The Exam Security System

Muzaffer DEMİRHAN

Berkecan Hamdi AKYÜZ

Muhammet Mirza BEKİROĞLU

230706004

230706003

220706022

Project: The Exam Security System
Date: 2026-01-09

Supervisor: Professor Emre OLCA

Table of Contents

1. Introduction

1.1 Purpose

This document provides a detailed Software Requirements Specification for the Exam Security System. Its purpose is to provide a complete overview of the system's intended functionality, limitations and goals. This SRS will serve as the main reference document for the development team and the official agreement between all stakeholders, ensuring a shared vision for the final product.

1.2 Scope

The Exam Security System is an advanced, web-based application designed to support the integrity and security of in-person examinations. The system's scope is focused on exam-day operations, providing tools for proctors and administrators to manage student verification efficiently, seating arrangements, and incident reporting.

The core functionalities within this scope are:

- **Identity Verification:** Utilizing a machine learning component to match a student's live photo against their registered ID photo.
- **Seating Plan Compliance:** Ensuring students occupy their designated seats.
- **Incident and Violation Recording:** Creating a digital log of any irregularities or academic misconduct that occurs.
- **Administrative Management:** Allowing exam coordinators to set up exams, manage student rosters, and produce post-exam reports.

1.3 Overview

This document is designed to lead the reader from a high-level conceptual overview to detailed, specific requirements. It begins with the context and limitations of the system, outlines key stakeholders and their roles, and then carefully specifies the functional and non-functional requirements. The following sections elaborate on the data model, system architecture, and user interface principles. This structure ensures clarity and provides a complete picture of the project's requirements.

2. System Overview

2.1 System Context

The Exam Security System is designed to operate within an academic institution's examination framework. It is a self-contained, web-based application that will be accessed via standard web browsers on devices like laptops or tablets. The system is intended to interface with a pre-existing student information system to retrieve student rosters and photos, which are necessary for the verification process. Its primary operational environment is the exam hall on the day of the examination.

2.2 Design Constraints

The development and implementation of the system are subject to the following constraints:

- **Database Management System:** The system must use MS SQL as its relational database for all data persistence.
- **Platform:** The application must be web-based**,** ensuring accessibility across different operating systems without requiring native client installation.
- **Machine Learning Implementation:** The face verification component must be implemented using a pre-existing, library-based approach. Training a complex deep learning model from scratch is explicitly outside the project's scope. The focus is on the successful integration and application of the ML model within the workflow.
- **Development Language:** The backend will be developed using Python, and the frontend will utilize a modern JavaScript framework.

3. Stakeholder Analysis

3.1 Identification

The system has three key user roles, each with definite permissions and goals:

- **Exam Coordinator (Admin):** The administrative user responsible for pre-exam setup and post-exam analysis.
- **Proctor (Invigilator):** The on-the-ground staff member responsible for executing exam-day security procedures.
- **Student:** The end-user subject to the verification and compliance checks, but not a direct system operator.

3.2 Responsibilities

Stakeholder	Responsibilities
Exam Coordinator (Admin)	- Manage user accounts for Proctors. - Create and configure exam sessions (exam name, room, date/time). - Import student rosters for each exam. - Create and assign seating plans. - Generate and review reports on check-ins, identity mismatches, and violations.
Proctor (Staff)	- Log into the system to access assigned exam sessions. - Execute the student check-in workflow for each student. - Capture or upload a live photo of the student. - Review the ML-based identity verification result. - Confirm the student is in the correct seat as per the system's seating plan. - Log any observed violations with detailed notes and optional photographic evidence.
Student	- Provide their student ID for lookup. - Consent to having their photo taken for verification. - Proceed to their assigned seat.

4. Business Rules

This section defines the business rules that limit and manage the behavior of the Exam Security System. Business rules serve as organizational policies, operational constraints, and decision logic that must be charged by the system in addition to functional requirements.

BR1: Authentication & Authorization Rules

BR1.1
Only authenticated users with valid personal details shall be allowed to access the system.

BR1.2
Users shall only be permitted to perform actions that correspond to their assigned role:

- Admin users may access all system functionalities.
- Proctor users may only access exam-day operational features (check-in and violation logging).

BR1.3
Proctors shall not be permitted to create, edit, or delete exams, rooms, or seating plans.

BR1.4
Inactive user sessions shall be automatically cancelled after a predefined timeout period to prevent unauthorized access.

BR2: Exam & Roster Management Rules

BR2.1
An exam session must be created before any student roster or seating plan can be assigned.

BR2.2
Each student may appear only once per exam roster**.**

BR2.3
A student cannot be assigned to an exam without a related room configuration.

BR2.4
Each seat code within a room shall be assigned to at most one student per exam**.**

BR2.5
Automatic seat assignment shall respect room capacity and layout constraints.

BR2.6
An exam roster cannot be modified once the exam has started.

BR3: Student Check-in Rules

BR3.1
A student must exist in the exam roster before a check-in attempt is allowed.

BR3.2
A student may not complete more than one successful check-in for the same exam.

BR3.3
Check-in attempts shall only be permitted during the scheduled exam time window.

BR3.4
Each check-in attempt must include:

- A live captured or uploaded photo
- An ML verification result
- A seat compliance confirmation

BR3.5
If the ML verification result is "No Match" or "Low Confidence", the check-in shall be marked accordingly and flagged for administrative review.

BR3.6
If the student is seated in a location different from their assigned seat, the system shall record a Seating Violation.

BR3.7
All check-in attempts shall be logged and saved for audit purposes, including unsuccessful attempts.

BR4: Machine Learning Verification Rules

BR4.1
The ML component shall not make final decisions independently; it shall return a confidence score and verification result to the system.

BR4.2
Verification outcomes shall be classified into one of the following states:

- Match
- No Match
- Low Confidence

BR4.3
Low-confidence results shall require manual review by an Admin.

BR4.4
The ML model itself shall not store student identity data; all data persistence shall be managed by the system database.

BR5: Violation Logging Rules

BR5.1
Only Proctors assigned to an active exam session may log violations for that exam.

BR5.2
A violation record must be associated with:

- A specific student
- A specific exam
- A reporting Proctor

BR5.3
Each violation must include a violation reason and timestamp.

BR5.4
Violation records cannot be edited or deleted after submission.

BR6: Reporting Rules

BR6.1
Reports shall only be accessible to Admin users.

BR6.2
Reports shall show the final, immutable state of check-ins and violations once an exam is completed.

BR6.3
Mismatch and violation reports shall include sufficient detail to support audit and disciplinary processes.

BR7: Data Integrity & Audit Rules

BR7.1
All critical system actions (check-ins, violations, ML decisions) shall be timestamped.

BR7.2
Historical check-in logs shall not be overwritten or deleted.

BR7.3
Sensitive data access (photos, embeddings) shall be restricted to authorized system processes only.

BR8: Operational Constraints

BR8.1
The system shall prevent check-in operations if the backend services or database are unavailable.

BR8.2
In the event of system failure, no partial check-in records shall be committed.

5. Functional Requirements (FR)

This section details the specific functionalities the system must provide, categorized by feature area.

FR1: Authentication and Access Control

- **FR1.1:** The system shall provide a secure login page for users.
- **FR1.2:** The system shall enforce role-based access control. Users logged in as "Exam Coordinator" shall have access to all administrative functions, while users logged in as "Proctor" shall only have access to exam-day operational functions (check-in, violation logging).
- **FR1.3:** The system shall include a secure session management mechanism, including session timeout for inactivity.
- **FR1.4:** The system shall enforce differentiated access permissions for Proctors, allowing read-only access to exam rosters and write access only to check-in and violation logging functionalities.

FR2: Exam and Roster Management (Admin)

- **FR2.1:** The system shall allow an Exam Coordinator to create a new exam session by specifying an exam name, room location, and the scheduled date and time.
- **FR2.2:** The system shall provide a mechanism for the Exam Coordinator to import a student roster for an exam from a CSV file. The file must contain at a minimum: Student ID, Student Name, and a path/URL to their official ID photo.
- **FR2.3:** The system shall allow an Exam Coordinator to create a seating plan for an exam. This can be done by defining the room layout (rows/columns) and assigning students to specific seat codes.
- **FR2.4:** The system shall allow an Exam Coordinator to manually enter or edit student roster information when CSV import is not used.
- **FR2.5:** The system shall support automatic seat assignment for students based on the defined room layout, with the option for manual adjustments by the Exam Coordinator.

FR3: Student Check-in Workflow (Proctor)

- **FR3.1:** The Proctor shall be able to select an active exam session and look up a student by entering their Student ID.
- **FR3.2:** Upon finding a student, the system shall display their registered information, including their name and official ID photo.
- **FR3.3:** The system shall provide an interface for the Proctor to capture a new, live photo of the student using the device's camera or upload a photo file.
- **FR3.4:** The system shall automatically send the live photo and the stored ID photo to the ML verification component. It shall display the result to the Proctor as a clear "Match" or "No Match" decision, potentially with a similarity score.
- **FR3.5:** The system shall display the student's assigned seat code. The Proctor must be able to visually confirm the student's seating and mark compliance in the system.
- **FR3.6:** The system shall record the outcome of the entire check-in along with a timestamp.
- **FR3.7:** The system shall support an additional ML verification outcome state indicating "Low Confidence" when the similarity score does not meet the acceptance threshold.

FR4: Violation Logging (Proctor)

- **FR4.1:** During an exam, a Proctor shall be able to select a student and log a new violation.
- **FR4.2:** The violation logging form must include a dropdown menu of common violation reasons, a text field for detailed notes, and an optional field to upload a photographic evidence image.
- **FR4.3:** Each logged violation shall be automatically timestamped and associated with the specific student and exam session.

FR5: Reporting (Admin)

- **FR5.1:** The Exam Coordinator shall be able to generate a "Check-in Report" for any completed exam session. This report must list all students on the roster and their final check-in status with timestamps.
- **FR5.2:** The system shall provide a "Mismatch Report" that filters and displays only the students whose identity verification resulted in a "No Match."
- **FR5.3:** The system shall provide a "Violations Report" that details all incidents logged during an exam, including the student's name, the violation reason, proctor's notes, and any evidence images.
- **FR5.4:** The system shall provide a real-time dashboard displaying high-level exam statistics, including the number of students checked in, pending check-ins, and logged violations.

6. Non-Functional Requirements (NFR)

NFR1: Performance

  - Student lookup and retrieval of details shall be completed in under 2 seconds.

- The ML-based face verification process (image upload, comparison, and result display) shall take no longer than 8 seconds.
- Report generation for an exam with up to 500 students shall be completed within 15 seconds.

NFR2: Reliability

- The system must guarantee an uptime of 99.5% during scheduled examination periods. It should include robust error handling to prevent crashes during critical operations like student check-in.

NFR3: Security

- All user passwords must be securely hashed and salted using a modern algorithm.
- All network communication between the client and server must be encrypted using TLS/SSL (HTTPS).
- Sensitive student data, particularly photographs, must be encrypted at rest. Access to raw image files on the server should be restricted.

NFR4: Usability

The interface for Proctors must be highly intuitive, optimized for speed, and usable on touch-based tablet devices. Workflows should require minimal clicks and navigation.

NFR5: Scalability

The system architecture must be capable of supporting at least 50 Proctors using the check-in functionality concurrently for a single large-scale exam.

7. Data Design

The data model is designed to support secure exam management, student identity verification, seating compliance, and violation tracking. The database schema is implemented using MS SQL and enforces data integrity through primary keys, foreign keys, and constraints.

User Table

Stores system users including Exam Coordinators (Admins) and Proctors.

Columns:

- **UserID** (PK): Unique identifier for each system user.
- **Username**: Unique login name.
- **PasswordHash**: Securely hashed user password.
- **Role**: User role (Admin, Proctor).
- **FullName**: Full name of the user.

