Group 9: Facial Recognition System for Class Attendance

Xavier Evans, Alan Jos and Connor Gurnham

CIS 453: Requirements Team Project

October 23rd, 2021

Group 9 Requirements Project - Software Requirements Document

Xavier Evans
Alan Jos
Connor Gurnham

Topic: Facial Recognition System to take Attendance in Class

Project Description

When determining a software engineering project, it is important to create a system that solves an issue or helps with a current system. Syracuse University is home to nearly 25,000 students and this can make systems such as attendance hard to manage. Thus, our idea is to create a biometric system which helps automate this process and make life easier for Professors on campus.This system will incorporate the current Syracuse University blackboard system. The goal is to create a safe and secure system that has an easier way to track attendance for Professors that include it as a key component of the overall grade. Inside of the classroom, a stationary camera will be mounted near the door such that students will walk towards it before class to check in. Since the system will connect to the current Blackboard and Syracuse University Registrar Database, the time, location and class section will contain a list of students who are supposed to attend class. When a student “signs in”, the system will mark them as present within Blackboard. From a user standpoint, it will be very straightforward and the Professor will not need to do any other work. If there are issues with the system, the Professor can always revert back to Blackboard attendance as we look to fix the temporary issues. Students that do not attend class will be marked as absent after 15 minutes of the class starting. As time goes on, the system will have multiple software updates and potentially interface updates so that it is compatible with the newest versions of Blackboard available.

Non-Functional Requirements

1. Operational Requirements

1.1. The system will be able to connect to Blackboard and enter in the attendance data for every student in a given course.

1.2. The system will use a stationary camera mounted near the classroom door to track attendance.

1.3. The system will operate on Blackboard versions 9.0 or later.

2. Performance Requirements:

2.1. The system will take no more than three seconds to verify a student's identity.

2.2. The system will update course attendance every thirty seconds.

2.3. The system should be able to recognize up to 300 students per course.

3. Security Requirements:

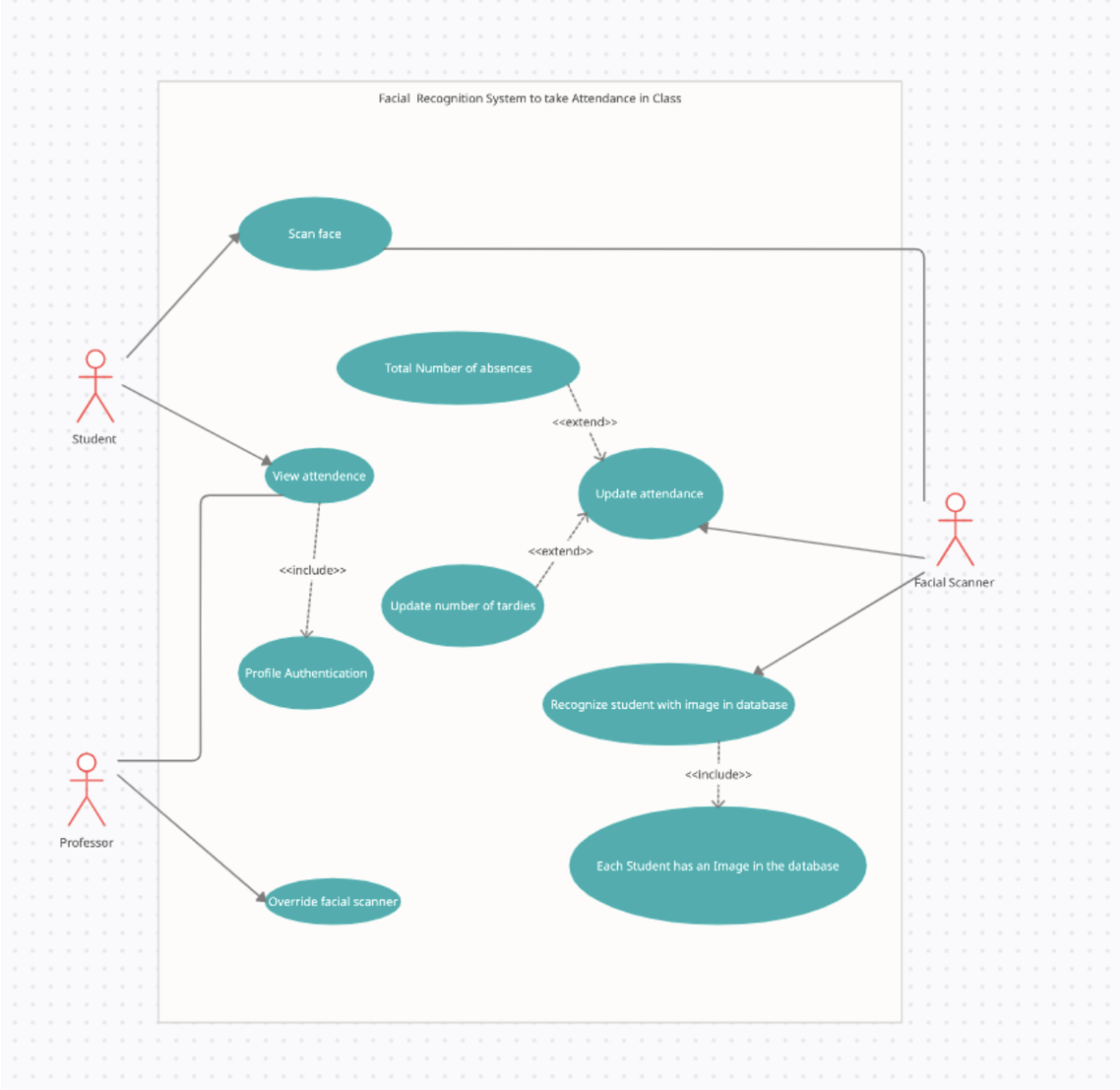
3.1. The system will not allow a student to take attendance on another's behalf.

