Class Attendance System Using Facial Recognition and QR Code Subsystems

Ricky Reyes-Pena
Georgia State University
Atlanta, GA 30302
rreyespena1@student.gsu.edu

Abstract

This project presents an innovative web-based attendance tracking system designed to modernize and streamline the process of recording student attendance in educational institutions. The system offers two distinct authentication methods: facial recognition and QR code verification, providing flexibility and reliability in attendance tracking.

The application is built using a React frontend and Django REST framework backend, implementing a modern client-server architecture. Professors can create and manage classes, while students can choose their preferred attendance verification method. The facial recognition feature utilizes deep learning for accurate face detection and matching, while the QR code system provides a quick and efficient alternative.

Key features include real-time attendance tracking, secure user authentication, automated student enrollment management, and an intuitive user interface. The system addresses common challenges in traditional attendance tracking methods, such as time consumption, accuracy, and potential manipulation, while maintaining data privacy and security.

This solution demonstrates the practical application of modern web technologies and biometric systems in educational settings, offering a scalable and user-friendly approach to attendance management.

1. Introduction

The Automated Attendance System is a comprehensive web application designed to modernize and streamline the process of tracking student attendance in educational settings. The system replaces traditional paper-based attendance methods with two efficient digital alternatives: facial recognition and QR code verification. This dual-approach system allows institutions to implement the attendance tracking method that best suits their needs while maintaining accuracy and security.

The motivation behind developing this application stems from the inherent inefficiencies and distractions associated with traditional attendance-taking methods. The common practice of passing around attendance sheets during lectures not only disrupts the learning environment but also consumes valuable teaching time. Additionally, physical attendance sheets are susceptible to proxy attendance and can be easily misplaced or damaged. This project aims to eliminate these issues by providing a digital solution that is both quick and minimally disruptive to the educational process.

The key components of this project include:

- User Interface Components: - Professor dashboard for class management and attendance tracking - Student interface for attendance verification method selection - Registration and authentication systems for both professors and students
- Backend Services: - REST API endpoints for handling requests and responses - Database man- agement system for storing professor, class, student, registration and attendance-logging records - Authentication and authorization services
- Technical Features: - Facial recognition system utilizing deep learning for accurate face detection and matching - QR code generation and scanning functionality - Real-time attendance verification and recording - Secure data storage and transmission
- Security Measures: - Encrypted data transmission - Secure storage of facial recognition data - Protected user authentication - Role-based access control

2. Database Details

Database Design: I designed the database with 6 tables that effectively manage the relationships between professors, students, classes, and attendance records. The design prioritizes data integrity, minimizes redundancy, and ensures efficient data retrieval. The tables are structured to handle both facial recognition and QR code attendance methods while maintaining proper relationships between entities.

The six core tables are:

1. Professor
    - Stores professor information including email and password
    - Primary key: professorId
    - Handles professor authentication and class ownership
2. Student
    - Manages student records including email and name
    - Primary key: studentId
    - Stores facial recognition data and QR code information
3. Class
    - Contains class information including CRN, name, and professor reference
    - Primary key: classId
    - Foreign key: professor (references Professor table)
4. ClassEnrollment
    - Manages the many-to-many relationship between students and classes
    - Primary key: classenrollmentId
    - Foreign key: classCRN, studentEmail
    - Foreign keys reference both Class and Student tables
5. AttendanceLog
    - Records attendance entries for each class session
    - Primary key: attendanceId
    - Foreign keys: classCRN, studentId
    - Includes timestamp and attendance status
6. AttendanceSystemRegistration
    - Stores student registration for a specific class. Face encoding and QR code files are some of the data being stored as well.
    - Primary key: attendancesystemregistrationId
    - Foreign keys: classCRN, studentId

