

ITBP481: Senior Graduation Project II
Final Project Report
ClassTrack

By:

Ali Alblooshi 202102812

Hamad Alketbi 202021767

Saif Alketbi 202022212

Saif Alkaabi 201915236

Ali Almerri 202100588

Supervision of:

Dr. Shayma Alkobaisi

Spring 2025

**A final year student project report submitted in partial fulfillment of
the requirement of Faculty of Information Technology, United Arab
Emirates University, for its bachelor degree program**

Executive Summary

The purpose of the ClassTrack system is to offer schools a secure, safe and smart platform for student attendance automation and exam cheating prevention. Combining RFID technology and computer vision tools like Mediapipe, YOLOv8-Face, and openCV into a single product guarantees the accuracy of attendance in class as well as testing procedures. The system is built on stable and scalable infrastructure, which includes web-based management system built on the MERN stack and real-time AI based video analysis built using Python Flask.

The Attendance Module quickly identifies students using RFID cards and a USB RFID reader, and then registers them in a database. Each student is provided a unique RFID card and is able to tap their card to automatically register their attendance in centrally monitored database. This further reduces or eliminates the need for manual roll-call and addresses the issue of proxy attendance.

The Cheating Detection Module detect potential test-takers foul play during tests through cutting-edge face and gesture recognition technologies. By using Retina Face, Insight Face, YOLOv8-Face model and Mediapipe framework on OpenCV which will be able to quickly detect if there are too many faces detected within a frame all at once, monitor facial movement inside the frame and monitor suspicious hand movement. This would enable us to watch multiple students at the same time making sure they were not cheating and record/screen for admin review.

This MERN stack Admin Dashboard also grants instructors full control over the attendance and monitoring/reporting on assessments. The facilitators are enabled to check attendance trends, keep a track of cheating alerts and create reports for those who administer the educational system. And, Flask and MERN are a great combination for synchronous response between the web interface and the AI modules

A holistic solution for classroom operation and supervision. ClassTrack is a combination of attendance system (via RFID card) and anti-cheating via AI. All the while providing additional security in the modern educational landscape, improving efficiencies within the institution and discouraging academic dishonesty.

Table of Contents

| | |
|---|-----------|
| CLASSTRACK..... | I |
| 1 INTRODUCTION..... | 1 |
| 1.1 PROBLEM STATEMENT AND PURPOSE | 1 |
| 1.2 PROJECT AND DESIGN OBJECTIVES | 1 |
| 1.3 FINAL OUTCOMES AND DELIVERABLES..... | 2 |
| 1.4 MOTIVATIONS..... | 2 |
| 1.5 SUMMARY OF REPORT STRUCTURE | 3 |
| 2 PROFESSIONAL STANDARDS AND PRACTICE CONSTRAINTS..... | 4 |
| 2.1. MANUFACTURABILITY CONSTRAINTS..... | 4 |
| 2.2. ECONOMIC CONSTRAINTS..... | 4 |
| 2.3. SUSTAINABILITY CONSTRAINTS | 5 |
| 2.4. ENVIRONMENTAL CONSTRAINTS..... | 5 |
| 2.5. HEALTH AND SAFETY CONSTRAINTS | 5 |
| 2.6. ETHICAL STANDARDS CONSTRAINTS | 6 |
| 2.7. SOCIAL VALUES CONSTRAINTS | 6 |
| 2.8. POLITICAL CONSTRAINTS | 6 |
| 3 SYSTEM ARCHITECTURE AND DETAILED DESIGN..... | 7 |
| 3.1 REQUIREMENT SPECIFICATIONS | 7 |
| 3.2 DESIGN SETUP..... | 8 |
| 3.3 STANDARDS AND CONSTRAINTS..... | 9 |
| 3.4 TOOLS AND METHODS | 10 |
| 3.5 DATA COLLECTION | 12 |
| 3.6 SYSTEM ARCHITECTURE..... | 13 |
| 3.7 FINAL COST ANALYSIS AND DISCUSSION..... | 15 |
| 4 EVALUATION AND IMPROVEMENTS | 16 |
| 4.1 IMPLEMENTATION | 16 |
| 4.2 TESTING | 20 |
| 4.3 ANALYSIS OF RESULTS..... | 22 |
| 4.3.1 AUTHENTICATION (ADMIN/FACULTY LOGIN) | 22 |
| 4.3.2 DASHBOARD (ATTENDANCE & MANAGEMENT) | 23 |
| 4.3.3 CHEATING DETECTION (AI MONITORING)..... | 24 |
| 4.4 POTENTIAL IMPROVEMENTS | 25 |
| 5 PROJECT MANAGEMENT..... | 26 |
| 5.1 TASKS AND SCHEDULE GANTT CHART..... | 26 |
| 5.2 PROBLEMS AND CHALLENGES..... | 27 |
| 5.3 RESOURCES USED DURING THE PROJECT..... | 29 |
| 6 CONCLUSIONS | 33 |
| 7 STATEMENT OF CONTRIBUTION | 34 |
| 8 REFERENCES..... | 36 |
| BIOGRAPHY | 38 |
| APPENDIX..... | 40 |

List of Figures

| | |
|--|-----------|
| FIGURE 1 SYSTEM ARCHITECTURE | 14 |
| FIGURE 2 LOGIN PAGE | 17 |
| FIGURE 3 INSTRUCTOR DASHBOARD | 17 |
| FIGURE 4 MONITORING SYSTEM | 18 |
| FIGURE 5 ADMIN DASHBOARD | 19 |
| FIGURE 6 LOGIN | 23 |
| FIGURE 7 INSTRUCTOR DASHBOARD | 23 |
| FIGURE 8 ADMIN DASHBOARD | 24 |
| FIGURE 9 MONITORING SCREEN | 24 |
| FIGURE 10 GANTT CHART | 27 |

List of Tables

| | |
|--|-----------|
| TABLE 1 TOOLS AND METHODS | 12 |
| TABLE 2 FINAL COST | 16 |
| TABLE 3 TESTCASE..... | 22 |

1 Introduction

1.1 Problem Statement and Purpose

Most challenges facing educational institutions are incorrect attendance and lack of detection of exam cheating. Manual attendance systems are vulnerable to errors and proxy grading, whereas traditional invigilators often do not catch cheating on large classrooms. These challenges undermine the effectiveness and equity of the educational system.

ClassTrack seeks to address these challenges through:

- Using **RFID cards** [1] with a **USB RFID reader/writer** to automatically and securely log student attendance.
- **Employing computer vision** (Mediapipe, YOLOv8-Face, RetinaFace, InsightFace, and OpenCV) [3] to monitor students during examinations and detect suspicious activities such as multiple faces, unusual head movements, or unauthorized device usage.
- Providing a centralized **web-based dashboard** for administrators and teachers to review attendance and examination monitoring reports.

This effort improves both academic standards and administrative efficiency by providing greater verification security, and an assurance to register attendance accurately and in real time.

1.2 Project and Design Objectives

- **Attendance Recording:** You register and store student attendance in a secure database with an RFID-based [2] attendance recording system to simplify record keeping and maintain the student's attendance as a simple procedure.
- **Cheating Detection:** Simply run AI-based monitoring during exams and in real time, where it detects abnormal behavior for notifying test administrators.
- **User-Friendly Dashboard:** You can provide faculty with access to attendance data, alerts and reports through a contemporary interface.
- **Scalability and Flexibility:** The system is designed to optimize attendance for a large body of students and can be modified for institutional specificity.
- **AI and Web Systems integration:** The design will ensure that the RFID module interacts seamlessly with the Flask AI detection to the MERN dashboard above the above
- **Accuracy and Reliability:** The system ensures data accuracy with compliant RFID protocols and state-of-the-art ML models for detection.

1.3 Final Outcomes and Deliverables

- **Dependable and Safe Attendance Tracker:** The product designed will be an effective web application, which would allow educational institutions to easily implement RFID technology to securely record attendance.
- **AI-Enabled Cheating Monitoring:** The system utilizes YOLOv8-Face, RetinaFace, InsightFace, Mediapipe and OpenCV for additional monitoring capabilities. It recognizes and analyzes student behavior during examinations, as well as strange events such as multiple faces, abnormal head or eye movements, or other suspicious gestures. When observed such behavior, the system immediately alerts the invigilator or teacher to what to do immediately.
- **Centralized Dashboard:** A web-based MERN dashboard where faculty and administrators can access attendance records, review exam monitoring alerts and generate detailed reports.
- **Seamless Integration:** Smooth communication between the RFID attendance module, Flask-based AI detection engine and MERN frontend to ensure reliability and real-time updates.
- **Scalable and Adaptable Platform:** A system designed to handle large student populations and configurable for various institutional needs.

1.4 Motivations

ClassTrack was created to deal with these problems head-on, responding to the growing stresses of maintaining academic integrity and overseeing large numbers of students. In addition to human error or proxy attendance, many hand-proctoring tests rarely use sophisticated cheating tactics; regular attendance procedures are cumbersome, in other words, human error or proxy attendance.

In addition to RFID and AI surveillance, the institutions now have the opportunity to apply innovative methods that will improve efficiency and fairness. ClassTrack was meant to:

- Ensure fair examinations by automatically identifying suspicious test-taking behavior.
- Provide a fast, reliable management system that reduces human error and deters proxy attendance
- Provide a trusted, centralized Data system for organizations to manage records and gain insights.
- Leverage Technologies such as RFID, vision, YOLOv8-face + Retina Face and InsightFace detection and OpenCV to increase academic integrity and trust.

By combining automation and artificial intelligence, ClassTrack response to the demands of learning environment today and showing that organisations can maintain their integrity and effectiveness.

1.5 Summary of Report Structure

- In **Chapter 1** we introduced the class track system including the problem statement the project purpose project and design objectives, the deliverables and final results, rationale and motivation and description of the overall report structure
- In **Chapter 2** we explain practice limitations and professional standards that were taken into consideration during the development of class track including ethical, technical, financial and environmental limitations
- In **Chapter 3** we explain the detail of design of the system architecture rules and restrictions, tools and methods, designs specification and final cost projection with discussion
- In **Chapter 4**, we present the implementation of the system, testing procedures and protocol analysis of results and possible improvement for future growth
- In **Chapter 5**, we discuss project management details, such as the Gantt chart task schedule throughout the project, obstacles encountered and resources and materials have used during the project
- In **Chapter 6**, we provide an overview of our ClassTrack results and conclude with the discussion of ClassTrack's potential role with attendance control and cheating detection
- In **Chapter 7**, we provide a contribution statement outlining the team members, the specific roles for each team member and percent contribution
- In **Chapter 8** we provide ClassTrack's references, biography, and appendices

2 Professional Standards and Practice Constraints

The ClassTrack project was established with an array of engineering principles and constraints considered, in order to maximize the chance of success and conform to industry standards. The design process for ClassTrack considered both technical and ethical components in order to allow students to be exposed early to professional design, project management, and effective communication. Each design decision was made not solely for the design itself, but also for the wider implications of each design, design value, sustainability, affordability, and legality. In the next sections we detail primary limitations of public facing, and some profession elements that influenced the project.

