STUDENT LIVE BEHAVIOUR MONITORING IN ONLINE CLASSES USING ARTIFICIAL INTELLIGENCE

By

AYESHA NAWAZ	**2021-GCUF-068713**
MUHAMMAD MANAN	**2021-GCUF-068654**

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

GOVERNMENT COLLEGE UNIVERSITY FAISALABAD

July 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 General Information

Information to be provided in this section gives a specific name to the project as well as pertinent information about the personal involved.

Project Name: Student Live Behaviour Monitoring in Online Classes Using Artificial Intelligence

Starting Date: ___________________ **Final Date:** ____________________

Controlling Agency: Department of Computer Science

Prepared By: Ayesha Nawaz & Muhammad Manan **Authorized by:** ___________________

1.2 Purpose

The purpose of this project is to develop an AI-based student behavior monitoring system for online classes. With the rapid shift to online education, instructors face challenges in monitoring student engagement, attention, and presence during virtual classes. This system aims to automate the monitoring process using computer vision and AI technologies to detect student face presence, eye status (for drowsiness detection), and head position (for attention tracking). The system will generate alerts for instructors when students appear distracted, drowsy, or absent, helping to improve student engagement and learning outcomes in online educational environments.

1.3 Document Conventions

Abbreviation	**Definition**
AI	Artificial Intelligence
CV	Computer Vision
EAR	Eye Aspect Ratio
GUI	Graphical User Interface
DB	Database
API	Application Programming Interface
ML	Machine Learning

1.4 Project Objective

The primary objective of this project is to develop a comprehensive and robust system for monitoring student behavior during online classes using artificial intelligence. The system aims to accurately detect student presence through advanced face detection techniques and monitor their attentiveness using eye tracking algorithms capable of identifying signs of drowsiness based on eye aspect ratio and blink patterns. Additionally, head pose estimation is integrated to determine whether students are focused on the screen or distracted. An intelligent alert system is designed to notify instructors in real time about instances of student inattention, drowsiness, or absence. The project also includes the development of a detailed reporting module that provides analytics and statistics on student attention trends over time. Emphasis is placed on ensuring that the entire system functions in real time with minimal latency, even on standard computer hardware. A user-friendly interface is created for both students and instructors to easily interact with and manage the system. Furthermore, strict data security and privacy protocols are implemented to safeguard all student-related information.

1.5 Intended Audience and Reading Suggestions

This document is intended for a diverse audience, including project supervisors and evaluators assessing the project, future developers who may enhance the system, educational technologists focused on student engagement, faculty members who might use the system, and IT administrators responsible for its implementation. Readers are encouraged to focus on sections relevant to their roles: technical stakeholders should refer to Sections 3–7, educational stakeholders may find Sections 1–2, 3.4–3.6, and 5 most useful, while implementation teams are advised to review all sections for a comprehensive understanding.

1.6 Project Scope

The scope of this project encompasses:

1. Development of a web-based application with user authentication and role-based access.
2. Implementation of computer vision algorithms for face detection, eye tracking, and head pose estimation.
3. Design and development of a real-time monitoring interface for tracking student behavior.
4. Creation of an alert system to notify about student behavior issues.
5. Development of a comprehensive reporting system with charts and statistics.
6. Implementation of session management to track student behavior over time.
7. Database design and implementation for storing monitoring sessions and alerts.

2. Overall Description

2.1 Product Perspective

The Student Live Behaviour Monitoring System is a standalone web application designed to operate alongside existing online learning platforms. It serves as a complementary tool that provides instructors with real-time insights into student engagement and attention during online classes.

1. The system consists of several integrated components:
2. A front-end web interface for users to interact with the system
3. A computer vision module for analyzing webcam feeds
4. A back-end server for processing and storing monitoring data
5. A database for maintaining user accounts and session data
6. A reporting and analytics module for generating insights

The product solves a critical gap in online education by providing instructors with the ability to monitor student engagement in virtual environments, which is typically difficult to achieve compared to physical classrooms.

