CLASSROOM AUTOMATIC ATTENDANCE SYSTEM PROJECT REPORT

Prepared by:
Berkay Aksoy
Student Number: 220706307

Hakkı Dökmeci
Student Number: 220706026

Course: SE 342 01 Software Validation and Testing
Instructor: Emre Olca

PROJECT CONTENTS

Classroom Automatic Attendance System

Requirements Analysis Document – Main Sections (EPICs)

EPIC 11- Deployment & Maintenance

1. System Requirements & Analysis

Story: Functional Requirements Definition

-Subtask: Prepare a *Functional Requirement Specification (FRS)* document describing user actions, system features, and data interactions.

Story: Non-Functional and System Constraints Analysis

-Subtask: Document *Non-Functional Requirements (NFR)* including performance, security, reliability, and scalability constraints.

2. Database Analysis & Setup

Story: Define Conceptual Database Design

-Subtask: Identify the main entities, attributes, and relationships required for the system before creating the ERD.

Story: Create Initial ERD Diagram

-Subtask: Create a structured list of entities, attributes, and relationship notes to be used as the ERD blueprint.

3. Initial System Foundations

Story: Database Development Setup

-Subtask: Create essential tables, relationships, and constraints according to the planned database model.

Story: Create Initial Application Screen Designs

-Subtask: Create basic wireframes and layouts for main screens such as login, dashboard, and attendance pages.

4. Course & Classroom Management

Story: Course Management Module
-Subtask: Develop CRUD (Create, Read, Update, Delete) operations for course data management..

Story: Classroom Assignment System

-Subtask: Build the classroom assignment feature; the system should automatically update timetables when changes occur.

5. Camera & Classroom Setup

Story: Camera Installation and Linking

-Subtask: Link each camera to its respective classroom and validate network accessibility.

Story: Camera Monitoring Dashboard

-Subtask: Build a real-time dashboard for monitoring camera status and live feed availability.

6. Attendance Session & Recognition Flow

Story: Automatic Attendance Session

-Subtask: Implement triggers to start and end attendance sessions based on class schedules.

Story: Face Recognition Attendance Flow

-Subtask: Integrate the face recognition module to automatically identify students and mark attendance.

7. Teacher Approval & Manual Adjustment

Story: Attendance Review Module

-Subtask: Create a user interface for teachers to review and confirm recognized attendance records.

Story: Manual Attendance Adjustment

-Subtask: Provide functionality for teachers to manually correct attendance errors or missing entries

8. Reporting & Semester Archive

Story: Attendance Reporting Module

-Subtask: Generate attendance reports by student, course, and semester.

Story: Semester Archive System

-Subtask: Develop a system to archive and export attendance data at the end of each term.

9. Privacy, Access, and Data Retention Policies

Story: Data Privacy Control

-Subtask: Define and implement role-based access control for teachers, administrators, and system operators.

Story: Data Retention & Deletion Policy

-Subtask: Add an automatic cleanup and secure data deletion mechanism after the retention period.

10. Testing & Evaluation

Story: System Integration Testing

-Subtask: Conduct integration tests between modules and document test results.

Story: Performance & Accuracy Evaluation

-Subtask: Measure face recognition accuracy, system response time, and performance metrics; compile results into a report.

11. Deployment & Maintenance

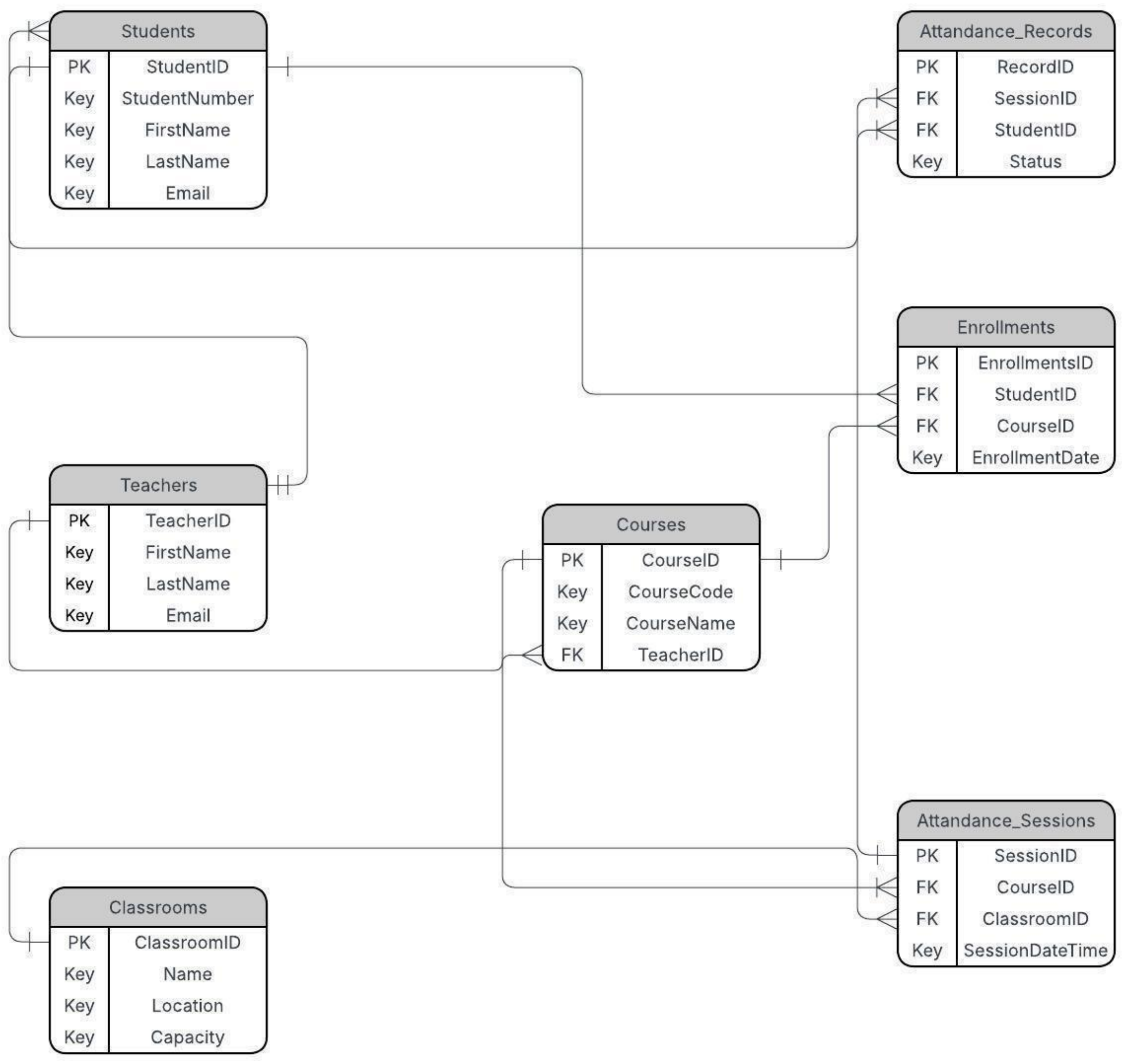
Story: System Deployment & Environment Setup

-Subtask: Deploy the system, configure environment variables, and verify installation integrity.

Story: System Maintenance & Update

-Subtask: Schedule regular maintenance, monitor system performance, and apply software updates.

2. Data Base Design



2.1 ERD Description

1. Students

- Stores student information such as ID, name, and email.
- Connected to **Enrollments** (courses they take).
- Linked to **Attendance_Records** (their attendance per session).

2. Teachers

- Holds teacher identity and contact information.
- Each teacher is responsible for multiple **Courses**.

3. Courses

- Defines course details, including code and name.
- Each course is taught by one **Teacher**.
- Linked to:
    - **Enrollments** (students registered)
    - **Attendance_Sessions** (scheduled attendance events)

4. Classrooms

- Contains physical classroom details such as name, location, and capacity.
- Each **Attendance_Session** occurs in one classroom.

5. Enrollments

- Resolves the many-to-many relationship between **Students** and **Courses**.
- Indicates which student is registered in which course.

