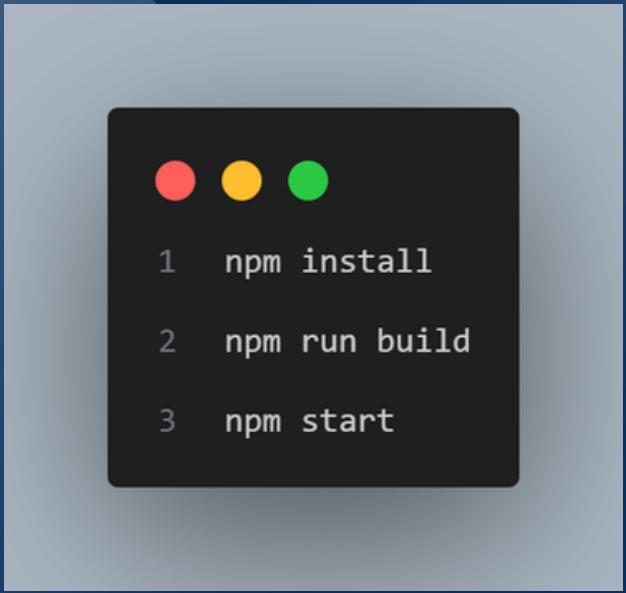
- Build production-optimized Next.js app.
- Configure .env.production with backend URLs.
- Deploy on a hosting service (Vercel, VPS, or Docker).
- Connect frontend modules to all backend APIs.
- Perform UI & functional testing with screenshots.

• Deployment Steps

1. Prepare Production Environment

```
NEXT_PUBLIC_API_ATTENDANCE=https://domain:5000
NEXT_PUBLIC_API_SUMMARIZER=https://domain:5001
NEXT_PUBLIC_API_VIOLENCE=https://domain:5002
NEXTAUTH_URL=https://domain
NEXTAUTH_SECRET=xxxxxxx
```

2. Build Frontend



3. Frontend Functional Integration

1 Attendance Dashboard

- Live camera preview
- Student recognition states

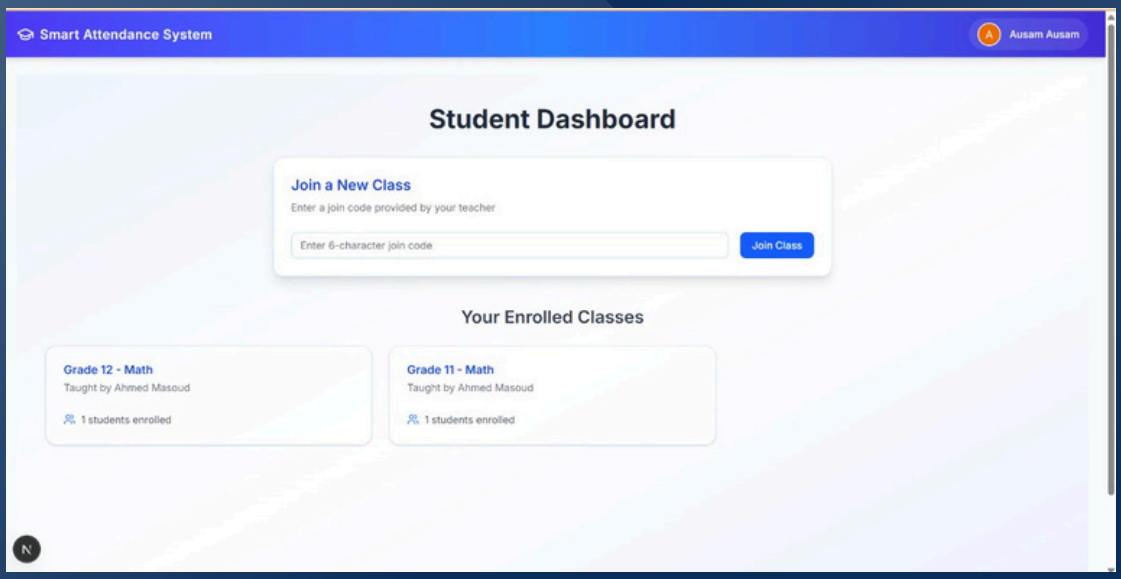
2. Audio & Quiz Section

- Audio upload or recording
- Transcript, summary, and quiz display
- Student quiz submission