2.2 Product Features

The key features of the Student Live Behavior Monitoring System include:

- Face detection and tracking
- Eye status monitoring for drowsiness detection
- Head pose estimation for attention tracking
- Real-time alerts for behavior issues
- Session management (start, pause, end)
- Statistical dashboard with attention metrics
- Comprehensive reports with charts and timelines
- User management with secure authentication

2.3 User Classes and Characteristics

The system supports three main user groups: students, instructors, and administrators. Students use the monitoring features with a simple interface, requiring basic technical skills and webcam-equipped devices, with privacy being a key concern. Instructors monitor multiple students, receive real-time alerts, and access detailed reports, focusing on both individual and group data. Administrators manage user accounts, system settings, and maintenance, requiring higher technical proficiency and access to system-wide analytics.

2.4 Operating Environment

The system is designed to operate in the following environment:

- **Web Browsers**: Chrome 80+, Firefox 75+, Safari 13+, Edge 80+
- **Operating Systems**: Windows 10+, macOS 10.14+, Ubuntu 18.04+
- **Hardware**: Computers with webcam capabilities, minimum 4GB RAM, 2GHz processor
- **Network**: Stable internet connection (minimum 1 Mbps upload/download)
- **Frontend Framework**: React.js (JavaScript) with Bootstrap for styling
- **Backend Server**: Node.js with Express.js and MongoDB using Mongoose ORM
- **AI Module**: Python-based Flask or FastAPI service for real-time face, eye, and head-pose analysis
- **Additional Requirements**: WebRTC-compatible browser for webcam access and Socket.IO support for real-time updates

2.5 Design and Implementation Constraints

The project encounters several constraints that influence its design and implementation. It must comply with educational privacy regulations such as FERPA, which restrict data collection and storage. Real-time computer vision algorithms need to be optimized to run efficiently on standard student hardware, while also ensuring compatibility across different web browsers due to variations in webcam access implementation. Network limitations, such as inconsistent internet speeds, may impact real-time monitoring performance. Security is a key concern, requiring robust encryption and access controls to protect student data. The project must also be completed within the academic semester, posing time constraints. Additionally, the development team has varying levels of expertise in computer vision and AI, and the system must be accessible to users with disabilities to ensure inclusivity.

2.6 User Documentation

The user documentation for the system will include several essential components to support various user needs. A System Installation Guide will be provided to assist with installing and configuring the system. A comprehensive User Manual will cater to all user types, offering detailed instructions on using the system. Video Tutorials will demonstrate key features through step-by-step walkthroughs, while an FAQ Section will address common questions and offer troubleshooting tips. Additionally, API Documentation will be available to support potential future integrations with other systems, and an Administrator Guide will provide detailed instructions for managing and maintaining the system.

2.7 Assumptions and Dependencies

The project operates under several assumptions and dependencies critical to its functionality. It assumes that users have access to devices with webcams, a stable internet connection, and are willing to grant camera permissions. It also expects that students and instructors have a basic understanding of using web applications and that lighting conditions are adequate for accurate face detection. The system depends on various technologies, including OpenCV for computer vision, dlib for facial landmark detection, and Flask for backend development. PostgreSQL is used for database management, while Chart.js supports data visualization. Additionally, the application relies on modern browsers with WebRTC support for webcam access and uses Bootstrap or similar frameworks to ensure a responsive and user-friendly interface.

3. System Features

3.1 Face Detection and Tracking

Face detection is a high-priority feature that serves as the foundation of the monitoring system by continuously tracking whether a student's face is visible in the webcam feed, thus enabling presence monitoring. When a student positions their face in front of the webcam, the system detects and begins tracking it; if the student moves away, the system registers the absence and triggers an alert after a configurable time threshold. Functionally, the system is designed to detect human faces in real-time, distinguish between presence and absence, and maintain tracking even during minor movements. It is optimized to operate under various lighting conditions and is capable of generating absence alerts after a default threshold of 30 seconds, while also recording timestamps for all face detection events.