6. Attendance_Sessions

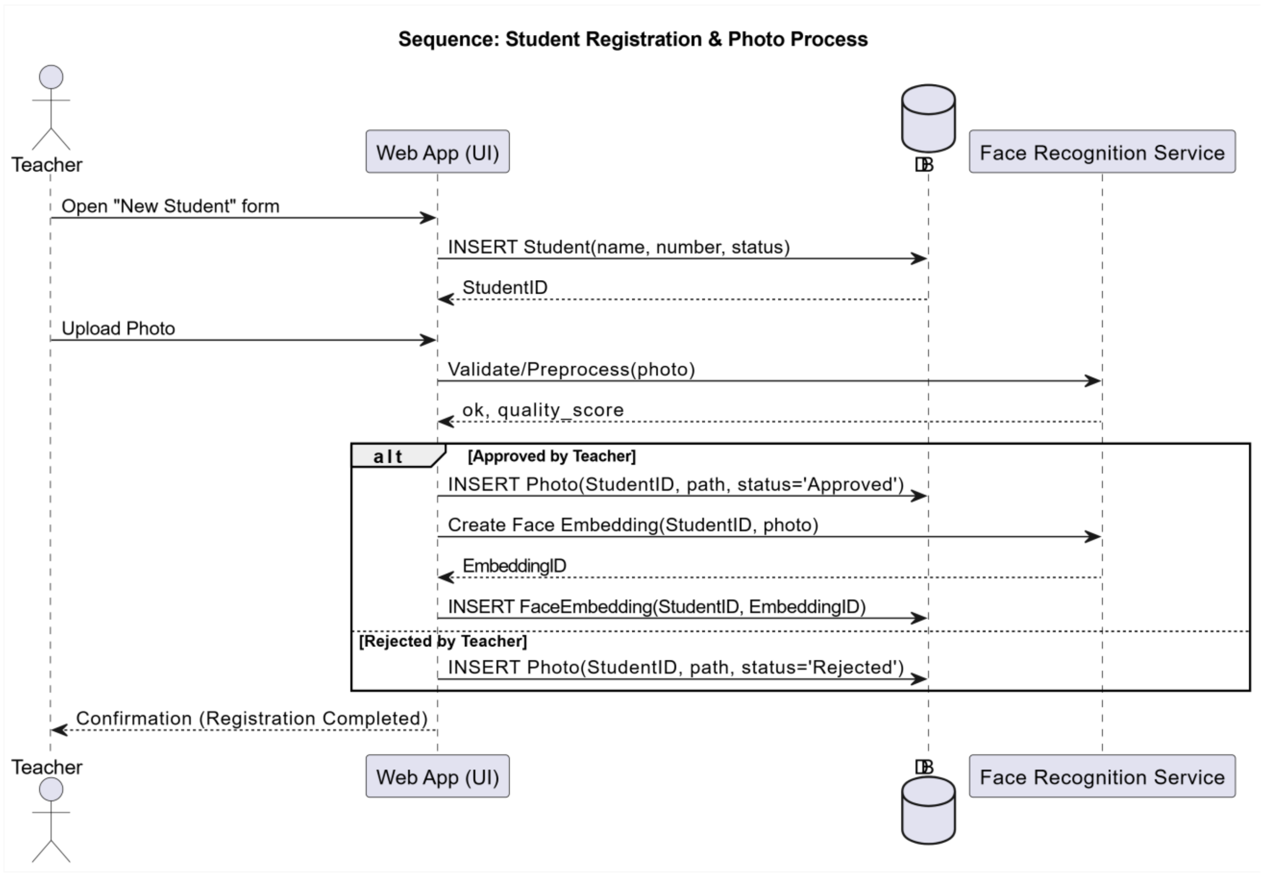
- Represents scheduled attendance sessions for a course.
- Linked to:
  - One **Course**
  - One **Classroom**
- Each session produces multiple **Attendance_Records**.

7. Attendance_Records

- Stores the attendance status (present/absent) for each student in each session.
- Links:
  - One **Student**
  - One **Attendance_Session**

3. Sequence Diagram

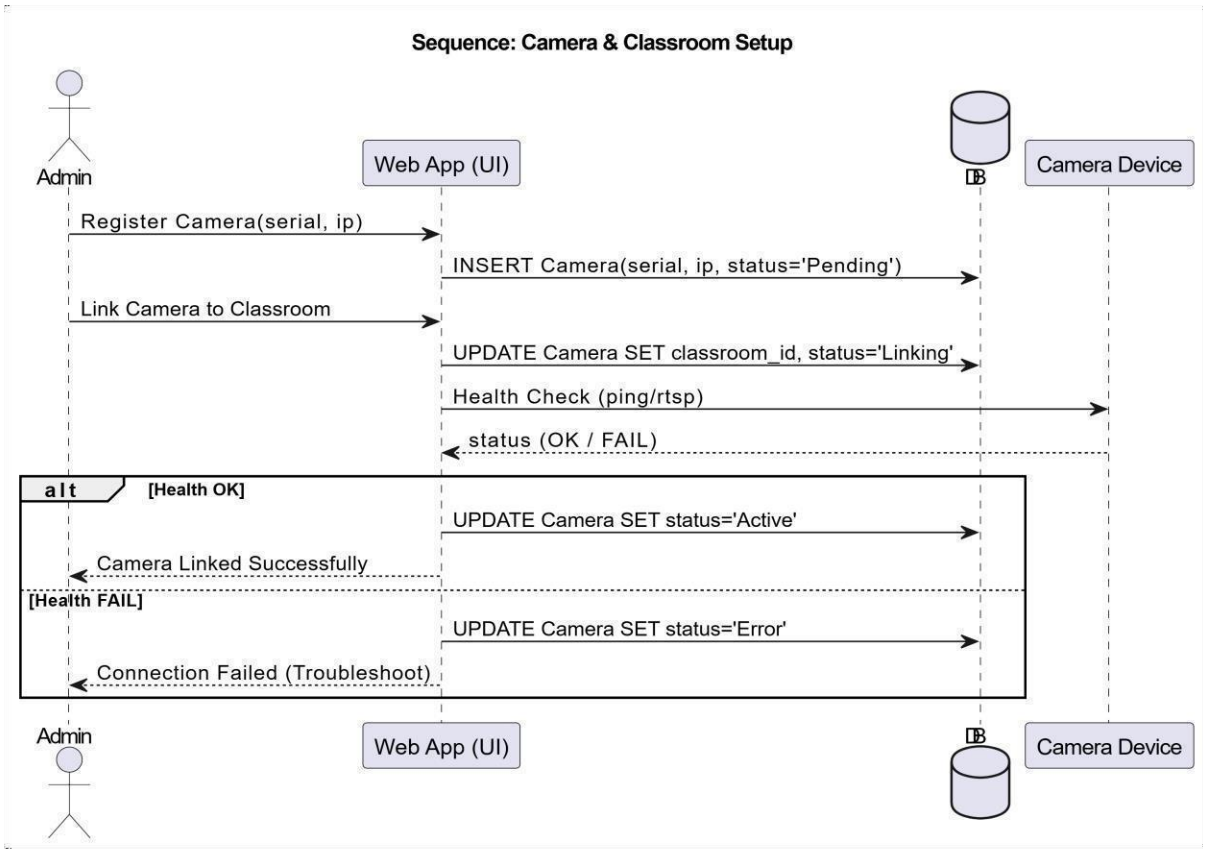
3.1 Sequence Diagram (Student Registration & Photo Process)


Sequence: Student Registration & Photo Process
Teacher
Web App (UI)
DB
Face Recognition Service
Open "New Student" form
INSERT Student(name, number, status)
StudentID
Upload Photo
Validate/Preprocess(photo)
ok, quality_score
alt
[Approved by Teacher]
INSERT Photo(StudentID, path, status='Approved')
Create Face Embedding(StudentID, photo)
EmbeddingID
INSERT FaceEmbedding(StudentID, EmbeddingID)
[Rejected by Teacher]
INSERT Photo(StudentID, path, status='Rejected')
Confirmation (Registration Completed)

Description:

- Thisworkflow describeshow a *teacher* addsa new student, uploadsa photo, and interacts with the face recognition service.
- It focuseson the **registration** stage before attendance begins.
- ***Goal:*** Prepare accuratestudent datafor later recognition.

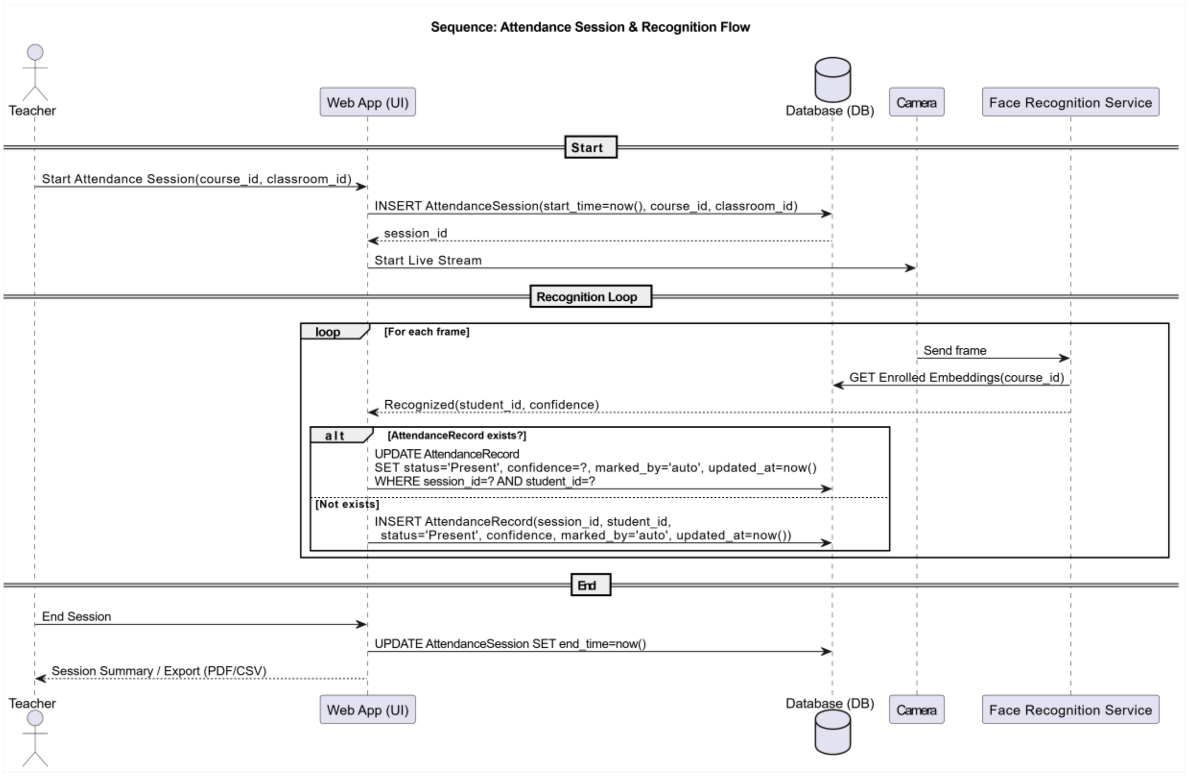
3.2 Sequence Diagram (Camera & Classroom Setup)