3.2. Only live motion will be used for image processing, not images of a person.

Functional Requirements

1. The system shall validate all students within a given course.
2. The system shall continuously check for students entering the room and mark them as late 15 minutes after the class has begun.
3. If a student is not present in the class, they will be marked as absent.
4. The system will store the data in an Excel sheet for bookkeeping purposes.
5. The system will alert the Professor if a student has been absent more than twice in a row.

Use-Case Diagram



Use-Case Descriptions

User Case Name: Scan Face

ID: 1

Importance level: High

Primary Actor: Facial Scanner

Use Case Type: Essential

Stakeholders and Interests:

Student: Wants to receive credit for attendance from the system

Professor: Wants to record attendance of the class

Brief description: This use case describes how the students face will be scanned for attendance.

Trigger: Students stands in front of scanner for facial scanning

Type: Internal

Relationships:

Association:

Include:

Extend:

Generalization:

Normal Flow of Events:

1. The student arrives at class and waits in line to use facial scanner
2. Student stands in front of scanner
3. Camera scans face
4. Facial scanner matches face with photo in database
5. Student goes to their seat

SubFlows:

Alternate/Exceptional Flows:

1. Scanner fails to scan face and tries again

User Case Name: View Attendance

ID: 2

Importance level: High

Primary Actor: Professor

Use Case Type: Essential

Stakeholders and Interests:

Student: Wants to see how many classes they've missed

Professor: Wants to keep track of how many classes each student misses

Brief description: This use case describes how the professor and student will be able to view attendance

Trigger: Student or professor try to access attendance through blackboard

Type: Internal

Relationships:

Association:

Include: Profile Authentication

Extend:

Generalization:

Normal Flow of Events:

1. Student or professor log onto blackboard
2. They navigate to the attendance section
3. They can view what days they have been marked there or absent

SubFlows:

Alternate/Exceptional Flows:

1. Blackboard has not updated yet and leaves the date blank until it does

---

User Case Name: Update Attendance

ID: 3

Importance level: High

Primary Actor: Facial Scanner

Use Case Type: Essential

Stakeholders and Interests:

Student: Wants to receive attendance from scan

Professor: Wants to record attendance of class

Brief description: This use case describes how the facial scanner will update the attendance of the students in blackboard

Trigger: Facial scan is successful

Type: Internal

Relationships:

Association: Facial Scanner

Include:

Extend: Total number of absences, Update number of tardies

Generalization:

Normal Flow of Events:

1. Facial scanner scans students face
2. Scanner determines which student is signing in
3. Scanner updates Blackboard for the student it recognizes

SubFlows:

Alternate/Exceptional Flows:

1. Student does not use facial scanner within first 15 minutes of class and gets marked as absent

---

User Case Name: Recognize student with image of them in the database

ID: 4

Importance level: High

Primary Actor: Facial Scanner

Use Case Type: Essential

Stakeholders and Interests:

Student: Wants to receive attendance from scan

Professor: Wants to record attendance of class

Brief description: This use case describes how the facial scanner will match the face it is scanning with the pictures of students in the database to mark the student present

Trigger: Students stands in front of scanner for facial scanning

Type: Internal

Relationships:

Association:

Include: Each student has an image in database

Extend:

Generalization:

Normal Flow of Events:

1. Facial scanner scans student face
2. Scanner then goes into database and searches through photos until it matches one of the students photos

SubFlows:

Alternate/Exceptional Flows:

1. Scanner fails to find a picture of the student in its database and returns error message

---

User Case Name: Override facial scanner

ID: 5

Importance level: Medium

Primary Actor: Professor

Use Case Type: Essential

Stakeholders and Interests:

Student: Wants to change an attendance marked absent

Professor: Wants to change an attendance of a student

Brief description: This use case describes how the professor will have the ability to go into blackboard and change what the scanner marked a student for attendance on any day

Trigger: Students stands in front of scanner for facial scanning

Type: Internal

Relationships:

Association:

Include:

Extend: Professor manually takes and enters student attendance

Generalization:

Normal Flow of Events:

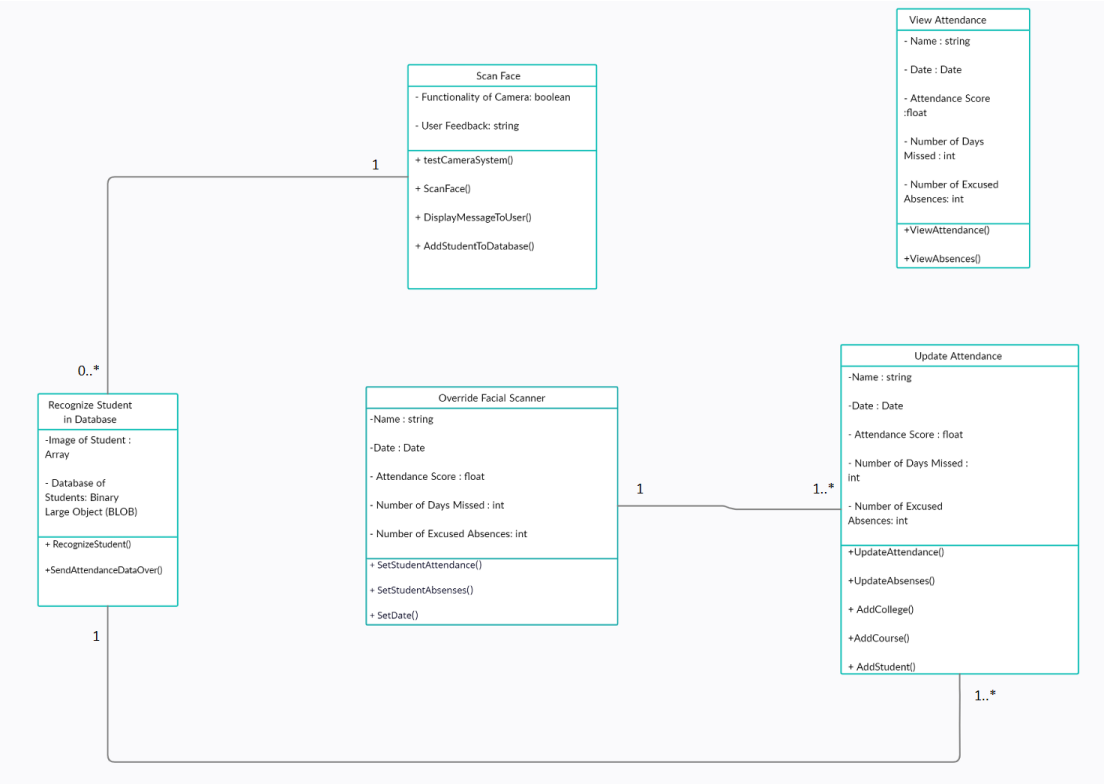
1. Professor logs into blackboard
2. Navigates to attendance
3. Chooses the student they want to change the attendance for
4. Changes the attendance
5. Clicks update

SubFlows:

Alternate/Exceptional Flows:

1. If the system goes down or updates, the professor will manually take attendance.

Class Diagram:



Relationships:

Scan face has a 1 to 0..1* relationship to recognize student in Database. This is because there is one face being scanned at a time and it is possible to have no relationship with any of the faces or to match a face in the database.

Recognize student in database has a 1 to 1..* relationship to Update attendance. This is because when a face is recognized in the database it updates any of the student's attendance in blackboard that it matches.

Override facial scanner has a 1 to 1..* relationship to update relationship. This is because only the professor is able to override the facial scanner to update any students' attendance.

View attendance does not have a relationship because none of the other classes are associated with it.

Front of CRC Card:

Class Name: Scan Face	ID: 1	Type: Concrete
Description: This class will explain how the students face will be scanned for attendance.		Associated Use Cases: 0
Responsibilities 1) Turn on the camera & test functionality. 2) Call the scanner method. 3) Scan face. 4) Notify student when their face has been scanned by the system or if an error occurred.		Collaborators 1. Recognize Student in Database

Back of CRC Card:

Attributes: - Confirmation display (string) - Functionality of camera (boolean)
Relationships: Generalization (a-kind-of): Aggregation (has-parts): Other Associations: Recognize Student in Database

Front of CRC Card:

Class Name: View Attendance	ID: 2	Type: Concrete
Description: This class will be used to give the professor and students access to view attendance.		Associated Use Cases: 0
Responsibilities 1) Allow students to view attendance history in Blackboard 2) Allow professor to view attendance history in Blackboard		Collaborators N/A

Back of CRC Card:

Attributes: - Name (string) - Date (date) - Attendance (float) - Total missed days (int) - Total excused absences (int)
Relationships: Generalization (a-kind-of): Aggregation (has-parts): Other Associations:

Front of CRC Card:

Class Name: Override Facial Scanner	ID: 3	Type: Concrete
Description: This class describes how the professor will have the ability to go into blackboard and change what the scanner marked a student for attendance on any day		Associated Use Cases: 1
Responsibilities 1) Allow professor to change student attendance from absent to present 2) Allow professor to change student attendance from present to absent 3) Allow professor to change the number of absences		Collaborators 1) Update Attendance

Back of CRC Card:

Attributes:

- Name (string)
- Date (date)
- Attendance Score (float)
- Number of Days Missed (int)
- Numbed of Excused Absences (int)

Relationships:

Generalization (a-kind-of):

Aggregation (has-parts): Update Attendance

Other Associations:

Front of CRC Card:

Class Name: Update Attendance	ID: 4	Type: Concrete
Description: The facial scanner will have the ability to scan the face of a student and update the attendance in blackboard with the associated attributes.		Associated Use Cases: 3
Responsibilities 1) Allow facial scanner to change student attendance from absent to present 2) Allow professor to change student attendance from present to absent and vise versa.		Collaborators 1) Facial Scanner 2) Override Facial Scanner