3.2 Eye Tracking for Drowsiness Detection

Eye tracking is a high-priority component of the system that monitors students' eye status to detect signs of drowsiness by calculating the eye aspect ratio (EAR) and analyzing blink patterns. When a student blinks normally, the system records the pattern without triggering any alerts. However, if the student's eyes remain closed or nearly closed for an extended period, the system detects a reduced EAR and generates a drowsiness alert. The system is capable of detecting and tracking eye landmarks in real-time, distinguishing between normal blinking and prolonged eye closure. It generates alerts after a configurable threshold, with a default of 3 seconds, and logs timestamps and durations of drowsiness events. Additionally, it is designed to function accurately even when users are wearing standard glasses.

3.3 Head Pose Estimation

Head pose estimation is a medium-priority feature that helps determine whether students are paying attention to the screen or are distracted by looking elsewhere. When a student looks directly at the screen, the system records them as attentive; however, if the student turns their head away, the system detects the change in head orientation and triggers a distraction alert after a configurable threshold, typically set to 5 seconds. The system estimates head orientation using facial landmarks and calculates the pitch, yaw, and roll angles to determine the student's focus direction. It tracks head movements in 3D space and records timestamps along with the direction of movement for monitoring and analysis.

3.4 Alerting System

The alerting system is a high-priority component designed to notify users in real-time about behavioral issues such as drowsiness, distraction, or absence. When such an issue is detected, the system generates a corresponding alert with an appropriate severity level. Once the user acknowledges the alert, it is marked as resolved. The system supports real-time alert generation, categorizes alerts by type and severity, and displays visual indicators for active alerts. It also maintains a history of alerts for each monitoring session, allows customization of alert thresholds, and provides users with the ability to acknowledge and dismiss alerts as needed.

3.5 Session Management

Session management is a high-priority feature that handles the creation, tracking, and control of monitoring sessions. When a user initiates a session, the system starts monitoring and logs the session details. Upon ending the session, the system finalizes the data and generates a summary. It supports tracking session duration and status, linking alerts and events to specific sessions, and calculating session-based statistics. Users can pause, resume, or end sessions, reset session stats, and access or delete past session records, ensuring flexibility and comprehensive monitoring control.

3.6 Reporting and Analytics

The reporting and analytics module is a medium-priority feature designed to offer insights into student behavior through detailed statistics and visualizations. When users request reports, the system generates session summaries including attention scores, alert distributions, and behavior trends over time. Users can apply filters such as date range and alert type to refine the data. The system visualizes session data, calculates averages, and presents a detailed event timeline, enabling instructors and administrators to assess engagement and identify patterns effectively.

4. External Interface Requirements

4.1 User Interfaces

The system will provide the following user interfaces:

1. **Login/Registration Interface:**
- Username and password fields
- Registration form with validation
- Password reset functionality

2. **Dashboard Interface:**
- Summary statistics and recent sessions
- Quick access to key features
- Notification area for system alerts

3. **Monitoring Interface:**
- live webcam feed display
- Real-time alert indicators
- Session controls (start, pause, stop)
- Current statistics display

4. **Reports Interface:**
- Filtering options (date, session type)
- Data visualizations (charts, graphs)
- Tabular data with sorting capabilities
- Export options for reports

5. **Session Detail Interface:**
- Comprehensive session statistics
- Timeline of alerts and events
- Visual representations of attention patterns

6. **Settings Interface:**
- Alert threshold configurations
- User profile management
- System preferences

4.2 Hardware Interfaces

The system interfaces with key hardware components including webcams, displays, and processing units. It accesses the user's webcam via browser APIs, requiring a minimum resolution of 640x480 and at least 15

FPS for effective monitoring, with permissions managed by browser security protocols. The display must support resolutions of at least 1280x720, with a responsive design to adapt to various aspect ratios and use hardware acceleration when available. Processing hardware considerations include optimized CPU usage for extended sessions, controlled memory usage to maintain browser performance, and optional GPU acceleration to enhance computer vision tasks.

