# MALTEPE UNIVERSITY

Faculty of Engineering and Natural Sciences - Software Engineering

## SE 342

## Software Validation and Testing

Instructor: Dr. Emre Atlıer Olca

*Prepared by:*

22 07 06 059 - Ceren Yaşar

22 07 06 028 - Mehmet Şenadlı

22 07 06 018 - Bilal Çifteci

## Table of Contents

# Smart Attendance System

*For detailed tasks and progress, see the Jira board: 🔗 Jira Link*

## 1. Introduction

### 1.1 Project Purpose

This system aims to automate the attendance process in an educational institution. The process begins with the teacher adding student records along with their photos to the system. The integrated camera system then detects student faces in the classroom and compares them with the registered photos. When a successful match occurs, attendance is automatically recorded, and a “Attendance recorded” notification is generated in the system.

### 1.2 Scope

The automated attendance system is developed to accelerate the attendance process and minimize errors in educational institutions. The system includes the following features:

- Adding student records to the system along with their photos,
- Recognizing and matching student faces in the classroom using the camera,
- Automatically recording attendance upon successful matches,
- Providing real-time notifications to teachers and administrators about the attendance status.

## 2. Requirements

### 2.1. Functional Requirements

**FR-1: Student Management and Registration Module (SCRUM-25)**

- **FR-1.1:** The system must allow the creation of new student registrations. (SCRUM-1)
- **FR-1.2:** A student registration form (name, number, department, class, photo) must be created. (SCRUM-5, 6)
- **FR-1.3:** Validation rules must be added for form fields. (SCRUM-7)
- **FR-1.4:** A student photo upload function must be developed. (SCRUM-8)
- **FR-1.5:** Uploaded photos must be stored in the system (database or file system). (SCRUM-10)
- **FR-1.6:** Backend integration for photo/file upload must be implemented. (SCRUM-9)
- **FR-1.7:** A message must be shown to the user after successful registration. (SCRUM-11)

**FR-2: Automatic Face Recognition and Attendance**

- **FR-2.1:** The system must have a camera integration system. (SCRUM-26, 4)
- **FR-2.2:** The system must be able to perform automatic face recognition and attendance tracking via camera. (SCRUM-2)
- **FR-2.3:** A face recognition and matching system must be established. (SCRUM-12)
- **FR-2.4:** The system must have an algorithm that compares the face from the camera with existing photos registered in the system. (SCRUM-13)
- **FR-2.5:** An accuracy threshold for matching must be determined. (SCRUM-14)
- **FR-2.6:** After a successful match, the attendance record must be created automatically. (SCRUM-15)
- **FR-2.7:** A real-time "Attendance taken" message must be displayed when attendance is recorded. (SCRUM-16)

**FR-3: Reporting and Admin Panel (SCRUM-27)**

- **FR-3.1:** There must be a reporting and management interface for attendance data. (SCRUM-3)
- **FR-3.2:** A page displaying daily attendance data must be created. (SCRUM-18)
- **FR-3.3:** The reporting page must have filtering and sorting features. (SCRUM-19)
- **FR-3.4:** Reports must be filterable by name. (SCRUM-20)
- **FR-3.5:** Reports must be sortable by department/class. (SCRUM-21)
- **FR-3.6:** The system must provide a warning for unrecognized students and allow manual attendance entry. (SCRUM-17)
- **FR-3.7:** The system must include a module for editing incorrect attendance records. (SCRUM-22)
- **FR-3.8:** Frontend integration for real-time updates in the admin panel must be implemented. (SCRUM-23)

## 2.2. Non-Functional Requirements (NFR)

**NFR-1: Database and Infrastructure**

- **NFR-1.1:** A database engine must be selected and installed. (SCRUM-34)
- **NFR-1.2:** Database setup and configuration must be performed. (SCRUM-33)
- **NFR-1.3:** A database connection configuration (connection string) must be defined. (SCRUM-35)
- **NFR-1.4:** An ORM (SQLAlchemy, Prisma, Django ORM, etc.) must be selected and developed. (SCRUM-38, 37)
- **NFR-1.5:** Database model classes (entities) must be created. (SCRUM-39)

