CS-223
SOFTWARE DESIGN DOCUMENT

for

Project 7
Classroom Visualisation App-1

Prepared by:
Group-12
Mitansh Jain - 160101042
Sujoy Ghosh - 160101073
Akul Agrawal - 160101085

March 15, 2018

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the design for the Classroom Visualization App software. This includes use case models, sequence diagrams, class diagrams and other supporting information. It will illustrate the complete declaration for the development of system along with the system constraints. This document is primarily intended to as a reference for developing the first version of the system for the development team.

1.2 Document Conventions

Term	Definition
Professor	Person who shall be using the software for monitoring
Student	Person who shall be monitored by the instructor
Users	Collectively refers to the professors or teaching assistants
Device	An electronic device using which the instructor is delivering their lecture
Android	A mobile operating system developed by Google
Google	An American multinational technology company
IDE	Integrated Development Environment
OpenCV	Open Source Computer Vision Library

Table 1.1: Document Conventions

1.3 Project Scope

The purpose of this project is to create convenient and easy-to-use android app for users. This app will enable professors to recognise the students present in the class and map them to their assigned states and augment a bounding box of particular color. The architecture of the app will allow it to be compatible to additional functionality for dynamic state allocation. The system is based on computer vision. Above all, we hope to provide a comfortable user experience along with best results.

2 Design Overview

2.1 Introduction

The Design Overview is the section to introduce and give a brief overview of the design. The System Architecture is a way to give the overall view of a system and to place it into context with external systems. This allows for the reader and user of the document to orient themselves to the design and see a summary before proceeding to the details of the design.

2.2 Background

This software will be a convenient and easy-to-use android application for the professors. This application will enable professors to recognize the students present in the class and map them to their assigned states, while augmenting a bounding box of particular color around their faces in real time. The architecture of the app will allow it to be compatible to additional functionality for dynamic state allocation.

2.3 System Architecture

The use case environment consists of a professor delivering lecture in a class. The professor is expected to add each student attending the lecture to the application database prior to the beginning of the lecture. He/She just switches on the application and opens the camera view facing the class. The professor is notified about the current states of the students in real-time with his/her corresponding roll number.

2.4 System Interfaces and Implementing Technologies

2.4.1 User Interfaces

The user interface will allow the professor to login. He would be able to add, edit and delete students attending his/her lecture. He will get the real-time statistics of the attention of the students.

2.4.2 Software Interfaces

The data collected will be stored in a relational MySQL database. The application will be written in Android Studio IDE and OpenCV package will be used to process images

and recognize faces of the students.

2.5 Design and Implementation Constraints

For the above purpose of implementing the app, we are forbidden from using the camera for measuring the states of the students. The students might portray themselves as paying attention even when they are not.

2.6 Assumptions and Dependencies

We assume that the device owned by the professor will be based on the Android Operating System. One assumption about the product is that it can always be used on mobile phone having good enough performance and camera. If the phone does not have enough hardware resources available for the application, for example the users might have allocated them with other applications, there may be scenarios where the application does not work as intended or even at all.