Sequence: Camera & Classroom Setup
Admin
Web App (UI)
DB
Camera Device
Register Camera(serial, ip)
INSERT Camera(serial, ip, status='Pending')
Link Camera to Classroom
UPDATE Camera SET classroom_id, status='Linking'
Health Check (ping/rtsp)
status (OK / FAIL)
alt
[Health OK]
UPDATE Camera SET status='Active'
Camera Linked Successfully
[Health FAIL]
UPDATE Camera SET status='Error'
Connection Failed (Troubleshoot)

Description:

- Thisworkflow shows how the *admin* installs and links cameras to classrooms.
- It's a completely different process – involveshardware checks and configuration, not student data.
  - ***Goal:*** Make sure the physical camera system is connected and ready to record sessions.

3.3 Sequence Diagram (Attandence Session & Recognition Flow)

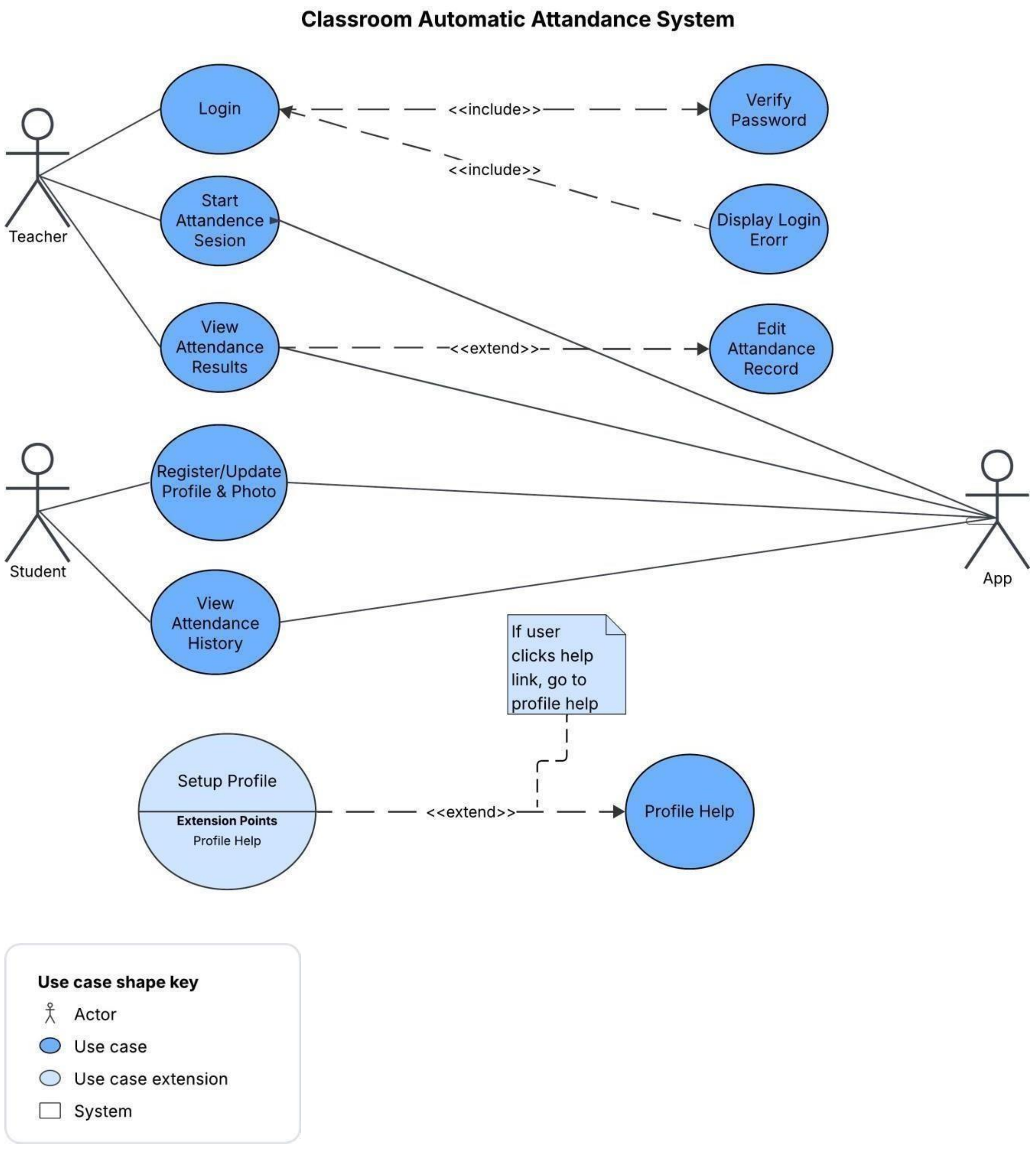

Sequence: Attendance Session & Recognition Flow
Teacher
Web App (UI)
Database (DB)
Camera
Face Recognition Service
Start
Start Attendance Session(course_id, classroom_id)
INSERT AttendanceSession(start_time=now(), course_id, classroom_id)
session_id
Start Live Stream
Recognition Loop
loop
[For each frame]
Send frame
GET Enrolled Embeddings(course_id)
Recognized(student_id, confidence)
alt
[AttendanceRecord exists?]
UPDATE AttendanceRecord
SET status='Present', confidence=?, marked_by='auto', updated_at=now()
WHERE session_id=? AND student_id=?
[Not exists]
INSERT AttendanceRecord(session_id, student_id,
status='Present', confidence, marked_by='auto', updated_at=now())
End
End Session
UPDATE AttendanceSession SET end_time=now()
Session Summary / Export (PDF/CSV)

Description:

- Thisis themain **operational phase** —where the teacher startsa session, the camera streams video, and the recognition service marks attendance.
- It dependson theprevious two processesbut runsas a separatedaily routine.

Goal: Automaticallydetect studentsand record attendance.

4. Use Case Diagram



4.1 Use-Case Diagram Description

This use-case diagram shows the main interactions in the *Classroom Automatic Attendance System*. **Teacher** and **Student** perform core actions such as logging in, managing profiles, starting attendance sessions, and viewing results or history. The **App** supports the system by handling automated tasks and optional extensions like login validation, error display, editing attendance records, and profile help. The diagram highlights mandatory steps using **include** and optional behaviors using **extend** relationships.

<<include>> — Role in the Diagram

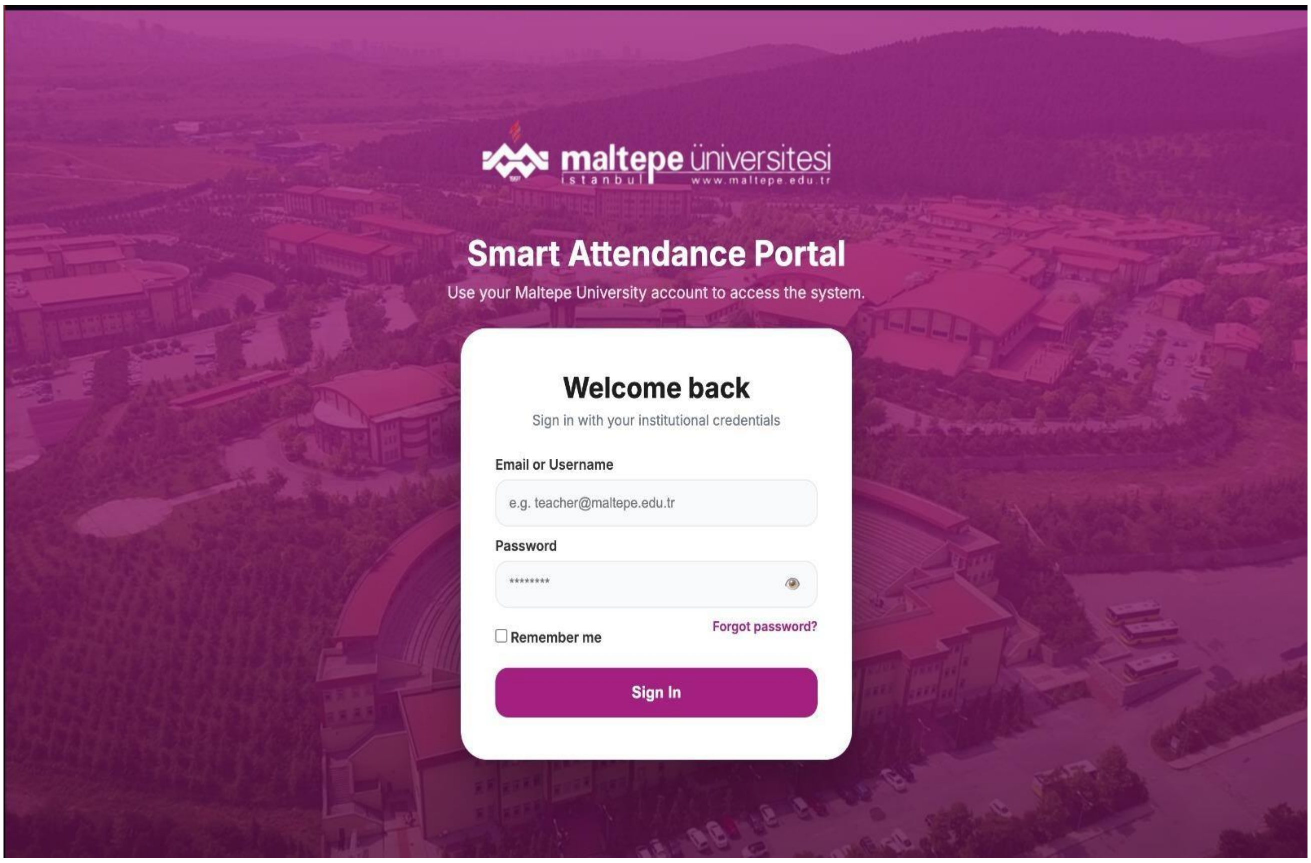
In the diagram, **<<include>>** is used to show **required sub-actions** that must occur as part of the main use case.
For example, the **Login** use case always includes **Verify Password**, and may include **Display Login Error** when needed.