Back of CRC Card:

Attributes:

- Date (date)
- Attendance (text)
- Student (text)
- Student ID (int)

Relationships:

Generalization (a-kind-of):

Aggregation (has-parts): Override Facial Scanner

Other Associations: Facial Scanner

Front of CRC Card:

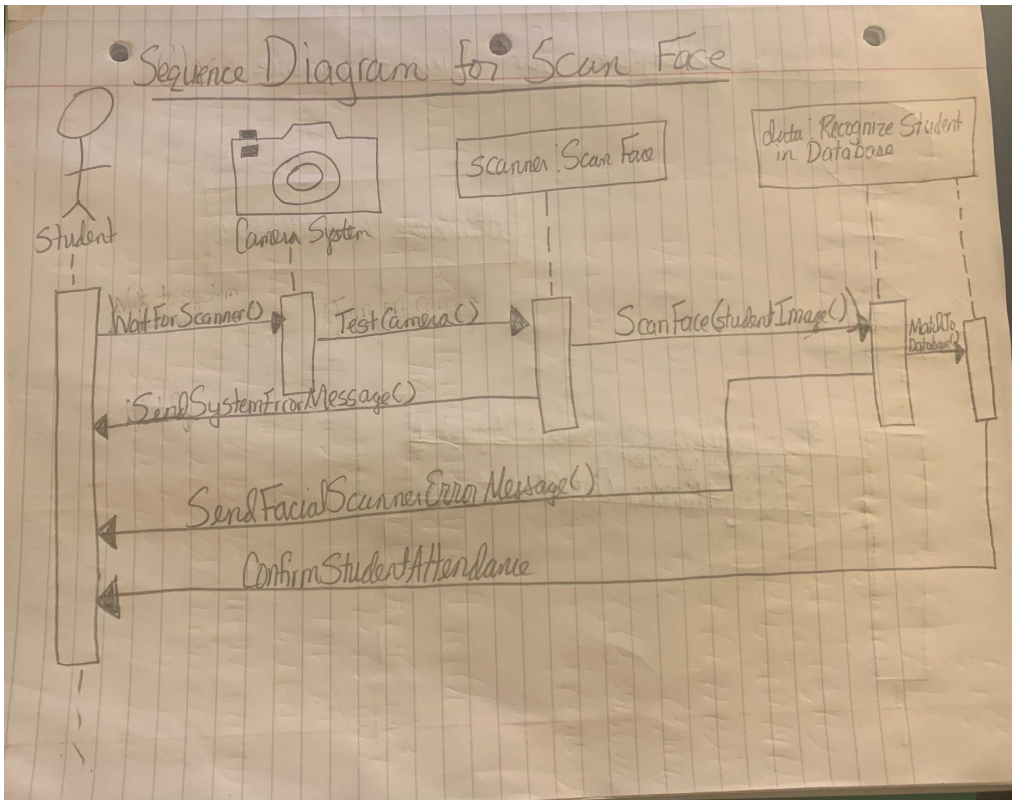
Class Name: Recognize student with image of them in the database	ID: 5	Type: Concrete
Description: The facial scanner will recognize the student with the prior image and use biometrics to analyze image.		Associated Use Cases: 0
Responsibilities 1) Allow system to be able to change reference image in database. 2) Allow for no other student to gain unauthorized access to the database. 3) Link with student's institution and blackboard. 4) Match student to their file in the database.		Collaborators 1. Update Attendance 2. Scan Face

Back of CRC Card:

Attributes: - Image - Student Name (text) - Student ID (int) - Database (SQL)
Relationships: Generalization (a-kind-of): Aggregation (has-parts): Update Attendance Other Associations: Scan Face