2.1. Manufacturability Constraints

ClassTrack was developed with long-term maintainability and scalability in mind. Current web development approaches—particularly the **MERN stack** combined with **Python Flask**—enable the overall solution to be deployed or hosted on local or online servers without major modifications to the libraries or architectural design. The system avoids vendor lock-in by being largely constructed from **open-source web frameworks**, and it supports seamless integration of **AI functionalities** using tools such as **Mediapipe**, **YOLOv8-Face**, **RetinaFace**, **InsightFace**, and **OpenCV**. Each of these could be replaced or incorporated in the future with alternative tools as technology evolves.

To change or modify the system, based on bugs, new features or the integration with the latest AI workflows, this transformation is safe and straightforward because there is a modular architecture embedded within the platform. The practices utilized reflect standards of professional software engineering, that emphasized modularity, adaptability and scalability as essential factors in the production of products in dynamic environments. Finally, this design mitigates technical hurdles for those schools that implement ClassTrack shifting toward collaborative planning and sustainable lifelong sustainability.

2.2. Economic Constraints

The development of ClassTrack was mostly based on affordability. The open-source frameworks used (Node.js, React, MongoDB, Express, and Python-based image recognition libraries) [9] significantly reduce licensing and availability fees for educational institutions that typically experience constrained budgets. The only large hardware investment is student ID cards and a low-cost RFID reader/writer [2], both of which are widely available and inexpensive. Additionally, effective agile project management practices further reduced financial exposure, as they required dividing work into smaller milestones, reducing waste in total development, and maintaining consistent resources for completion. ClassTrack is an example of how professional engineering practices have provided a permanent approach to satisfy institutional needs while remaining within budgetary constraints and a commitment to functional cost-effectiveness.

2.3. Sustainability Constraints

Long-term sustainability was considered through both technical design decisions and the system as a whole. Building on MERN and Flask will develop in universally accepted frameworks that will likely be up to date and of good documentation for many years in the future. Incremental upgrades can now be supported by a modular design that extends the lifespan of the product, and considerably reduce the need for a total redesign of the system. Because of cloud scalability, educational institutions can expand their student capacity without entering into new major investment hardware or design upgrades. ClassTrack also directly reduces administrative waste and promotes sustainability as it eliminates the requirement for manual monitoring of exams and paper-based attendance. Further, mere contemporary engineering practice and principles, sustainability as an environment consideration is viewed as a metric of a system's long-term viability.

2.4. Environmental Constraints

Even though ClassTrack is basically a software solution, environmental impact played into the design. The primary driver of the system's energy footprint is its server use, which can be minimized, to some extent, through either institutional servers that utilize efficient processors or cloud server provisions. All of the major cloud providers utilize some form of renewable energy sources which have greatly diminished the environmental effects of digital solutions for institutions. The use of AI-powered, automated services for exam and attendance monitoring decreases the resource costs required for human labor, paper record maintenance, and in-person proctoring. The need for hardware manufacturing is further diminished because RFID [10] readers and cards consume very little resources to manufacture, thereby reducing the environmental impact related to the manufacturing process. This suggestion relates to responsible engineering practices that seek to balance environmental responsibility and innovation.

2.5. Health and Safety Constraints

ClassTrack was created around principles of privacy, safety and health. The use of secure logging and storage processes means that the system complies with institutional and legal requirements while protecting sensitive student data. The use of RFID-only attendance now means calls can be done without crowding and therefore reduces the risk of safety issues in large lecture halls or classrooms. The AI [11] served surveillance is designed to be non-intrusive and only searches for any signs of academic dishonesty rather than collecting extraneous personal information. This keeps the students' intellectual integrity, is less stressful, and is protected by dignity. This comes from this commitment to safety and wellbeing that reinforce engineers' professional responsibility to protect end users in every element of system design.

2.6. Ethical Standards Constraints

The ethical foundation of the ClassTrack project was ethics. The system is simply collecting the information needed to function, such as attendance reporting and AI generated notifications, to prevent unnecessary surveillance or obstruction. Data retention is done according to privacy law and institutional policy that provides accountability and transparency in record keeping. Because students can be certain their data will not be stolen, the administrators are fully responsible for their data ownership. In fact, a surveillance process has been designed to promote academic integrity without presenting a hazard to trust. The project adopts ethical principles developed within the project in line with the practice of engineering practice that promotes the rights of the user and privacy.

2.7. Social Values Constraints

ClassTrack was designed to help empower equality, diversity and trust in learning environments. The robust RFID [2] storage ensures that all children are treated properly and that all visitors to the facility receive constant care. The AI monitoring tool allows for equitable assessments and maintains academic integrity by alerting the agent of cheating. The system was easily understood and easy to access through the dashboard and user interface to technical and non-technical staff. ClassTrack fosters integrity and accountability that is reflected in the larger social values of education and builds trust among students, educators and the educational system.

2.8. Political Constraints

The ClassTrack was also a followup to political and regulatory principles. By integrating a system architecture that limits risk to distressed institutions and has accountability for data governance under the terms of GDPR and various other local data protection legislation, it developed its own system architecture whose legal protections are based on compliance with GDPR and other local data protection regulations. Under those conditions the institution is accountable and credible, and providing robust defense for the rights of its users. The construction of ClassTrack also allows for e-learning, accountability and transparency, including political practices that move the digital transformation of education through the national and global arena. These political perspectives contribute to ClassTrack's utility and present it as a progressive educational solution that can be applied with regulations.

3 System Architecture and Detailed Design

3.1 Requirement Specifications

The goal of ClassTrack was to design a solution that would allow schools to be able to adequately, reliably and safely account for student attendance so that the students could actually attend a test while not being cheated by cheating on the exam. Neither of the two standard methods of attendance accounting—paper-based sign-in or manual roll call—was done adequately, efficient, and can be altered. Second, providing a manual examination invigilation system that is difficult to keep an eye on, inconsistent with the requirements of other test proctors and whose many items may be exposed to subtle cheating. These limitations can be overcome by ClassTrack and its RFID-supported attendance recording system with an AI-based exam monitoring solution that addresses accountability, fairness, and accuracy in educational environments.

To describe what the system functions, we identified the following characteristics:

- **RFID-Attendance Feature:** The system should allow students to record their attendance by swiping their RFID [13] cards with a USB RFID reader/writer. It is recommended that attendance is recorded on the central database once scanning, and no humans would be required to enter the information into the system. This will enable attendance monitoring accurately and correctly, especially for large classrooms that cannot do this manually.
- **Detection of Cheating:** Video monitoring is the feature of the system that can detect suspicious behavior or abnormalities during examinations. It uses Mediapipe, YOLOv8-Face, RetinaFace, InsightFace, and OpenCV to analyze video analysis and behavioral tracking. The system can identify key indicators such as multiple faces on one frame, irregular or repetitive head movements, and other banned behaviors. This is useful because the invigilator can easily identify any possible cheating attempts and can assist in the detection process and help stabilize the test-taker's accuracy.
- **User Authentication:** Faculty and administrators must validate their login to secure and limit access. A list of user roles that can be described to allow only appropriate personnel to access the system to view attendance records, manage an exam, and update settings in the system.
- **Centralized Dashboard:** The MERN-based web interface will provide a comprehensive dashboard that directs users to their own cheating alerts, generate reports, and provide attendance logs; all in one user-friendly dashboard. The dashboard will allow the user to manage records efficiently by filtering, searching and exporting information.
- **Notifications and Alerts:** Administrators must send up real-time alerts to the administrators when cheating activity is detected during an exam. Alerts should be presented directly on the dashboard and, in cases of need, delivered via email or

push notifications. It also allows for timely action in response to detectable suspicious activities.

- **Data Storage and Management:** All monitoring and attendance data are to be stored in a MongoDB database that provides restrictions, scalability, and indexing. A robust backup system should be designed to prevent data loss. The storage system must provide confidentiality, integrity and compliance with institutional requirements.
- **Scalability:** The system must have a capacity to cope with large student populations of multiple classes and exam rooms simultaneously without losing performance or degradation. This includes managing multiple video feeds, multiple RFID. [2] scans and extensive data retrieval for reporting.

3.2 Design Setup

The ClassTrack design showcases professional engineering practices of modularity, scalability, and security. The system's architecture comprises microservices, based on AI, with web technologies, to provide a robust and flexible solution. Each design decision was considered maintainability, cost-efficiency and overall consistency with industry best practices.

React.js was used to build the frontend of the system; the front end is designed in React.js to be a simple, dynamic and responsive application for academics and administrators. The component-based design of React provides the developer team with reusable user interface components that can be used for maintenance and quick changes to feature. Access is easily possible by users with differing technical backgrounds and has been built in to access the dashboard attendance logs and exam alarms.

The backend has been built using Node.js and Express.js and it is centralized for communication, API calls, and user authentication. This design choices, and implementation, leverages the non-blocking event-driven architecture of Node.js that ensures benefits to applications for scale, and concurrent connections, such as RFID scans, and real-time monitor updates, to scale and multiple application applications. Express.js provides lightweight and modular framework for RESTful API creation that makes it possible to integrate with other system components and frontend.

MongoDB was chosen as the storage database for data because it is used for storing large amounts of unstructured and semistructured data. MongoDB's document-oriented format provides the flexibility for data processing, with ClassTrack producing both structured attendance records and unstructured monitoring results. A database could grow as big organizations need it, but could also provide quick query performance due to indexing, replication, and sharing.

The Python Flask [7] microservice uses AI features for face recognition and cheating detection to operate independently but links well to the MERN application. Because it is

modular, the AI service can process live video feeds utilizing Mediapipe, YOLOv8-Face, RetinaFace, InsightFace, and OpenCV, and send them back to the main dashboard in real time via face recognition outputs or cheating alerts. This is so that while a system remains stable and free of errors, it is able to maintain the stability and flexibility of the Flask microservice by decoupling AI operations inside the microservice and it can be changed or replaced within the system without altering the architecture of the system.