<<extend>> — Role in the Diagram

In the diagram, **<<extend>>** is used to show **optional or additional actions** that can be triggered depending on specific conditions within the main use case.
For example, **View Attendance Results** can be extended with **Edit Attendance Record**, and **Setup Profile** can be extended with **Profile Help** if needed.

5. Page Designs

5.1 Login Page



Smart Attendance Portal

Use your Maltepe University account to access the system.

Welcome back

Sign in with your institutional credentials

Email or Username

e.g. teacher@maltepe.edu.tr

Password

****

Remember me

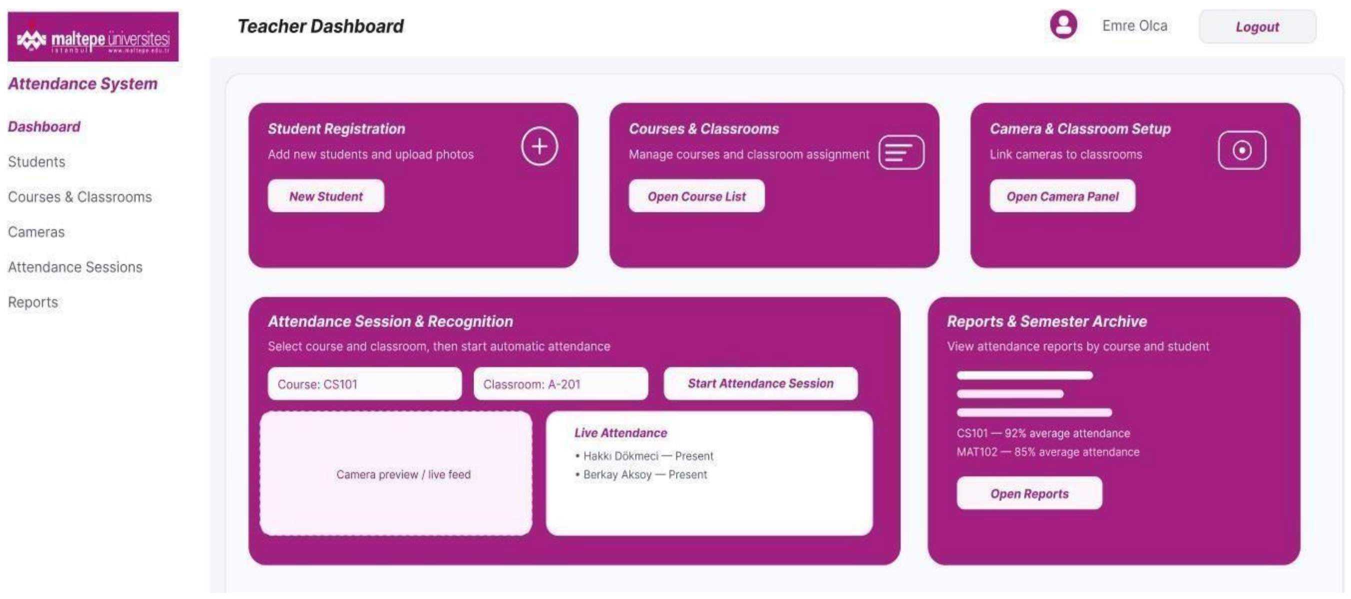
Forgot password?

Sign In

Description:
The Login Page allows teachers to securely sign in with their Maltepe University

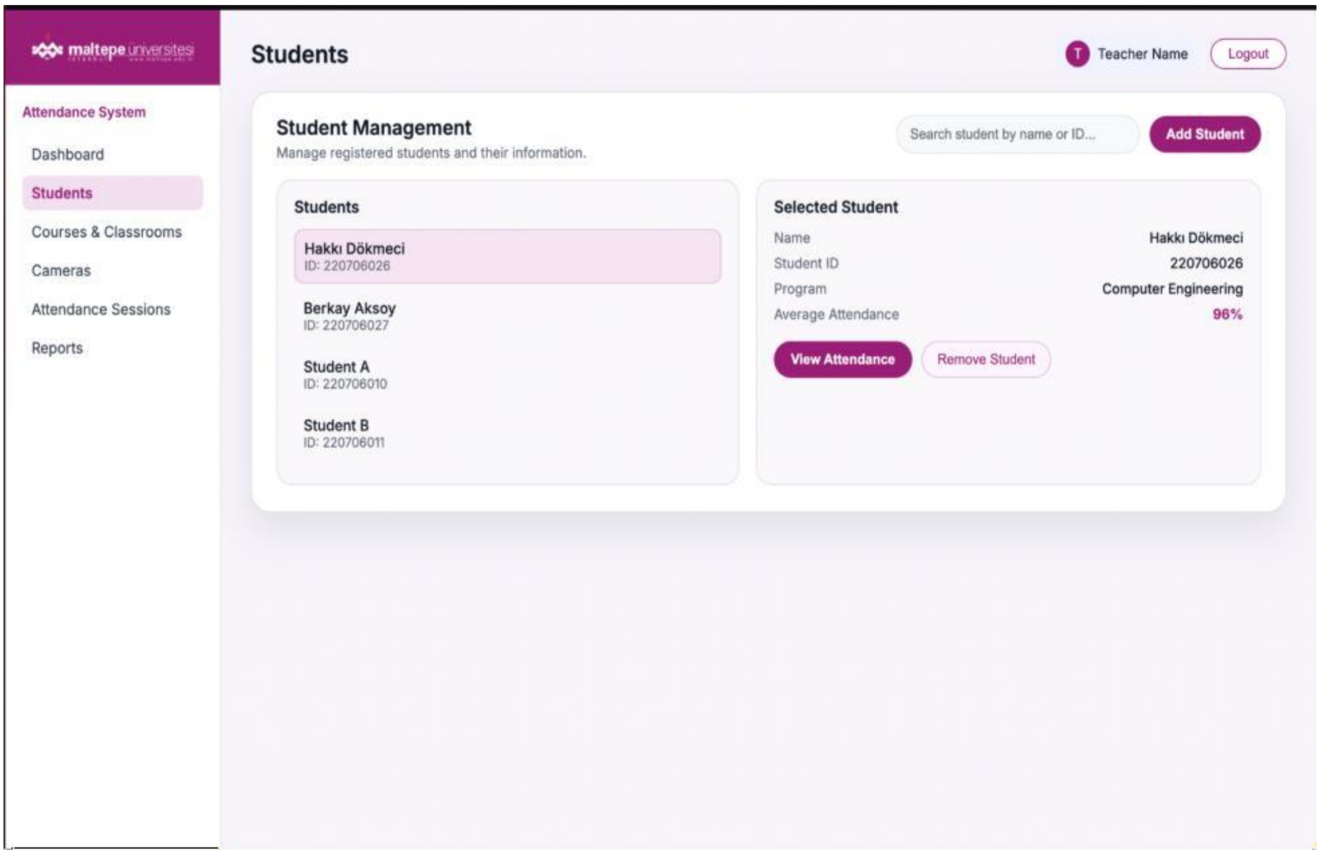
credentials. It provides a clean interface with username and password fields, a "Remember me" option, and a modern layout styled with the university's colors.