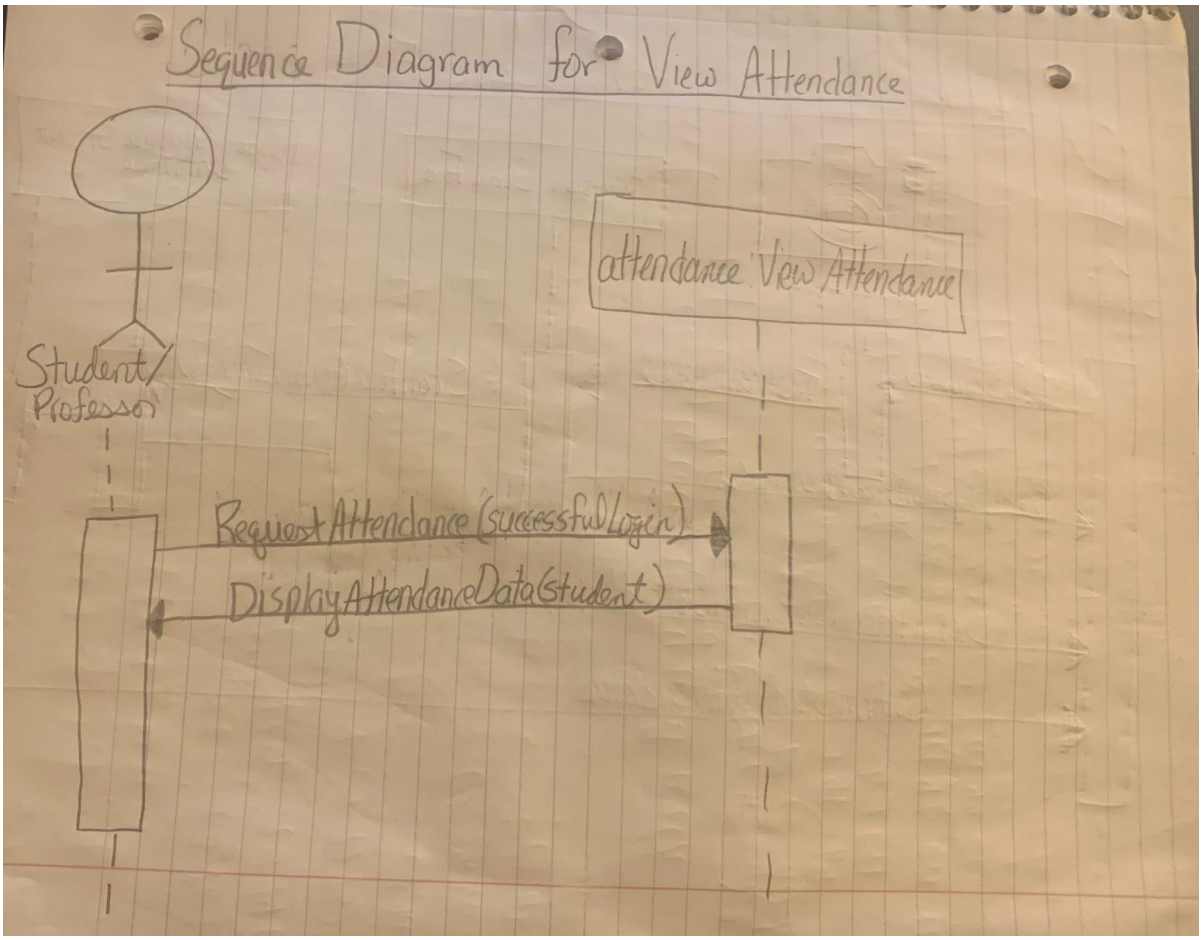
Sequence Diagrams:

1. Shown below is the sequence diagram for behaviors associated with the 'Scan Face' class. First a student will walk into the classroom and wait for the system. Next, the system will (given the presence of a student) test the system by turning on the camera and ensuring everything functions correctly. If this isn't the case, an error message will be sent to the student notifying them that the scanner isn't working. If the system is functioning correctly, the camera will start to scan for a face. Once successful, the face will be sent to the 'Recognize Student in Database' class, where it will be matched to the database of students before a confirmation message will be displayed to the student. If the student isn't recognized in the database, an error message will be displayed indicating that the student is not recognized in the system.