The system is divided into three main sections:

- **Attendance Module:** With a responsibility to implement RFID this module collects data on student attendance and saves it safely in the MongoDB database. It offers quick RFID scan processing and instant confirmation for instructors and students.
- **Cheating Detection Module:** The cheating detection module is based on Flask-based AI service and analyzes video feeds to detect unusual behavior. The function of the software is to notify the dashboard of unusual behavior, send notifications that are sent to the dashboard and communicate with the backend through the APIs.
- **Dashboard Module:** Developed with React and backed by Node.js/Express, this system's user-facing component gives administrators and professors access to real-time insights, logs, and reports. Additionally, this module has alerting mechanisms for prompt exam reaction.

Every part of the system is rooted in security. All sensitive information including monitoring findings and student attendance, is secured in the database and secured in transition. Access control policies avoid the risk of misuse of privilege, and authentication systems enable only authenticated users to access system functionality. Using these methods, ClassTrack poses the engineering professional requirements regarding securing and protecting data.

3.3 Standards and constraints

- ClassTrack has been designed on the basis of defined standards in order to ensure compatibility, security and long-term sustainability, along with real-world limitations that affect system performance and application.
- **Web Development Standards**
ClassTrack is a professional web development framework that draws on the well-known MERN stack - MongoDB, Express, React, Node.js - with its MERN stack. This ensures scalability, support, and cross platform support for desktop and mobile browsers. With this approach, the system can be extended in the future and the design is modular.
- **AI and Computer Vision Standards**
The cheating detection module uses YOLOv8-Face, RetinaFace, InsightFace,

Mediapipe and OpenCV in conformance with industry recognized standards for accuracy and performance in computer vision. The system is built upon pre-trained and highly specialized models, and maintains reliable and accurate detection outcomes, reducing the risk of untested or unstable AI tools. This means that the cheating detection process is robust and reliable in its application to best practices of AI-driven surveillance and behavioral analysis.

- **RFID Integration Standards**

RFID components are guaranteed to be compatible with Standard student ID cards and readers. This allows vendors to be protected and hardware supply is flexible and affordable.

- **Security Standards**

Encrypted APIs ensure all component interactions and data management is compliant with local privacy laws and the GDPR. Mechanisms for authentication and access control guard against illegal use of private academic information..

- **Constraints**

Additionally, the system must function under specific constraints. If several exam rooms are monitored on a small amount of hardware, performance could suffer. Centralized reporting and real-time monitoring depend on a reliable internet connection. Additionally, the efficacy of AI-based cheating detection might be impacted by external factors like lighting and camera quality.

3.4 Tools and methods

ClassTrack's design incorporates modern development tools and frameworks to attain dependability, scalability, and efficiency. Every element was chosen with care to guarantee that the system satisfies institutional standards while continuing to be simple to maintain and grow in the future.

- **Frontend**

React.js was used to create the web-based dashboard because of its capacity to create dynamic, modular, and responsive user interfaces. Whether using a desktop or mobile browser, React guarantees that academics and administrators can engage with the system without any issues. Also, its component-based architecture makes fixes and subsequent improvements easier.

- **Backend**

Node.js and Express.js operate the backend of the system, which is primarily driven by Node.js and Express.js, which provides a solid foundation for managing RESTful APIs. These APIs ensure that logic is processed correctly and that communication between front-end, database, and AI modules is made efficient. Node.js is event driven and nonblocking, so the backend can be effectively scaled

to handle multiple concurrent requests, such as large exams or large attendance logs.

- **Database**

The main database, MongoDB, contains AI monitoring records, student information, and attendance records. Unlike MongoDB which only accepts schema and indexing capabilities, which allow real-time querying, MongoDB also allows for more advanced operations for large datasets. That is vital for universities with a growing student population and many exam cycles.

- **Integration of RFID** RFID Student attendance is tracked automatically using the ISO/IEC certified RFID cards and USB RFID reader/writer. While the student scans their card, they communicate directly with the backend, recording attendance information for that individual. This automation eliminates administrative burden but will still provide fairness and accuracy in attendance records.

- **AI and Computer Vision** ClassTrack uses Mediapipe, YOLOv8-Face. [1] The Python Flask microservice provides access to a combination of RetinaFace, InsightFace and OpenCV and AI in the exam-monitoring module. These technologies enable a ClassTrack system to efficiently recognize multiple faces, to identify unusual head movements, and to understand the possible gestures associated with prohibited behavior during a series of examinations. The solution also creates modularity in that it is able to organize AI functions within Flask services so it can be updated and replace, model replacement, or upgrade detection models without altering the core application or the stability.

- **Authentication**

Role-based authentication is used to safeguard student privacy through administrators and faculty. JWT (JSON Web Tokens) is used to protect access, guaranteeing session integrity and preventing unwanted access to system configurations, reports, and warnings. This strategy is in line with industry standards for web application security.

- **Reporting and Analytics**

Faculty and administrators can analyze cheating alerts and create attendance records using the MERN dashboard. Filtering, visualizing, and exporting data can help in decision-making and institutional policy compliance. By keeping clear records of student activity, the analytics feature also increases responsibility.

| Category | Tools and Methods |
|-----------------------|---|
| Development Framework | React.js (Frontend), Node.js + Express.js (Backend) |
| Database | MongoDB (Attendance, student and monitoring data) |
| RFID Integration | USB RFID Card Reader/Writer (ISO/IEC compatible) |
| AI & Computer Vision | Python Flask, Mediapipe, YOLOv8-Face, Retina Face, Insight Face, OpenCV |
| Authentication | JWT-based authentication for faculty/admin access |
| Reporting & Analytics | MERN Dashboard with role-based access and reporting features |

Table 1 Tools and methods

3.5 Data Collection

Only the information required for the ClassTrack system's primary purposes of admin reporting, exam monitoring, and attendance tracking is gathered. The solution maintains compliance with organizational standards and privacy standards while enabling enhanced functionality by restricting data gathering to necessary categories. The following kinds of information are gathered and safely stored:

- Attendance Data**
 RFID card scans are used to automatically record student attendance. The system records important information such as the student ID, timestamp, and class session details when a student taps their RFID [2] card on the USB reader/writer. This produces an accurate digital record that is accessible at any time and removes inaccuracies that are frequently found in manual roll calls.
- Student Information**
 The system keeps basic institutional information such the student's name, enrollment number, and assigned RFID card ID in order to map attendance data to specific students. The database does keep only the bare minimum of data required to function and avoids storing excessive personal information.
- Cheating Detection Data**
 The AI module analyzes live camera footage on tests and detects anomalous behavior such as multiple faces in a frame, odd or regular head movements, or suspicious movements. ClassTrack doesn't keep raw video footage in order to save

storage costs and preserve privacy. Rather, it documents detection occurrences, system-generated logs and flagged alerts, which are sufficient for administrators to investigate potential cases of misconduct.

- **Authentication Data**

Secure authentication procedures are used to safeguard administrator and faculty accounts. The system is secured with a JWT-based authentication system and retains only the most important credentials, including login, email, and hashed passwords, according to current best practice for cybersecurity. Private data is never stored in plain text.

- **System Logs**

Operational logs are stored for both auditing purposes and traceability. System logs contain events such as RFID scans (11), detections, login attempts, and dashboard activity. System logs are important for troubleshooting, understanding usage patterns over time, and being accountable in a dispute.

Information collected is stored in MongoDB, which is secured with role-based access policies to prevent unauthorized access. Procedures are in place to backup information to prevent loss of data and if sensitive information like login credentials are captured, the information is stored encrypted or hashed.

3.6 System Architecture

ClassTrack's design is layered and modular, which results in improved security, maintainability, and scalability. The platform is designed to compartmentalize each function into its own discrete layer to ensure that any updates and improvements can be done without affecting the system in its whole. This design also enables seamless communication between elements so that it is easy to incorporate RFID attendance data, AI detection results, and reporting features.

- **Front-End Presentation Layer:** This layer is created using React.js and provides users with an intuitive experience for academics and administrators. Users can check attendance records, monitor exam alerts, and generate analytics reports via the web-based dashboard. Responsive design allows users to interact with the system via desktop, laptop, and tablet devices.
- **Business Logic Layer:** The system's functions run through the layers Node.js and Express.js. This layer handles authentication requests, directs API [17] communications, processes the Web attendance data from RFID scans, and connects the front-end to other application modules. Its application logic provides ClassTrack with the ability to ensure that important functions remain intact while also expanding when needed in institutions.

- **AI Processing Layer**

This layer is built around Python Flask, with an additional computer vision pipeline, Mediapipe, YOLOv8-Face, RetinaFace, InsightFace, and OpenCV, to detect cheating in real time. It analyzes the emerging video frames from examination sessions looking for obvious behavior like multiple faces, irregular head movements, or other suspicious behaviors. The video inputs a live video input and then the businesslogic layer sends a data set that is in real time and translated into a single, interpretable, product for further processing. The AI component is designed to be fully modular, allowing it to function as an independent microservice—ensuring that it can be easily updated, replace or improvise detection models and algorithms without altering the architecture of the system.

- **Backend Data Layer**

MongoDB, in addition to its MongoDB hosting servers, stores data regarding student information, attendance records, login credentials, and results of the AI detection. Its document-based storage provides fast querying across large structured datasets and flexible data input. The system has the capacity to handle more students, more exam rooms, and larger data set scale-out capability of MongoDB compared to MongoDB.

Both functional layers perform the same function to create a seamless process of AI-driven exam proctoring and RFID-based attendance. Within the modular system, institutions will be able to securely capture, monitor and access canonical, real-time data through a central dashboard and, via the modular system, maintain uptime and maintenance.

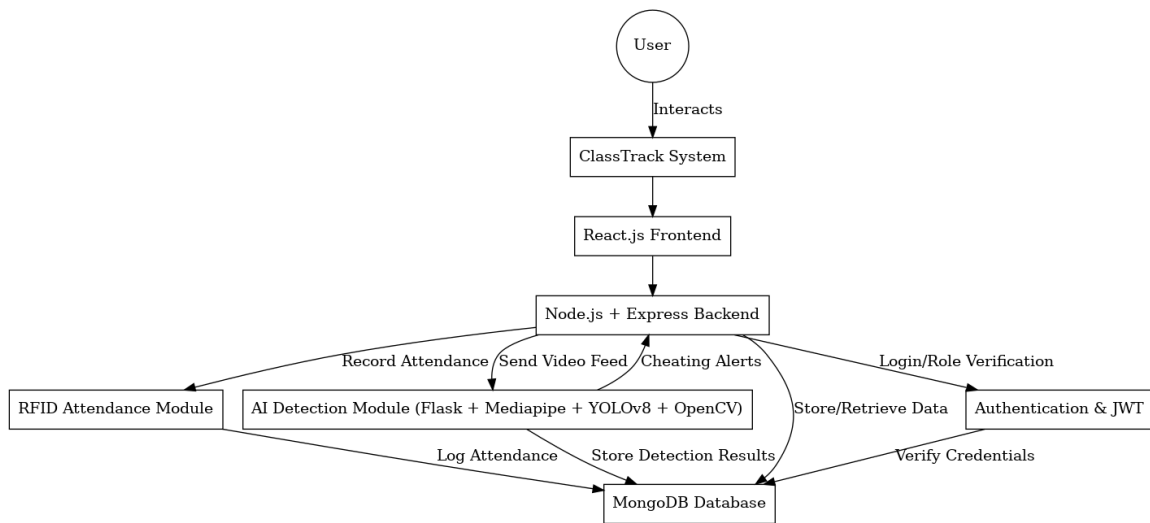


Figure 1 System Architecture

3.7 Final Cost Analysis and Discussion

ClassTrack uses open-source technologies and tools to maximize the development costs of their application. The system uses Python-based technologies such as Flask, Mediapipe, YOLOv8-Face, RetinaFace, InsightFace, and OpenCV [5] to implement AI-driven web applications and web development through the MERN stack (MongoDB, Express.js, React.js, and Node.js). This would avoid licensing fees associated with proprietary solutions. While these open-source technologies are often relatively cheap, large developers groups and detailed documentation can speed development, speed troubleshooting, and thus save money on technical support.