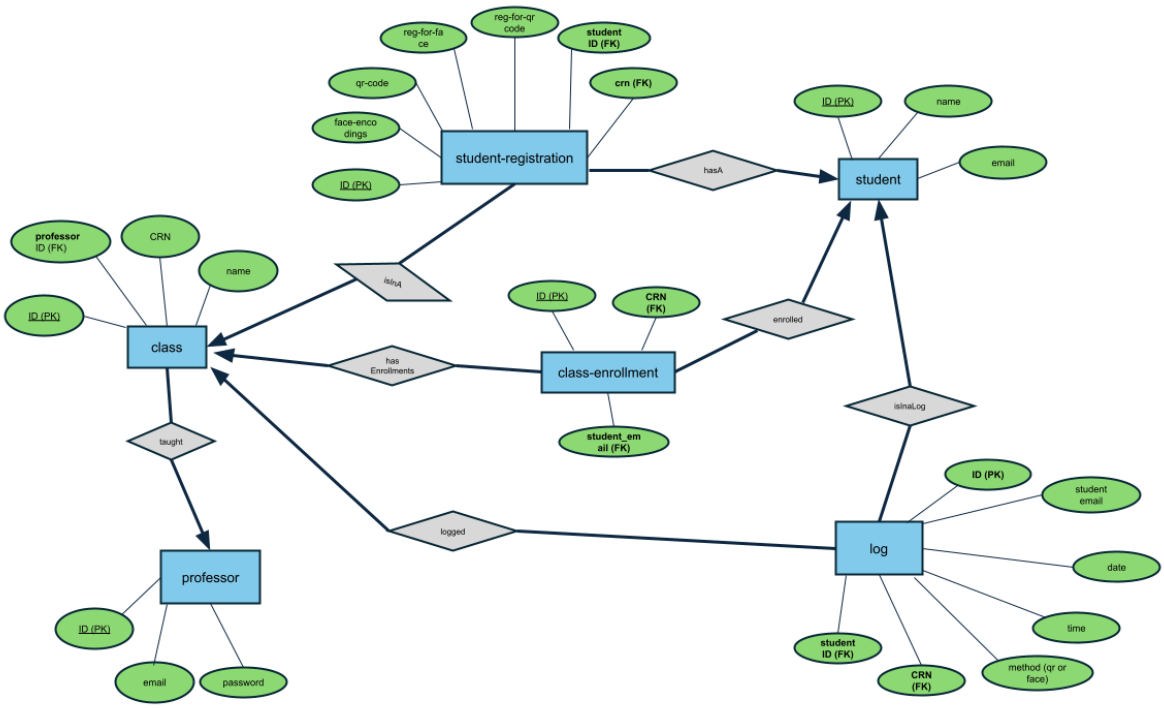

Figure 1. ER Model of the Database

Normalization and Functional Dependencies: The database design adheres to Third Normal Form (3NF) as:

1. All tables satisfy First Normal Form (1NF) with atomic values
2. There are no partial dependencies (2NF)
3. No transitive dependencies exist between non-prime attributes (3NF)

For example:

- In the Class table, all attributes depend only on the CRN (primary key)
- In the Attendance table, attributes depend on the composite of classCRN, student, and timestamp
- The ClassEnrollment table eliminates many-to-many relationship redundancies

Additional Constraints:

1. Referential Integrity:
    - Foreign key constraints ensure data consistency
    - Cascade delete operations where appropriate
    - Null constraints on mandatory fields
2. Business Rules:
    - Unique email addresses for professors and students
    - Valid CRN format enforcement
    - Timestamp constraints for attendance records

3. Data Validation:
    - Email format validation
    - Required field constraints
    - Image size and format restrictions for facial recognition data
4. Temporal Constraints:
    - Attendance records include timestamps
    - Class creation and modification timestamps

3. Functionality Details

The application implements both basic and advanced functions to manage attendance tracking effectively. These functions are divided into professor-related (basic) and student-related (advanced) functionalities.

Basic Functions (Professor-Related):

1. Professor Authentication
    - Login functionality using email/password
    - Session management for authenticated professors
    - Secure password handling and validation
    - Error handling for invalid credentials
    - Automatic session timeout for security
2. Class Management
    - Create Class:
        - Input validation for class details (CRN, name)
        - Bulk student email upload via CSV file
        - Automatic student record creation if email doesn't exist
        - Class enrollment generation for each student
        - Error handling for duplicate CRNs
        - Validation of email format and uniqueness
    - Delete Class:
        - Verification of professor ownership
        - Cascade deletion of related records:
            * Class enrollments
            * Attendance records
            * QR codes
            * Facial recognition data
        - Confirmation process to prevent accidental deletion
3. Attendance Report Generation
    - Comprehensive data retrieval options:
        - Filter by date range
        - Sort by student name/email
        - Group by attendance status
    - Statistical analysis:
        - Attendance percentage calculations

    - Absence tracking
    - Late arrival monitoring
- Export functionality for data analysis

Advanced Functions (Student-Related):

1. Facial Recognition System
    - Registration:
        - Enrollment verification against class records
        - Multiple facial image capture for accuracy
        - Image quality validation:
            * rightness check
            * Face detection confirmation
            * Multiple angle verification
        - Secure storage of facial data
        - Duplicate registration prevention
    - Login/Attendance Marking:
        - Real-time face detection and analysis
        - Multi-step verification:
            * Initial face detection
            * Feature extraction
            * Pattern matching with stored data
            * Confidence score calculation
        - Attendance record creation with:
            * Timestamp
            * Verification method
            * Success/failure status
        - Prevention of duplicate attendance entries
2. QR Code System
    - Registration:
        - Student verification against class enrollment
        - Dynamic QR code generation containing:
            * Student email
            * Username
            * CRN
            * Timestamp
        - Secure encoding of student data
        - QR code image generation and storage
        - Validation of registration eligibility
    - Login/Attendance Marking:
        - Real-time QR code scanning
        - Data extraction and validation:
            * Code authenticity verification
            * Student enrollment confirmation
            * Timestamp validation