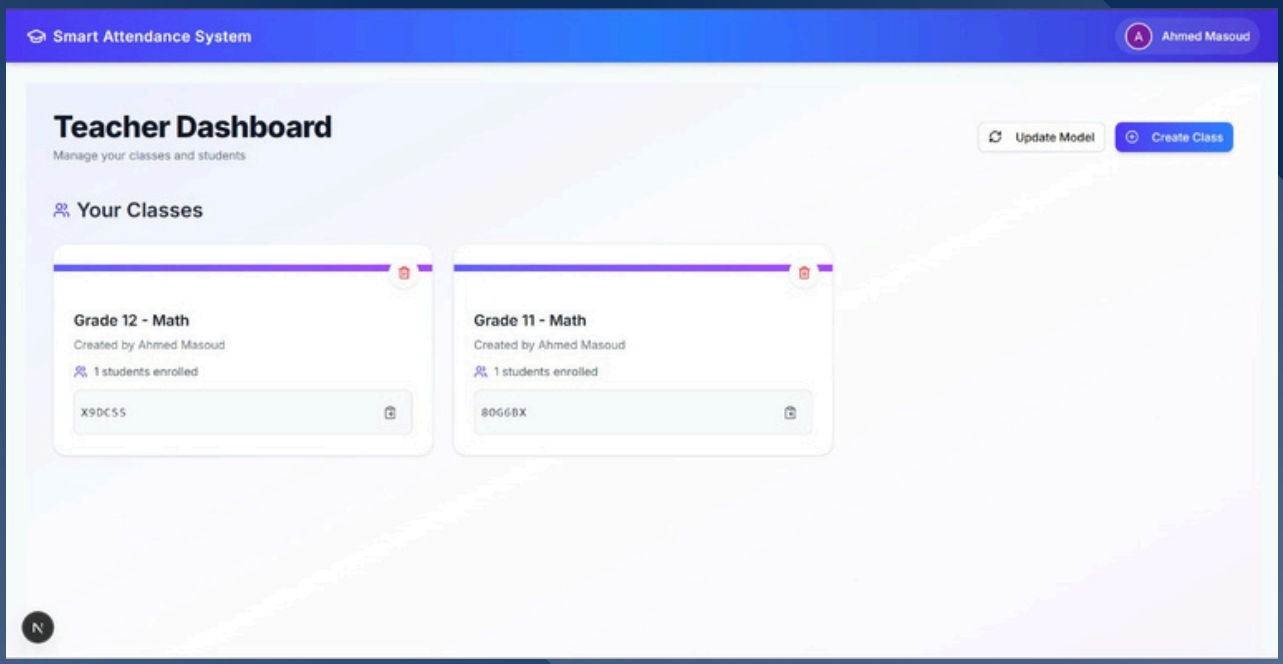
3 Frontend Final Testing & Validation

- User authentication
- Student portal
- Teacher portal
- Real-time data updates from React Query