**NFR-2: Security**

- **NFR-2.1:** Database security and access management must be ensured. (SCRUM-40)
- **NFR-2.2:** User roles must be defined and authorization implemented. (SCRUM-41)
- **NFR-2.3:** Encryption and access policies for sensitive data must be applied. (SCRUM-42)
- **NFR-2.4:** The system must have connection security (e.g., SSL). (SCRUM-43)
- **NFR-2.5:** Database access tests must be conducted. (SCRUM-36)

**NFR-3: Performance and Maintenance**

- **NFR-3.1:** Database backup and performance management must be planned. (SCRUM-44)
- **NFR-3.2:** An automatic backup plan must be created. (SCRUM-45)
- **NFR-3.3:** Query performance analyses must be conducted. (SCRUM-47)
- **NFR-3.4:** Necessary indexing strategies must be determined. (SCRUM-46)

**NFR-4: Analysis and Documentation (DONE)**

- **NFR-4.1:** Requirements analysis and documentation for the project must be completed. (SCRUM-48, 49, 50, 51, 52, 53, 54)
- **NFR-4.2:** A requirements document (SRS) must be created. (SCRUM-55)

# 3. Sprints

## 3.1 Sprint 0: Software Requirement and Analysis

- **Purpose:** To determine and document the project's scope, objectives, and functional/non-functional requirements.
- **Tasks:**
  - SCRUM-48: Software Requirement and Analysis
  - SCRUM-49: Gather Requirements from the Client
  - SCRUM-51: Schedule a meeting with the client and define the project scope...
  - SCRUM-52: Document needs and expectations
  - SCRUM-53: Prioritize requirements by categorizing them
  - SCRUM-50: Requirement Analysis and Documentation
  - SCRUM-54: Classify functional and non-functional requirements...
  - SCRUM-55: Create requirements document (SRS)

## 3.2 Sprint 1: Database Design and Management

- **Purpose:** To design, install, configure the database schema, and set up security and performance settings.
- **Tasks:**
  - SCRUM-28: Database Design and Management
  - SCRUM-29: Database Schema Design
  - SCRUM-30: Create ER Diagram
  - SCRUM-31: Determine Table and Column Types
  - SCRUM-32: Test Relationships (1-N, N-M)
  - SCRUM-33: Database Setup and Configuration
  - SCRUM-34: Select and Install Database Engine
  - SCRUM-35: Connection Configuration (connection string)
  - SCRUM-37: Develop ORM / Database Connection Layer
  - SCRUM-39: Create Model Classes
  - SCRUM-38: Select ORM Structure (SQLAlchemy, Prisma, Django ORM)
  - SCRUM-40: Database Security and Access Management

  - SCRUM-43: Connection Security (SSL Environment)
  - SCRUM-42: Apply Encryption and Access Policies
  - SCRUM-41: Define User Roles
  - SCRUM-44: Database Backup and Performance Management
  - SCRUM-47: Analyze Query Performance
  - SCRUM-46: Determine Indexing Strategies
  - SCRUM-45: Create Automatic Backup Plan
  - SCRUM-36: Perform Access Tests

## 3.3 Sprint 2: Student Management and Registration Module

- **Purpose:** To develop the necessary forms and backend integrations for administrators to create new student registrations.
- **Tasks:**
  - SCRUM-25: Student Management and Registration Module
  - SCRUM-1: Create Student Registration
  - SCRUM-5: Create student registration form
  - SCRUM-6: Add form fields (name, number, department, class, photo)
  - SCRUM-7: Add validation rules for form fields
  - SCRUM-8: Develop photo upload function
  - SCRUM-10: Store photos in the system
  - SCRUM-11: Display message after successful registration
  - SCRUM-9: File upload backend integration