5.2 Teacher Dashboard Page


maltepe üniversitesi
Attendance System
Dashboard
Students
Courses & Classrooms
Cameras
Attendance Sessions
Reports
Teacher Dashboard
Emre Olca
Logout
Student Registration
Add new students and upload photos
New Student
Courses & Classrooms
Manage courses and classroom assignment
Open Course List
Camera & Classroom Setup
Link cameras to classrooms
Open Camera Panel
Attendance Session & Recognition
Select course and classroom, then start automatic attendance
Course: CS101
Classroom: A-201
Start Attendance Session
Camera preview / live feed
Live Attendance
• Hakki Dökmeci — Present
• Berkay Aksoy — Present
Reports & Semester Archive
View attendance reports by course and student
CS101 — 92% average attendance
MAT102 — 85% average attendance
Open Reports

Description:
The Teacher Dashboard serves as the central control panel for instructors, providing quick access to all core features of the Smart Attendance System. From this interface, teachers can navigate to student management, course and classroom settings, camera setup, attendance sessions, and reporting tools. The dashboard offers an organized overview that helps instructors efficiently manage daily academic tasks and monitor classroom activities.

5.3 Students Page



Student Management

Manage registered students and their information.

Students

Hakkı Dökmeci
ID: 220706026

Berkay Aksoy
ID: 220706027

Student A
ID: 220706010

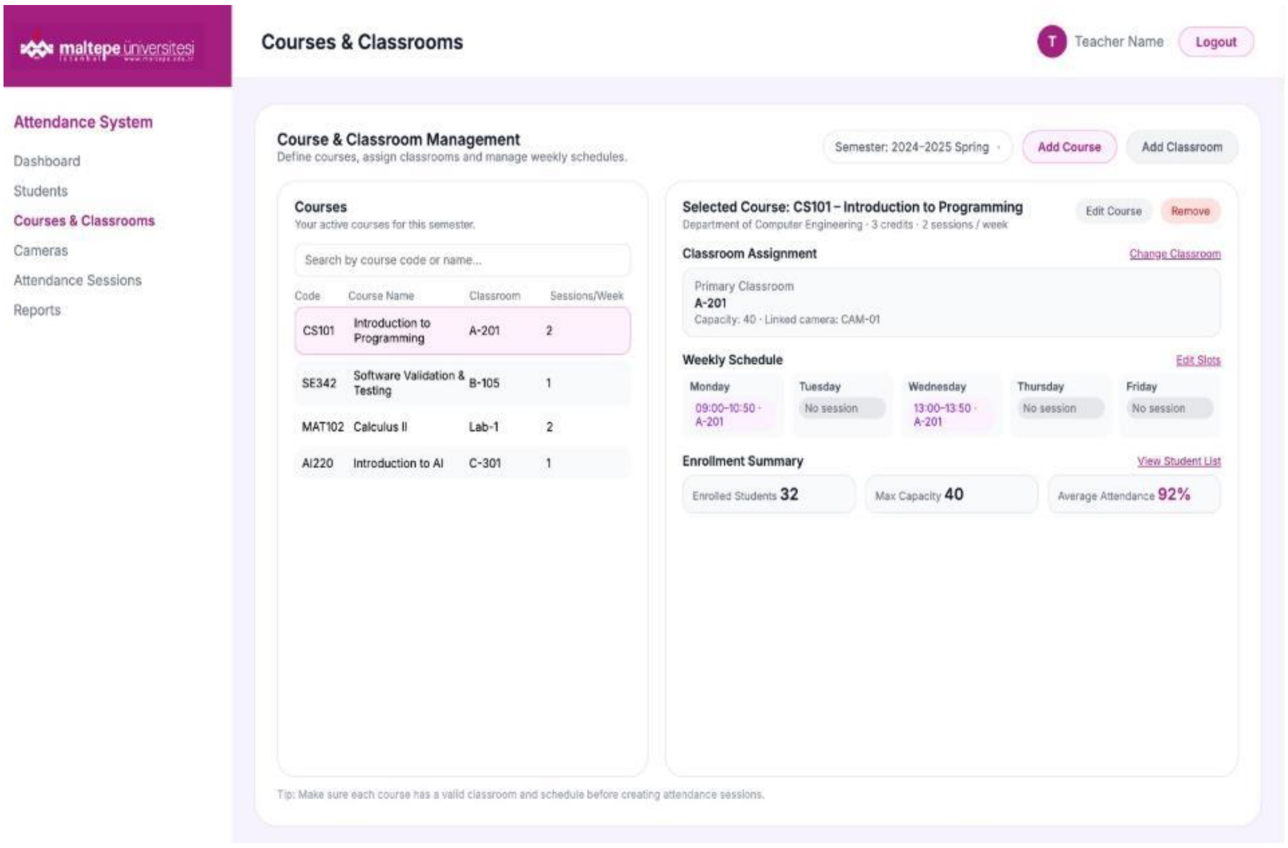
Student B
ID: 220706011

Selected Student

Name	Hakkı Dökmeci
Student ID	220706026
Program	Computer Engineering
Average Attendance	96%

Description:
The Students Page allows instructors to view and manage all registered students in the system. It provides a searchable list of students and displays detailed information—such as program, student ID, and attendance performance—for the selected student. This interface helps teachers quickly access student data and monitor academic participation.

5.4 Courses & Classrooms Page



Courses & Classrooms

Course & Classroom Management

Define courses, assign classrooms and manage weekly schedules.

Courses

Your active courses for this semester.

Search by course code or name...

Code	Course Name	Classroom	Sessions/Week
CS101	Introduction to Programming	A-201	2
SE342	Software Validation & Testing	B-105	1
MAT102	Calculus II	Lab-1	2
AI220	Introduction to AI	C-301	1

Selected Course: CS101 – Introduction to Programming

Department of Computer Engineering · 3 credits · 2 sessions / week

Classroom Assignment

Primary Classroom

A-201

Capacity: 40 · Linked camera: CAM-01

Weekly Schedule

Monday	Tuesday	Wednesday	Thursday	Friday
09:00–10:50 · A-201	No session	13:00–13:50 · A-201	No session	No session

Enrollment Summary

Enrolled Students 32

Max Capacity 40

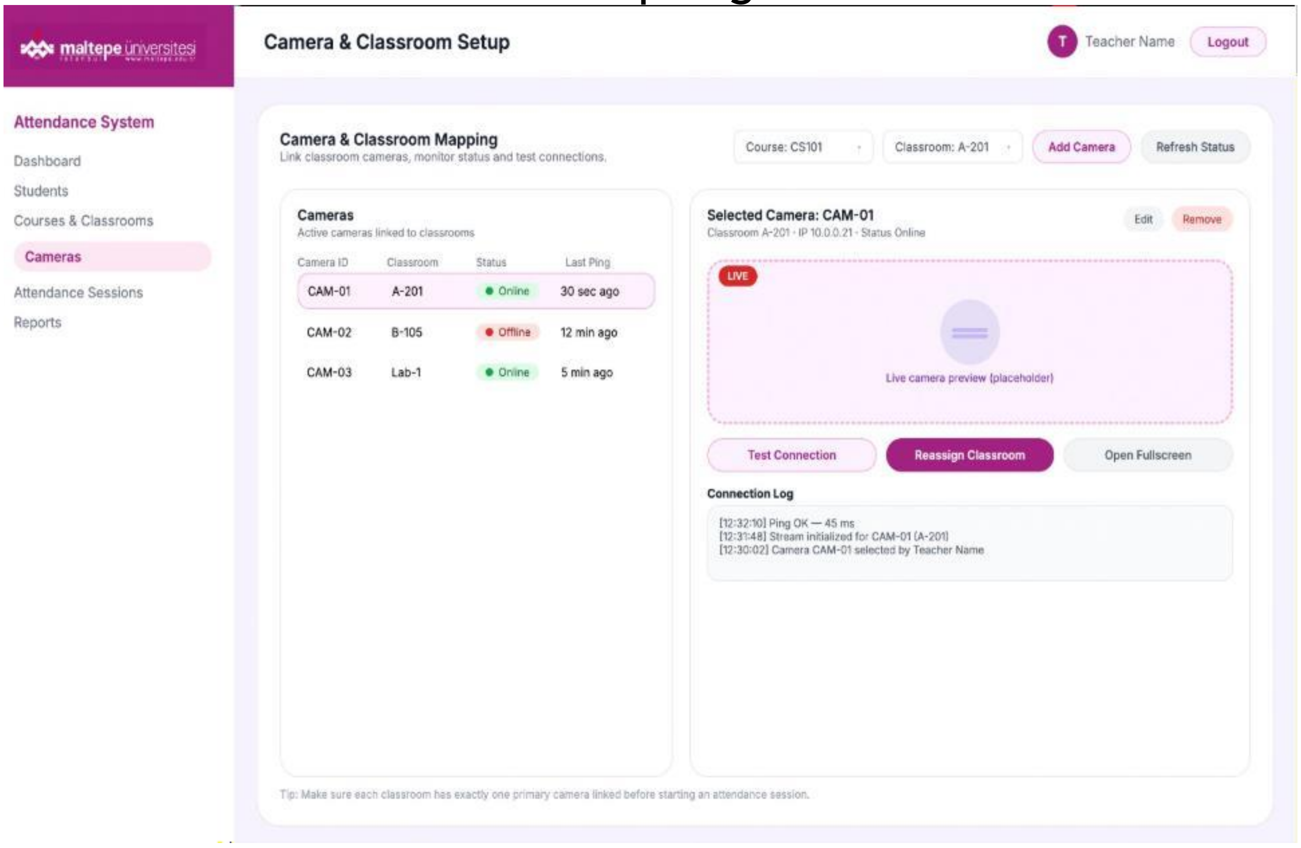
Average Attendance 92%

Tip: Make sure each course has a valid classroom and schedule before creating attendance sessions.