- **Costs of Doing Business**

For ClassTrack, there is no physical physical space, and only cloud infrastructures like AWS, Google Cloud or institutional servers that offer a highly scalable and resilient platform to help it scale and survive and for a significantly less costly setup and maintenance costs. The cloud platforms also create a system with flexibility to accommodate different budgets and allow organizations to start cheap, but increase in size as they need to. The primary operational costs of attendance records, AI detection logs, and concurrent users will depend on the amount of attendance records, AI detection logs and concurrent users and the possible need to require higher level services in the future.

- **Costs of Labor**

The system required development expertise across multiple domains, including full-stack web development, database design and computer vision. The use of open-source frameworks and modular architecture reduced development effort by allowing different modules (attendance, AI detection, dashboard) to be integrated without redundancy. But, the initial configuration of AI models and RFID hardware needed additional development hours, including testing and optimization.

- **Costs of Hardware**

ClassTrack, unlike only software-based platforms, requires little or no hardware to operate. This includes USB RFID readers/writers, students' RFID cards and webcams for exam monitoring. These are low upfront costs, but the hardware is reusable over the years, making them a low long-term cost. The standardization and affordability of RFID technology means replacement or expansions can be purchased cheaply without the need to ask specific vendors.

Costs of Security

This is important to Data Protection, as ClassTrack contains sensitive academic information. At development, security measures such as JWT authentication,

encrypted APIs and secure storage practices were adopted, some cost added but necessary in order to adhere to GDPR and institutional policies. While cost-effective initial initiatives are effective, enduring security should be updated, vulnerability tests and audits, and is necessary to maintain a strong security over time with a monthly cost but vital effort to enhance trust and compliance.

| Component | Cost Description | Estimated Cost |
|-----------------------|--|----------------|
| Development Cost | Web app design, coding, RFID integration, AI model setup and testing | \$4,500 |
| Open-Source Libraries | MERN stack, Flask, Mediapipe, YOLOv8, OpenCV (free) | \$0 |
| RFID Hardware | USB RFID Reader/Writer and 100 RFID Cards | \$150 |
| Camera Equipment | Standard HD webcams for exam monitoring (per classroom) | \$200 |
| Cloud Hosting | MongoDB Atlas + cloud server deployment (based on usage tier) | \$100 |
| Server Maintenance | Updates, bug fixes and monitoring | \$150 |
| Testing Tools | Debugging tools and cloud-based testing environments | \$120 |

Table 2 Final cost

4 Evaluation and Improvements

4.1 Implementation

In phases, ClassTrack was developed in phases with its own set of challenges and some need to be examined cautiously. Frontend development, backend integration, RFID hardware configuration, AI model deployment, and system testing were all included in the project. This is a brief overview of the main phases and challenges faced in implementation.

- **Front-End Development (React.js):**
 - **UI/UX Design Consistency:** Providing a clean and intuitive design for admin and faculty dashboards was one of the main concerns. The challenge was to display large datasets, such as attendance log, AI detection alerts, responsively and consistently across devices.

- **Data Visualization:** Implementing charts and tables for attendance analytics required third-party React libraries, which had to be optimized for speed and clarity.

```

Login.jsx
pages > Login.jsx
26 }
33 return (
34   <div className="max-w-md mx-auto mt-20 p-8 bg-white rounded shadow-md border border-gray-200">
36     <form onSubmit={onSubmit} className="space-y-4">
37       <div>
27
28     } catch (e) {
29       setError(e?.response?.data?.message || "Login failed");
30     }
31   };
32
33   return (
34     <div className="max-w-md mx-auto mt-20 p-8 bg-white rounded shadow-md border border-gray-200">
35       <h2 className="text-2xl font-bold mb-6 text-center">Login</h2>
36       <form onSubmit={onSubmit} className="space-y-4">
37         <div>
38           <label className="block text-sm font-medium text-gray-700 mb-1">Email</label>
39           <input
40             className="w-full px-4 py-2 border rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"
41             value={email}
42             onChange={(e) => setEmail(e.target.value)}
43             type="email"
44             required
45           />
46         </div>
47         <div>
48           <label className="block text-sm font-medium text-gray-700 mb-1">Password</label>
49           <input
50             className="w-full px-4 py-2 border rounded-md focus:outline-none focus:ring-2 focus:ring-blue-500"
51             value={password}

```

Figure 2 Login Page

```

InstructorDashboard.jsx
pages > instructor > InstructorDashboard.jsx
33 <main className="max-w-6xl mx-auto px-6 py-12">
34   { /* Header */ }
35   <header className="mb-12 flex items-center gap-4">
36     <div className="h-12 w-12 rounded-full bg-blue-100 flex items-center justify-center text-blue-700 font-bold text-lg">
37       {displayName[0]?.toUpperCase() || "I"}
38     </div>
39     <div>
40       <h1 className="text-3xl font-bold text-gray-900">
41         Instructor Dashboard
42       </h1>
43       <p className="text-gray-600">
44         Welcome back, { " " }
45         <span className="font-medium text-blue-700">{displayName}</span>
46       </p>
47     </div>
48   </header>
49
50   { /* Navigation Grid */ }
51   <section className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 gap-6">
52     {navItems.map(({ to, label, icon: Icon, description }) => (
53       <Link
54         key={to}
55         to={to}
56         className="group bg-white border border-gray-200 rounded-xl p-6 shadow-sm hover:shadow-md hover:border-blue-500
57         focus:outline-none focus:ring-2 focus:ring-blue-400 transition-all"
58       >
59         <div className="flex items-center justify-center h-12 w-12 rounded-lg bg-blue-50 text-blue-600 mb-4 group-hover:bg-blue-100
60         group-hover:scale-105 transition-transform">
61           <Icon size={24} />

```

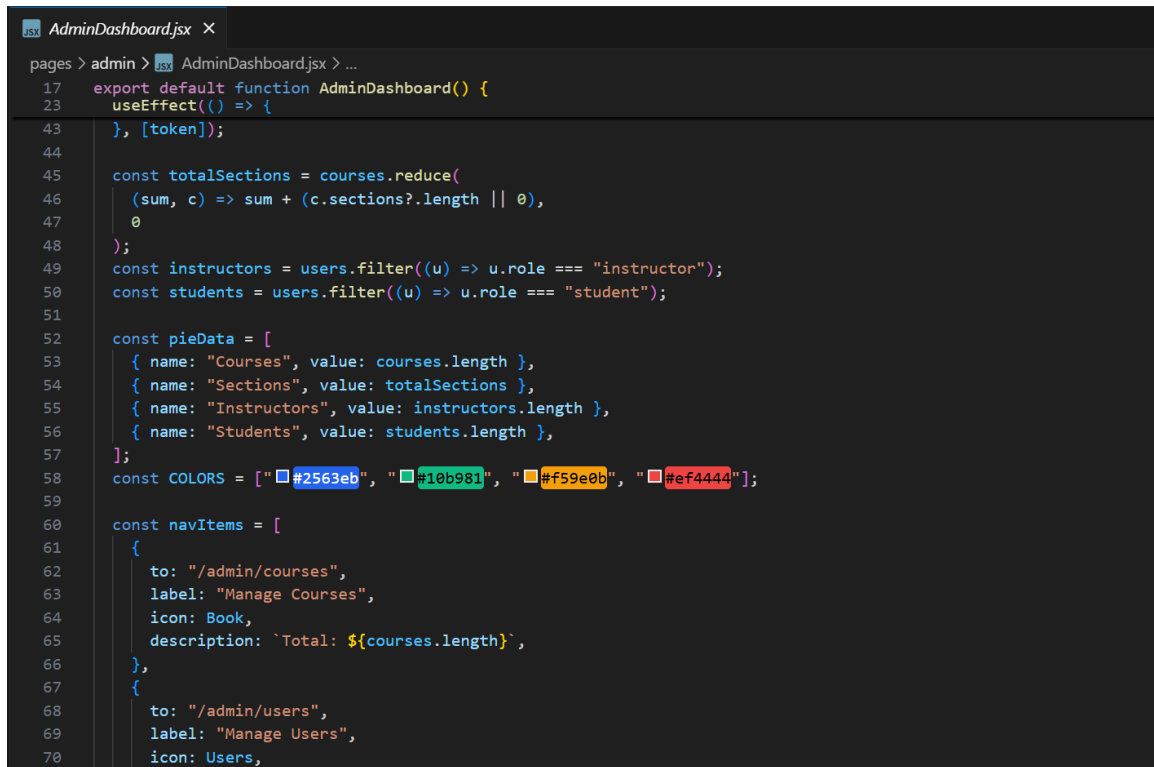
Figure 3 Instructor Dashboard

```

InstructorMonitoring.jsx
pages > instructor > InstructorMonitoring.jsx > ...
117     setEnding(false);
118   }
119 };
120
121 return (
122   <div className="p-6 max-w-7xl mx-auto">
123     <div className="flex flex-col lg:flex-row lg:items-center lg:justify-between gap-3 mb-6 sticky top-0 bg-white z-10 pb-2 border-b">
124       <div>
125         <h2 className="text-2xl font-bold flex items-center gap-2">
126           <Video className="text-blue-600" />
127           Live Monitoring
128         </h2>
129         {sessionInfo && (
130           <p className="text-gray-700 mt-1">
131             {sessionInfo.course?.code} -{" "}
132             <b>{sessionInfo.course?.name}</b>{" "}
133             <span className="text-gray-500">
134               {(sessionInfo.section?.name)}
135             </span>
136           </p>
137         )}
138       </div>
139       <div className="flex gap-3 items-center">
140         {!detecting ? (
141           <button
142             onClick={startDetection}
143             className="px-5 py-2 bg-green-600 text-white rounded shadow hover:bg-green-700"
144           >
145             Start Detection
146         ) : (
147           <button
148             onClick={stopDetection}
149             className="px-5 py-2 bg-red-600 text-white rounded shadow hover:bg-red-700"
150           >
151             Stop Detection
152         )}
153       </div>
154     </div>
155   </div>
156 )