## 3.4 Sprint 3: Camera Integration System

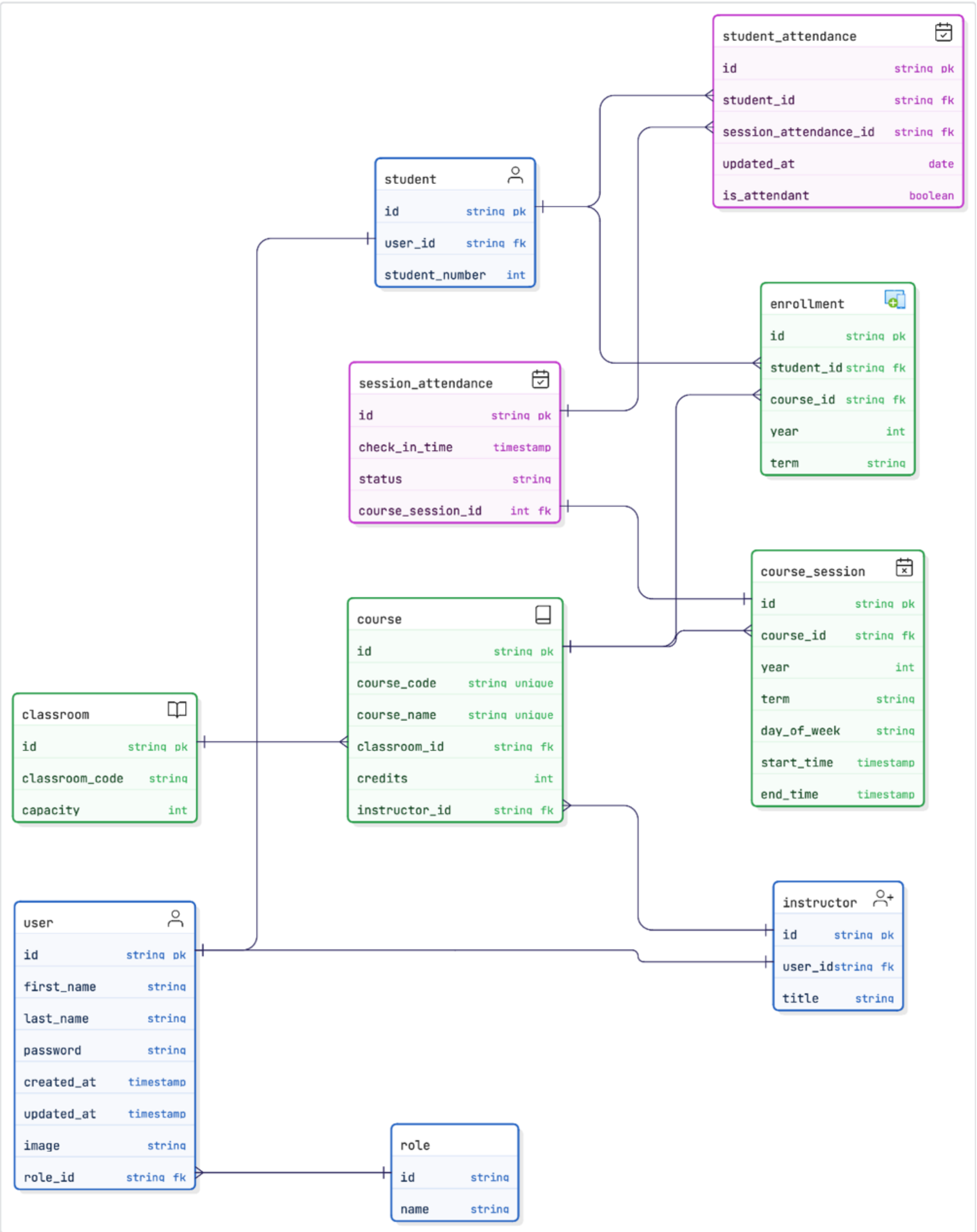
- **Purpose:** To integrate the camera system, set up face recognition and matching algorithms to automate attendance recording.
- **Tasks:**
  - SCRUM-26: Camera Integration System
  - SCRUM-2: Automatic Face Recognition and Attendance with Camera
  - SCRUM-4: Add camera integration
  - SCRUM-12: Set up face recognition and matching system
  - SCRUM-13: Comparison algorithm with existing photos
  - SCRUM-14: Determine the match accuracy threshold
  - SCRUM-15: Create attendance record automatically
  - SCRUM-16: Display real-time "Attendance taken" message