Student Table

Stores student identity information and primary reference data for identity verification.

Columns:

- **StudentID** (PK): Unique identifier for each student.
- **UniversityID**: Institution-issued unique student identifier.
- **FullName**: Student’s full name.
- **Email**: Student contact email.
- **ReferencePhotoUrl**: Path or URL to the primary stored identity photo.
- **FaceEmbedding**: Stored facial feature representation (JSON or encoded string).

StudentPhoto Table

Stores multiple reference photos for each student to improve identity verification reliability.

Columns:

- **PhotoID** (PK): Unique identifier for each photo.
- **StudentID** (FK): Reference to the associated student.
- **PhotoUrl**: Path or URL to the stored photo.
- **CreatedAt**: Timestamp indicating when the photo was added.

Room Table

Stores examination room information and high-level seating layout configurations.

Columns:

- **RoomID** (PK): Unique identifier for each room.
- **RoomName**: Human-readable room name or code.
- **Capacity**: Maximum number of students allowed in the room.
- **LayoutConfig**: JSON-based layout definition used as a legacy or cached representation.

Seat Table

Defines the normalized seating structure for each exam room.

Columns:

- **SeatID** (PK): Unique identifier for each seat.
- **RoomID** (FK): Reference to the room containing the seat.
- **RowLabel**: Row identifier (e.g., “A”, “1”).
- **ColLabel**: Column identifier (e.g., “1”, “2”).

- **IsAccessible**: Indicates whether the seat is usable.
- **Status**: Operational state of the seat (Active, Maintenance).

Exam Table

Stores exam session details including scheduling, room assignment, and ownership.

Columns:

- **ExamID** (PK): Unique identifier for each exam.
- **CourseCode**: Academic course identifier.
- **ExamName**: Name of the exam.
- **StartTime**: Scheduled start date and time.
- **DurationMinutes**: Length of the exam in minutes.
- **RoomID** (FK): Assigned exam room.
- **CreatedBy** (FK): Admin user who created the exam.

ExamRoster Table

Maps students to exams and manages seating assignments and attendance status.

Columns:

- **RosterID** (PK): Unique identifier for each exam-student assignment.
- **ExamID** (FK): Associated exam.
- **StudentID** (FK): Associated student.
- **AssignedSeat**: Seat code assigned to the student (e.g., “B12”).
- **Status**: Attendance status (Scheduled, Present, Absent).
- **CheckInTime**: Timestamp of successful check-in.

Constraints:

- Each student may appear only once per exam.

CheckInLog Table

Records all student check-in attempts for auditing and validation purposes.

Columns:

- **LogID** (PK): Unique identifier for each check-in log entry.
- **RosterID** (FK): Reference to the exam roster entry.
- **Timestamp**: Date and time of the check-in attempt.
- **MLConfidenceScore**: Similarity score returned by the ML verification service.
- **IsMatch**: Indicates whether identity verification succeeded.
- **IsSeatCorrect**: Indicates whether the student was seated correctly.
- **CapturedImagePath**: Path or URL to the captured live photo.

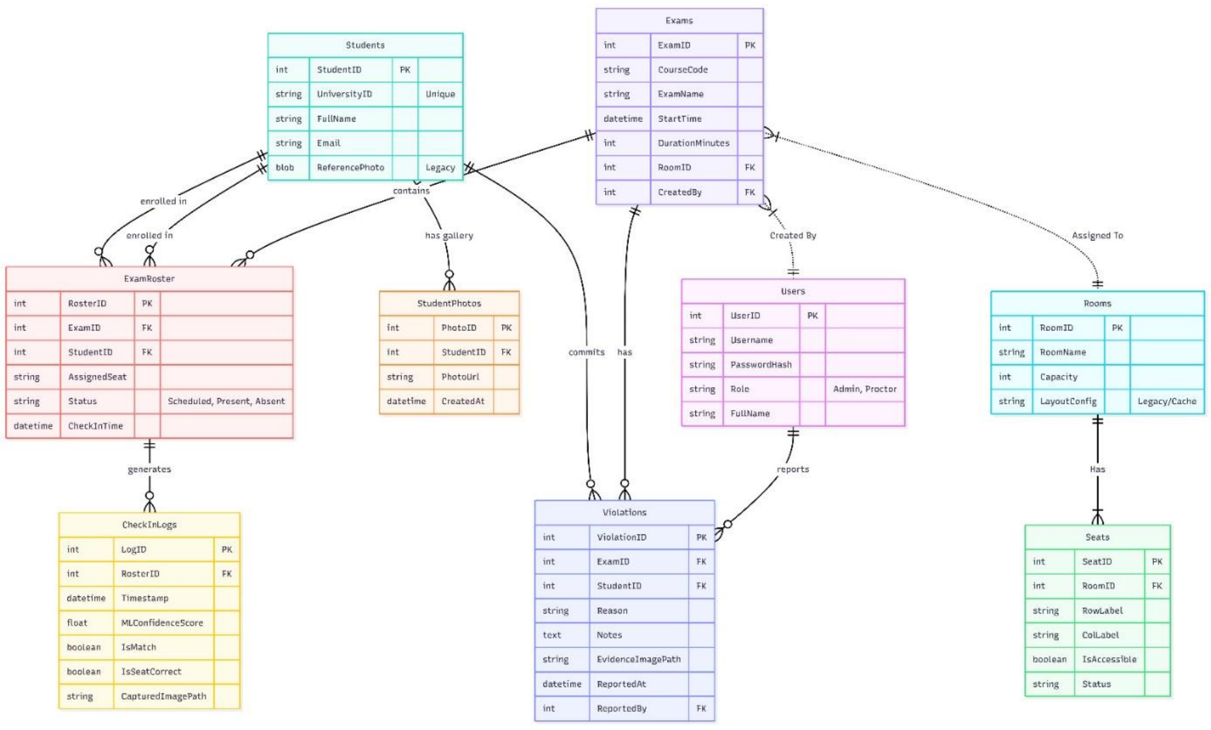
Violation Table

Stores logged exam violations along with evidence and reporting details.

Columns:

- **ViolationID** (PK): Unique identifier for each violation.
- **ExamID** (FK): Associated exam.
- **StudentID** (FK): Associated student.
- **Reason**: Violation category.
- **Notes**: Detailed description provided by the Proctor.
- **EvidenceImagePath**: Optional path or URL to photographic evidence.
- **ReportedAt**: Timestamp of the violation report.
- **ReportedBy** (FK): Proctor who reported the violation.

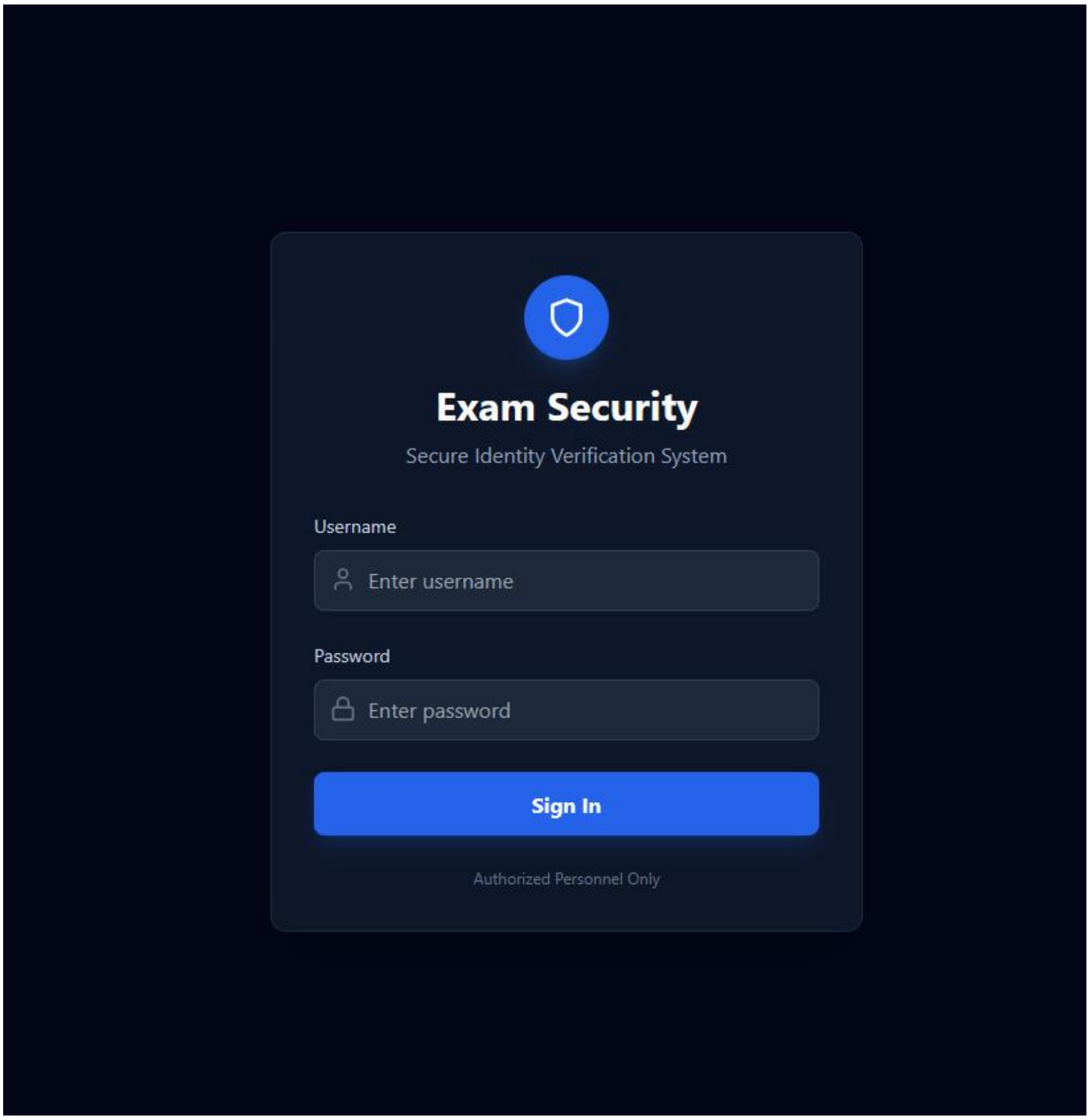
ERD Diagram



8. UI (User Interface)

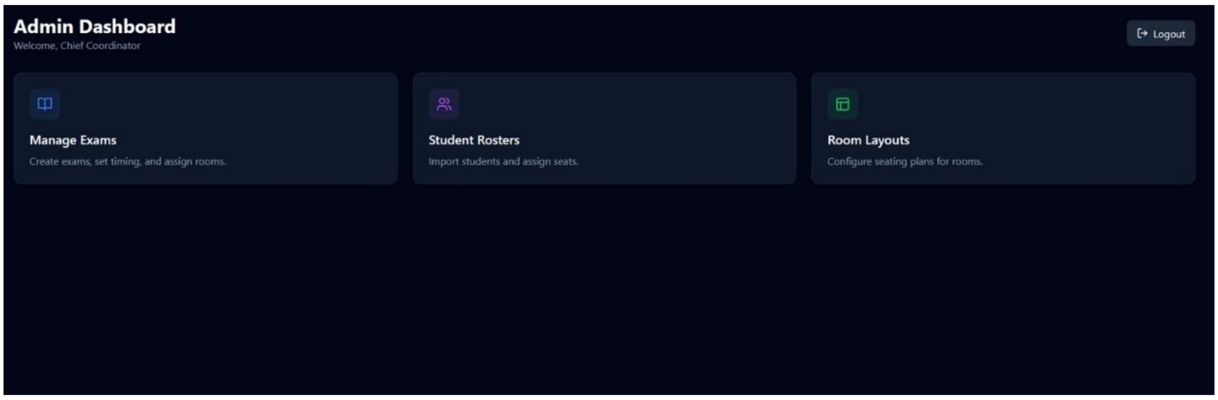
Login

- This section is for loginning in to the system. It includes headline text, username input, password input and sign-in button.