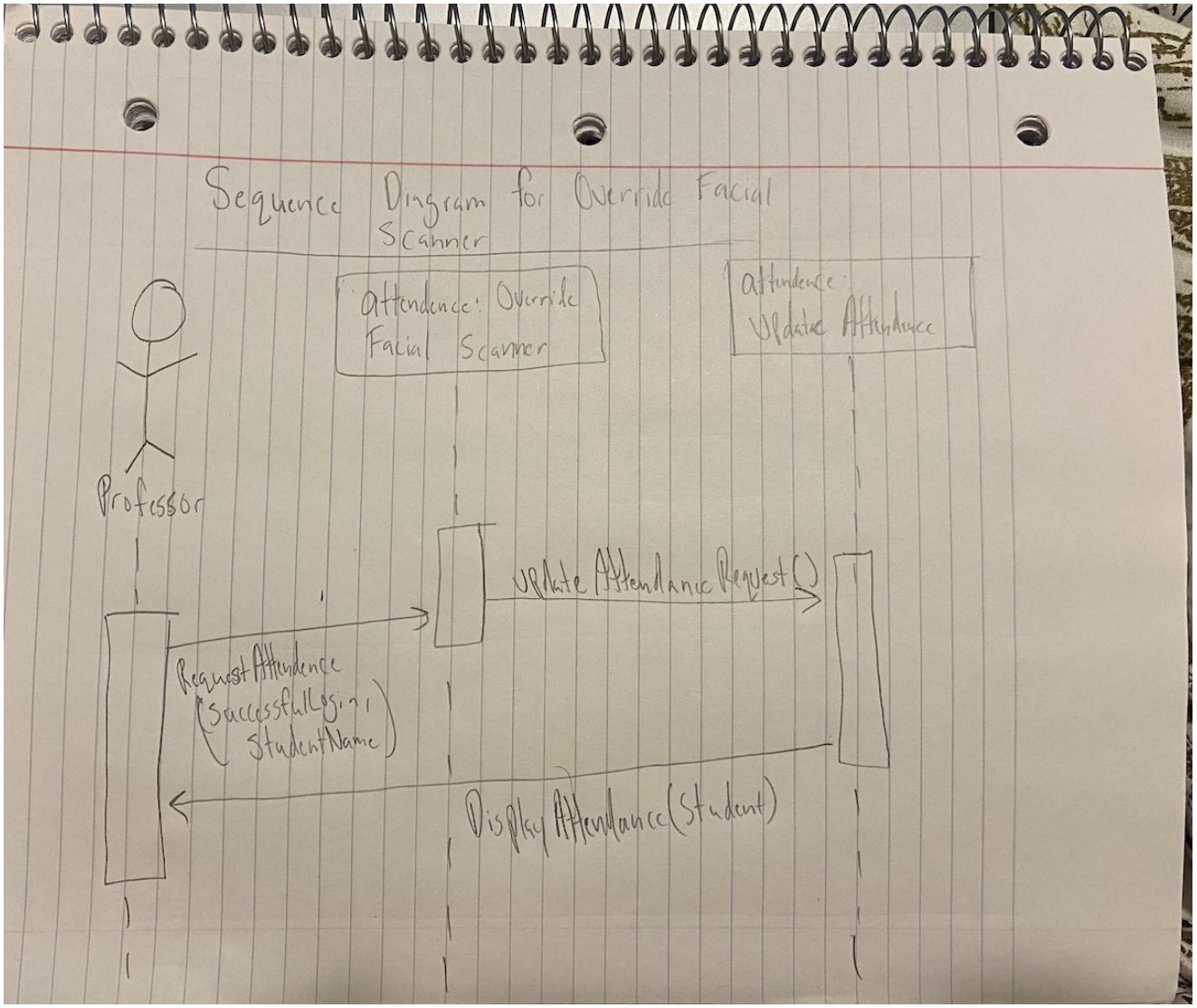


2. Shown below is the sequence diagram for behaviors associated with the 'View Attendance' class. Although this sequence diagram isn't complex, it plays a vital role in allowing administrators and students to access student attendance. The trigger for this diagram is the student or professor requesting to view their attendance, given a successful login to Blackboard for the appropriate course. Next, the class will send the most current attendance information

(total attendance score, total missed days & total excused absences) back to the user that requested it.

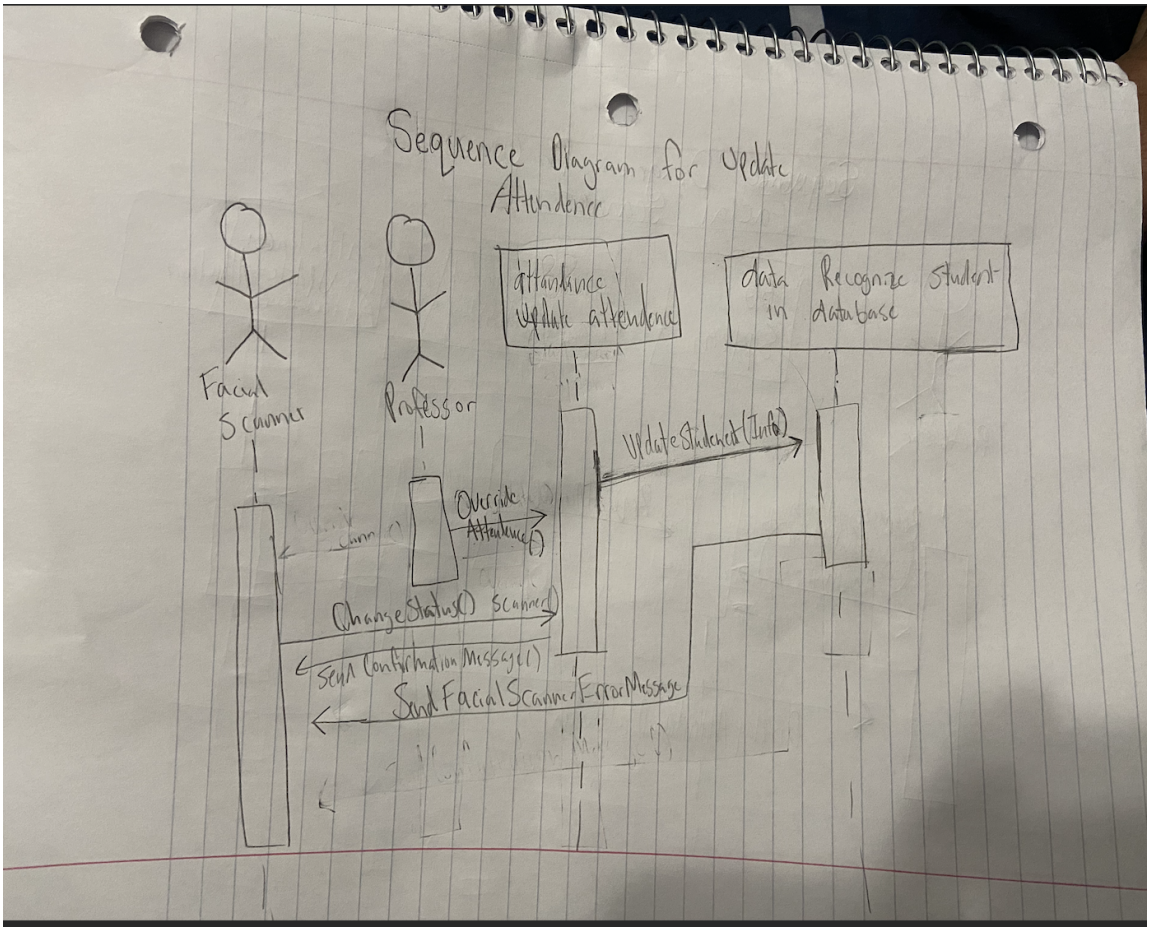


3. Shown below is the sequence diagram for behaviors associated with the 'Override Facial Scanner' class. This class is integrated with a more simple outline. Since this process is mainly how the professor can override the facial scanner, it is a simple relationship with the attendance: update attendance that helps the professor change attendance through a status and the update attendance will display the changes made. Additionally the professor signs into blackboard and requests attendance change(successful login, student name) and from there the override facial scanner class would send the updated information to the update attendance class.