## 3.5 Sprint 4: Reporting and Admin Panel

- **Purpose:** To create an interface for administrators to view, filter, and manage the attendance records taken.
- **Tasks:**
  - SCRUM-27: Reporting and Admin Panel
  - SCRUM-3: Attendance Reporting and Management
  - SCRUM-18: Create daily attendance viewing page
  - SCRUM-19: Add filtering and sorting feature

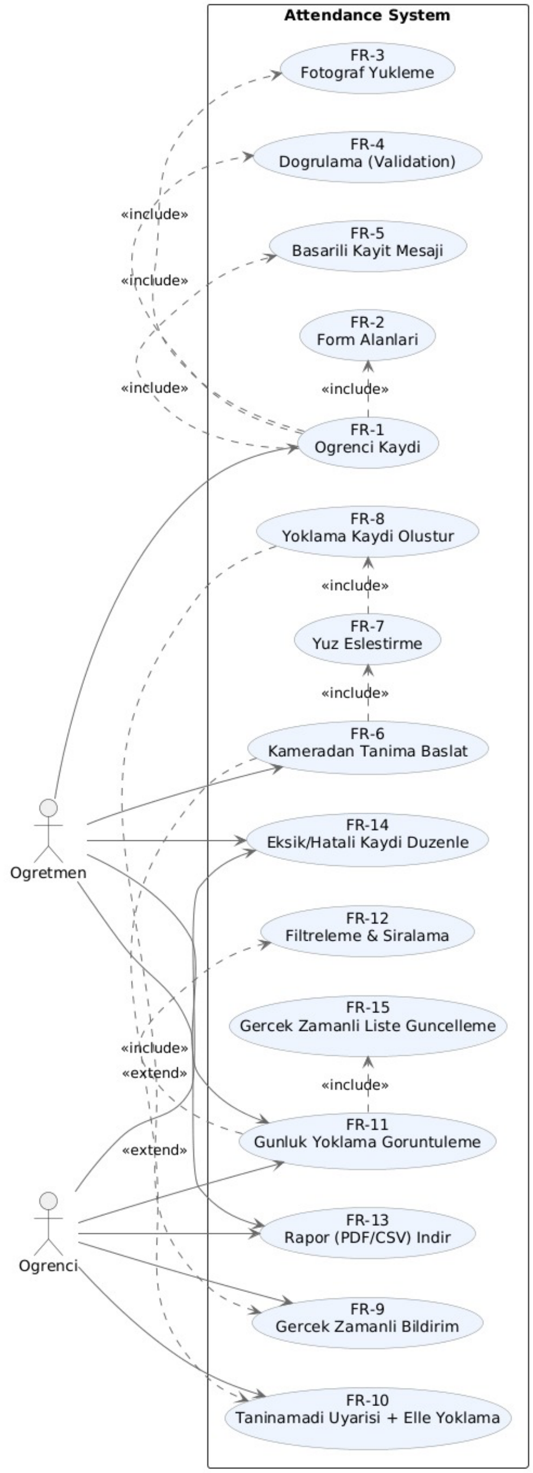
- SCRUM-20: Filter by name
- SCRUM-21: Sort by department/class
- SCRUM-17: Warning for unrecognized student and manual attendance entry
- SCRUM-22: Add module to edit incorrect attendance
- SCRUM-23: Frontend integration for real-time updates