Admin

- This section is Admin’s main dashboard. It includes headline text, log-out button, manage exams button, student rosters button and room layouts button.



Admin Dashboard

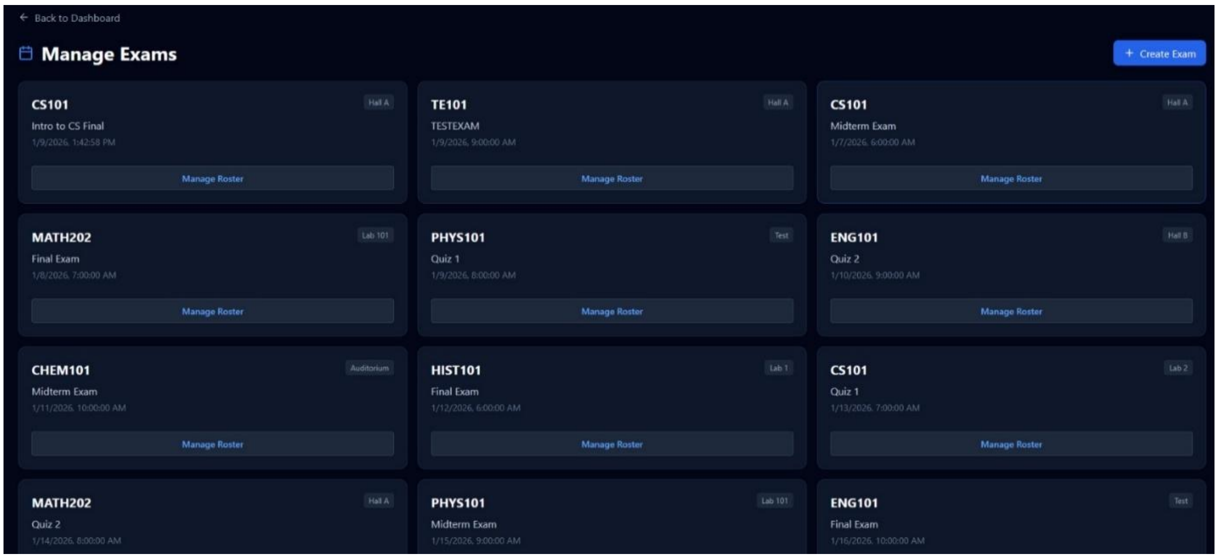
Welcome, Chief Coordinator

Manage Exams – Create exams, set timing, and assign rooms.

Student Rosters – Import students and assign seats.

Room Layouts – Configure seating plans for rooms.

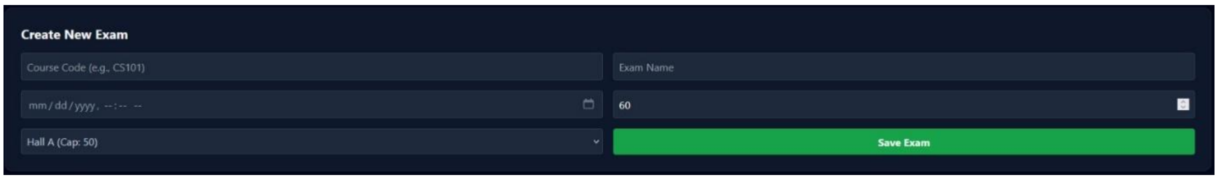
- This section is to manage exams and create new ones. It includes headline text, create exam button, Exam managemenet section



Manage Exams

CS101 (Hall A) Intro to CS Final 1/9/2026, 1:42:58 PM	TE101 (Hall A) TESTEXAM 1/9/2026, 9:00:00 AM	CS101 (Hall A) Midterm Exam 1/7/2026, 6:00:00 AM
MATH202 (Lab 101) Final Exam 1/8/2026, 7:00:00 AM	PHYS101 (Test) Quiz 1 1/9/2026, 8:00:00 AM	ENG101 (Hall B) Quiz 2 1/10/2026, 9:00:00 AM
CHEM101 (Auditorium) Midterm Exam 1/11/2026, 10:00:00 AM	HIST101 (Lab 1) Final Exam 1/12/2026, 6:00:00 AM	CS101 (Lab 2) Quiz 1 1/13/2026, 7:00:00 AM
MATH202 (Hall A) Quiz 2 1/14/2026, 8:00:00 AM	PHYS101 (Lab 101) Midterm Exam 1/15/2026, 9:00:00 AM	ENG101 (Test) Final Exam 1/16/2026, 10:00:00 AM

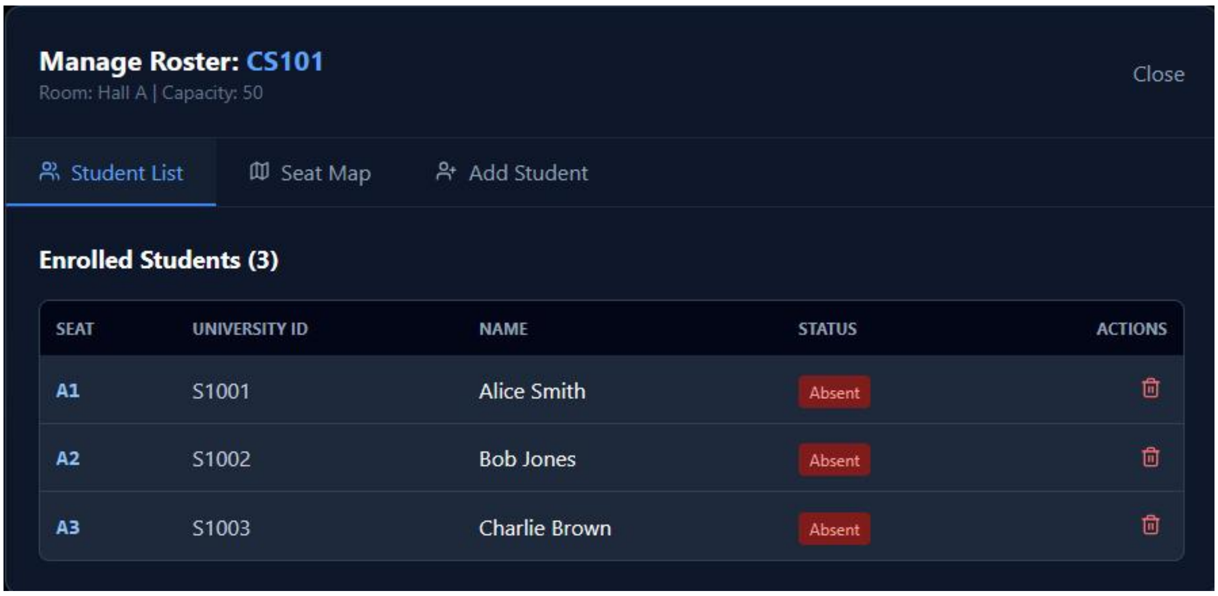
- This section is to create new exams. It includes headline text, course code input, exam name input, exam date input, exam time input, exam room input and save exam button.



Create New Exam

Course Code (e.g., CS101)	Exam Name
mm / dd / yyyy, --:-- --	60
Hall A (Cap: 50)	Save Exam

- This section is to manage student, seats and new students of the exam. It includes headline text, close button, student list section, seat map section, add student section, seats section and student input section.



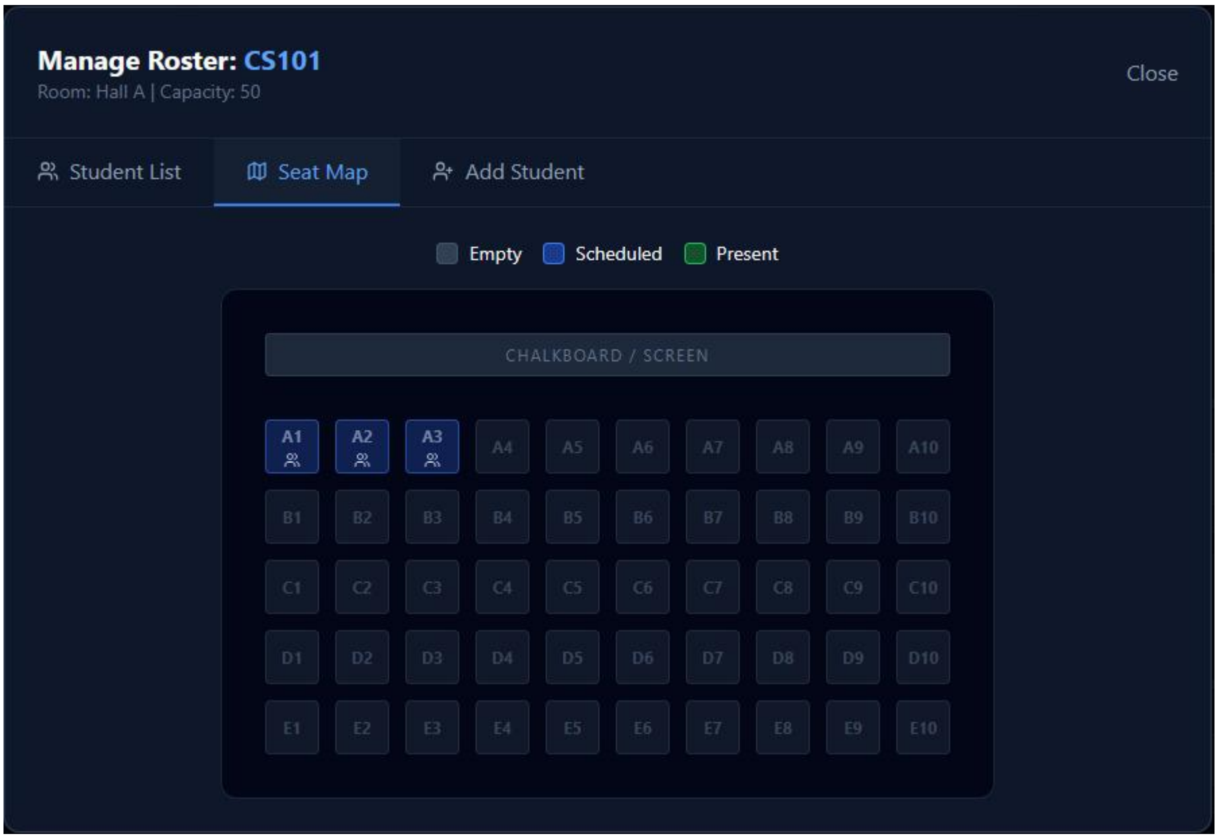
Manage Roster: CS101
Room: Hall A | Capacity: 50

Close

Student List | Seat Map | Add Student

Enrolled Students (3)

SEAT	UNIVERSITY ID	NAME	STATUS	ACTIONS
A1	S1001	Alice Smith	Absent	🗑
A2	S1002	Bob Jones	Absent	🗑
A3	S1003	Charlie Brown	Absent	🗑