Description:

This page allows teachers to manage their courses, assign classrooms, and view weekly schedules. It provides a clear overview of course details, classroom assignments, and student enrollment, helping instructors organize their teaching activities efficiently.

5.5 Camera & Classroom Setup Page



Camera & Classroom Mapping
Link classroom cameras, monitor status and test connections.

Cameras
Active cameras linked to classrooms

Camera ID	Classroom	Status	Last Ping
CAM-01	A-201	● Online	30 sec ago
CAM-02	B-105	● Offline	12 min ago
CAM-03	Lab-1	● Online	5 min ago

Selected Camera: CAM-01
Classroom A-201 · IP 10.0.0.21 · Status Online

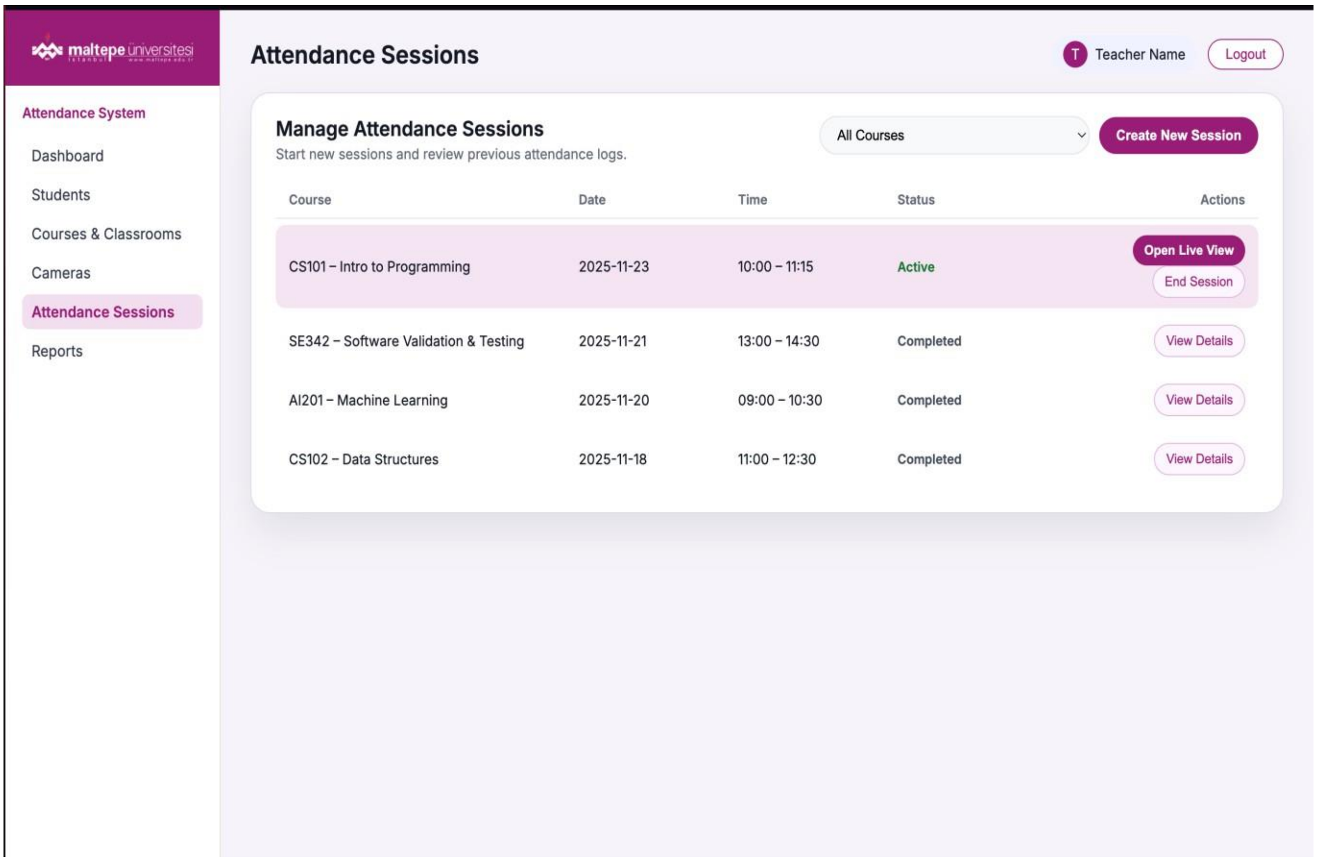
Connection Log

```
[12:32:10] Ping OK — 45 ms
[12:31:48] Stream initialized for CAM-01 (A-201)
[12:30:02] Camera CAM-01 selected by Teacher Name
```

Tip: Make sure each classroom has exactly one primary camera linked before starting an attendance session.

Description:
This page allows teachers to manage classroom cameras by monitoring their status, reviewing live previews, and assigning cameras to specific classrooms. It provides tools for testing connections, updating camera links, and viewing real-time activity logs to ensure reliable attendance recording.

5.6 Attendance Sessions Page



Attendance Sessions

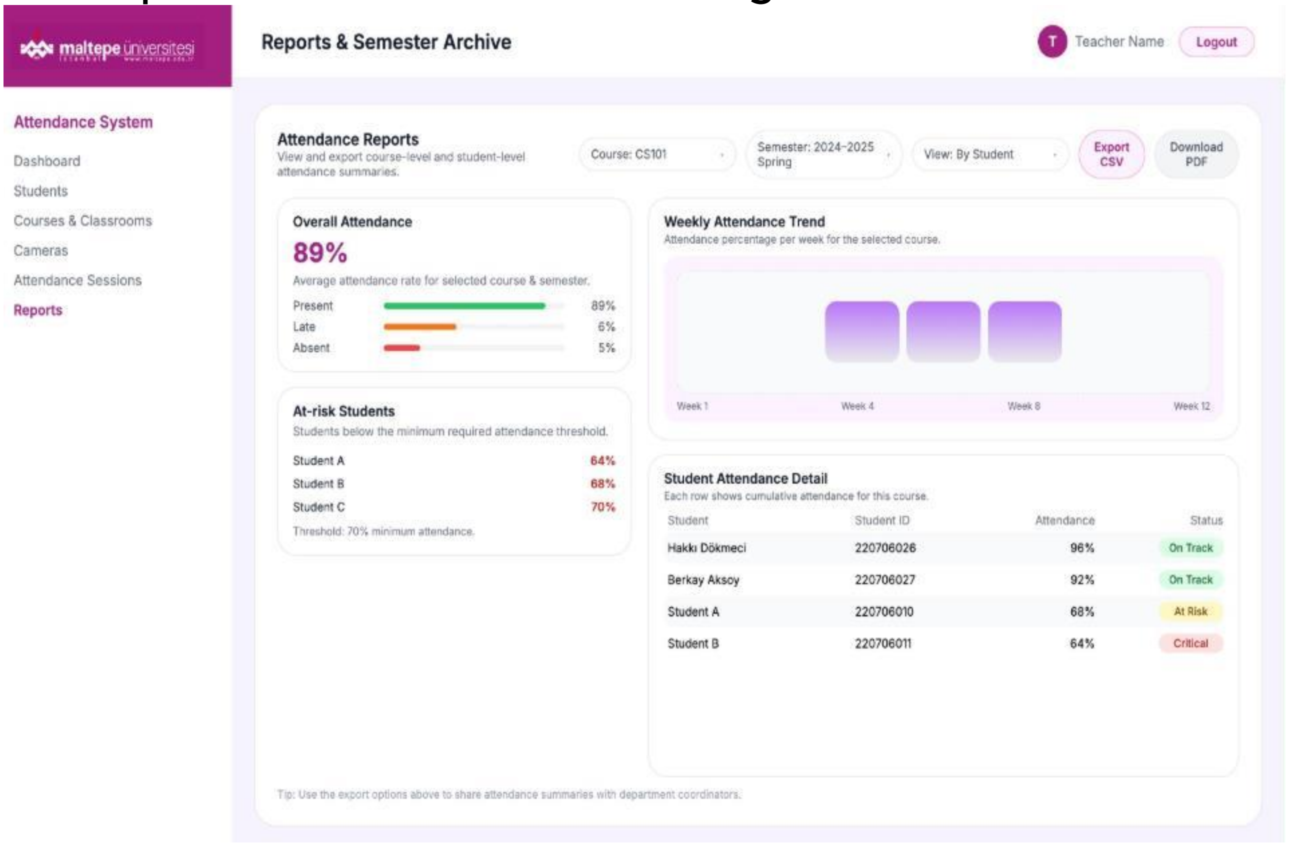
Manage Attendance Sessions

Start new sessions and review previous attendance logs.

Course	Date	Time	Status	Actions
CS101 – Intro to Programming	2025-11-23	10:00 – 11:15	Active	Open Live View / End Session
SE342 – Software Validation & Testing	2025-11-21	13:00 – 14:30	Completed	View Details
AI201 – Machine Learning	2025-11-20	09:00 – 10:30	Completed	View Details
CS102 – Data Structures	2025-11-18	11:00 – 12:30	Completed	View Details

Description:
The Attendance Sessions Page allows instructors to start new attendance sessions, monitor active sessions, and review completed ones. It provides a structured list of all sessions with details such as course name, date, time, and session status. Teachers can quickly open live camera views, end active sessions, or review past attendance records through this interface.