## 4.ERD (Entity Relationship Diagram)



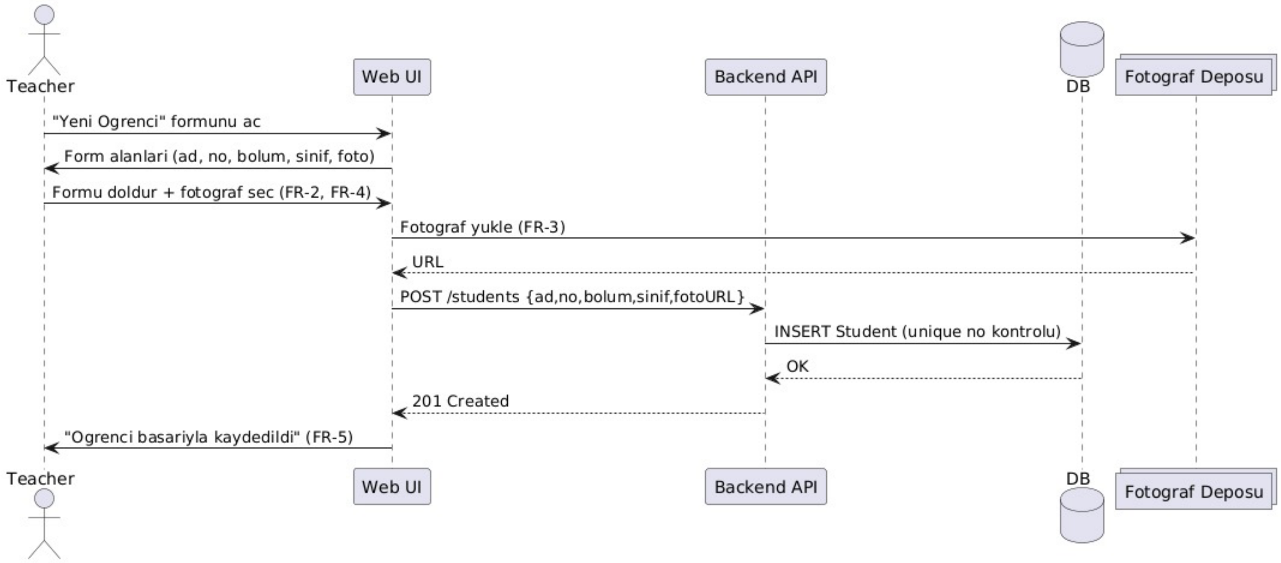
# 5. UML DIAGRAMS

## 5.1 Use Case Diagram

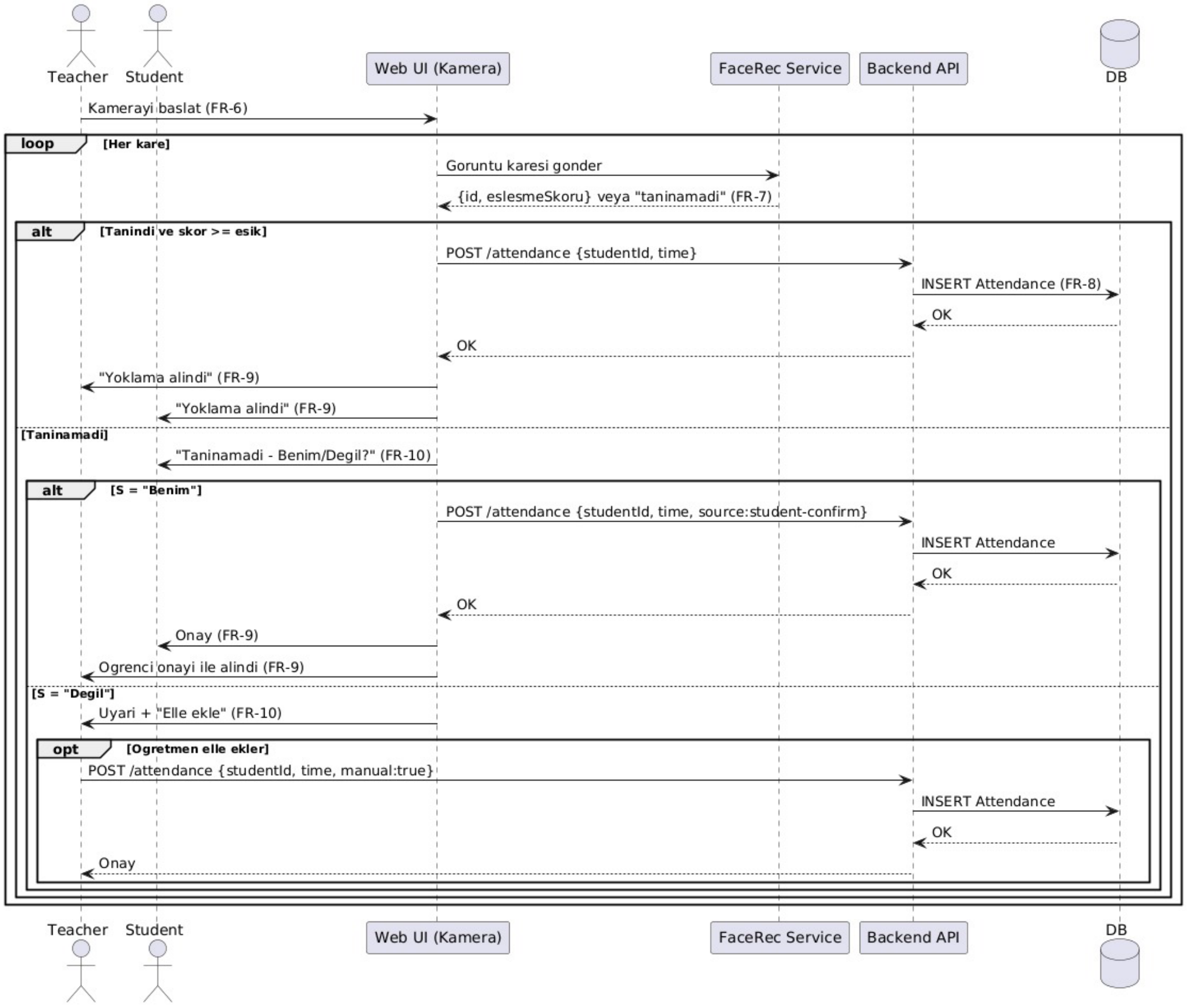


## 5.2 Sequence Diagrams

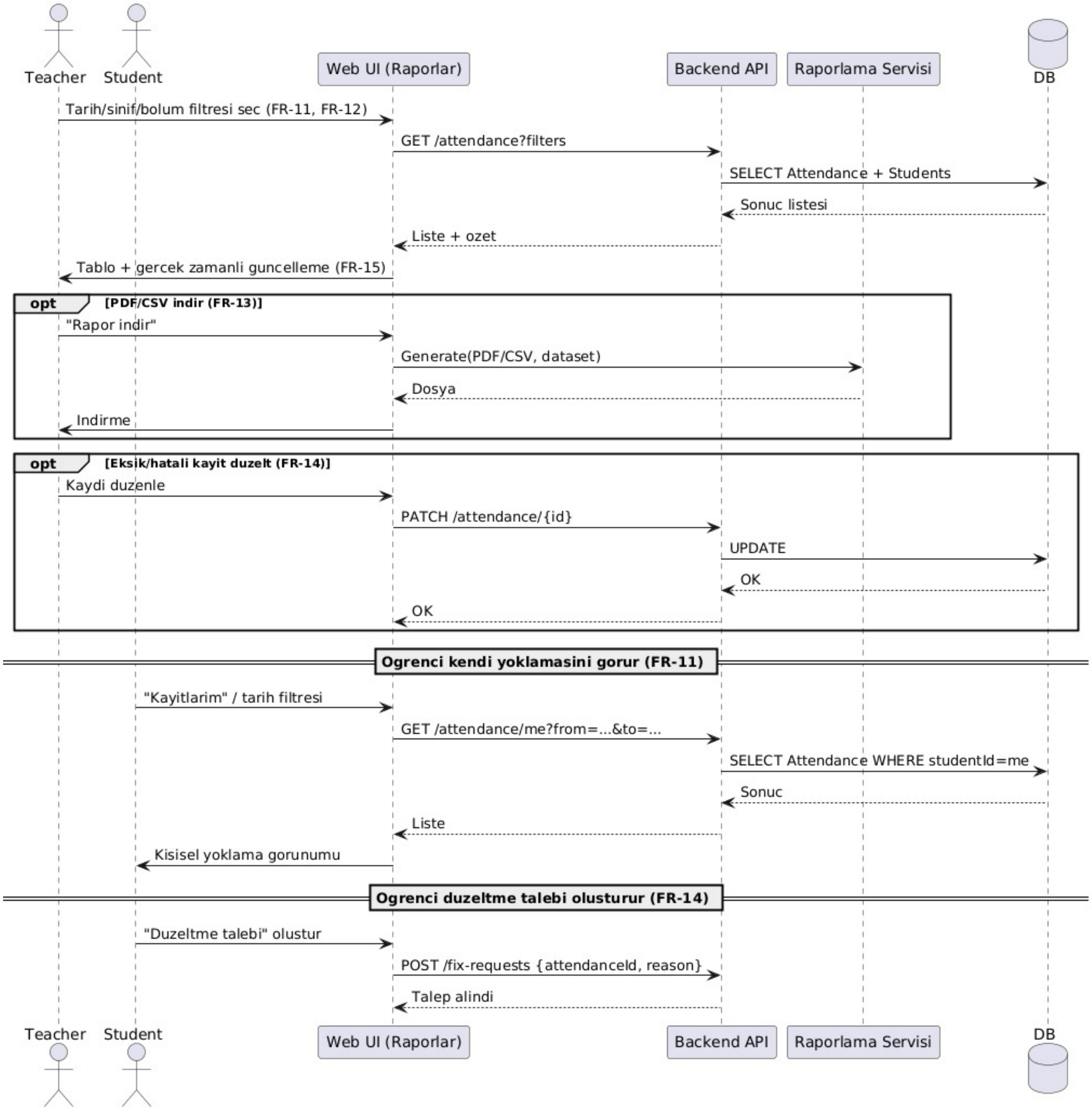
### 5.2.1 Student Registration Sequence


Teacher
Web UI
Backend API
DB
Fotograf Deposu
"Yeni Ogrenci" formunu ac
Form alanlari (ad, no, bolum, sinif, foto)
Formu doldur + fotograf sec (FR-2, FR-4)
Fotograf yukle (FR-3)
URL
POST /students {ad,no,bolum,sinif,fotoURL}