Manage Roster: CS101
Room: Hall A | Capacity: 50

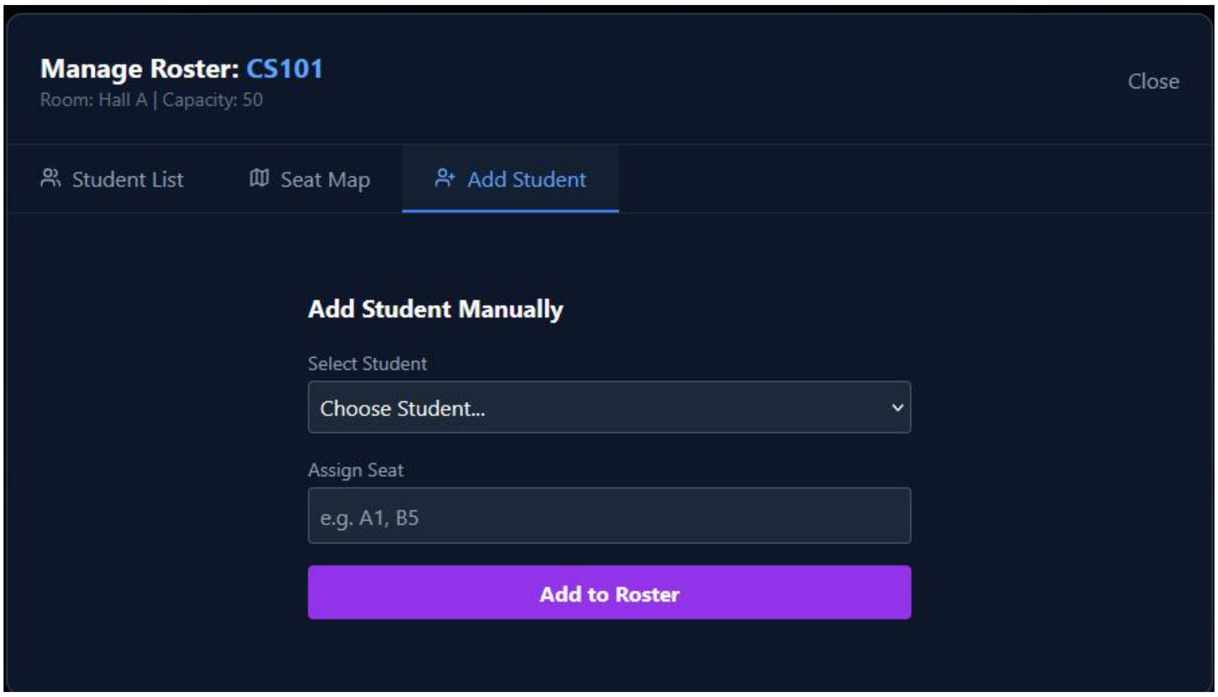
Close

Student List | Seat Map | Add Student

▨ Empty 🟦 Scheduled 🟩 Present

CHALKBOARD / SCREEN

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
E1	E2	E3	E4	E5	E6	E7	E8	E9	E10



Manage Roster: CS101

Room: Hall A | Capacity: 50

Close

Student List | Seat Map | Add Student

Add Student Manually

Select Student

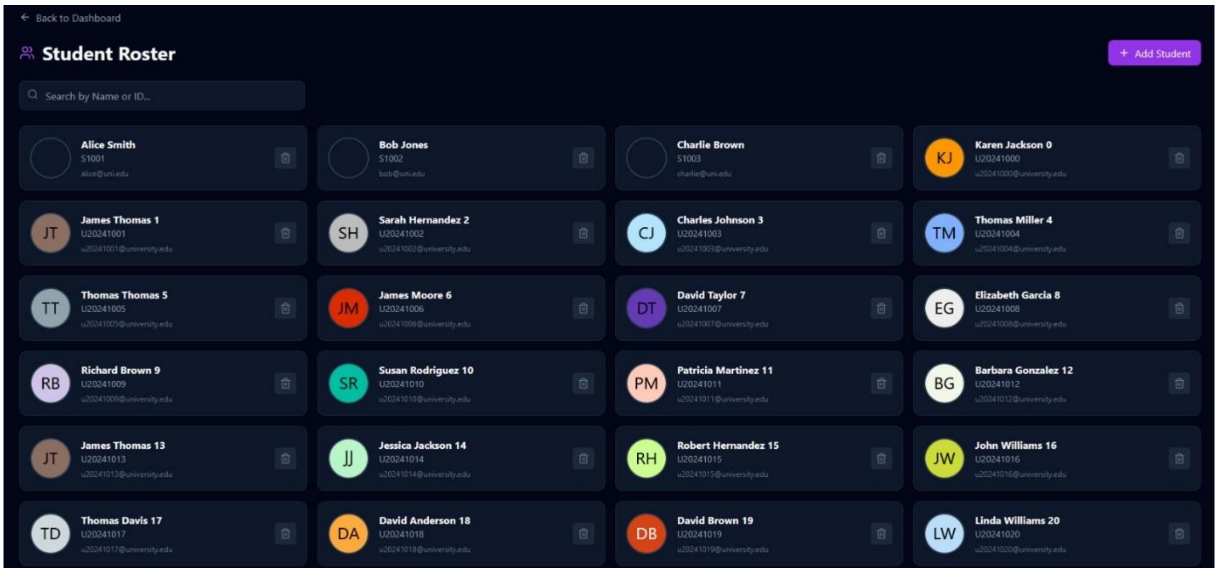
Choose Student...

Assign Seat

e.g. A1, B5

Add to Roster

- This section is to manage students and add new students. It includes headline text, add student button and students section.



← Back to Dashboard

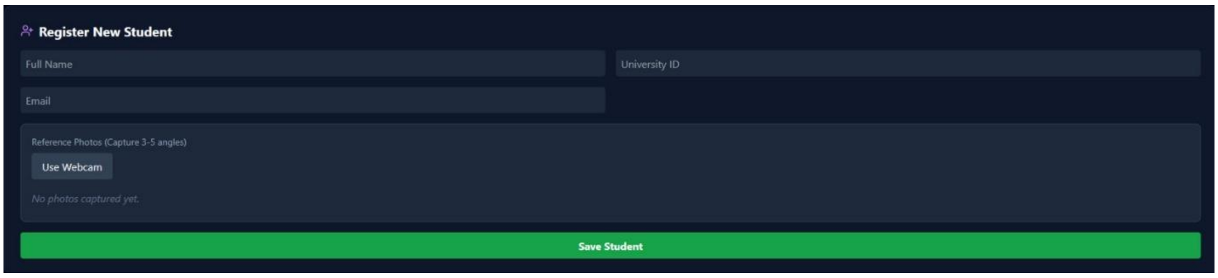
Student Roster

+ Add Student

Search by Name or ID...

Alice Smith S1001 alice@uni.edu	Bob Jones S1002 bob@uni.edu	Charlie Brown S1003 charlie@uni.edu	KJ Karen Jackson 0 U20241000 u20241000@university.edu
JT James Thomas 1 U20241001 u20241001@university.edu	SH Sarah Hernandez 2 U20241002 u20241002@university.edu	CJ Charles Johnson 3 U20241003 u20241003@university.edu	TM Thomas Miller 4 U20241004 u20241004@university.edu
TT Thomas Thomas 5 U20241005 u20241005@university.edu	JM James Moore 6 U20241006 u20241006@university.edu	DT David Taylor 7 U20241007 u20241007@university.edu	EG Elizabeth Garcia 8 U20241008 u20241008@university.edu
RB Richard Brown 9 U20241009 u20241009@university.edu	SR Susan Rodriguez 10 U20241010 u20241010@university.edu	PM Patricia Martinez 11 U20241011 u20241011@university.edu	BG Barbara Gonzalez 12 U20241012 u20241012@university.edu
JT James Thomas 13 U20241013 u20241013@university.edu	JJ Jessica Jackson 14 U20241014 u20241014@university.edu	RH Robert Hernandez 15 U20241015 u20241015@university.edu	JW John Williams 16 U20241016 u20241016@university.edu
TD Thomas Davis 17 U20241017 u20241017@university.edu	DA David Anderson 18 U20241018 u20241018@university.edu	DB David Brown 19 U20241019 u20241019@university.edu	LW Linda Williams 20 U20241020 u20241020@university.edu

- This section is add new student. It includes headline text, student name input, student id input, student e-mail input, student reference photo input and save student button.


Register New Student

Full Name

University ID

Email

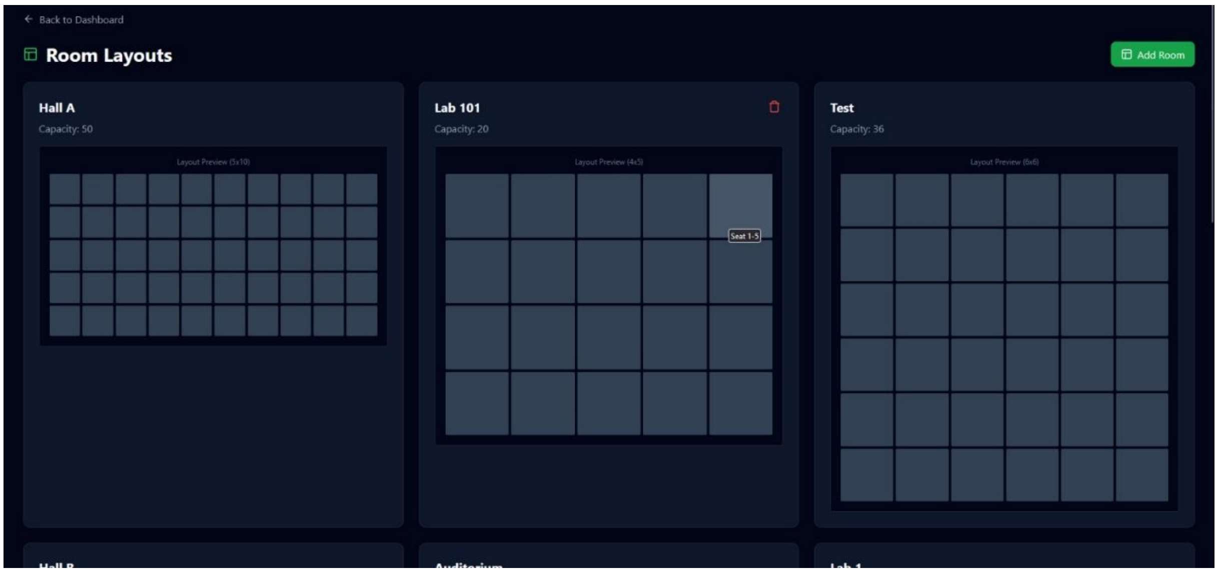
Reference Photos (Capture 3-5 angles)

Use Webcam

No photos captured yet.

Save Student

This section is to manage room layouts and add new ones. It includes headline text, add room button, rooms section.


Back to Dashboard

Room Layouts

Add Room

Hall A

Capacity: 50

Layout Preview (5x10)

Lab 101

Capacity: 20

Layout Preview (4x5)

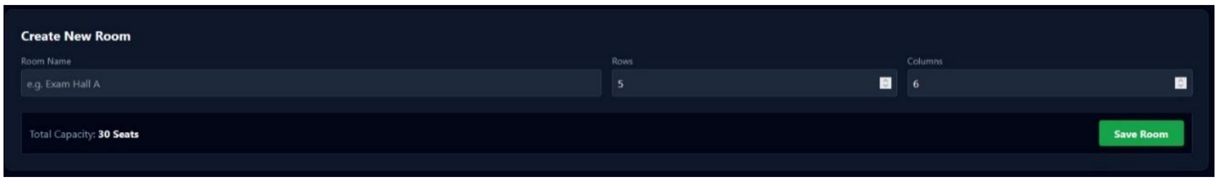
Seat 1-5

Test

Capacity: 36

Layout Preview (6x6)

- This section is add room section. It includes headline text, room name input, room rows input, rom columns input, total capacity section and save room button.