```

Figure 4 Monitoring System



```

AdminDashboard.jsx
pages > admin > AdminDashboard.jsx > ...
17 export default function AdminDashboard() {
23   useEffect(() => {
43     }, [token]);
44
45     const totalSections = courses.reduce(
46       (sum, c) => sum + (c.sections?.length || 0),
47       0
48     );
49     const instructors = users.filter((u) => u.role === "instructor");
50     const students = users.filter((u) => u.role === "student");
51
52     const pieData = [
53       { name: "Courses", value: courses.length },
54       { name: "Sections", value: totalSections },
55       { name: "Instructors", value: instructors.length },
56       { name: "Students", value: students.length },
57     ];
58     const COLORS = ["#2563eb", "#10b981", "#f59e0b", "#ef4444"];
59
60     const navItems = [
61       {
62         to: "/admin/courses",
63         label: "Manage Courses",
64         icon: Book,
65         description: `Total: ${courses.length}`,
66       },
67       {
68         to: "/admin/users",
69         label: "Manage Users",
70         icon: Users,

```

Figure 5 Admin Dashboard

- **Backend Development (Node.js + Express):**
 - **API Design and Integration:** Testing safe APIs for attendance recording, cheating detection alerts and authentication required rigorous testing to make sure that communication between the MERN stack and Flask AI module would be seamless.
 - **Authentication and Role Management:** Testing safe APIs for attendance recording, cheating detection alerts and authentication required rigorous testing to make sure that communication between the MERN stack and Flask AI module would be seamless.
- **RFID Attendance Integration:**
 - **Hardware Communication:** Using USB RFID reader/writer with backend on the backend, real-time updates were not feasible. It was necessary to be sure that device drivers and serial communication protocols were configured properly to ensure that logs would not drift out from logs.
 - **Scalability Issues:** Backend optimization is needed to avoid duplicate or missed records using multiple RFID scans during an enormous classroom entry.

• **AI Cheating Detection Module (Flask + Mediapipe + YOLOv8-Face + RetinaFace + InsightFace + OpenCV):**

- **Model Deployment:** YOLOv8-Face, RetinaFace and InsightFace are resource intensive combined with live video streams. This required improving the AI pipeline in order to detect real-time detection while ensuring accuracy.
 - **Edge Cases in Detection:** Detection accuracy was affected by movement, camera resolution, and variable lighting conditions. Some testing and threshold adjustment needed to alleviate false positives and improve reliability.
 - **Communication with MERN App:** Efficiencies were used to ensure that alerts and detection are transmitted automatically and over the web from the Flask AI microservice in real time to the Node.js backend while facilitating smooth integration within the system.
- **Database Setup (MongoDB):**
- **Efficient Data Storage:** Schemas on student data, attendance data, and AI detection logs needed to balance speed and scalability. It was used to manage a great deal of data. indexing techniques were employed.
 - **Data Security:** Enforcing role-based access and encrypting sensitive fields (such login passwords) were essential for adhering to privacy regulations.
- **System Testing and Deployment:**
- **Integration Testing:** several synchronization issues were reported with the RFID hardware, Flask AI module, MERN dashboard, which had to be fixed before deployment.
 - **Deployment Challenges:** Cloud instances needed enough GPU/CPU power to accommodate AI tasks for both the Flask AI service and MERN application.

4.2 Testing

Testing was crucial for ClassTrack's success in order to ensure the system worked as planned, maintained accurate data, and was reliable in real-world classroom or testing. Unit testing, integration testing, white box testing, and black box testing were all used.

• **Black Box Testing:**

Without examining the internal code structure, this testing approach was about verifying the systems' requirements for function. The tests were also tested on basic aspects, including:

- Identified student attendance by RFID card scans.
- Registration and verification of login credentials for faculty/admin.
- Accurate display of attendance data on the dashboard.

- Correct triggering of AI cheating detection alerts, such as multiple faces, abnormal head movements.
- Responsiveness of the user interface during high-volume RFID scans.
- **White Box Testing:**
This approach validated the internal logic and algorithms of ClassTrack. Specific checks included:
 - Accuracy of database queries and indexing for attendance retrieval.
 - Communication flow between Node.js backend and Flask AI module for real-time cheating detection.
 - Correct functioning of JWT-based authentication and role verification.
 - Error handling for cases such as RFID scan failures, network connectivity issues and AI module timeouts.
- **Unit Testing:**
Individual functions and modules were independently tested to ensure reliability. Examples are:
 - RFID reader integration and parsing of student IDs.
 - API endpoints for attendance login, authentication and reporting.
 - AI module for detection of faces and suspicious behavior using YOLOv8-Face and Mediapipe.
 - Secure storage and retrieval of attendance and exam monitoring logs from MongoDB.
- **Integration Testing:**
This phase was essential to ensure synchronized interaction between subsystems. Testing validated it.
 - RFID hardware communicated correctly with the backend and updated MongoDB in real time.
 - Flask AI detection results were received by the Node.js backend and displayed accurately on the React dashboard.
 - Faculty/admin authentication controlled access properly across the system.
 - Reports generated from attendance and AI logs matched the actual collected data.

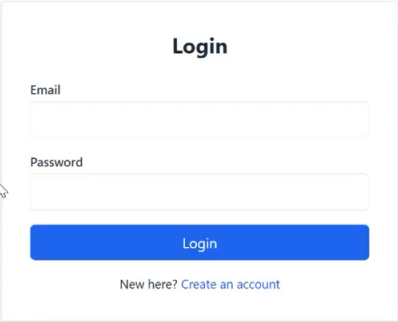
| Test Case ID | Module | Input | Expected Output | Actual Result | Status |
|--------------|-----------------------|--|--|---------------|--------|
| TC-01 | RFID Attendance | Student scans valid RFID card | Attendance is logged with correct student ID, timestamp, and class session | As Expected | Pass |
| TC-02 | RFID Attendance | Student scans invalid/unregistered RFID card | System rejects scan and displays error message | As Expected | Pass |
| TC-03 | Authentication | Admin enters correct username & password | Admin successfully logged in and redirected to dashboard | As Expected | Pass |
| TC-04 | Authentication | Admin enters incorrect password | System denies login and shows "Invalid credentials" | As Expected | Pass |
| TC-05 | AI Cheating Detection | Camera feed with a single student face | No cheating alert triggered | As Expected | Pass |
| TC-06 | AI Cheating Detection | Camera feed with multiple student faces | Cheating alert triggered and logged in database | As Expected | Pass |
| TC-07 | AI Cheating Detection | Student making abnormal head movements | Alert triggered with label "Suspicious Behavior" | As Expected | Pass |
| TC-08 | Database Storage | Retrieve attendance report for a given date | Report generated with accurate student IDs and timestamps | As Expected | Pass |
| TC-09 | Dashboard Analytics | Faculty requests summary attendance report for one month | Graph/summary generated correctly | As Expected | Pass |
| TC-10 | System Integration | RFID scan while AI module is actively monitoring exam | Both attendance and cheating logs updated in real time | As Expected | Pass |

Table 3 Testcase

4.3 Analysis of Results

4.3.1 Authentication (Admin/Faculty Login)

- The JWT-based authentication system allowed secure login for authorized faculty and admin users.
- Incorrect credentials were correctly rejected, while role-based access ensured admins had management privileges.



A login form titled "Login" with two input fields: "Email" and "Password". Below the fields is a blue "Login" button. At the bottom, there is a link that says "New here? [Create an account](#)".

Figure 6 Login

4.3.2 Dashboard (Attendance & Management)

- The React.js dashboard displayed attendance data clearly and allowed for search/filtering for students or dates.
- Administrators managed courses, sections and users on the dashboard in all cases which confirmed that role-based permissions were indeed properly employed.