4.3 Software Interfaces

The system interfaces with several software components to ensure smooth operation. It supports modern web browsers like Chrome, Firefox, Safari, and Edge, requiring WebRTC for camera access, JavaScript for dynamic features, and local storage or session storage to maintain session state.
The backend is developed using Node.js with the Express.js framework, offering RESTful API communication via JSON. Data is stored in MongoDB using Mongoose ORM with connection pooling for performance. The AI module operates as a Python-based Flask or FastAPI microservice.
Computer vision tasks rely on OpenCV and Dlib libraries for face, eye, and head pose detection. The frontend visualizations are implemented using Chart.js, while Axios handles API communication. Socket.IO is used for real-time bi-directional alert delivery between client and server.

4.4 Communication Interfaces

The system uses several communication interfaces to ensure secure and efficient data exchange. All client-server communication occurs over secure HTTPS using RESTful API endpoints with session-based authentication. WebRTC enables real-time webcam access by directly capturing video streams within the browser without server relay. WebSockets may be used optionally for instant, bidirectional alerts and notifications between client and server. Database connections to PostgreSQL are secured and optimized with connection pooling. Additionally, all communication is encrypted with TLS/SSL to protect data privacy and security.

5. Nonfunctional Requirements

5.1 Performance Requirements

The system is designed to meet key performance requirements across various dimensions. User interface actions should respond within 1 second, while face detection must process frames in under 100 milliseconds, and alerts should be generated within 3 seconds of issue detection. The system must support real-time webcam processing at a minimum of 15 frames per second and handle at least 100 database transactions per minute. It should accommodate at least 50 concurrent users per server instance and store up to 1,000 monitoring sessions per user, with the database capable of holding over 1 million alert records. On the client side, memory usage should stay below 500MB, CPU utilization under 50% on average systems, and bandwidth use should not exceed 1 Mbps per active session. The architecture must support horizontal scaling to handle increasing user load, and the database should be optimized for managing large datasets efficiently.

5.2 Safety Requirements

The system incorporates several important safety considerations to ensure a secure and supportive user experience. For psychological safety, it avoids excessive or overly intrusive monitoring that might lead to student anxiety, clearly indicates when monitoring is active, and includes features to reduce "surveillance fatigue." In terms of physical safety, the system issues warnings to help prevent eye strain, enforces session time limits with recommended breaks, and offers guidance for proper ergonomic workstation setup. To protect data safety, the system performs regular backups, includes mechanisms for data recovery, and uses transaction management techniques to prevent data corruption.

5.3 Security Requirements

The system incorporates robust security measures across multiple areas. For authentication, it uses strong password hashing (e.g., BCrypt), supports optional multi-factor authentication, enforces account lockouts after repeated failed logins, and includes session timeouts after inactivity. Data security is ensured through encryption at rest and in transit using TLS/SSL, along with regular security audits and adherence to the principle of least privilege. Privacy protections include compliance with data regulations, clear privacy policies, user control over data deletion, and anonymization for analytics. Application-level security addresses OWASP Top 10 threats, with input validation, secure API authentication, and regular updates and patches.

5.4 Software Quality Attributes

The system is designed to demonstrate key quality attributes that ensure a robust, user-friendly, and maintainable experience. In terms of reliability, it is expected to maintain 99.5% uptime during active use, support graceful degradation when certain components fail, and automatically recover from transient errors. For availability, the system should remain accessible 24/7, with minimal downtime restricted to scheduled maintenance periods. Maintainability is supported through a modular architecture, detailed documentation, consistent coding practices, and comprehensive automated testing. Usability is emphasized through intuitive navigation, consistent UI patterns, helpful guidance, and compliance with WCAG 2.1 accessibility standards. Lastly, the system ensures portability with cross-browser compatibility, responsive design for various screen sizes, and minimal reliance on platform-specific features.