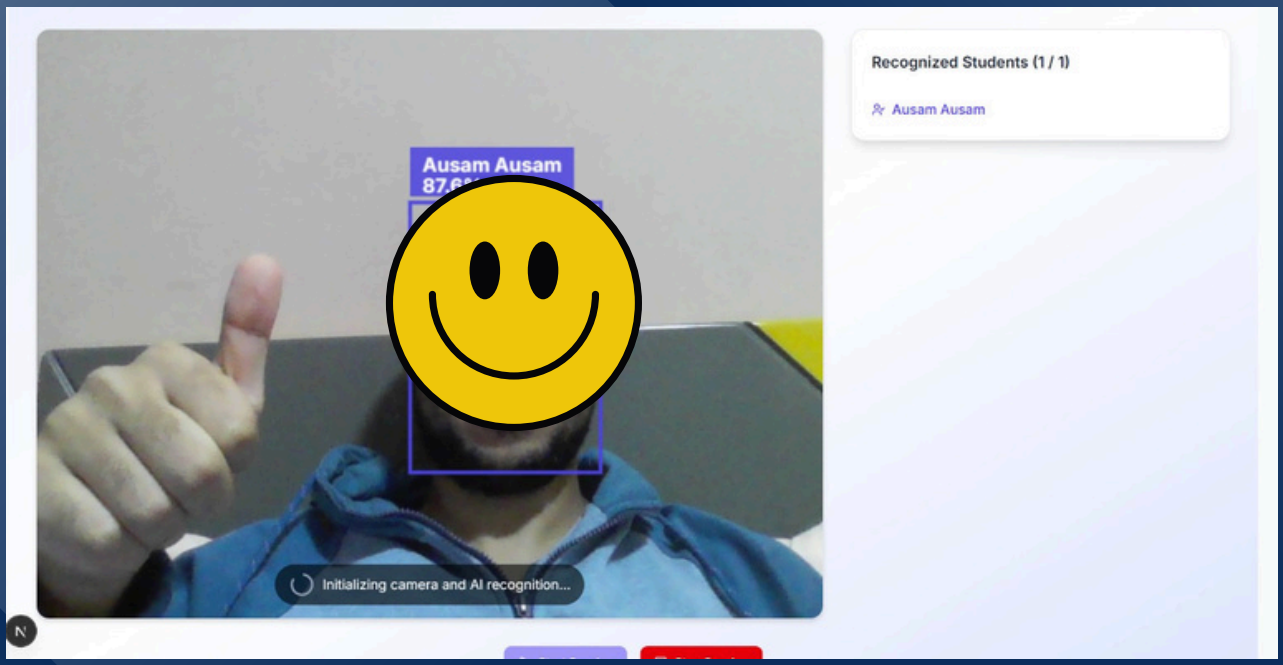
Screenshot — Attendance Dashboard UI



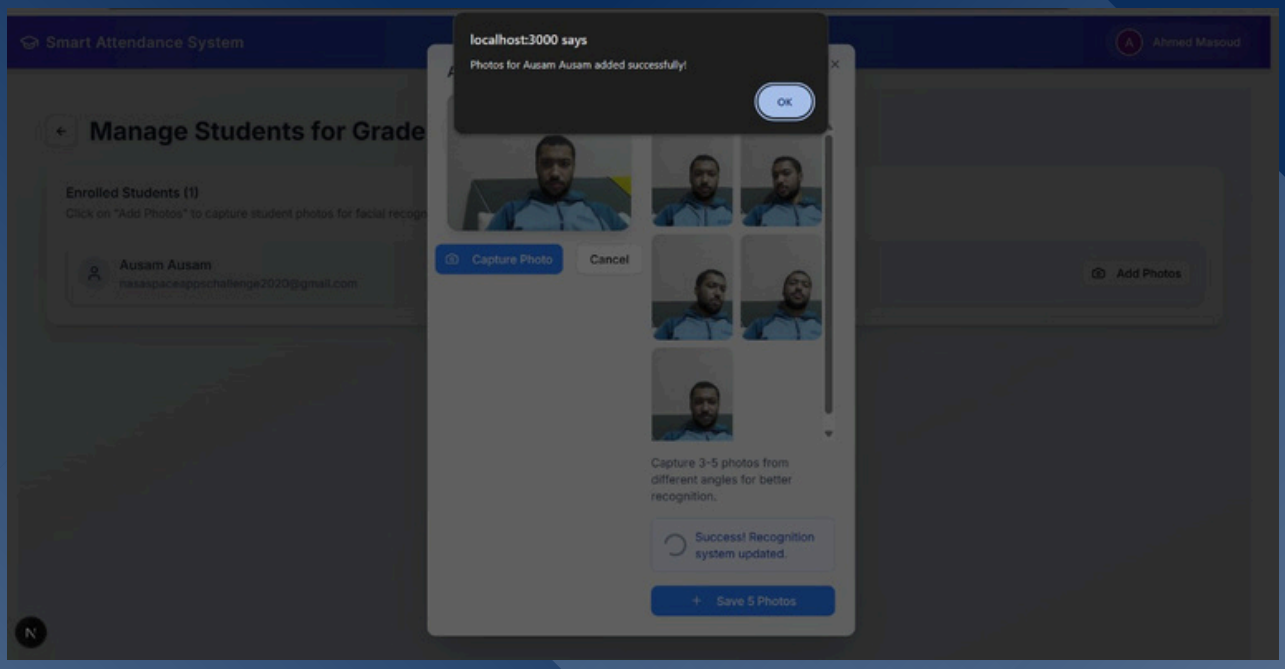
Screenshot — Teacher Portal Dashboard



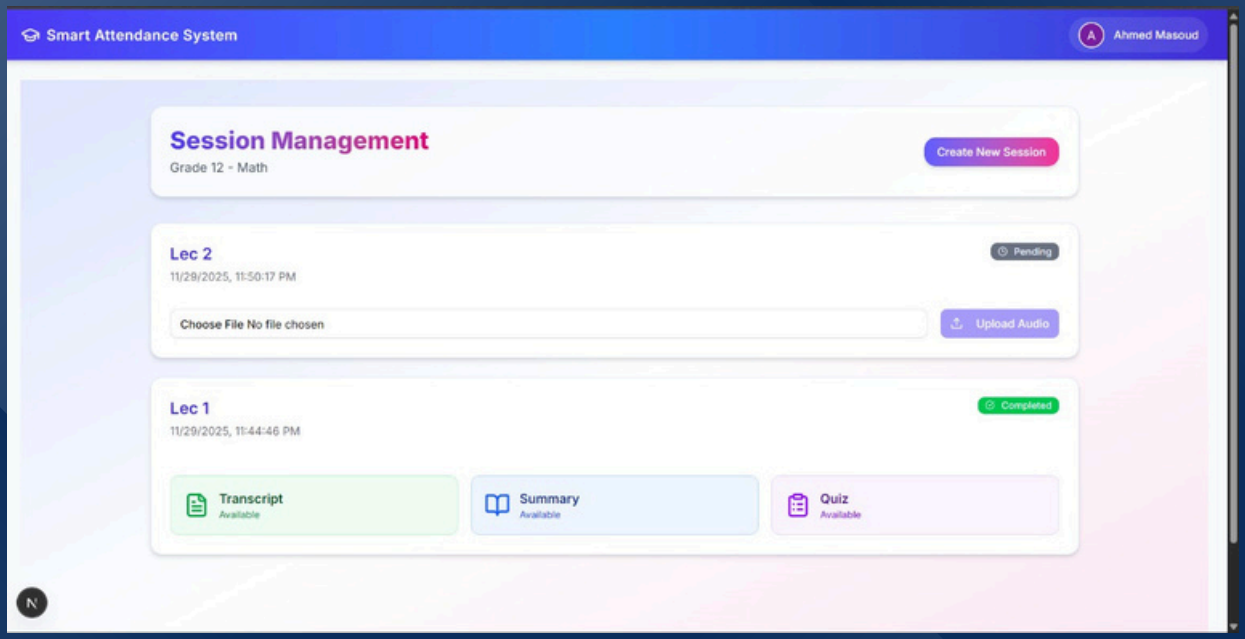
Screenshot — Student Recognition



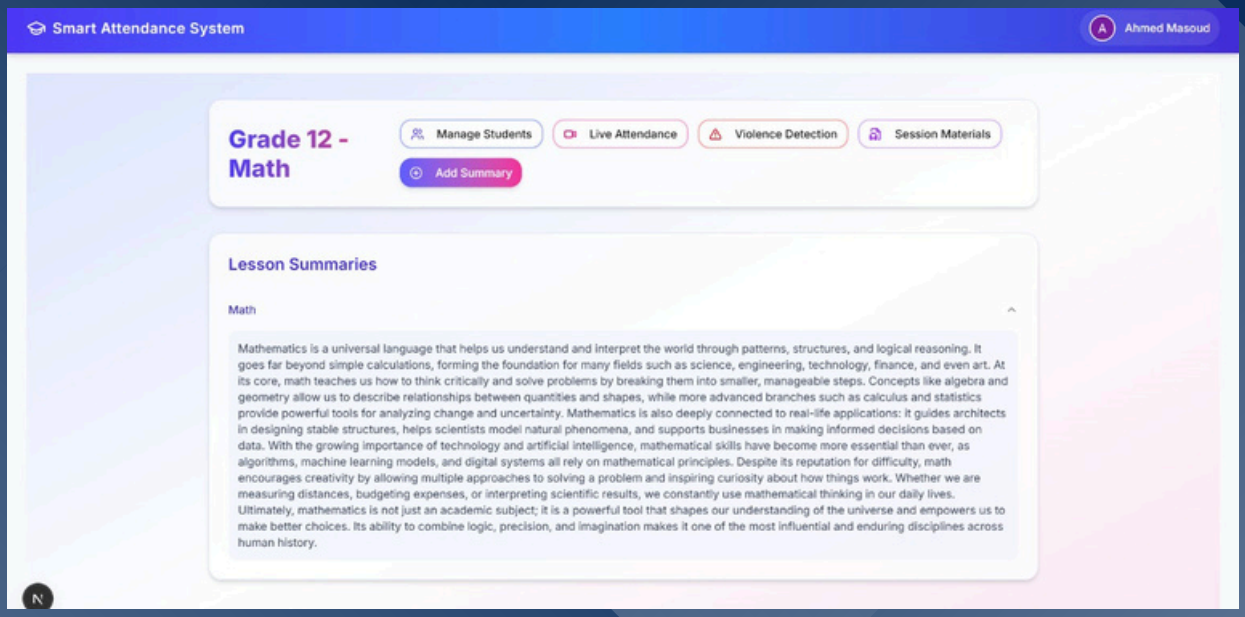
Screenshot — Uploading Student Picture



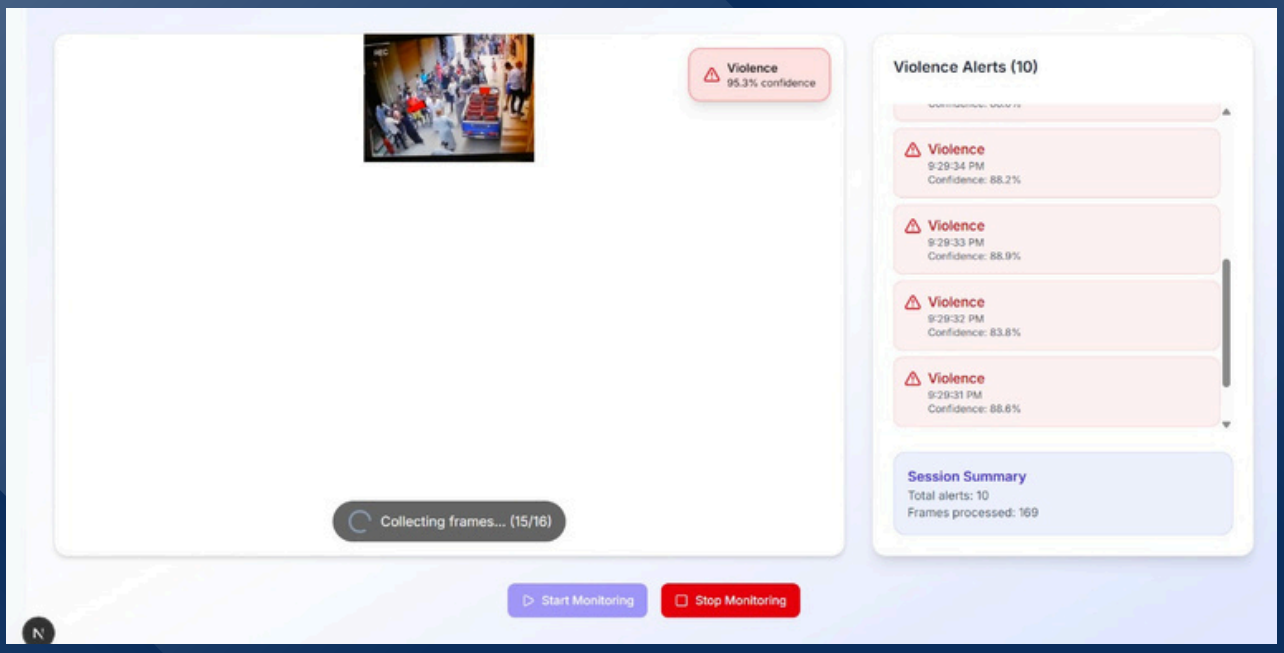
Screenshot — Audio Upload UI