5.7 Reports & Semester Archive Page



Reports & Semester Archive

T Teacher Name Logout

Attendance System

- Dashboard
- Students
- Courses & Classrooms
- Cameras
- Attendance Sessions
- Reports

Attendance Reports

View and export course-level and student-level attendance summaries.

Course: CS101 | Semester: 2024-2025 Spring | View: By Student | Export CSV | Download PDF

Overall Attendance

89%

Average attendance rate for selected course & semester.

Present	89%
Late	6%
Absent	5%

At-risk Students

Students below the minimum required attendance threshold.

Student A	64%
Student B	68%
Student C	70%

Threshold: 70% minimum attendance.

Weekly Attendance Trend

Attendance percentage per week for the selected course.

Student Attendance Detail

Each row shows cumulative attendance for this course.

Student	Student ID	Attendance	Status
Hakkı Dökmeci	220706026	96%	On Track
Berkay Aksoy	220706027	92%	On Track
Student A	220706010	68%	At Risk
Student B	220706011	64%	Critical

Tip: Use the export options above to share attendance summaries with department coordinators.

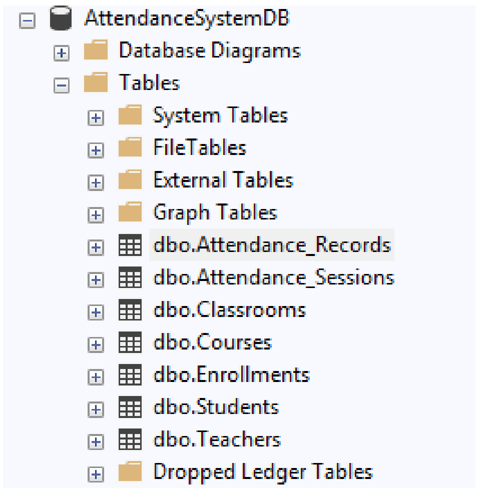
Description:
This page provides detailed attendance analytics for each course and semester. Teachers can view overall attendance rates, weekly attendance trends, at-risk students, and individual student records. It also offers export options such as CSV and PDF for sharing attendance summaries with academic departments.

6. Database development

The database layer of the system was implemented based on the previously designed ERD (Entity Relationship Diagram). All entities, attributes, relationships, and constraints defined in the ERD were successfully translated into a fully normalized relational database structure using Microsoft SQL Server. The final schema includes seven core tables—Students, Teachers, Courses, Classrooms, Enrollments, Attendance_Sessions, and Attendance_Records—ensuring data integrity, scalability, and support for the system's functional requirements.

6.1 Database Storage and Management Environment

The database structure designed based on the ERD was successfully created and stored in the Microsoft SQL Server (MSSQL) environment. As shown in the figure, all related tables are organized under the **AttendanceSystemDB** database, including Students, Teachers, Courses, Classrooms, Enrollments, Attendance_Sessions, and Attendance_Records. These tables are managed within SQL Server to ensure data consistency, relational integrity, secure storage, and efficient query execution throughout the system.

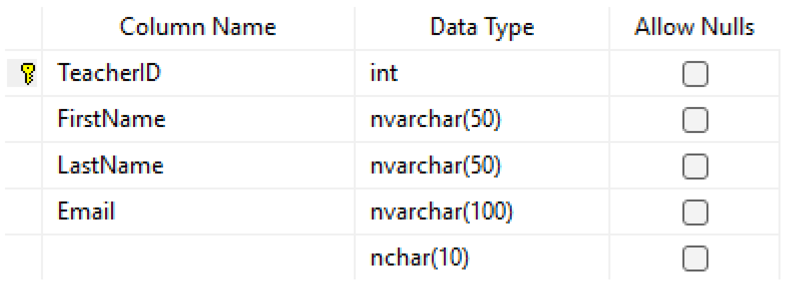

AttendanceSystemDB
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Attendance_Records
dbo.Attendance_Sessions
dbo.Classrooms
dbo.Courses
dbo.Enrollments
dbo.Students
dbo.Teachers
Dropped Ledger Tables

6.2 Database Tables Description

This section provides a detailed explanation of all database tables created as part of the attendance management system. Each table has been implemented based on the finalized Entity Relationship Diagram (ERD), ensuring proper normalization, data integrity, and relational consistency. The tables collectively support user management, course assignment, enrollment tracking, session scheduling, and attendance recording processes within the system.

Teachers Table

Stores mandatory personal and contact information for instructors responsible for teaching courses.

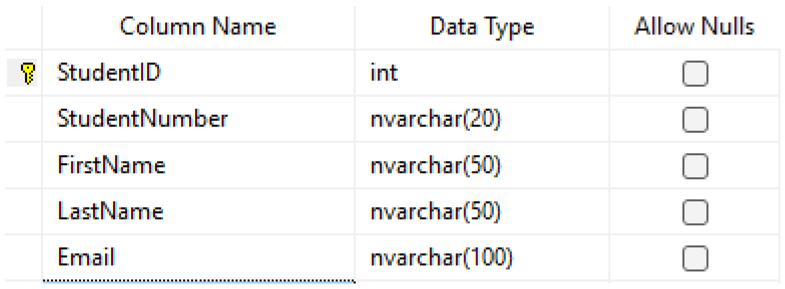


Column Name	Data Type	Allow Nulls
TeacherID	int	☐
FirstName	nvarchar(50)	☐
LastName	nvarchar(50)	☐
Email	nvarchar(100)	☐
	nchar(10)	☐

Column	Data Type	Purpose
TeacherID	INT	Primary Key, unique identifier
FirstName	NVARCHAR(50)	Instructor's first name
LastName	NVARCHAR(50)	Instructor's last name
Email	NVARCHAR(100)	Instructor's unique email address

Students Table

Stores academic identity and personal contact information for enrolled students.



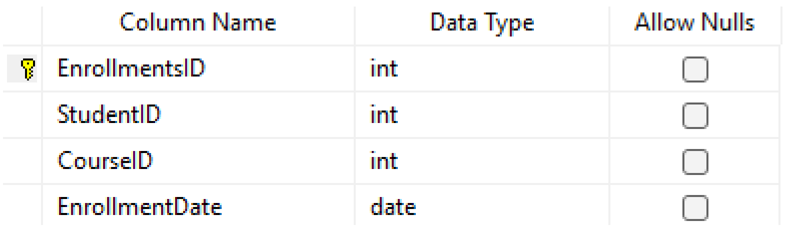
Column Name	Data Type	Allow Nulls
StudentID	int	☐
StudentNumber	nvarchar(20)	☐
FirstName	nvarchar(50)	☐
LastName	nvarchar(50)	☐
Email	nvarchar(100)	☐

Column	Data Type	Purpose
StudentID	INT	Primary Key
StudentNumber	NVARCHAR(20)	Unique institutional student number
FirstName	NVARCHAR(50)	Student's first name
LastName	NVARCHAR(50)	Student's last name

`Email`	`NVARCHAR(100)`	Student’s email, must be unique

Enrollments Table

Represents the **bridge table** managing the Many-to-Many relationship between Students and Courses.

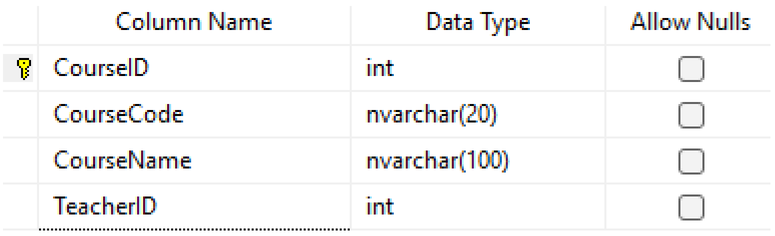


	Column Name	Data Type	Allow Nulls
🔑	EnrollmentsID	int	☐
	StudentID	int	☐
	CourseID	int	☐
	EnrollmentDate	date	☐

Column	Data Type	Purpose
`EnrollmentsID`	`INT`	Primary Key
`StudentID`	`INT`	Foreign Key referencing `Students`
`CourseID`	`INT`	Foreign Key referencing `Courses`
`EnrollmentDate`	`DATE`	Course enrollment date

Courses Table

Defines institution courses and links each one to the assigned teacher.

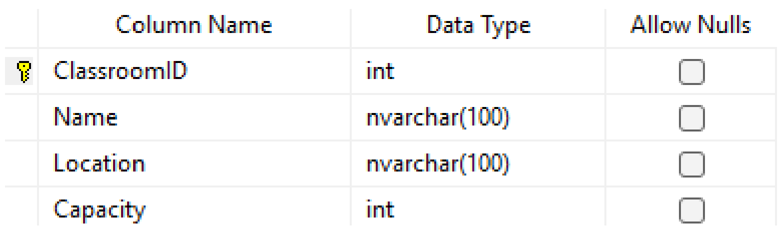


	Column Name	Data Type	Allow Nulls
🔑	CourseID	int	☐
	CourseCode	nvarchar(20)	☐
	CourseName	nvarchar(100)	☐
	TeacherID	int	☐

Column	Data Type	Purpose
`CourseID`	`INT`	Primary Key
`CourseCode`	`NVARCHAR(20)`	Unique course identifier
`CourseName`	`NVARCHAR(100)`	Full course title
`TeacherID`	`INT`	Foreign Key referencing `Teachers`

Classrooms Table

Stores physical classroom information used for scheduling teaching sessions.