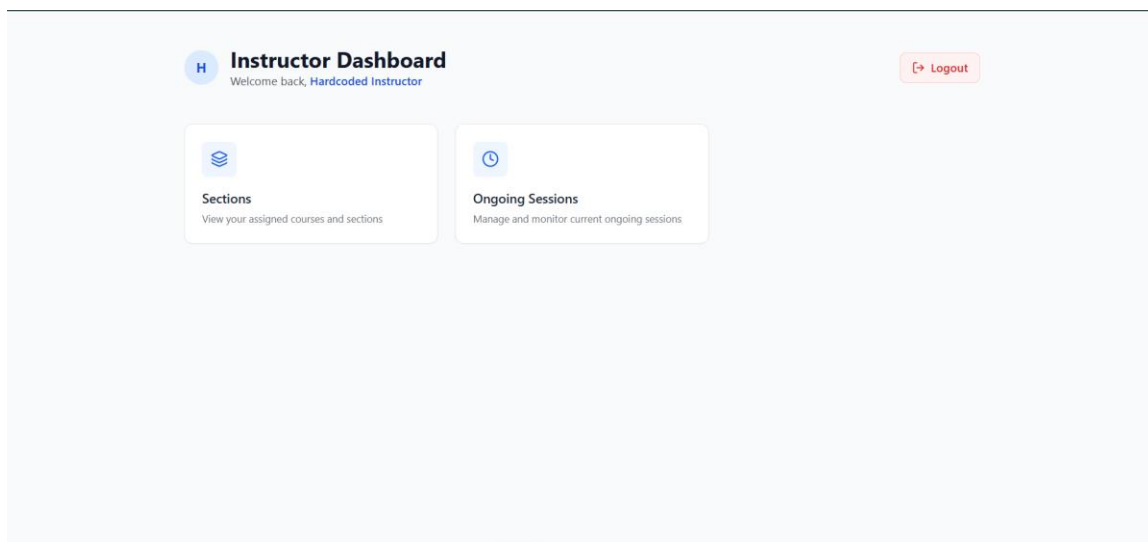


Figure 7 Instructor Dashboard

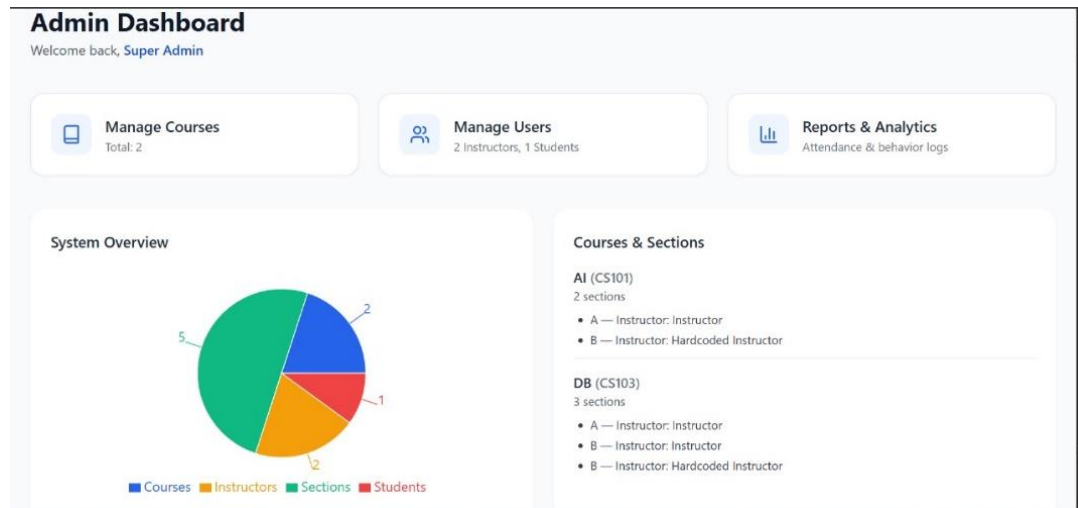


Figure 8 Admin Dashboard

4.3.3 Cheating Detection (AI Monitoring)

- The Mediapipe, YOLOv8-Face, RetinaFace, InsightFace and OpenCV performed multiple faces accurately in real time at high precision with a high accuracy.
- suspicious behaviors, such as abnormal head movement, blinking, or other strange movements, were tracked as possible indicators of cheating.
- Alerts were immediately sent to the web-based dashboard, where they appeared in the monitoring log for review by invigilators or administrators.

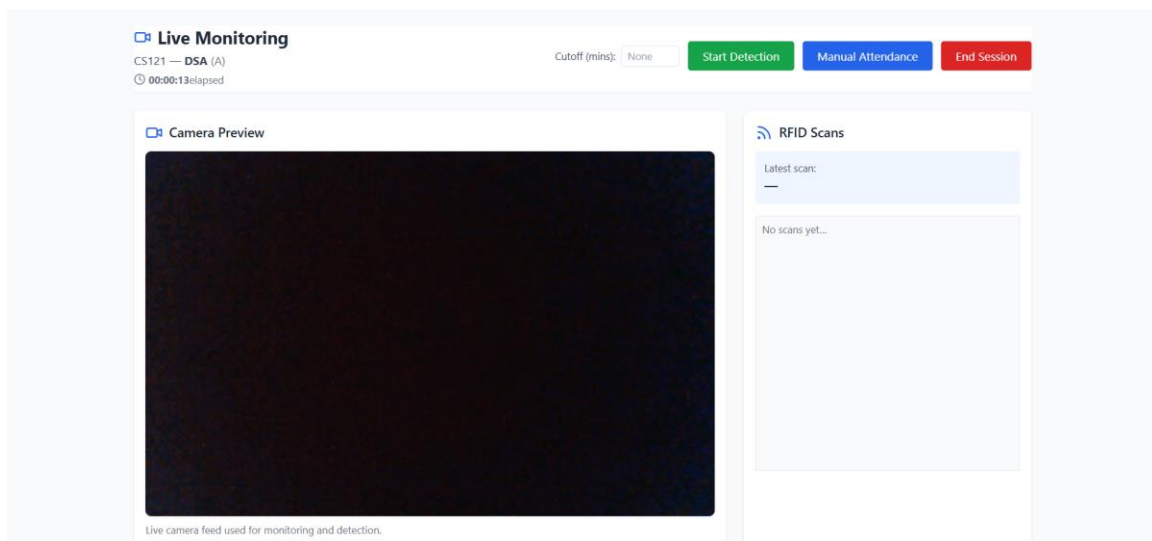


Figure 9 Monitoring Screen

4.4 Potential Improvements

- **Advanced Multi-Factor Authentication:**
Additional applications such as email verification or authentication software could further enhance system security, but not just for JWT based login.
- **UI and UX Enhancements:**
The existing dashboard is functional, but future improvements could enhance user experience by providing more customisable report views and streamlined navigation.
- **AI Model Enhancement:**
A further extension to the cheat detection module, detecting more sophisticated behaviors such as object detection for unauthorized devices, would enhance exam monitoring further.
- **Scalability for Large Classrooms:**
Considering the system would have worked well for controlled tests, improving performance across multiple test halls would make ClassTrack more suitable for large institutions.
- **Offline Attendance Mode:**
Offline RFID scanning support (with subsequent synchronization when internet is available) would be beneficial to institutions with low Internet connectivity.
- **Cross-Institution Integration:**
Adding multi-institution support could allow universities and schools to share the platform while maintaining separate databases.

5 Project Management

5.1 Tasks and Schedule Gantt chart

The Gantt chart for the ClassTrack project presents the stages of development, from the first planning phase to testing of the system.

1. **Project Initiation:**

This marks the beginning of the project. These included setting up the scope of ClassTrack, developing a development team, conducting preliminary research on RFID and AI-based cheating detection, and preparing the development environment.

2. **Requirements Gathering:**

At this stage, the task was to gather and define the features and function of ClassTrack. It uncovered user needs for systems including RFID attendance tracking, AI surveillance, and reporting features to guide systems development.

3. **Design and Architecture:**

Overall system design and architecture was produced during this phase. MERN stack was selected for the web application and Flask for AI integration. Each step was completed with data flow diagrams, database schemas and layered architecture.

4. **Constructing and Executing:**

This was the moment ClassTrack was actually built. These tasks included integrating the RFID attendance module backend functionality, Node.js and MongoDB backend support, Flask-based AI modules for cheating detection, creating React.js frontend dashboard for administrators and faculty using Flask-based AI modules.

5. **Testing and Quality Assurance:**

In that stage, testing of the system was conducted to ensure its accuracy, reliability and usability. This was used as test cases for RFID scanning, faculty verification, AI cheating detection and dashboard performance. Bugs were identified and corrected to improve system stability.

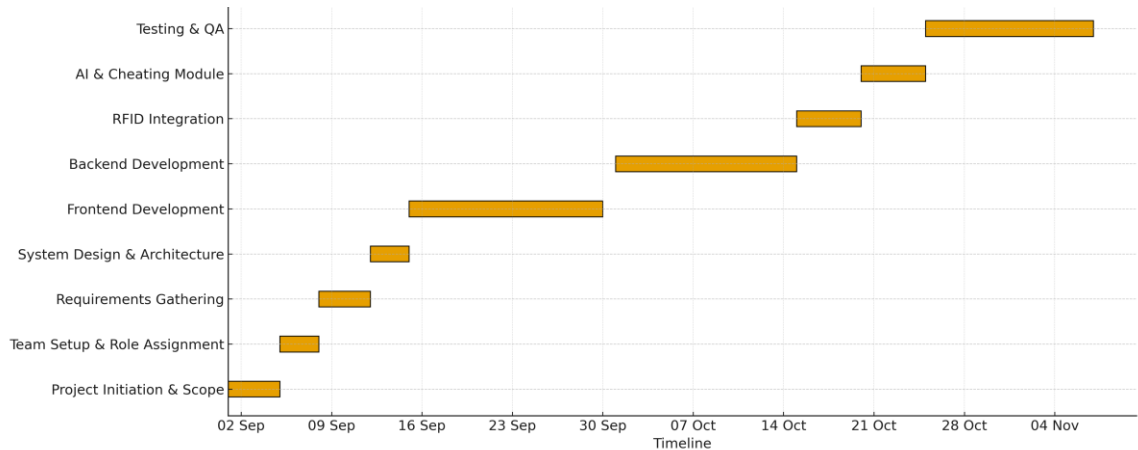


Figure 10 Gantt chart

5.2 Problems and Challenges

- This system was developed multiple times in different phases; design and integration, testing and deployment; and some problems and problems that they encountered during the development of the ClassTrack system. When dealing with such needs, planning and testing and sometimes compromise, it was necessary to address them. Here is an overview of the main challenges facing these countries:
- **RFID Hardware Integration**
 - **Problem Faced:** The use of multiple card scans quickly in the backend was difficult, particularly when several cards were being scan in rapid succession, such as during big classroom entry. In some instances duplicate or missed logs were found.
 - **Solution:** Optimized device drivers and implemented queue-based logging to ensure each scan was uniquely captured and correctly recorded in the MongoDB database.

5.2.1 Real-Time AI Monitoring (Mediapipe, YOLOv8-Face, RetinaFace, InsightFace, OpenCV)

- **Problem Faced:** Running AI models in real time on live video feeds took much computational/GPU resources. This has been reported to cause detection delays and false positives, especially under difficult lighting conditions or with poor quality cameras.
- **Solution:** Determining detection thresholds was determined, redundant detections narrowed to only a limited number, and preprocessing techniques, such as face alignment and lighting normalization, were applied. These measures improved

detection accuracy and reduced false alerts, thus enabling reliable, real-time monitoring during tests.

- **Backend Integration (MERN + Flask)**

- **Problem Faced:** Synchronizing communication between the Node.js backend and the Flask-based AI service was challenging. Delays in sending and receiving detection results occasionally disrupted real-time monitoring.
- **Solution:** Used lightweight REST APIs and asynchronous calls to improve data flow, ensuring that AI alerts were sent instantly to the React dashboard.

- **Database Performance (MongoDB)**

- **Problem Faced:** With growing datasets (attendance logs and AI detection alerts), query performance slowed down, particularly when retrieving large attendance reports.
- **Solution:** Appropriate indexing techniques and schema optimization were put into place, greatly enhancing query performance and system responsiveness.

- **Authentication and Role Management**

- **Problem Faced:** It was difficult to provide role-based access (professor versus admin) with secure login. Unauthorized access or exposure of private logs could result from improper setting.
- **Solution:** Role-based access control, which also incorporated RBAC, was used at the API level to ensure strict permissions, and encrypted tokens were used as proof of JWT authentication.

- **Testing AI in Real Environments**

- **Problem Faced:** During test runs, the AI module occasionally had trouble with packed classes or pupils seated at odd angles, which resulted in missed detections and false positives.
- **Solution:** To increase robustness, more training data was included and several test runs were carried out in various settings.

- **Deployment Challenges**

- **Problem Faced:** More server resources were needed to host the Flask AI service and the MERN application simultaneously. Complexity was increased by setting up GPU/CPU instances in the cloud.