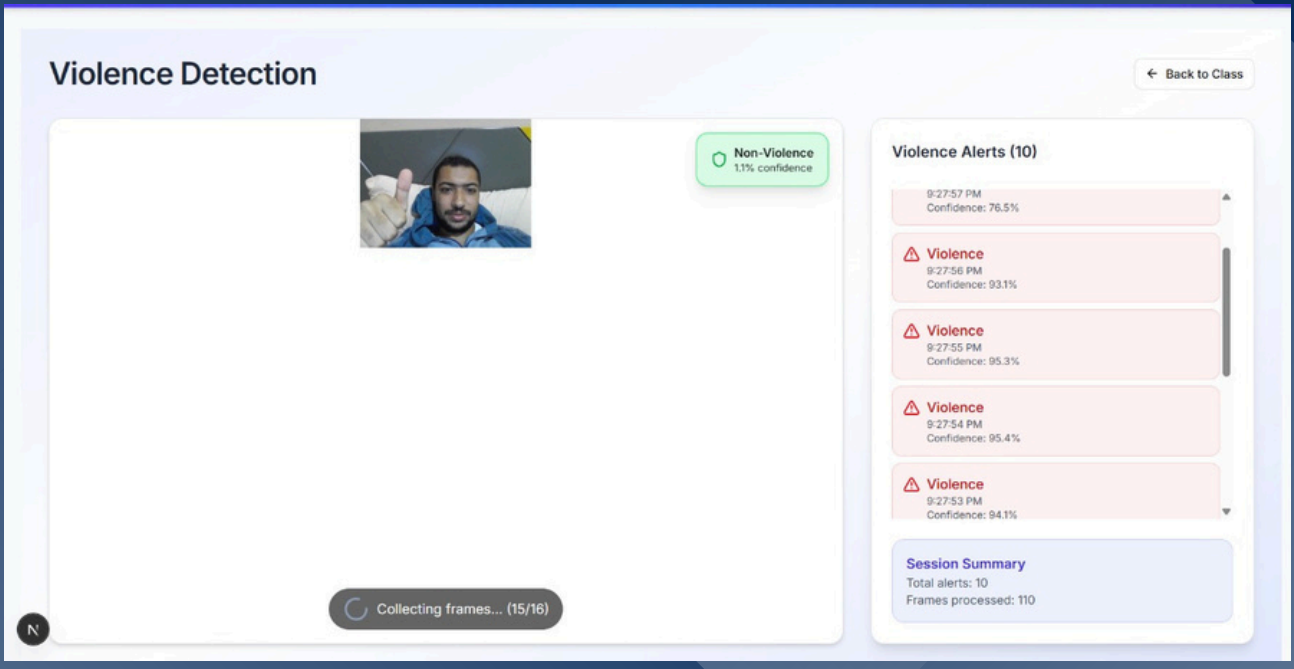
Screenshot — Session Summary Page



Screenshot — Violence Detection Live Monitor



Screenshot — Non Violence Detection



Deployment Summary

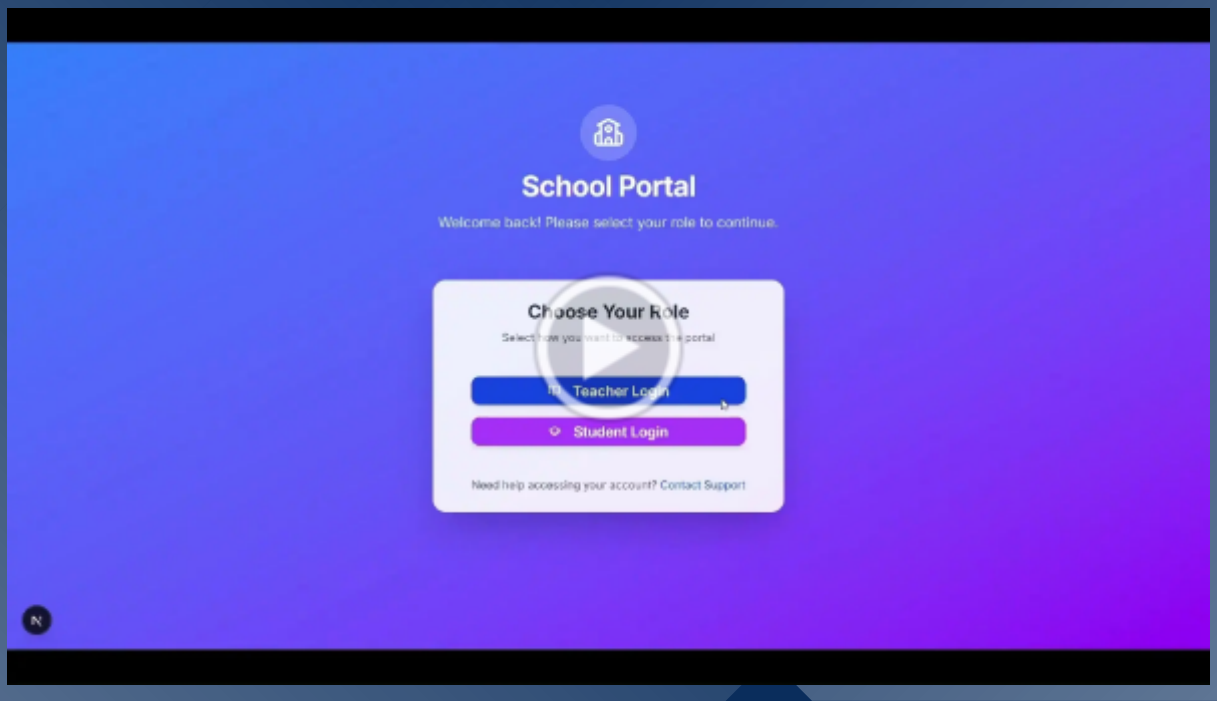
Multiple deployment platforms were evaluated during the deployment phase. Initial attempts on Evennode were unsuccessful because the machine-learning models required system-level libraries that were not available on the free tier. A similar issue occurred on PythonAnywhere, where the platform did not provide the necessary dependencies to run the model components.

Subsequently, deployment was attempted on Azure, where the application was successfully uploaded; however, execution failed remotely due to the same missing system libraries required by the models.Azure provided a Docker-based deployment option that allowed a custom environment to be configured for the application.

During the Docker build and upload process, several issues were encountered due to the limited hardware resources available on the local development machine, particularly when building large images that included heavy models and dependencies. These constraints resulted in repeated build failures, conflicts, and slowdowns during the image creation and upload phases.

As a result, the deployment process was not completed successfully because the local system resources were insufficient to handle the Docker build requirements for a model of this size.

Take a look at the final result



2025

THANK YOU FOR YOUR ATTENTION