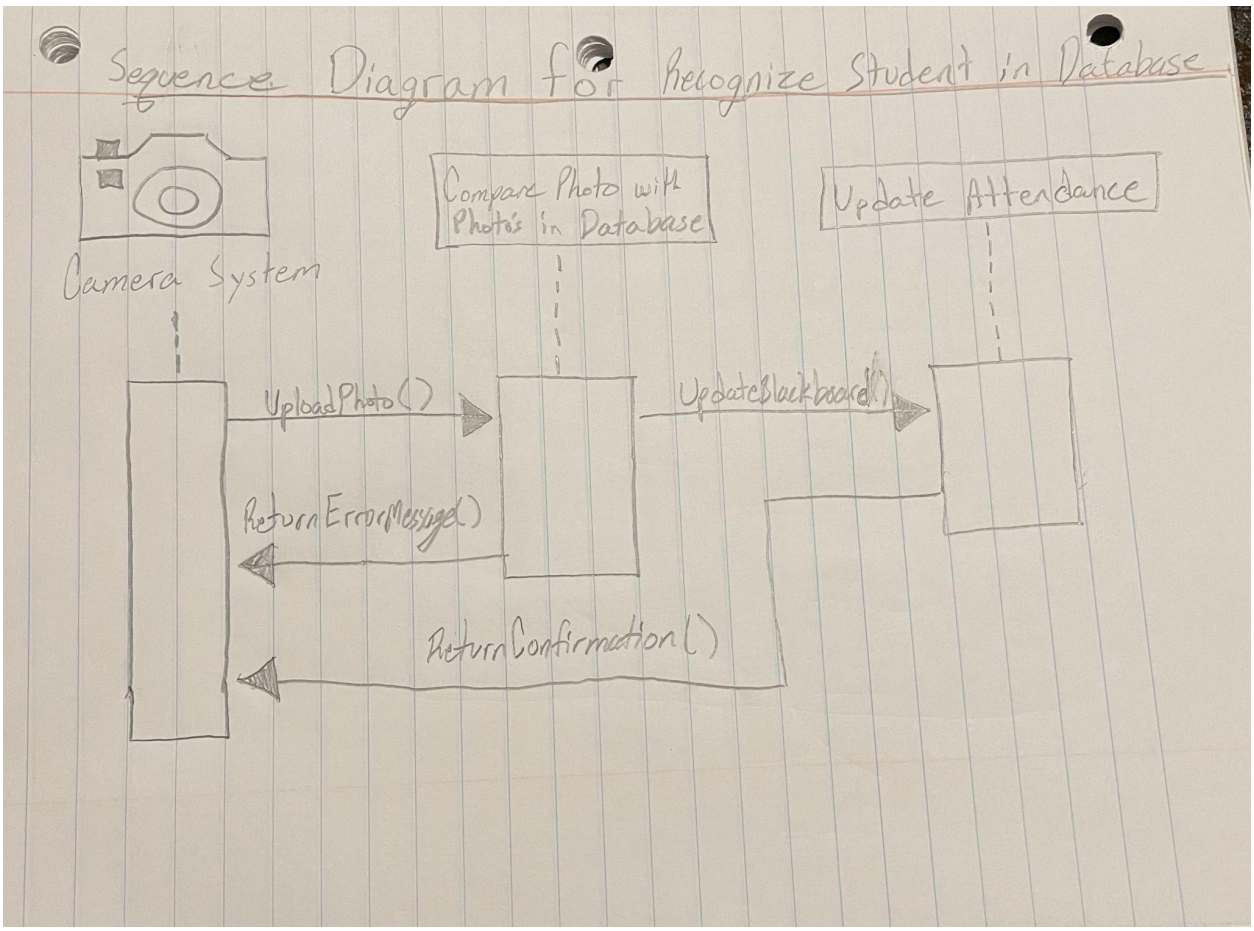
4. Shown below is the sequence diagram for behaviors associated with the 'Update Attendance' class. This class is more complex as it includes two actors and an object class for data recognizing students in the database. This diagram shows the relationship for the sequence of updating attendance between the facial scanner, professor, blackboard system and an object. When done correctly, the facial scanner and professor can change the status of students in the blackboard, where the blackboard system can update student information and recognize

students in the database. The database can send messages of invalid student scan and update attendance can send confirmation messages to the scanner.