INSERT Student (unique no kontrolu)
OK
201 Created
"Ogrenci basariyla kaydedildi" (FR-5)
Teacher
Web UI
Backend API
DB
Fotograf Deposu

## 5.2.2 Automatic Attendance Sequence



### 5.2.3 Attendance Reporting Sequence


Teacher
Student
Web UI (Raporlar)
Backend API
Raporlama Servisi
DB
Tarih/sinif/bolum filtresi sec (FR-11, FR-12)
GET /attendance?filters
SELECT Attendance + Students
Sonuc listesi
Liste + ozet
Tablo + gercek zamanli guncelleme (FR-15)
opt
[PDF/CSV indir (FR-13)]
"Rapor indir"
Generate(PDF/CSV, dataset)
Dosya
Indirme
opt
[Eksik/hatali kayit duzelt (FR-14)]
Kaydi duzenle
PATCH /attendance/{id}
UPDATE
OK
OK
Ogrenci kendi yoklamasini gorur (FR-11)
"Kayitlarim" / tarih filtresi
GET /attendance/me?from=...&to=...
SELECT Attendance WHERE studentId=me
Sonuc
Liste
Kisisel yoklama gorunumu
Ogrenci duzeltme talebi olusturur (FR-14)
"Duzeltme talebi" olustur
POST /fix-requests {attendanceId, reason}
Talep alindi
Teacher
Student
Web UI (Raporlar)
Backend API
Raporlama Servisi
DB