Create New Room

Room Name

e.g. Exam Hall A

Rows

5

Columns

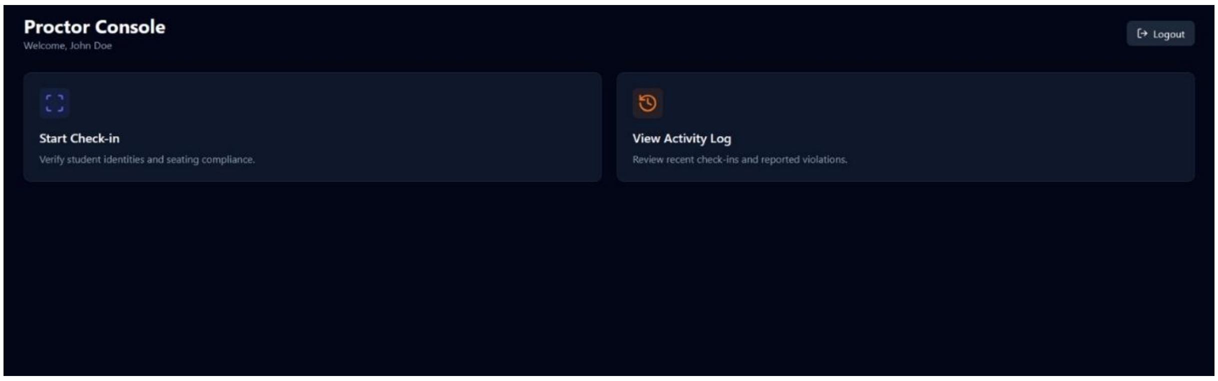
6

Total Capacity: **30 Seats**

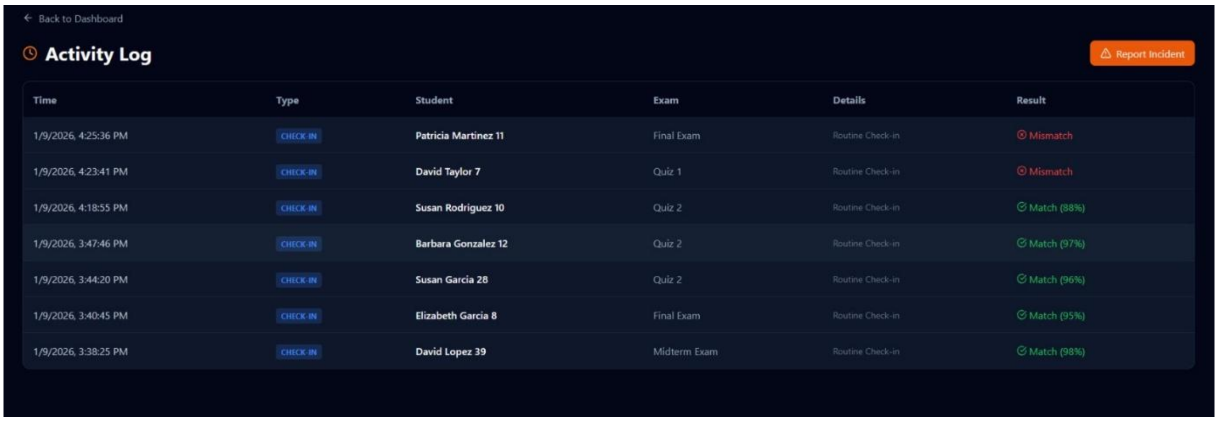
Save Room

Proctor

- This section is Proctor’s main dashboard. It includes headline text, log-out button, start check-in button and view activity log button.

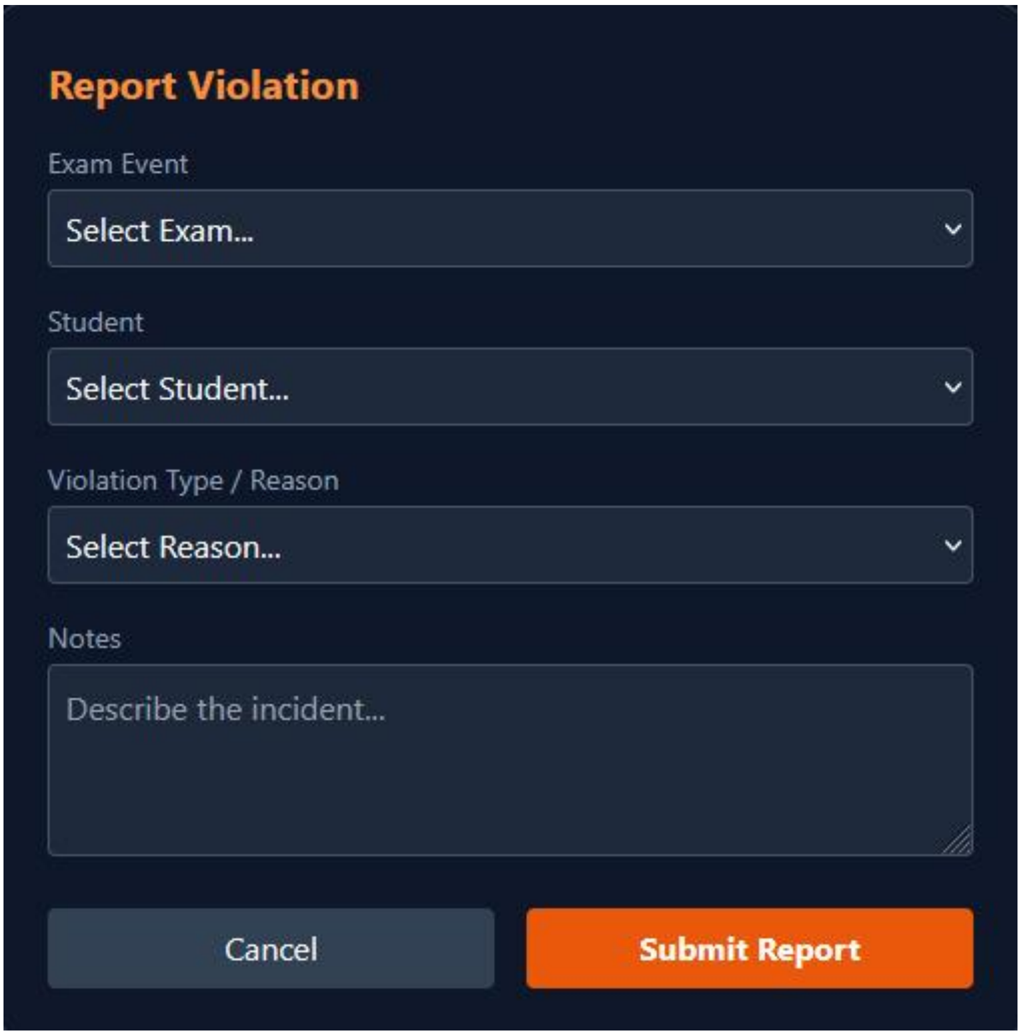

Proctor Console
Welcome, John Doe
Logout
Start Check-in
Verify student identities and seating compliance.
View Activity Log
Review recent check-ins and reported violations.

This section is activity log section. You can see which student is match or mismatch. You can report an incident too. It includes headline text, report incident button and activity log section.



Time	Type	Student	Exam	Details	Result
1/9/2026, 4:25:36 PM	CHECK-IN	Patricia Martinez 11	Final Exam	Routine Check-in	Mismatch
1/9/2026, 4:23:41 PM	CHECK-IN	David Taylor 7	Quiz 1	Routine Check-in	Mismatch
1/9/2026, 4:18:55 PM	CHECK-IN	Susan Rodriguez 10	Quiz 2	Routine Check-in	Match (88%)
1/9/2026, 3:47:46 PM	CHECK-IN	Barbara Gonzalez 12	Quiz 2	Routine Check-in	Match (97%)
1/9/2026, 3:44:20 PM	CHECK-IN	Susan Garcia 28	Quiz 2	Routine Check-in	Match (96%)
1/9/2026, 3:40:45 PM	CHECK-IN	Elizabeth Garcia 8	Final Exam	Routine Check-in	Match (95%)
1/9/2026, 3:38:25 PM	CHECK-IN	David Lopez 39	Midterm Exam	Routine Check-in	Match (98%)

This section is report violation section. It includes headline text, exam input, student input, violation type input, description input, cancel button and submit report button.


Report Violation

Exam Event

Select Exam...

Student

Select Student...

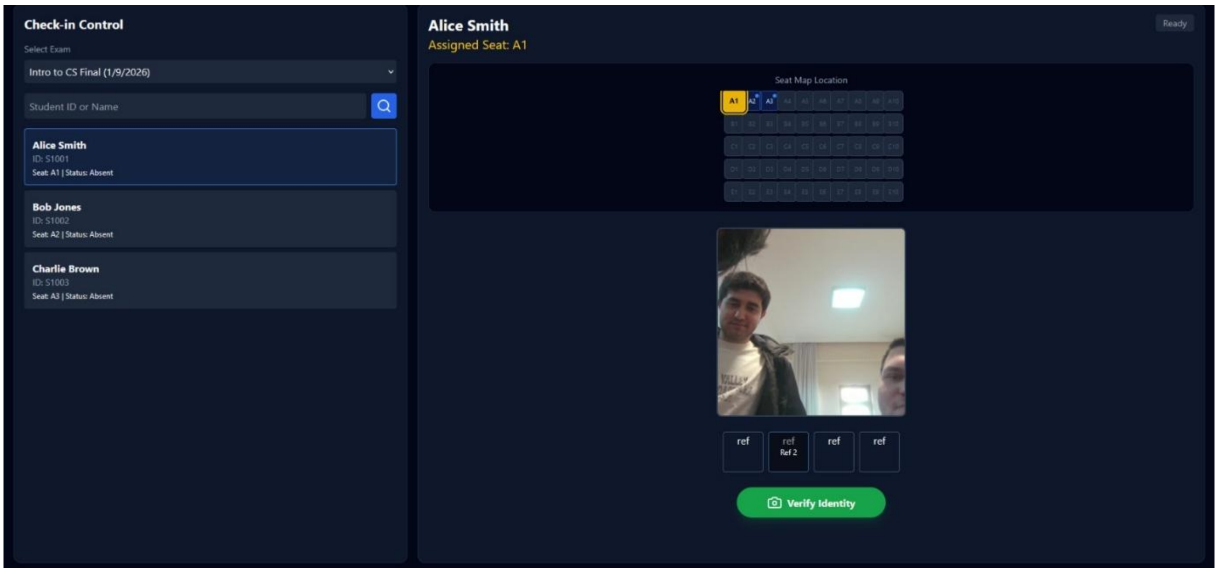
Violation Type / Reason

Select Reason...

Notes

Describe the incident...

This section is start check-in section. It checks if the student is matched and at the correct seat. It includes headline text, student search filter, students section, seat section, camera section and verify button which concludes if it is match or not.