6. WBS Project Management

Project Phases:

Phase No	**Task Description**	**Duration**
1.	Requirements and Planning	2 week
2.	Core Development	3 week
3.	Feature Development	3 week
4.	Testing and Integration	3 week
5.	Documentation and Deployment	1 week

Total Duration: Approx. **12 week**

7. Tools & Technologies

7.1 Programming Languages

- **Python** for AI microservices and computer vision tasks
- **JavaScript** for frontend and backend development
- **HTML/CSS** for responsive web interface

7.2 Databases/Data storages

- **MongoDB** for data storage
- **Mongoose ORM** for schema-based database operations

7.3 Operating System

The system is designed to be platform-independent, with development and deployment on:

- **Server Environment**:
  - Node.js runtime with Express.js and MongoDB
  - Python (Flask or FastAPI) for AI microservice

- **Client Environment**:
  - Any modern OS with supported browsers
  - Mobile OS support is not prioritized for the initial release

7.4 Libraries & Frameworks

- React.js with JavaScript for the frontend
- Bootstrap for responsive UI design
- Chart.js for interactive data visualization
- Axios for REST API communication
- Socket.IO for real-time alerting
- OpenCV and Dlib for computer vision tasks in the AI module

Appendix A: Glossary

1. CDN- Content Delivery Network
2. DB- Database
3. HTML- Hyper Text Markup Language
4. CSS- Cascading Style Sheet
5. JS- Java Script
6. HTTPS- Hypertext Transfer Protocol Secure
7. TLS- Transport Layer Security
8. SSL- Secure Sockets Layer
9. MongoDB- Mongo Database
10. Mongoose ORM- Mongoose Object Relational Mapping
11. REST API- Representational State Transfer Application Programming Interface
12. OpenCV- Open Source Computer Vision Library
13. API- Application Programming Interface
14. CPU-Central Processing Unit
15. GPU- Graphic Processing Unit
16. JSON- JavaScript Object Notation
17. WebRTC- Web Real Time Communication
18. EAR- Export Administration Regulation
19. FERPA- Family Educational Rights and Privacy Act
20. IT- Information Technology
21. FPS- Frames Per Second
22. OWASP- Open Web Application Security Project
23. WCAG- Web Content Accessibility Guideline
24. UI- User Interface
25. OS- Operating System

Appendix B: Analysis Models

Check List	Yes	No
I. Starting/Ending Dates		
II. Project Scope		
III. Product modules (covering all aspects of scope)		
IV. System Features (covering scope)		
V. Interface Requirements		
VI. Non-Functional Requirements		
VII. WBS		
VIII. Tools and Technologies Detail (for implementation)		
IX. Plagiarism Report		

Appendix C: Check List

Check List	**Yes**	**No**
I. Starting/Ending Dates		
II. Project Scope		
III. Product modules (covering all aspects of scope)		
IV. System Features (covering scope)		
V. Interface Requirements		
VI. Non-Functional Requirements		
VII. WBS		
VIII. Tools and Technologies Detail (for implementation)		
IX. Plagiarism Report		

Appendix D: Supervisory Committee

For Approval of any two Consultant Teachers	
Teacher Consulted **Name:** ______________________________ **Designation:** _________________________ **Comments:** __________________________ ____________________________________ ____________________________________ ____________________________________ ____________________________________ **Signature:** ___________________________	**Teacher Consulted** **Name:** ______________________________ **Designation:** _________________________ **Comments:** __________________________ ____________________________________ ____________________________________ ____________________________________ ____________________________________ **Signature:** ___________________________

(For office use only)

Date: ___________________

- ➢ **Approved** **Group ID:** ________________
- ➢ **Meeting Required:** **Date:** ____________ **Time:** ____________ **Place:** __________________
- ➢ **Rejected**

Remarks:____

____

____

Project Title (If Revised):

____

Project Coordinator _________________