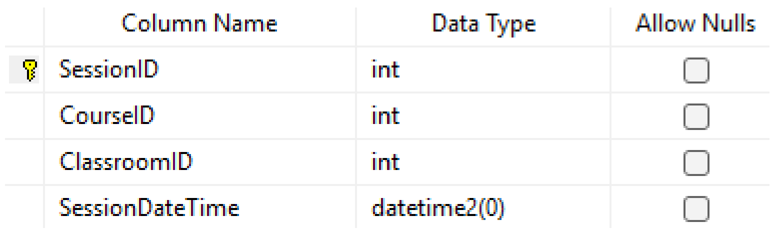


	Column Name	Data Type	Allow Nulls
🔑	ClassroomID	int	☐
	Name	nvarchar(100)	☐
	Location	nvarchar(100)	☐
	Capacity	int	☐

Column	Data Type	Purpose
`ClassroomID`	`INT`	Primary Key
`Name`	`NVARCHAR(100)`	Classroom title/label
`Location`	`NVARCHAR(100)`	Physical building/room identification
`Capacity`	`INT`	Maximum number of students allowed

6-Attendance_Sessions Table

Records scheduled course sessions, connecting a lesson, location, and scheduled time.



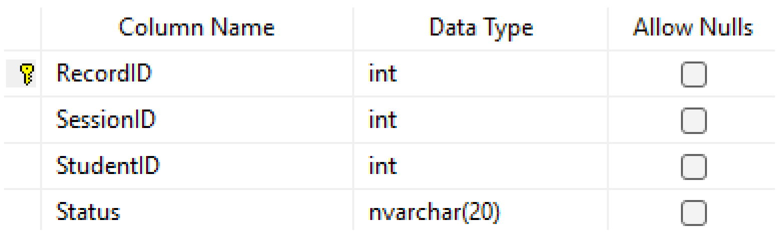
	Column Name	Data Type	Allow Nulls
🔑	SessionID	int	☐
	CourseID	int	☐
	ClassroomID	int	☐
	SessionDateTime	datetime2(0)	☐

Column	Data Type	Purpose
`SessionID`	`INT`	Primary Key
`CourseID`	`INT`	Foreign Key referencing `Courses`

ClassroomID	INT	Foreign Key referencing Classrooms
SessionDateTime	DATETIME2(0)	Exact start time of the session

7-Attendance_Records Table

Stores attendance results for each student and each session.



	Column Name	Data Type	Allow Nulls
🔑	RecordID	int	☐
	SessionID	int	☐
	StudentID	int	☐
	Status	nvarchar(20)	☐

Column	Data Type	Purpose
`RecordID`	`INT`	Primary Key
`SessionID`	`INT`	Foreign Key referencing `Attendance_Sessions`
`StudentID`	`INT`	Foreign Key referencing `Students`
`Status`	`NVARCHAR(20)`	Attendance status (`Present`, `Absent`, `Late`)

7.Backend API Integration & Data Flow Documentation

Overview of Backend Architecture

The backend of the Attendance Management System is built using ASP.NET Core Web API. Its main purpose is to process requests coming from the frontend, validate the input, and store the data in the SQL Server database. The backend operates as the connection point between the user interface and the data layer, following a simple UI → API → Database flow.

Database Integration via Entity Framework Core

Entity Framework Core is used to communicate with the SQL Server database. The existing database schema was imported into the project through EF Core scaffolding, which automatically generated the model classes. These models represent tables such as Students, Teachers, and Courses, allowing the backend to interact with the database in a structured and consistent way.

API Endpoints Implemented

The backend provides endpoints that allow new system records to be created. Each endpoint receives form data, checks the required fields, and saves the information to the database.
Implemented operations include:
Adding a new teacher
Adding a new student
Adding a new course
After successful processing, the API returns a simple confirmation message to the frontend.

Data Flow Between Frontend and Backend

When a user submits a form on the web interface, an HTTP POST request is sent to the appropriate API endpoint. The backend controller receives the data, performs basic validation, and uses Entity Framework Core to insert the new record into the SQL Server database. Once the operation is completed, the system sends a success response back to the user interface.

Backend Interaction Sequence

The interaction follows this basic sequence:
User enters information into the form
The frontend sends the data via POST
The API processes and saves the data
A confirmation message is returned
This ensures a clear and consistent flow for all create operations.

Technologies Used

The backend is developed using:
ASP.NET Core Web API
C#
Entity Framework Core
Microsoft SQL Server
HTML form integration for data submission
Swagger for testing endpoints

8.Page Test Scenarios

This section outlines the functional test scenarios for the application's core management interfaces. It focuses on validating the Create, Read, Update, and Delete operations, ensuring data integrity through validation rules, and verifying user interactions across the Student, Course, and Teacher pages.

8.1 Students Page Test

ID	Test Case Name	Type	Precondition	Steps	Expected Result
STU-01	Create Student with Valid Data	Positive	Student page is open and backend API is running.	1. Click “Add New Student”. 2. Enter valid First Name, Last Name, Student Number, and Email. 3. Click “Save”.	Student is successfully created and displayed in the student list.
STU-02	Create Student with Duplicate Student Number	Negative	A student with the same student number already exists.	1. Click “Add New Student”. 2. Enter an existing Student Number. 3. Fill other fields with valid data. 4. Click “Save”.	System prevents saving and displays an error message indicating the student number must be unique.
STU-03	Update Existing Student Information	Positive	At least one student exists in the list.	1. Select a student from the list. 2. Click “Edit”. 3. Update student information. 4. Click “Save”.	Student information is updated successfully.
STU-04	Delete Student	Positive	At least one student exists.	1. Select a student. 2. Click “Delete”. 3. Confirm deletion if required.	Student is removed from the system and no longer appears in the list.
STU-05	Search Student by Name	Positive	At least one student exists in the system.	1. Enter a student name in the search box. 2. Perform search.	Student list is filtered according to the search criteria.

This section covers the functional test scenarios for the Student Management module. It aims to verify the successful execution of Create, Read, Update, Delete for student records. Additionally, the test scope includes validation checks to ensure the system preserves data integrity by displaying appropriate error messages when duplicate student numbers are entered.

8.2 Courses Page Test

ID	Test Case Name	Type	Precondition	Steps	Expected Result
CRS-01	Create Course with Valid Data	Positive	Course page is open.	1. Click “Add New Course”. 2. Enter valid Course Code, Course Name and Credits. 3. Click “Save”.	Course is created successfully and displayed in the course list.
CRS-02	Create Course with Missing Course Code	Negative	Course page is open.	1. Click “Add New Course”. 2. Leave Course Code empty. 3. Click “Save”.	System displays validation error and does not save the course.

CRS-03	Create Course with Duplicate Course Code	Negative	A course with the same course code already exists.	1. Click “Add New Course”. 2. Enter an existing Course Code. 3. Click “Save”.	System prevents duplicate course creation and displays an error message.
CRS-04	Update Course Credits	Positive	At least one course exists.	1. Select a course. 2. Click “Edit”. 3. Update credit value. 4. Click “Save”.	Course credit value is updated successfully.
CRS-05	Search Course by Course Code	Positive	At least one course exists.	1. Enter course code in search box. 2. Perform search.	Course list is filtered according to the entered course code.

The following table is designed to verify the accuracy of operations on the Course page.It details functions such as defining new courses, updating credits, and performing searches. Specifically, these scenarios ensure that the system correctly handles mandatory field checks (e.g., empty course codes) and enforces validation rules for unique course codes.

8.3 Teachers Page Test

ID	Test Case Name	Type	Precondition	Steps	Expected Result
TCH-01	Create Teacher with Valid Data	Positive	Teacher page is open and backend API is running.	1. Click “Add New Teacher”. 2. Enter valid First Name, Last Name and Email. 3. Click “Save”.	Teacher is created successfully and displayed in the teacher list.

TCH-02	Create Teacher with Invalid Email Format	Negative	Teacher page is open.	1. Click “Add New Teacher”. 2. Enter an invalid email format (e.g., missing “@”). 3. Click “Save”.	System displays an email validation error and does not save the teacher.
TCH-03	Update Teacher Information	Positive	At least one teacher exists in the list.	1. Select a teacher from the list. 2. Click “Edit”. 3. Update teacher information. 4. Click “Save”.	Teacher information is updated successfully.
TCH-04	Delete Teacher	Positive	At least one teacher exists.	1. Select a teacher. 2. Click “Delete”. 3. Confirm deletion if required.	Teacher is removed from the system and no longer appears in the list.
TCH-05	Search Teacher by Last Name	Positive	At least one teacher exists.	1. Enter last name in the search box. 2. Perform search.	Teacher list is filtered according to the entered last name.

This section tests user interactions and data entry rules within the Teacher module. It analyzes the functionality of specific data validation rules—such as email format validation—alongside basic operations like adding, editing, and deleting teachers. Additionally, these tests verify whether the search filter produces accurate results.