- **Solution:** By leveraging Docker to containerize the system, MERN and Flask services may be independently deployed, minimizing dependency conflicts and streamlining scaling.
- **Data Privacy and Security**
 - **Problem Faced:** Since ClassTrack handled sensitive student data (attendance logs and exam monitoring), compliance with privacy regulations such as GDPR was essential. Improper handling could lead to legal and ethical issues.
 - **Solution:** Sensitive data was encrypted before storage, only minimal personal information was collected and strict access controls were enforced to protect privacy.
- **Network Reliability**
 - **Problem Faced:** The system required stable internet connectivity for syncing attendance logs and streaming AI detection data. In environments with weak or unstable networks, data syncing failed intermittently.
 - **Solution:** Implemented a temporary offline caching mechanism for RFID scans and AI logs, which synchronized with the database once internet connectivity was restored.

5.3 Resources used during the project

- **Development Tools and Frameworks**
 - **MERN Stack (MongoDB, Express.js, React.js, Node.js):**
 - **Purpose:** Used as the primary framework for building the ClassTrack web application. React provided a responsive frontend, while Node.js and Express powered the backend. MongoDB served as the database for attendance logs and cheating detection alerts.
 - **Impact:** Enabled fast development with a single technology stack, ensured scalability for large student datasets and simplified integration between components.
 - **Python Flask (AI Module):**
 - **Purpose:** Deployed as a microservice to integrate AI models for cheating detection using Mediapipe, YOLOv8-Face, RetinaFace, InsightFace and OpenCV.
 - **Impact:** Provided modularity and real-time communication with the web app for detecting cheating behaviors.
- **Security Tools**
 - **JWT Authentication:**

- **Purpose:** Used to authenticate and authorize faculty and administrators with role-based access control.
 - **Impact:** Secured sensitive data like attendance logs and monitoring results from unauthorized access.
- **Encryption Libraries:**
 - **Purpose:** Implemented for secure API communication between the backend, RFID module and AI module.
 - **Impact:** Ensured privacy and compliance with data protection standards (e.g., GDPR).
- **Testing Tools**
 - **Jest and Mocha (JavaScript Testing):**
 - **Purpose:** Used for unit testing and integration testing of the MERN stack application.
 - **Impact:** Validated system stability and ensured correct interaction between frontend, backend and database.
 - **PyTest (Python Testing):**
 - **Purpose:** Used to test the Flask-based AI detection services.
 - **Impact:** Verified detection accuracy and reduced false positives during trial runs.
- **Design and Prototyping Tools**
 - **Figma:**
 - **Purpose:** Used to design the user interface of the dashboard, including reports and analytics pages.
 - **Impact:** Provided clear wireframes and prototypes for early design feedback, improving usability.
 - **Lucidchart / Draw.io:**
 - **Purpose:** Used to prepare architectural diagrams and system workflows.
 - **Impact:** Helped visualize system architecture for better implementation planning.
- **Hardware Resources**
 - **RFID Equipment:**
 - **Purpose:** USB RFID reader/writer and RFID cards for attendance logging.
 - **Impact:** By providing reliable contactless attendance tracking and eliminating manual effort, it had a positive impact.

- **Cameras:**
 - **Purpose:** standard HD webcams used for AI-based cheating detection during exams.
 - **Impact:** Live video feeds were provided for the AI module to track student behavior.
- **Development Machines:**
 - **Purpose:** Laptops and desktops with MERN and Flask development environments.
 - **Impact:** Implemented code development, AI training and testing.
- **Knowledge Resources**
 - **Official Documentation (MongoDB, React, Node.js, Flask, YOLOv8, Mediapipe, RetinaFace, InsightFace, OpenCV):**
 - **Impact:** Provided instructions for supporting complex RFID and computer vision programs.
 - **Stack Overflow and GitHub Repositories:**
 - **Impact:** Offered troubleshooting solutions, coding best practices and reusable code snippets.
 - **Online Courses (Coursera, Udemy):**
 - **Purpose:** Used to strengthen knowledge in AI model deployment, MERN stack and real-time system integration.
 - **Impact:** Rapid development and better implementation.
- **Version Control and Project Management Tools**
 - **Git and GitHub:**
 - **Purpose:** Version control system for managing source code.
 - **Impact:** Ensured collaborative coding, easy rollback of changes and centralized repository management.
 - **Trello:**
 - **Purpose:** Project management tool for tracking tasks, assigning responsibility and monitoring timelines.
 - **Impact:** organized and followed up the project as needed and coordinated with the team.
- **Libraries and Packages**
 - **Mediapipe, YOLOv8-Face, RetinaFace, InsightFace:** AI libraries for face detection, recognition and gesture analysis in examinations.
 - **OpenCV:** Computer vision library for images and video processing, including frame capture, face alignment and lighting normalization.
 - **Mongoose:** ODM (Object Data Modeling) library for handling MongoDB schemas and interaction between databases.

- **Express Middleware (Helmet, CORS, Body-Parser):** Secure, efficient and smooth communication between the backend and other system components.
- **Impact:** These libraries, with software available for AI detection, data modeling and secure communication, reduced development time.
- **Financial Resources**
 - **Cloud Hosting (MongoDB Atlas & Node.js/Flask Deployment):**
 - **Purpose:** Backend and AI module hosting services.
 - **Impact:** Live monitoring and attendance in real-time without the need for self-management.
 - **RFID Hardware & Camera Setup:**
 - **Purpose:** To monitor exams and record practical attendance.
 - **Impact:** Despite the utility of ClassTrack in practice, there was a small cost of initial execution.

6 Conclusions

All the solutions that schools need for safely and effectively monitoring student attendance and promoting academic integrity in assessment are available in ClassTrack. One of the most challenging problems in classroom management is the reliability of attendance records and academic fairness, which ClassTrack seeks to address by tracking attendance and AI-based cheating detection.

Its attendance feature, using RFID cards and a USB RFID reader/writer, is one of the most innovative components of ClassTrack. Though attendance took a heavy weight, manual rolls calls that sometimes produced human errors, ClassTrack lets the student RFID cards accurately measure attendance without even touching or texting. The students are assigned their own ID card and stay safe and anonymous in the database, because they are all assigned their own unique RFID card.

ClassTrack uses a Python Flask microservice to use best practices in computer vision technology like Mediapipe, YOLOv8-Face, and OpenCV to monitor the exam process. These technology can also detect suspicious behavior in real time – such as attempts to see off from the screen, head movements reminiscent of those around other eyes or multiple faces in the video feed. An equitable test, based on low malpractice rates and AI proctoring, yields better test outcomes.

Plus, it has a modular architecture that is one of the other benefits of the system. With the MERN stack behind the student and administrator to be able to access attendance logs and cheating alerts, the web interface is easily accessible and easy-to-use. Flask does not, but, work with the independent AI module of Flask, but does integrate seamlessly with the web. This separation facilitates scaling, debugging, and upgrading parts of the system while operating normally.

The team encountered many technical problems at work designing the system including migrating RFID hardware back to the MERN backend, using AI models to identify a real-time cheating in real time and ensuring the system performance under load was stable. For real-time analysis, it needed efficient algorithms to prevent delays, a particular caution was exercised when dealing with video streams. The solution to these problems was better automating query queries, lessening processing complexity via AI micro-service, and creating a modular system.

Tests were the critical components of ClassTrack's reliability and efficacy. In the case of quick succession, one of the students were tested on the RFID attendance module to ensure good readings occurred regardless of short-run. To verify robustness, the AI cheating identification was tested during several scenarios, including multiple students in the frame, low lighting and several camera settings. Functional, integration and stress tests demonstrated that the system properly tracked exams and attendance.

The results were consistent and continued to overload:

- Logged attendance quickly and accurately through RFID cards.
- Detected suspicious activities during exams with reasonable accuracy.
- Provided administrators with clear reports and dashboards for decision-making.

In conclusion, ClassTrack serves as a twofold solution that promotes exam integrity with AI monitoring and automatically tracks attendance. This two-fold method of alternative authentic learning assessment is the duality of standard forms of implementation as well as the array of teacher "decisions" regarding alternative assessment of learners. Future enhancements will implement a multi-factor authentication for learning individuals. Additional RFID hardware will further enhance a learning individual use case interaction with other subjects, including P.E. With AI models to help direct cheating models, these alterations could boost performance, even if the current version reliably works. ClassTrack builds the bridge to a smarter classroom and an equitable learning experience by coming together hardware, software and Artificial Intelligence.

7 Statement of Contribution

| Task | Contribution |
|--------------------------------|---------------------|
| System Design and Architecture | |

| | <table> <tr> <th>Student</th><th>Contribution Percentage</th></tr> <tr> <td>Student 1</td><td>30%</td></tr> <tr> <td>Student 2</td><td>25%</td></tr> <tr> <td>Student 3</td><td>20%</td></tr> <tr> <td>Student 4</td><td>15%</td></tr> <tr> <td>Student 5</td><td>10%</td></tr> </table> | Student | Contribution Percentage | Student 1 | 30% | Student 2 | 25% | Student 3 | 20% | Student 4 | 15% | Student 5 | 10% |
|--|--|---------|-------------------------|-----------|-----|-----------|-----|-----------|-----|-----------|-----|-----------|-----|
| Student | Contribution Percentage | | | | | | | | | | | | |
| Student 1 | 30% | | | | | | | | | | | | |
| Student 2 | 25% | | | | | | | | | | | | |
| Student 3 | 20% | | | | | | | | | | | | |
| Student 4 | 15% | | | | | | | | | | | | |
| Student 5 | 10% | | | | | | | | | | | | |
| Frontend and Backend Development | <table> <tr> <th>Student</th><th>Contribution Percentage</th></tr> <tr> <td>Student 1</td><td>30%</td></tr> <tr> <td>Student 2</td><td>20%</td></tr> <tr> <td>Student 3</td><td>15%</td></tr> <tr> <td>Student 4</td><td>20%</td></tr> <tr> <td>Student 5</td><td>15%</td></tr> </table> | Student | Contribution Percentage | Student 1 | 30% | Student 2 | 20% | Student 3 | 15% | Student 4 | 20% | Student 5 | 15% |
| Student | Contribution Percentage | | | | | | | | | | | | |
| Student 1 | 30% | | | | | | | | | | | | |
| Student 2 | 20% | | | | | | | | | | | | |
| Student 3 | 15% | | | | | | | | | | | | |
| Student 4 | 20% | | | | | | | | | | | | |
| Student 5 | 15% | | | | | | | | | | | | |
| AI Cheating Detection and RFID integration | <table> <tr> <th>Student</th><th>Contribution Percentage</th></tr> <tr> <td>Student 1</td><td>15%</td></tr> <tr> <td>Student 2</td><td>20%</td></tr> <tr> <td>Student 3</td><td>20%</td></tr> <tr> <td>Student 4</td><td>15%</td></tr> <tr> <td>Student 5</td><td>20%</td></tr> </table> | Student | Contribution Percentage | Student 1 | 15% | Student 2 | 20% | Student 3 | 20% | Student 4 | 15% | Student 5 | 20% |
| Student | Contribution Percentage | | | | | | | | | | | | |
| Student 1 | 15% | | | | | | | | | | | | |
| Student 2 | 20% | | | | | | | | | | | | |
| Student 3 | 20% | | | | | | | | | | | | |
| Student 4 | 15% | | | | | | | | | | | | |
| Student 5 | 20% | | | | | | | | | | | | |

| Student ID | Student Name | Signature |
|------------|---------------|-----------|
| 202102812 | Ali Alblooshi | |
| 202021767 | Hamad Alketbi | |
| 202022212 | Saif Alketbi | |
| 201915236 | Saif Alkaabi | |
| 202100588 | Ali Almerri | |

