Exam Security System Analysis

Project Analysis & Requirements Specification

Project Title: Exam Security System (Attendify Integration) Course: Software Testing & Validation Module: Final Project

1. Project Overview

The Exam Security System is a web-based application designed to ensure the integrity of academic examinations. The system integrates with the existing "Attendify" platform to provide identity verification using Machine Learning (Computer Vision), seating plan compliance checking, and violation logging.

The core objective is to prevent exam fraud by verifying that the student entering the exam room matches their registered biometric data and is sitting in their assigned seat.

2. Actors and Roles

Based on the system analysis, the following actors are defined:

2.1. Human Actors

Exam Coordinator (Admin):

Responsible for global system configuration.

Creates exams, defines classrooms, and manages student rosters.

Generates and assigns the Seating Plan for each exam.

Views post-exam reports regarding attendance and security violations.

Proctor (Invigilator/Teacher):

Responsible for operational security during the exam.

Performs the Student Check-in using the application interface.

Captures student photos for real-time verification.

Manually overrides ML decisions if necessary and logs violations (e.g., "Wrong Seat", "Identity Mismatch").

Student:

The subject of the verification process.

Enters the exam room, presents themselves for check-in, and occupies the assigned seat.

2.2. System Actors

ML Service (Face Recognition):

An internal service wrapper that processes the live photo captured by the Proctor.

Compares the live image against the stored biometric encoding in the database and returns a match/no-match decision.

3. Functional Requirements

3.1. Authentication & Management

REQ-01: The system must allow users to log in with secure credentials (Username/Password).

REQ-02: The system must enforce Role-Based Access Control (RBAC). Only Admins can create exams; only Proctors can perform check-ins.

3.2. Exam & Seating Setup (Admin)

REQ-03: The Admin must be able to create an exam entity linked to a specific Class, Date, Time, and Room.

REQ-04: The Admin must be able to generate a Seating Plan, assigning specific seat numbers to students stored in the database.

REQ-05: The system must support importing or syncing student rosters from the main user database.

3.3. Check-in Workflow (Proctor)

REQ-06: The system must allow the Proctor to select a specific ongoing exam and view the list of eligible students.

REQ-07: The system must capture a live photo of the student via the device interface or file upload.

REQ-08 (ML Integration): The system must send the captured photo to the ML Service. The service compares it with the facial encoding data stored in the database.

Success: System marks identity as verified.

Failure: System alerts the Proctor of a mismatch.

3.4. Compliance & Violations

REQ-09 (Seating Check): Upon check-in, the system must automatically compare the student's current seat against the assigned seat in the seating plan.

REQ-10 (Violation Logging): If an identity mismatch occurs or seating compliance fails, the system must allow the Proctor to log a Violation.

Violations must include a reason (e.g., "Cheating", "Wrong ID", "Phone Usage") and optional notes.

4. Business Rules & Logic

These rules define the constraints for the Activity and Sequence diagrams.

Duplicate Check-in Rule: A student cannot be checked in twice for the same exam session. The system must reject duplicate attempts with a warning.

ML Threshold Rule: The ML comparison must meet a defined minimum confidence threshold to be automatically considered a "Match".

Manual Override Rule: If the ML Service fails (False Negative), the Proctor has the authority to manually approve the student but is required to verify their physical ID card first.

Seating Integrity Rule: A student cannot be marked as "Present" without a seat validation check. If the seat is incorrect, the system must prompt for correction or violation logging.

5. Data Requirements

The system relies on the following key entities:

Users: Stores Admin and Proctor credentials and roles.

Students: Stores student profile data and unique identifiers.

Student_Photos: Stores the reference images and encoded biometric data for ML verification.

Exams: Links a Class to a Date, Time, and Room.

Exam_Seating: Maps a Student to a specific Seat Number for a specific Exam.

Attendances / Details: Stores the final check-in status (present, absent) and timestamps.

Feedbacks (Violations): Stores incident reports, violation types, and evidence notes.

6. Diagram References

The following technical diagrams support this analysis and are located in the project documentation:

Use Case Diagram: Defines user interactions and system boundaries.

ER Diagram (ERD): Defines the database schema and entity relationships.

Sequence Diagram: Details the API calls, Check-in flow, and ML verification process.

Activity Diagram: Illustrates the decision logic for validation, violations, and manual overrides.

7. Jira Entegration

Epic 1: Analysis and Design

Sprint 1: Foundation & Modeling

- **Story: System Requirements Specification**
    - **Task:** Document user roles (Admin, Proctor, Student) and access levels.
    - **Task:** Define business validation rules for exam seating and check-in.
- **Story: Architectural Visualizations**
    - **Task:** Design **Use Case Diagrams** for actor-system interactions.
    - **Task:** Design **Entity Relationship Diagram (ERD)** including Exams, Students, and Violations.
    - **Task:** Design **Sequence Diagrams** for the Check-in data flow.
    - **Task:** Design **Activity Diagrams** for general system workflow.

Epic 2: Infrastructure and Database

Sprint 1: Foundation & Modeling

- **Task: Repository Environment Setup**
  - Initialize Bitbucket repository with structured folders (docs, src, tests).
- **Story: Database Layer Implementation**
  - **Task:** Execute SQL scripts to create the schema based on the ERD.
  - **Task:** Develop seed scripts to populate dummy data (students, classes, exams).

Epic 3: Implementation - Core Features

Sprint 2: Core Development

- **Story: Authentication & RBAC (Role-Based Access Control)**
  - **Task:** Implement secure login for Admin and Proctor roles.
  - **Task:** Develop middleware for role-based route protection.
- **Story: Exam Management Module**
  - **Task:** Create CRUD operations for Exams and Seating Plans.
  - **Task:** Implement Student Roster import/upload functionality.

Epic 4: Implementation - ML and Check-in

Sprint 3: Advanced Features & Integration

- **Story: ML Service Integration**
  - **Task:** Develop a wrapper for the face verification/image processing library.
- **Story: Intelligent Check-in Workflow**
  - **Task:** Implement the image upload and real-time verification logic.
  - **Task:** Build the seat-matching validation service.
- **Story: Violation Incident Management**
  - **Task:** Implement automated logging for mismatch/seat violations.
  - **Task:** Create a reporting interface for proctors to review incidents.

Epic 5: Testing and Validation

Sprint 4: Quality Assurance & Delivery

- **Story: Quality Assurance Documentation**
  - **Task:** Write comprehensive Test Cases (Functional, Negative, Edge Cases).

- **Story: Automated Testing Suite**
    - **Task:** Implement Unit Tests for seating algorithms and business rules.
    - **Task:** Develop Mock Tests for ML service dependencies.
- **Task: Final System UAT (User Acceptance Testing)**
    - Conduct end-to-end verification of the check-in process.