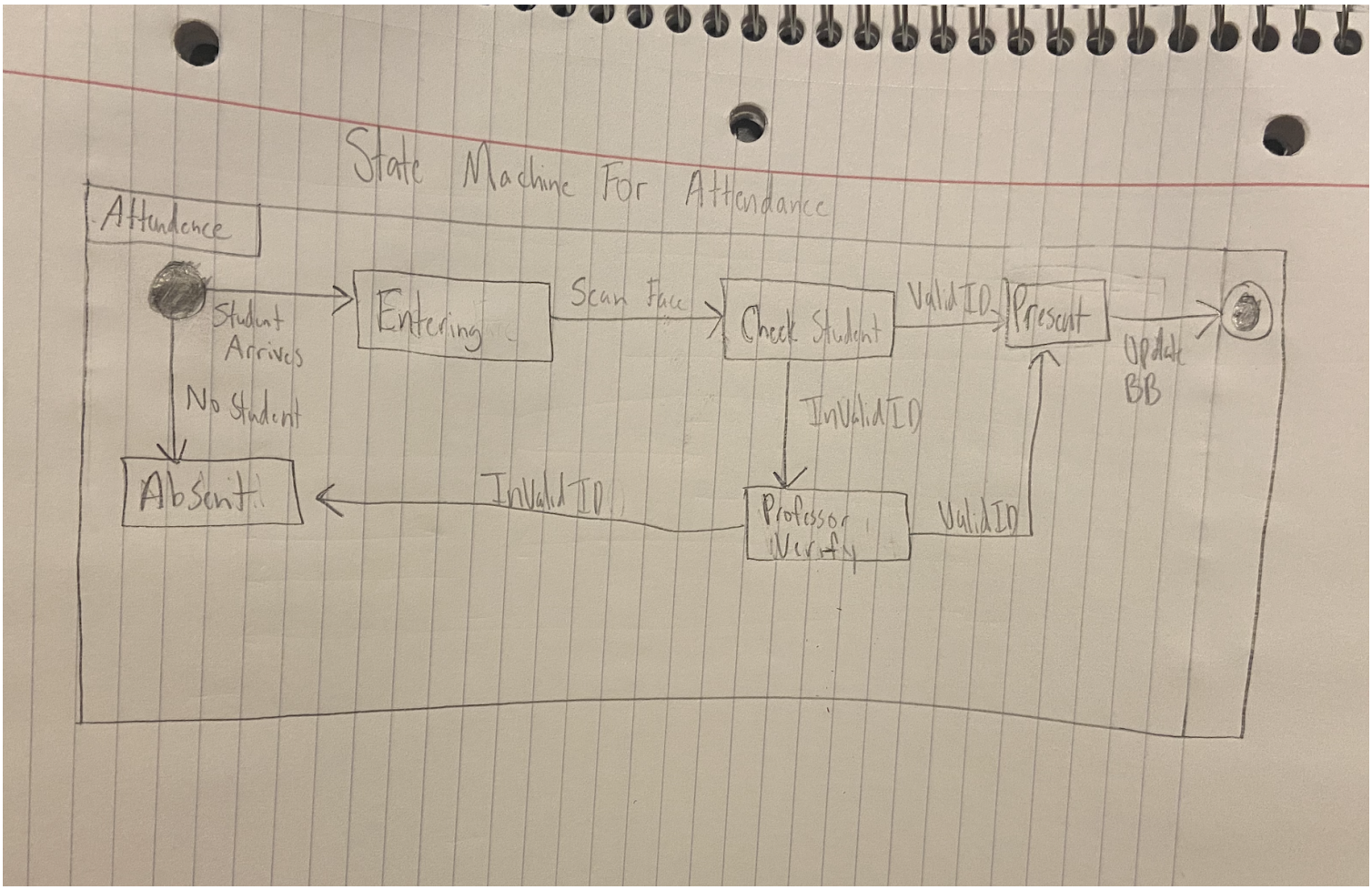
5. Shown below is the sequence diagram for behaviors associated with the ‘Recognize Student in Database’ class. This diagram shows the relationship of sequences for recognizing a student in the database between the camera system and Blackboard. The class is triggered when the camera takes a photo of a student. The system uploads teh photo to the database that has saved photos of all the students in class. It then begins to compare the photo with all the other photos until it finds a match or makes it through all the photos without finding a match. If a match is found then it updates attendance in blackboard and then returns a confirmation to

the camera system to let the student know he was marked present. If the system does not find a match then it returns an error message to the camera system to let the student know.



State Diagram:

State Machine for Attendance:



A state machine contains frames, initial state, events, transitions and the final state. The state machine above shows the different stages of the attendance instance class. It shows the states that validate if a student is present or absent. Following this, it updates to Blackboard which is described as its final state.

Crude Matrix:

	Student Actor	Professor Actor	Camera System Actor	Scan Face Class	View Attendance Class	Recognize Student in Database Class	Override Facial Scanner Class	Update Attendance Class
Student Actor					**R**			
Professor Actor	**C, R, D**				**R**		**U, D, E**	
Camera System Actor				**R, E**				
Scan Face Class	**R**					**R**		
View Attendance Class					**R**			
Recognize Student in Database Class	**C**			**R**				**U**
Override Facial Scanner Class								**U, D, E**
Update Attendance Class	**R**					**R**	**R**	**U, D**

The crude matrix above describes how the actors and classes collaborate with each other. The actors are the student, professor and camera system while the classes are for scan face, view attendance, recognize student in database, override facial scanner and update attendance. The matrix is read in row:column order, meaning that the item in the row performs an action (creates, reads, updates, deletes, executes) on the item in the corresponding column.