8 References

- [1] Agarwal and R. Patni, “RFID technology for smart attendance monitoring system,” *Int. J. Adv. Res. Comput. Sci.*, vol. 10, no. 2, pp. 45–49, 2019.
- [2] K. Ahsan, H. Shah and P. Kingston, “RFID applications: An introductory and exploratory study,” *Int. J. Comput. Sci. Issues*, vol. 7, no. 1, pp. 1–7, 2010.
- [3] S. Alotaibi and D. Argles, “Online exam and authentication security system,” in *Proc. Int. Conf. Educ. e-Learning Innovations (ICEELI)*, 2011, pp. 1–6.
- [4]] E. Balan, A. Jivoi and A. Pasarica, “Face recognition methods: Applications in access control and surveillance,” *Procedia Comput. Sci.*, vol. 159, pp. 577–586, 2019.
- [5] A. Bochkovskiy, C. Y. Wang and H. Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [6] S. Choudhury and T. Bhowmik, “Biometric-based proctoring for online exams: A survey,” *Int. J. Comput. Appl.*, vol. 179, no. 25, pp. 1–7, 2018.
- [7] R. Girdhar and A. Rathi, “Automated classroom attendance system using face recognition,” *Int. J. Emerg. Technol. Learn. (iJET)*, vol. 15, no. 9, pp. 54–68, 2020.
- [8] K. He, X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.
- [9] A. G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [10] M. Kaur and M. Sandhu, “RFID technology principles, advantages, limitations & its applications,” *Int. J. Comput. Electr. Eng.*, vol. 6, no. 1, pp. 17–20, 2014.
- [11] A. Mathur and R. Tripathi, “Cloud-based RFID attendance system for educational institutes,” *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, pp. 1768–1772, 2017.
- [12] S. Mehta and R. Mehta, “Anti-cheating examination monitoring system using machine learning,” *Int. J. Innov. Technol. Explor. Eng. (IJITEE)*, vol. 8, no. 12, pp. 3106–3110, 2019.

- [13] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [14] K. Shah, H. Patel, D. Sanghvi and M. Shah, “A comparative analysis of logistic regression, random forest and KNN models for the text classification,” *Augment. Hum. Res.*, vol. 5, no. 1, pp. 1–16, 2020.
- [15] M. Simon and B. Tiddeman, “Real-time facial landmark detection using Mediapipe framework,” *Int. J. Comput. Vis. Appl.*, vol. 34, no. 2, pp. 85–93, 2018.
- [16] S. Srivastava and R. Sharma, “Biometric authentication and AI-enabled proctoring for secure online examinations,” *J. Inf. Secur. Res.*, vol. 9, no. 2, pp. 67–75, 2018.
- [17] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2001, vol. 1, pp. 511–518.
- [18] A. Wibowo and S. Lestari, “Student attendance system using RFID and GSM networks,” *Procedia Comput. Sci.*, vol. 124, pp. 93–99, 2017.
- [19] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, 2016
- [20] Q. Zhou and Y. Tang, “AI-based online examination monitoring: Design and implementation,” *Int. J. Emerg. Technol. Learn. (iJET)*, vol. 15, no. 20, pp. 223–233, 2020.

Biography

Background and Goals

The growing need for classroom management systems that are more trustworthy, automated, and secure inspired ClassTrack. Although monitoring exams often doesn't uphold fairness and deter cheating, traditional methods of taking attendance (e.g., human roll calls or sheets of paper) are cumbersome and prone to error. A system of exam integrity detection by computer vision-based cheating detection and internet using RFID to record attendance could emerge out of artificial intelligence and IoT advances.

ClassTrack was designed to connect the two critical components of learning to a user-friendly platform. With ClassTrack schools can manage their students intelligently and efficiently, recording attendance in an efficient way using RFID cards and RFID readers to enable touchless attendance, and AI models for observing exams in real-time.

Process of Development

The system structure consists of two components:

- In the MERN stack (MongoDB, Express.js, React.js and Node.js), an online application designed to provide administrators and teachers a way to view attendance logs, create reports and monitor accuracy of student conduct during exams.
- An AI microservice developed in Python Flask integrated Mediapipe, YOLOv8-Face, RetinaFace, InsightFace and OpenCV for real-time cheating detection. The microservice processed live video feeds, participated in behavioral and facial analysis, notified and alerted users immediately. It communicated seamlessly with the web application so that events had been tracked on the dashboard for an early review by invigilators or administrators.

RFID integration was successfully enabled by incorporating a unique student ID card and USB RFID reader/writer device that was secure, encrypted attendance data. MongoDB is data storage which enables scalability and simple retrieval by storing attendance information and cheating alerts on attendance in the form of data storage.

Key Features

- **RFID Attendance:** To effortlessly record attendance for every student, we give them an RFID card that they simply touch on the USB reader to be noted present. This ensures we have attendance that is instant, accurate, and secure.
- **Cheating Detection:** The AI module utilizes Mediapipe, YOLOv8-Face, RetinaFace, InsightFace and OpenCV to detect cheating behaviours in the exam. This includes rapid or abnormal head movements, multiple faces on the screen, and

obvious attempts to distract attention from unauthorized screens or devices. The system provides real-time notifications to the dashboard that allow invigilators to see and respond quickly to alerts.

- **Administrative Dashboard:** The faculty can access an online dashboard built in React to review attendance, create reports, and receive cheating alerts.
- **Secure Storage:** Claims data and attendance records are securely stored in MongoDB to ensure data reliability and privacy.

Tests and Obstacles

One obstacle encountered during development was a number of difficulties:

- **Hardware Integration:** Ensuring that the USB RFID reader/writer worked seamlessly with the MERN backend required custom middleware and API handling.
- **AI Functionality:** Because of low hardware resources, running Mediapipe, YOLOv8-Face, RetinaFace and InsightFace in real time suffered lags. This problem was solved by model optimization, preprocessing techniques, such as face alignment, lighting normalization, and the use of AI components as Flask microservices, which did a much more efficient work, while at the same time not potentially impacting the main application itself.
- **Cross-Platform Testing:** the AI module was tested by testing it at various low light conditions, variable camera quality, and many users.
- **Scalability:** The system needed to handle multiple concurrent users and not compromise performance by minimizing the MongoDB queries and node.js asynchronous handling.

Unit testing for the RFID and AI modules, integration testing for Flask-MERN stack communication, and stress testing all took place during testing during long class periods.

Impact and Future Enhancements

ClassTrack gives schools one solution for academic integrity and attendance automation, two issues that are often treated separately. With reliable record-keeping, fair assessments, and lower administrative time through a single RFID and AI platform.

Although ClassTrack has delivered on its foundational goals, there are always opportunities for improvement:

- **Multi-Camera Support** for larger exam halls.
- **Integration with Learning Management Systems (LMS)** to provide a seamless academic ecosystem.
- **Mobile App for Faculty** to monitor attendance and exam sessions on the go.
- **Cloud-Based Deployment** for scalability across multiple institutions.

Appendix

User Interface Mockups

The proposed layout of the ClassTrack web app user interface can be found in the appendix. The mockups also show the faculty dashboard, attendance, alarms when an exam is going to occur, reporting panels and login screen. These mockups show us the system's layout and how users will interact with the system ensuring accessibility and usability.

Cheating Detection Algorithm

This appendix gives a detailed description of the AI-powered cheating detection models used in ClassTrack. It outlines Mediapipe, YOLOv8-Face, RetinaFace, InsightFace, and OpenCV as the face detection and head tracking system, as well as assessment of behavior for exam. In addition, it addresses other limitations of these models such as lighting, camera quality, and student movement; and explains strategies for controlling false positives using preprocessing, threshold tuning, and model optimization to minimize false positives and improve accuracy and reliability of the system.

User Guide

The user guide appendix provides detailed instructions for faculty, administrators and students. It covers:

- Register for an account and log into.
- RFID cards in attendance registration.
- Accessing and evaluating attendance reports.
- Observe live exams through the cheat detection dashboard.
- Interpreting alerts and exporting reports.

Testing Findings

This appendix includes testing documentation for ClassTrack. It includes:

- Test cases for RFID attendance record.
- Test scenarios for real-time cheating detection.
- Execution logs and error reports.
- Issues found in integration testing, such as hardware faults, AI detection error, and solutions.

Performance Metrics

The data received during performance and stress tests is included in this appendix.:

- **RFID Scan Speed:** Average time to log student attendance.
- **AI Processing Latency:** Time taken to detect and flag suspicious behavior from live video input.
- **System Throughput:** Maximum number of students supported in a live exam session without delays.
- **Resource Utilization:** CPU/GPU usage of the AI microservice and database query efficiency under high load.

Legal and Compliance Documentation

This Appendix includes documents about data protection and compliance.

- Privacy policy for handling student data.
- Institutional compliance with **FERPA** (Family Educational Rights and Privacy Act) and **GDPR** (General Data Protection Regulation).
- Ethical guidelines for the use of AI-based monitoring in classrooms.

Glossary of Terminology

This appendix defines key technical terms in ClassTrack including :

- **RFID (Radio-Frequency Identification):** Tracks student ID cards for automated attendance.
- **YOLOv8-Face:** Real-time face detection AI model.
- **RetinaFace:** Accurate face detection and landmark localization model.
- **InsightFace:** AI framework for face recognition and verification.
- **Mediapipe:** Framework for face tracking, gesture recognition, and pose estimation.
- **Flask Microservice:** Modular Python service for AI detection, communicating with the main app.
- **False Positive (AI Detection):** When AI incorrectly flags normal behavior as suspicious.
- **MongoDB Atlas:** Cloud database for storing student and exam data.