    - Automatic attendance recording
    - Prevention of duplicate scans
    - Error handling for invalid codes
  - QR Code Retrieval:
    - Student identity verification
    - Validation of existing registration
    - Secure QR code regeneration
    - Access control verification
    - Rate limiting for security

Each function includes:

- Input validation
- Error handling
- Success/failure notifications
- Secure data processing
- Audit logging
- Rate limiting where appropriate

4. Implementation Details

Technology Stack:

1. Frontend Development:
   - React.js framework for building the user interface
   - JavaScript/ES6+ for application logic
   - CSS3 for styling and animations
   - Framer Motion for enhanced animations
   - HTML5 for structure
   - Libraries:
     - react-modal for modal dialogs
     - react-toastify for notifications
     - html5-qrcode for QR code scanning
     - react-router-dom for navigation
2. Backend Development:
   - Django REST Framework (Python)
   - SQLite database (development)
   - Libraries:
     - qrcode for QR code generation
     - Pillow for image processing
     - face-recognition for facial recognition
     - JWT for authentication

Frontend Implementation:

1. Component Structure:
    - Modular components for reusability
    - Page-specific views:
        - StudentChoosingPage for attendance method selection
        - FacialRecognitionPage for face recognition
        - QRCodePage for QR code operations
        - ProfessorOptionsPage for class management
2. State Management:
    - React hooks (useState, useEffect) for local state
    - Props for component communication
    - Context API for global state where needed
3. User Interface Design:
    - Responsive design using CSS flexbox
    - Glassmorphism styling for modern aesthetics
    - Consistent color scheme and typography
    - Interactive animations and transitions
    - Modal dialogs for forms and notifications
4. Client-Side Features:
    - Real-time webcam integration
    - QR code scanning
    - Form validation
    - Error handling
    - Loading states
    - Success/error notifications

Backend Implementation:

1. API Architecture:
    - RESTful endpoints for all operations
    - URL patterns:
        - /professor/* for professor operations
        - /student/* for student operations
        - /class/* for class management
2. Database Models:
    - Professor model for instructor data
    - Student model for student information
    - Class model for course details
    - ClassEnrollment for student-class relationships
    - Attendance for tracking records

- StudentImage for facial recognition data

3. View Logic:
    - Class-based views for complex operations
    - Function-based views for simpler endpoints
    - Decorators for authentication and permissions

Frontend-Backend Interaction:

1. API Communication:
    - Fetch API for HTTP requests
    - JSON data format for request/response
    - Async/await for handling asynchronous operations
    - Javascript Example flow:

```
const handleRegisterSubmit = async () => {
      try {
        const response = await fetch('/api/student/register', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify({ username, email, crn }),
        });
        const data = await response.json();
        // Handle response
      } catch (error) {
        // Handle error
      }
    };
```

2. Data Flow:
    - Frontend sends requests to backend endpoints
    - Backend processes requests and interacts with database
    - Response sent back to frontend
    - Frontend updates UI based on response

3. Security Measures:
    - CORS configuration
    - CSRF token validation
    - JWT authentication
    - Input sanitization
    - Error handling at both ends

4. Performance Optimization:
    - Lazy loading of components
    - Image optimization
    - Efficient database queries
    - Caching where appropriate

5. Experiences

Learning Outcomes: Through this project, I gained extensive experience with modern web development technologies and libraries. On the frontend, I learned to build dynamic user interfaces using React.js, implement smooth animations with Framer Motion, and integrate real-time scanning capabilities using HTML5-QRCode. The project enhanced my understanding of component-based architecture and state management in React applications.

On the backend, I gained proficiency with Django REST Framework and various Python libraries. Working with face-recognition for biometric authentication, qrcode for QR code generation, and Pillow for image processing expanded my knowledge of complex backend operations. The project also deepened my understanding of RESTful API design and secure authentication systems.

Problem-Solving Approaches: The most challenging aspects of the project included:

1. Facial Recognition Implementation
    - Solved accuracy issues by implementing multiple angle capture
    - Optimized image processing for better performance
    - Optimized image processing for better performance
2. Real-time QR Code Processing
    - Resolved scanning reliability issues
    - Implemented secure data encoding/decoding
    - Added duplicate scan prevention
3. System Integration
    - Coordinated frontend-backend communication
    - Managed complex state across components
    - Handled asynchronous operations effectively

Future Extensions:

1. Technical Enhancements:
    - Mobile application development
    - Cloud-based deployment
    - Real-time attendance analytics
    - Automated reporting system
2. Feature Additions:
    - Multiple course section support
    - Attendance pattern analysis
    - Integration with learning management systems
    - Automated absence notifications
3. Security Improvements:
    - Two-factor authentication
    - Advanced encryption methods
    - Biometric data protection
    - Enhanced audit logging

References