College Portal Project

---

1. Exam Portal with Cheating Detection

Description:

The Exam Portal enables students to take online exams securely with integrated anti-cheating mechanisms. It monitors student behavior during exams, ensures fair code submission for programming tasks, and logs activities for review.

Detailed Features:

- **Webcam Monitoring**: Uses OpenCV for real-time face detection to verify student identity and detect if the student leaves the frame or multiple faces are present.
- **Code Evaluation**: Integrates Judge0 API or a Python sandbox environment to evaluate programming submissions, supporting multiple languages and test cases.
- **Tab Switch Detection**: JavaScript-based monitoring to detect when a student switches tabs or applications, flagging suspicious activity.
- **Timers and Logs**: Implements strict exam timers and logs all activities (e.g., start/end times, tab switches) for post-exam analysis.
- **Random Question Set**: Supports randomized question sets to minimize cheating through collaboration.
- **Proctoring Alerts**: Notifies faculty in real-time if cheating is suspected (e.g., face not detected, tab switch).

Technical Considerations:

- Use OpenCV with Python for webcam processing, optimized for browser compatibility via WebRTC.
- Judge0 API requires secure API key management and rate-limiting handling.
- JavaScript event listeners (e.g., `visibilitychange`, `blur`) for tab switch detection.
- Store logs in a secure database (e.g., PostgreSQL) with encryption for sensitive data.

---

2. Attendance Tracking

Description:

The Attendance Tracking feature allows faculty to record student attendance efficiently and provides students with real-time visibility into their attendance status. It enforces institutional attendance policies (e.g., 75% attendance rule).

Detailed Features:

- **automated attendence**: Faculty can mark attendance manually via a web interface for automated check-ins during classes.
- **Color-Coded Status**: Displays attendance status (e.g., green for present, red for absent) on student dashboards.
- **Eligibility Tracking**: Automatically calculates attendance percentage and flags students below the required threshold (e.g., 75%).
- **Reports**: Generates downloadable attendance reports for faculty and admin use.

- **Notifications**: Sends alerts to students nearing the eligibility threshold.

Technical Considerations:

- Store attendance data in a relational database with indexes for fast retrieval.
- Implement role-based access to prevent unauthorized modifications.

---

3. Faculty-Student Chat

Description:
The Faculty-Student Chat feature facilitates real-time communication between students and faculty, restricted to enrolled students for privacy and relevance.

Detailed Features:

- **Real-Time Messaging**: Uses WebSocket-based communication for instant messaging.
- **Typing Indicators**: Shows when the other party is typing to enhance user experience.
- **Message History**: Stores chat history for reference, with pagination for performance.
- **Read Receipts**: Indicates when messages are read.
- **Enrollment-Based Access**: Restricts chat access to students enrolled in a faculty's course or also if he wants to chat with another faculty.
- **File Sharing**: Allows sharing of small files (e.g., PDFs, images) for academic purposes.

Technical Considerations:

- Use Django Channels for WebSocket handling and scalability.
- Implement end-to-end encryption for sensitive communications.
- Store messages in a NoSQL database (e.g., MongoDB) for flexible schema and scalability.
- Validate file uploads to prevent malicious content (e.g., restrict file types, scan for viruses).

---

4. Online Result Viewing & Progress Visualization

Description:
This feature allows students to view their exam results, ranks, and academic progress through interactive visualizations, with options to download reports.

Detailed Features:

- **Result Access**: Students can view their marks, grades, and class ranks securely.
- **Progress Graphs**: Uses Chart.js or Plotly to display trends (e.g., semester-wise performance, subject-wise scores).
- **PDF Reports**: Generates downloadable PDF progress reports with detailed breakdowns.

- **Comparison Metrics**: Optionally shows anonymized class averages for context.
- **Secure Access**: Ensures students only see their own data.

Technical Considerations:

- Use Chart.js for lightweight, responsive graphs or Plotly for interactive visualizations.
- Generate PDFs server-side using libraries like `reportlab` or `pdfkit`.
- Implement role-based authentication to restrict data access.
- Cache frequently accessed data (e.g., results) using Redis for performance.

---

5. Centralized Notes & Notice Upload

Description:
The Centralized Notes & Notice Upload feature enables faculty to share organized academic materials and announcements, with robust search and versioning capabilities.

Detailed Features:

- **File Uploads**: Faculty can upload notes (e.g., PDFs, Word documents) and notices.
- **Organization**: Categorizes files by course, subject, or semester for easy access.
- **Search and Filter**: Allows students to search notes by keywords or filter by metadata (e.g., upload date, subject).
- **File Versioning**: Tracks versions of updated notes to prevent confusion.
- **Notifications**: Alerts students when new notes or notices are uploaded.

Technical Considerations:

- Implement versioning by appending version numbers to file metadata.
- Ensure file size limits and validation to prevent abuse.

---

6. Feedback System

Description:
The Feedback System allows students to provide anonymous feedback on faculty performance, with aggregated reports generated for administrative review.

Detailed Features:

- **Anonymous Feedback**: Students submit feedback via a form, with fields for rating (e.g., 1-5 stars) and comments on teaching quality, communication, etc.
- **Feedback Reports**: Generates periodic reports for each faculty member, summarizing ratings and comments (anonymized).
- **Admin Oversight**: Admins can review feedback and flag inappropriate comments.
- **Submission Limits**: Restricts feedback to one submission per student per faculty per semester to prevent spam.
- **Notifications**: Notifies faculty when new feedback is available (without revealing student identities).

Technical Considerations:

- Use a form library (e.g., Formik with React) for feedback submission.
- Store feedback in a secure database with anonymization (e.g., no student IDs linked).
- Generate reports using a backend script (e.g., Python with Pandas) and visualize with Chart.js.
- Implement rate-limiting to enforce submission restrictions.

Implementation Suggestion:

- Start with a simple feedback form and admin report generation.
- Add notifications and advanced report visualizations later.

---

7. Parent Communication System

Description:
The Parent Communication System automates the delivery of parent-related notices, student progress reports, and complaint notifications via WhatsApp, integrated through the college portal.

Detailed Features:

- **Notice Sharing**: Sends WhatsApp messages to parents for notices tagged as parent-related (e.g., fee deadlines, parent-teacher meetings).
- **Progress Reports**: Automatically sends periodic progress reports (e.g., marks, attendance) to parents via WhatsApp.
- **Complaint Notifications**: Notifies parents of any disciplinary complaints or academic issues via WhatsApp.
- **Website Integration**: Parents can opt-in/out and manage preferences through the portal.
- **Message Templates**: Uses predefined templates for consistency and personalization (e.g., including student name).

Technical Considerations:

- Integrate with a WhatsApp Business API provider (e.g., Twilio or Meta's WhatsApp API) for message delivery.
- Store parent contact details securely with encryption.
- Implement a queuing system (e.g., Celery with RabbitMQ) for reliable message delivery.