Check-in Control

Select Exam

Intro to CS Final (1/9/2026)

Student ID or Name

Alice Smith
ID: S1001
Seat: A1 | Status: Absent

Bob Jones
ID: S1002
Seat: A2 | Status: Absent

Charlie Brown
ID: S1003
Seat: A3 | Status: Absent

Alice Smith
Assigned Seat: A1

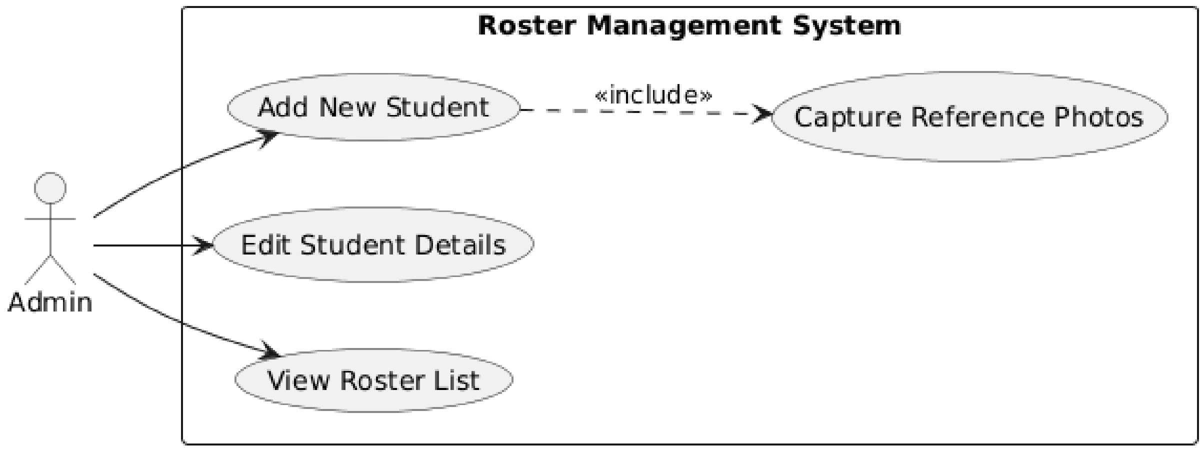
9. References

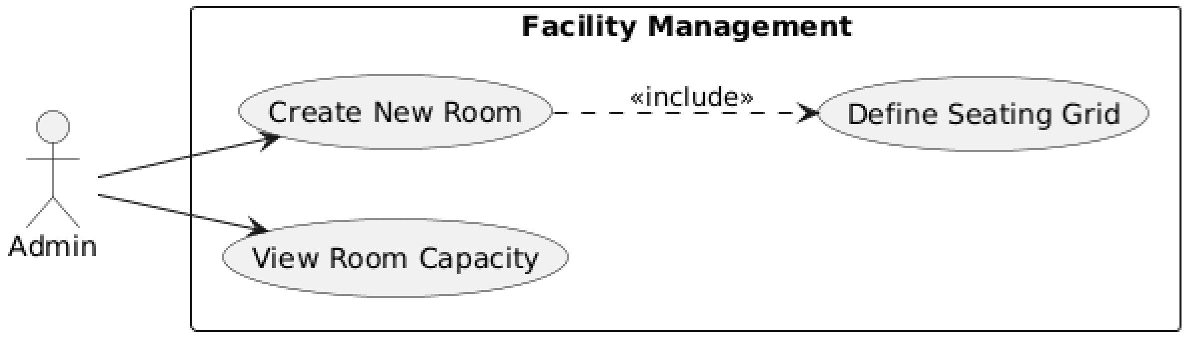
PlantUML for diagrams.
https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa700003

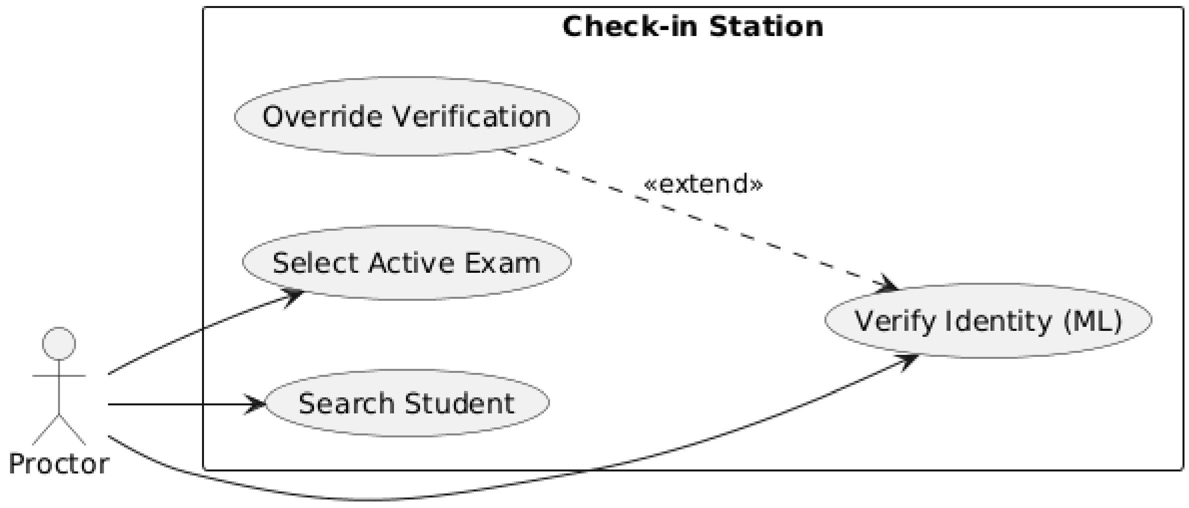
Mermaid Chart for ERD diagram.
https://www.mermaidchart.com/

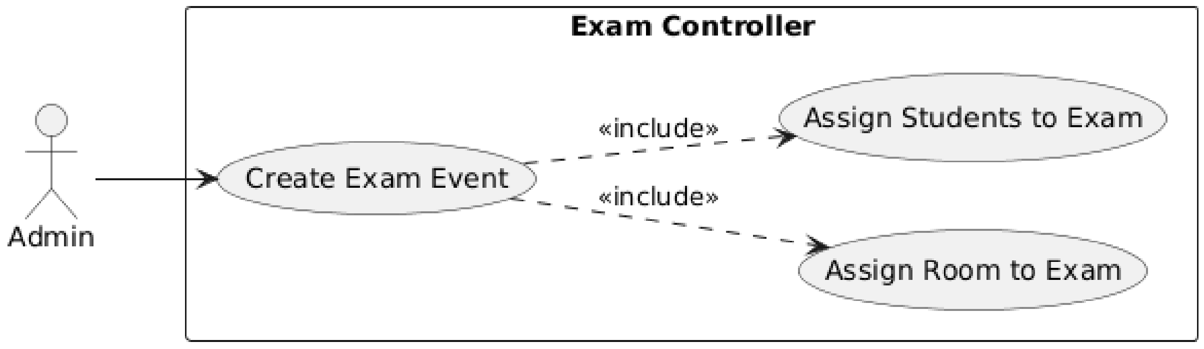
10. Appendices

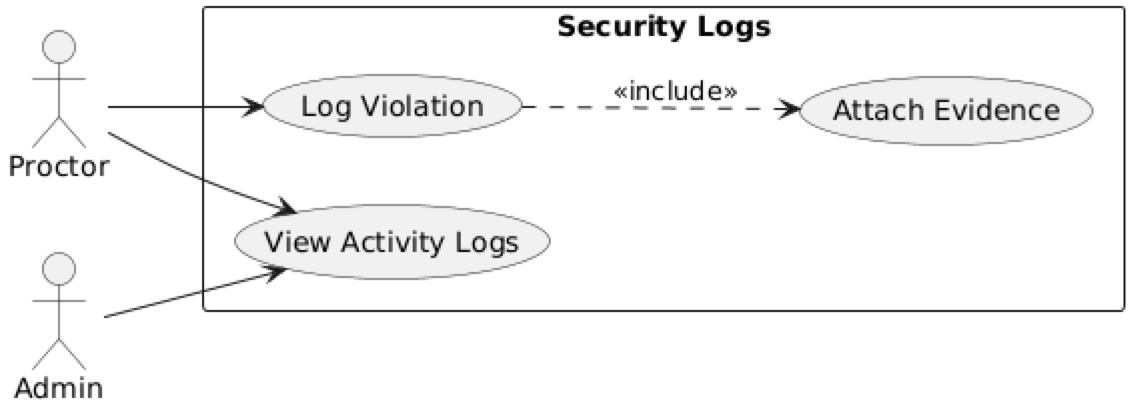
A. Use Case Diagrams


Roster Management System
Admin
Add New Student
«include»
Capture Reference Photos
Edit Student Details
View Roster List


Facility Management
Admin
Create New Room
«include»
Define Seating Grid
View Room Capacity

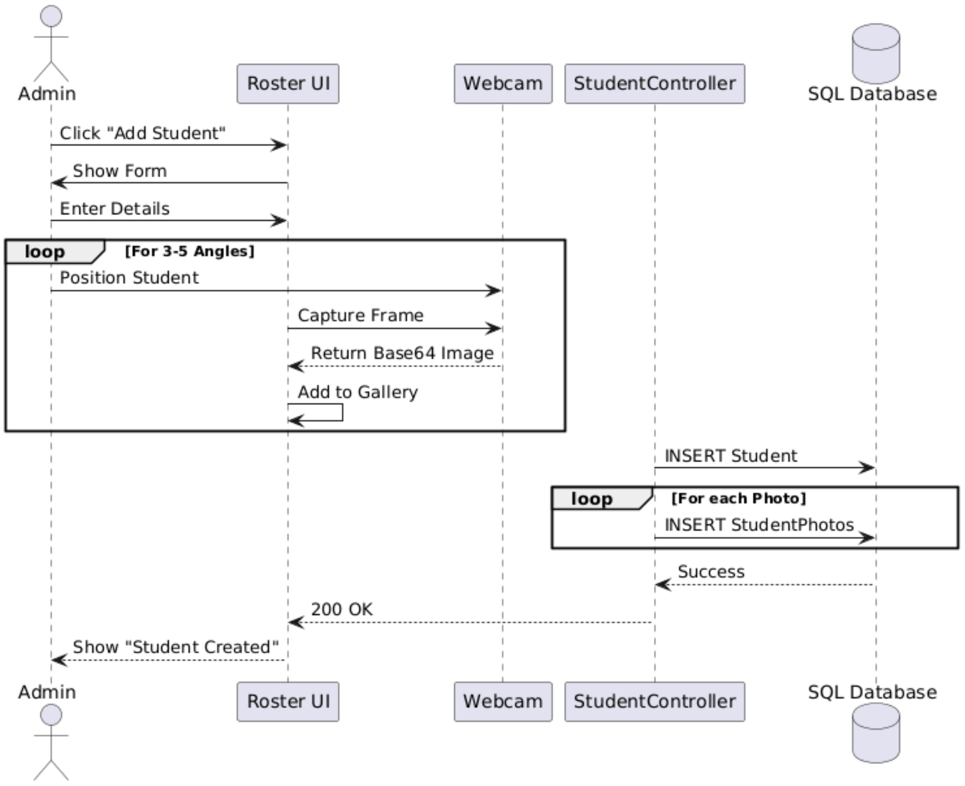

Check-in Station
Override Verification
«extend»
Select Active Exam
Verify Identity (ML)
Search Student
Proctor



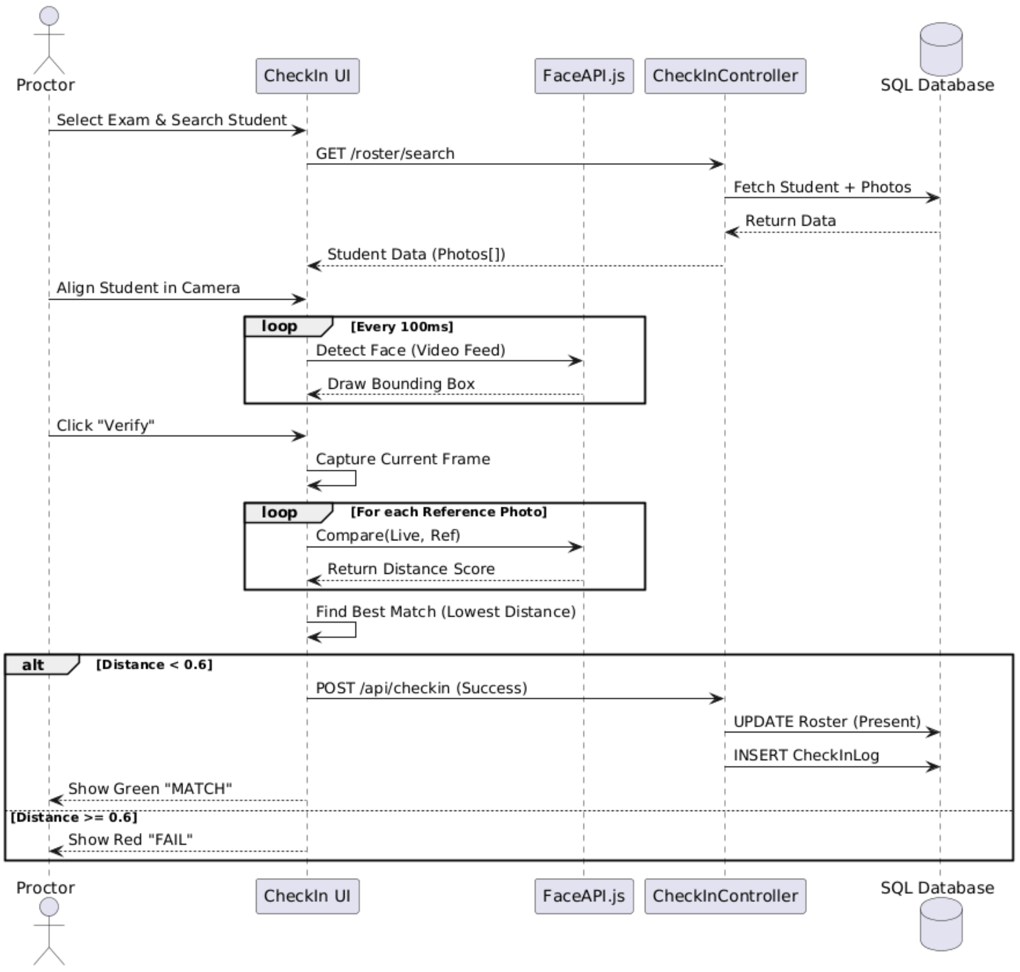


B. Sequence Diagrams

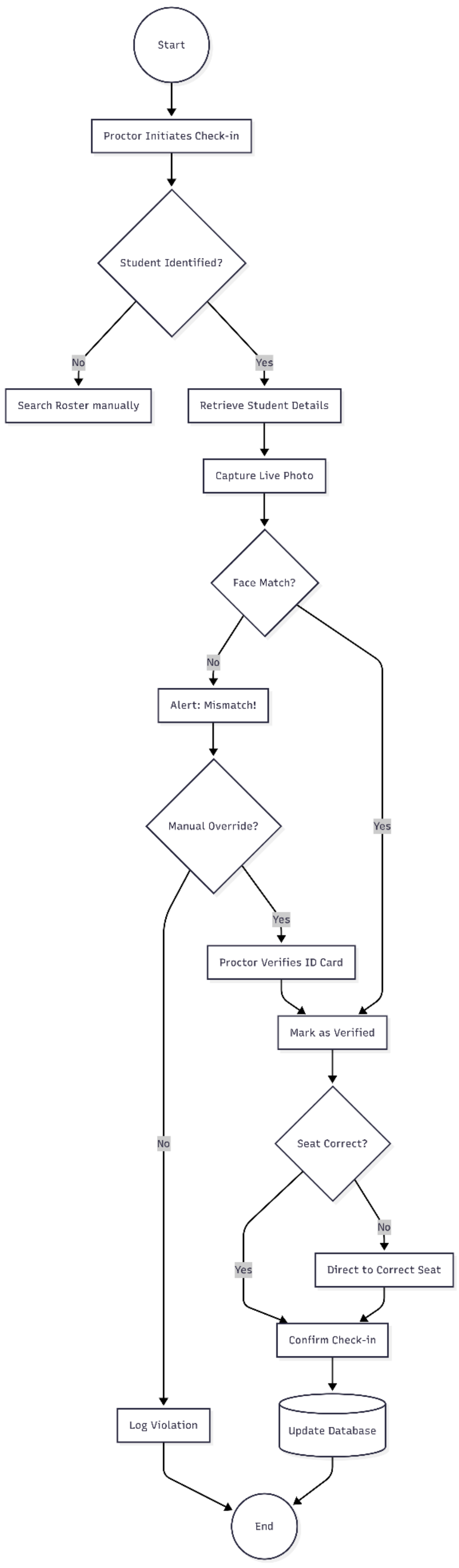
SQ-01: Multi-Photo Student Registration



SQ-02: AI-Powered Check-in Verification



C. Activity Diagram



D. Data Dictionary

Data Definition Commands

These commands create the tables, define primary keys, foreign keys, and necessary constraints by MS SQL.

1. User Table

```sql
CREATE TABLE User (
    UserID INT IDENTITY(1,1) PRIMARY KEY,
    Username NVARCHAR(50) NOT NULL UNIQUE,
    PasswordHash NVARCHAR(255) NOT NULL,
    Role NVARCHAR(20) NOT NULL CHECK (Role IN ('Admin', 'Proctor')),
    FullName NVARCHAR(100)
);
```

2. Student Table

```sql
CREATE TABLE Student (
    StudentID INT IDENTITY(1,1) PRIMARY KEY,
    UniversityID NVARCHAR(50) NOT NULL UNIQUE,
    FullName NVARCHAR(100) NOT NULL,
    Email NVARCHAR(100),
    ReferencePhotoUrl NVARCHAR(MAX), -- Path or URL to stored image
    FaceEmbedding NVARCHAR(MAX) -- Stored as JSON string if needed, or rely
on client-side matching
);
```

3. StudentPhoto Table

```sql
CREATE TABLE StudentPhoto (
    PhotoID INT IDENTITY(1,1) PRIMARY KEY,
    StudentID INT FOREIGN KEY REFERENCES Students(StudentID),
    PhotoUrl NVARCHAR(MAX) NOT NULL,
    CreatedAt DATETIME DEFAULT GETDATE()
);
```

4. Room Table

```
CREATE TABLE Room (
    RoomID INT IDENTITY(1,1) PRIMARY KEY,
    RoomName NVARCHAR(50) NOT NULL,
    Capacity INT,
    LayoutConfig NVARCHAR(MAX) -- JSON string describing rows/cols
);
```

5. Seat Table

```
CREATE TABLE Seats (
    SeatID INT IDENTITY(1,1) PRIMARY KEY,
    RoomID INT FOREIGN KEY REFERENCES Rooms(RoomID) ON DELETE CASCADE,
    RowLabel NVARCHAR(10),
    ColLabel NVARCHAR(10),
    IsAccessible BIT DEFAULT 1, -- 1=Normal, 0=Broken or Gap
    Status NVARCHAR(20) DEFAULT 'Active'
);
```

5. Exam Table

```
CREATE TABLE Exam (
    ExamID INT IDENTITY(1,1) PRIMARY KEY,
    CourseCode NVARCHAR(20) NOT NULL,
    ExamName NVARCHAR(100) NOT NULL,
    StartTime DATETIME NOT NULL,
    DurationMinutes INT NOT NULL,
    RoomID INT FOREIGN KEY REFERENCES Rooms(RoomID),
    CreatedBy INT FOREIGN KEY REFERENCES Users(UserID)
);
```

6. ExamRoster Table

```
CREATE TABLE ExamRoster (
    RosterID INT IDENTITY(1,1) PRIMARY KEY,
    ExamID INT FOREIGN KEY REFERENCES Exams(ExamID),
    StudentID INT FOREIGN KEY REFERENCES Students(StudentID),
    AssignedSeat NVARCHAR(20),
    Status NVARCHAR(20) DEFAULT 'Scheduled' CHECK (Status IN ('Scheduled',
'Present', 'Absent')),
    CheckInTime DATETIME,
    UNIQUE(ExamID, StudentID)
);
```

7. CheckInLog Table

```
CREATE TABLE CheckInLog (
    LogID INT IDENTITY(1,1) PRIMARY KEY,
    RosterID INT FOREIGN KEY REFERENCES ExamRoster(RosterID),
    Timestamp DATETIME DEFAULT GETDATE(),
    MLConfidenceScore FLOAT,
    IsMatch BIT,
    IsSeatCorrect BIT,
    CapturedImagePath NVARCHAR(MAX)
);
```

8. Violation Table

```
CREATE TABLE Violation (
    ViolationID INT IDENTITY(1,1) PRIMARY KEY,
    ExamID INT FOREIGN KEY REFERENCES Exams(ExamID),
    StudentID INT FOREIGN KEY REFERENCES Students(StudentID),
    Reason NVARCHAR(100),
    Notes NVARCHAR(MAX),
    EvidenceImagePath NVARCHAR(MAX),
    ReportedAt DATETIME DEFAULT GETDATE(),
    ReportedBy INT FOREIGN KEY REFERENCES Users(UserID)
);
```

Data Manipulation Commands (Sample Data)

These commands populate the tables with meaningful sample data.

1. Seed Users

```
INSERT INTO Users (Username, PasswordHash, Role, FullName) VALUES ('admin',
'$2b$10$YourHashedPasswordHere', 'Admin', 'System Administrator');

INSERT INTO Users (Username, PasswordHash, Role, FullName) VALUES
('proctor1', '$2b$10$YourHashedPasswordHere', 'Proctor', 'Jane Doe
Proctor');
```

2. Seed Rooms

```
DECLARE @Rooms TABLE (Name NVARCHAR(50), Cap INT, Cfg NVARCHAR(MAX));
INSERT INTO @Rooms VALUES
('Hall A', 30, '{"rows":5,"cols":6}'),
('Hall B', 64, '{"rows":8,"cols":8}'),
('Auditorium', 150, '{"rows":10,"cols":15}'),
('Lab 1', 20, '{"rows":4,"cols":5}'),
('Lab 2', 20, '{"rows":4,"cols":5}'),
('Lecture Hall 101', 60, '{"rows":6,"cols":10}'),
('Seminar Room A', 24, '{"rows":3,"cols":8}');

MERGE INTO Rooms AS Target
USING @Rooms AS Source
ON Target.RoomName = Source.Name
WHEN NOT MATCHED BY TARGET THEN
    INSERT (RoomName, Capacity, LayoutConfig) VALUES (Source.Name,
Source.Cap, Source.Cfg);
```

3. Seed Students

```
DECLARE @ManualStudents TABLE (
    UniversityID NVARCHAR(50),
    FullName NVARCHAR(100),
    Email NVARCHAR(100),
    ReferencePhotoUrl NVARCHAR(MAX)
);

INSERT INTO @ManualStudents (UniversityID, FullName, Email,
ReferencePhotoUrl) VALUES
('U2024001', 'James Smith', 'u2024001@university.edu', 'https://ui-
avatars.com/api/?name=James+Smith&background=0D8ABC&color=fff'),
('U2024002', 'Mary Johnson', 'u2024002@university.edu', 'https://ui-
avatars.com/api/?name=Mary+Johnson&background=1D9ABC&color=fff'),
('U2024003', 'John Williams', 'u2024003@university.edu', 'https://ui-
avatars.com/api/?name=John+Williams&background=2DABC0&color=fff'),
('U2024004', 'Patricia Brown', 'u2024004@university.edu', 'https://ui-
avatars.com/api/?name=Patricia+Brown&background=3DBCC4&color=fff'),
('U2024005', 'Robert Jones', 'u2024005@university.edu', 'https://ui-
avatars.com/api/?name=Robert+Jones&background=4DCDD8&color=fff'),
('U2024006', 'Jennifer Garcia', 'u2024006@university.edu', 'https://ui-
avatars.com/api/?name=Jennifer+Garcia&background=5DDDE0&color=fff'),
('U2024007', 'Michael Miller', 'u2024007@university.edu', 'https://ui-
avatars.com/api/?name=Michael+Miller&background=6DEEF4&color=fff'),
('U2024008', 'Linda Davis', 'u2024008@university.edu', 'https://ui-
avatars.com/api/?name=Linda+Davis&background=7DFF08&color=fff'),
('U2024009', 'William Rodriguez', 'u2024009@university.edu', 'https://ui-
avatars.com/api/?name=William+Rodriguez&background=8E001C&color=fff'),
('U2024010', 'Elizabeth Martinez', 'u2024010@university.edu', 'https://ui-avatars.com/api/?name=Elizabeth+Martinez&background=9F1120&color=fff'),
('U2024011', 'David Hernandez', 'u2024011@university.edu', 'https://ui-avatars.com/api/?name=David+Hernandez&background=A02234&color=fff'),
('U2024012', 'Barbara Lopez', 'u2024012@university.edu', 'https://ui-avatars.com/api/?name=Barbara+Lopez&background=B13348&color=fff'),
('U2024013', 'Richard Gonzalez', 'u2024013@university.edu', 'https://ui-avatars.com/api/?name=Richard+Gonzalez&background=C2445C&color=fff'),
('U2024014', 'Susan Wilson', 'u2024014@university.edu', 'https://ui-avatars.com/api/?name=Susan+Wilson&background=D35570&color=fff'),
('U2024015', 'Joseph Anderson', 'u2024015@university.edu', 'https://ui-avatars.com/api/?name=Joseph+Anderson&background=E46684&color=fff'),
('U2024016', 'Jessica Thomas', 'u2024016@university.edu', 'https://ui-avatars.com/api/?name=Jessica+Thomas&background=F57798&color=fff'),
('U2024017', 'Thomas Taylor', 'u2024017@university.edu', 'https://ui-avatars.com/api/?name=Thomas+Taylor&background=0688AC&color=fff'),
('U2024018', 'Sarah Moore', 'u2024018@university.edu', 'https://ui-avatars.com/api/?name=Sarah+Moore&background=1799C0&color=fff'),
('U2024019', 'Charles Jackson', 'u2024019@university.edu', 'https://ui-avatars.com/api/?name=Charles+Jackson&background=28AAD4&color=fff'),
('U2024020', 'Karen Martin', 'u2024020@university.edu', 'https://ui-avatars.com/api/?name=Karen+Martin&background=39BBE8&color=fff'),
('U2024021', 'Christopher Lee', 'u2024021@university.edu', 'https://ui-avatars.com/api/?name=Christopher+Lee&background=4ACCF0&color=fff'),
('U2024022', 'Lisa Perez', 'u2024022@university.edu', 'https://ui-avatars.com/api/?name=Lisa+Perez&background=5BDD04&color=fff'),
('U2024023', 'Daniel Thompson', 'u2024023@university.edu', 'https://ui-avatars.com/api/?name=Daniel+Thompson&background=6CEE18&color=fff'),
('U2024024', 'Nancy White', 'u2024024@university.edu', 'https://ui-avatars.com/api/?name=Nancy+White&background=7DFF2C&color=fff'),
('U2024025', 'Paul Harris', 'u2024025@university.edu', 'https://ui-avatars.com/api/?name=Paul+Harris&background=8E0040&color=fff'),
('U2024026', 'Betty Sanchez', 'u2024026@university.edu', 'https://ui-avatars.com/api/?name=Betty+Sanchez&background=9F1154&color=fff'),
('U2024027', 'Mark Clark', 'u2024027@university.edu', 'https://ui-avatars.com/api/?name=Mark+Clark&background=A02268&color=fff'),
('U2024028', 'Margaret Ramirez', 'u2024028@university.edu', 'https://ui-avatars.com/api/?name=Margaret+Ramirez&background=B1337C&color=fff'),
('U2024029', 'Donald Lewis', 'u2024029@university.edu', 'https://ui-avatars.com/api/?name=Donald+Lewis&background=C24490&color=fff'),
('U2024030', 'Sandra Robinson', 'u2024030@university.edu', 'https://ui-avatars.com/api/?name=Sandra+Robinson&background=D355A4&color=fff'),
('U2024031', 'George Walker', 'u2024031@university.edu', 'https://ui-avatars.com/api/?name=George+Walker&background=E466B8&color=fff'),
('U2024032', 'Ashley Young', 'u2024032@university.edu', 'https://ui-avatars.com/api/?name=Ashley+Young&background=F577CC&color=fff'),
('U2024033', 'Kenneth Allen', 'u2024033@university.edu', 'https://ui-avatars.com/api/?name=Kenneth+Allen&background=0688E0&color=fff'),
('U2024034', 'Kimberly King', 'u2024034@university.edu', 'https://ui-avatars.com/api/?name=Kimberly+King&background=1799F4&color=fff'),
('U2024035', 'Steven Wright', 'u2024035@university.edu', 'https://ui-avatars.com/api/?name=Steven+Wright&background=28AA08&color=fff'),
('U2024036', 'Donna Scott', 'u2024036@university.edu', 'https://ui-avatars.com/api/?name=Donna+Scott&background=39BB1C&color=fff'),
('U2024037', 'Edward Torres', 'u2024037@university.edu', 'https://ui-avatars.com/api/?name=Edward+Torres&background=4ACC30&color=fff'),
('U2024038', 'Michelle Nguyen', 'u2024038@university.edu', 'https://ui-avatars.com/api/?name=Michelle+Nguyen&background=5BDD44&color=fff'),
('U2024039', 'Brian Hill', 'u2024039@university.edu', 'https://ui-avatars.com/api/?name=Brian+Hill&background=6CEE58&color=fff'),
('U2024040', 'Emily Flores', 'u2024040@university.edu', 'https://ui-avatars.com/api/?name=Emily+Flores&background=7DFF6C&color=fff'),
('U2024041', 'Ronald Green', 'u2024041@university.edu', 'https://ui-avatars.com/api/?name=Ronald+Green&background=8E0080&color=fff'),
('U2024042', 'Dorothy Adams', 'u2024042@university.edu', 'https://ui-avatars.com/api/?name=Dorothy+Adams&background=9F1194&color=fff'),
('U2024043', 'Anthony Nelson', 'u2024043@university.edu', 'https://ui-avatars.com/api/?name=Anthony+Nelson&background=A022A8&color=fff'),
('U2024044', 'Melissa Baker', 'u2024044@university.edu', 'https://ui-avatars.com/api/?name=Melissa+Baker&background=B133BC&color=fff'),
('U2024045', 'Kevin Hall', 'u2024045@university.edu', 'https://ui-avatars.com/api/?name=Kevin+Hall&background=C244D0&color=fff'),
('U2024046', 'Deborah Rivera', 'u2024046@university.edu', 'https://ui-avatars.com/api/?name=Deborah+Rivera&background=D355E4&color=fff'),
('U2024047', 'Jason Campbell', 'u2024047@university.edu', 'https://ui-avatars.com/api/?name=Jason+Campbell&background=E466F8&color=fff'),
('U2024048', 'Stephanie Mitchell', 'u2024048@university.edu', 'https://ui-avatars.com/api/?name=Stephanie+Mitchell&background=F5770C&color=fff'),
('U2024049', 'Matthew Carter', 'u2024049@university.edu', 'https://ui-avatars.com/api/?name=Matthew+Carter&background=068820&color=fff'),
('U2024050', 'Rebecca Roberts', 'u2024050@university.edu', 'https://ui-avatars.com/api/?name=Rebecca+Roberts&background=179934&color=fff');

-- Merge into Students (Idempotent Insert)
MERGE INTO Students AS Target
USING @ManualStudents AS Source
ON Target.UniversityID = Source.UniversityID
WHEN NOT MATCHED BY TARGET THEN
    INSERT (UniversityID, FullName, Email, ReferencePhotoUrl)
    VALUES (Source.UniversityID, Source.FullName, Source.Email, Source.ReferencePhotoUrl);

-- Populate Photos Table (Idempotent)
INSERT INTO StudentPhotos (StudentID, PhotoUrl)
SELECT s.StudentID, s.ReferencePhotoUrl
FROM Students s
WHERE s.ReferencePhotoUrl IS NOT NULL
```

```
AND NOT EXISTS (SELECT 1 FROM StudentPhotos WHERE StudentID = s.StudentID);
```

4. Seed Exams

```
DECLARE @AdminID INT = (SELECT TOP 1 UserID FROM Users WHERE Role =
'Admin');
-- Rooms
DECLARE @HallA INT = (SELECT RoomID FROM Rooms WHERE RoomName = 'Hall A');
DECLARE @HallB INT = (SELECT RoomID FROM Rooms WHERE RoomName = 'Hall B');
DECLARE @Audit INT = (SELECT RoomID FROM Rooms WHERE RoomName =
'Auditorium');

IF NOT EXISTS (SELECT 1 FROM Exams WHERE CourseCode = 'CS101' AND ExamName =
'Midterm Exam')
   INSERT INTO Exams (CourseCode, ExamName, StartTime, DurationMinutes,
RoomID, CreatedBy) VALUES
   ('CS101', 'Midterm Exam', DATEADD(DAY, 2, GETDATE()), 90, @HallA,
@AdminID);

IF NOT EXISTS (SELECT 1 FROM Exams WHERE CourseCode = 'MATH202' AND ExamName
= 'Calculus Final')
   INSERT INTO Exams (CourseCode, ExamName, StartTime, DurationMinutes,
RoomID, CreatedBy) VALUES
   ('MATH202', 'Calculus Final', DATEADD(DAY, 4, GETDATE()), 120, @HallB,
@AdminID);

IF NOT EXISTS (SELECT 1 FROM Exams WHERE CourseCode = 'ENG101' AND ExamName
= 'Literature Quiz')
   INSERT INTO Exams (CourseCode, ExamName, StartTime, DurationMinutes,
RoomID, CreatedBy) VALUES
   ('ENG101', 'Literature Quiz', GETDATE(), 60, @Audit, @AdminID);
```

5. Seed Roster

```
-- CS101 (Hall A, Cap 30) - Add 25 students
DECLARE @Exam1 INT = (SELECT TOP 1 ExamID FROM Exams WHERE CourseCode =
'CS101' ORDER BY ExamID DESC);

INSERT INTO ExamRoster (ExamID, StudentID, AssignedSeat, Status)
SELECT TOP 25
    @Exam1, StudentID,
    CONCAT(CHAR(65 + (ROW_NUMBER() OVER(ORDER BY StudentID)-1)/6),
(ROW_NUMBER() OVER(ORDER BY StudentID)-1)%6 + 1),
    'Scheduled'
FROM Students
```

```
WHERE StudentID BETWEEN (SELECT MIN(StudentID) FROM Students) AND (SELECT
MIN(StudentID) + 24 FROM Students)
AND NOT EXISTS (SELECT 1 FROM ExamRoster WHERE ExamID = @Exam1 AND StudentID
= Students.StudentID);

-- MATH202 (Hall B, Cap 64) - Add next 25 students
DECLARE @Exam2 INT = (SELECT TOP 1 ExamID FROM Exams WHERE CourseCode =
'MATH202' ORDER BY ExamID DESC);

INSERT INTO ExamRoster (ExamID, StudentID, AssignedSeat, Status)
SELECT TOP 25
    @Exam2, StudentID,
    CONCAT(CHAR(65 + (ROW_NUMBER() OVER(ORDER BY StudentID)-1)/8),
(ROW_NUMBER() OVER(ORDER BY StudentID)-1)%8 + 1),
    'Scheduled'
FROM Students
WHERE StudentID >= (SELECT MIN(StudentID) + 25 FROM Students)
AND NOT EXISTS (SELECT 1 FROM ExamRoster WHERE ExamID = @Exam2 AND StudentID
= Students.StudentID);
```

E. Links

Jira :

https://oswinberke9.atlassian.net/jira/software/projects/ESS/summary

Bitbucket :

https://bitbucket.org/berkecanakyuz/examsecuritysystem/src/main/