3 Use Cases

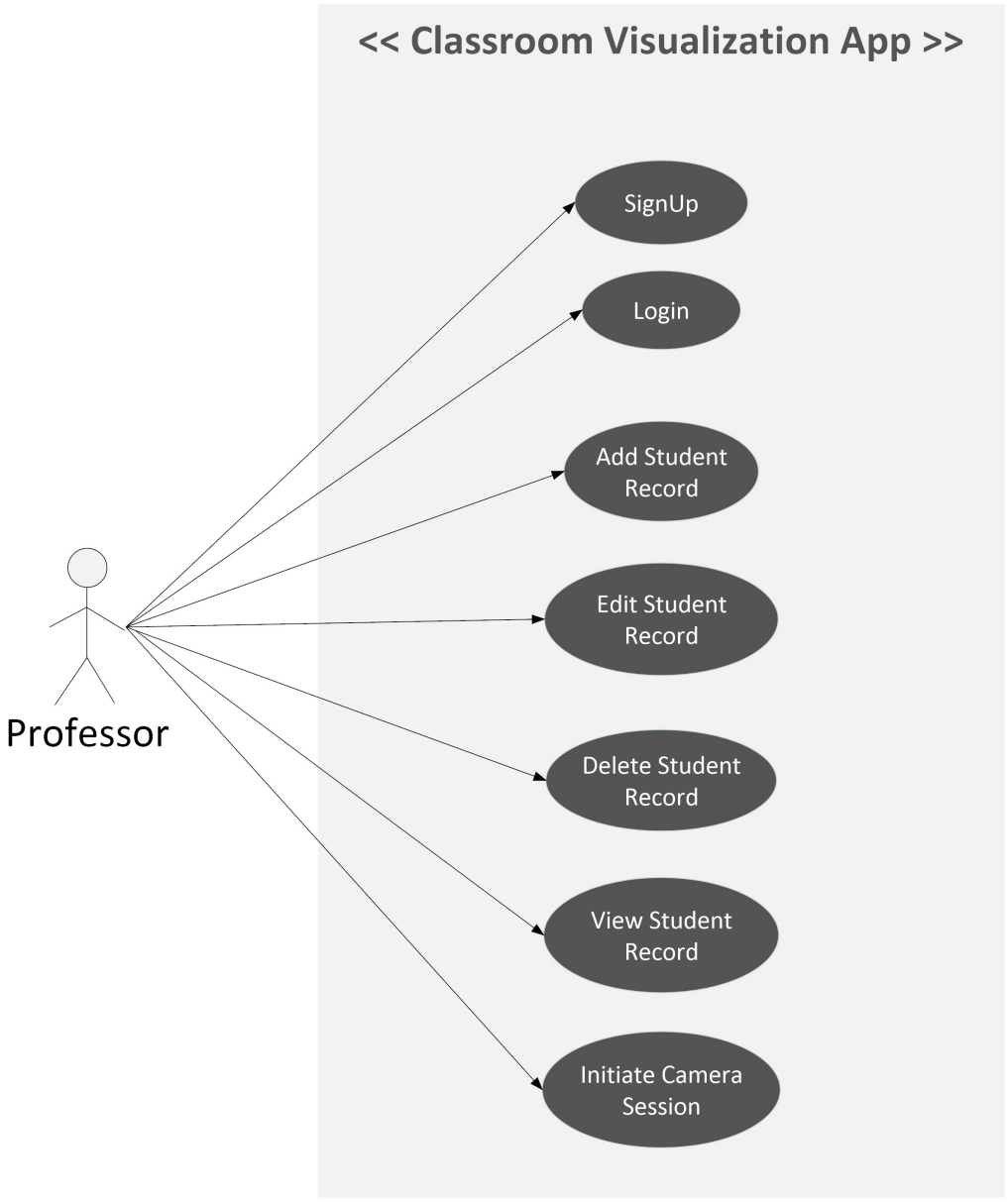

Figure 3.1: Use case diagram

3.1 U1: Sign Up

- **Actors:** Professor
- **Precondition :** The user should not have signed up already.
- **Scenario 1:** Mainline Sequence:
    1. **Professor:** Requests the register page.
    2. **System:** Prompts to enter details.
    3. **Professor:** Enters the details i.e. username and password.
    4. **System:** Displays acknowledgement for successful registration.
- **Scenario 2:** At step 4 in mainline:

    4. **System:** Displays error in case any required input field is empty and prompts to enter those details.

3.2 U2: Log In

- **Actors:** Professor
- **Precondition :** The user must have signed up already.
- **Scenario 1:** Mainline Sequence:
    1. **Professor:** Requests the login page.
    2. **System:** Prompts to enter username and password.
    3. **Professor:** Enters the creadentials i.e. username and password.
    4. **System:** Displays login acknowledgement and available options.
- **Scenario 2:** At step 4 in mainline:

    4. **System:** Displays error in case any required input field is left empty and prompts to enter those details.
- **Scenario 3:** At step 4 in mainline:

    4. **System:** Displays error in case of invalid credentials.

3.3 U3: Add Student Record

- **Actors :** Professor
- **Precondition :** The user must have logged in with his credentials.
- **Scenario 1 :** Mainline Sequence:

1. **Professor :** Selects 'Add Student' option.

2. **System :** Displays prompt to enter student details including name, roll number and course ID.

3. **Professor :** Enters the required values and selects 'Next' option.

4. **System :** Prompts user to add 20 images of the new student from camera.

5. **Professor :** Adds the required number of images of the new student.

6. **System :** Displays acknowledgement message for addition of new student.

- **Scenario 2 :** At step 4 of mainline sequence:

  4. **System :** Displays error if the student is already present.

- **Scenario 3 :** At step 4 of mainline sequence:

  4. **System :** Displays error if some of the required data has not been entered and prompts to enter those data.

- **Scenario 4 :** At step 4 of mainline sequence:

  4. **System :** Displays error if entered data(Roll number and/or course Id) is not valid.

3.4 U4: Edit Student Record

- **Actors :** Professor
- **Precondition :** The user must have logged in with his credentials.
- **Scenario 1 :** Mainline Sequence:

  1. **Professor :** Selects 'Update Student Records' option.

  2. **System :** Displays prompt to enter student's roll number.

  3. **Professor :** Enters the required value and selects 'next'

  4. **System :** Prompts user to enter new details of the student.

  5. **Professor :** Enters the new details of the student and selects 'Complete Editing' option.

  6. **System :** Displays acknowledgement message for successful editing of student's data.

- **Scenario 2 :** At step 4 of mainline sequence:

  4. **System :** Displays error if the student's roll number is absent in database or invalid.

- **Scenario 3 :** At step 6 of mainline sequence:

6. System : Displays error if entered data is not valid.

- **Scenario 4 :** At step 6 of mainline sequence:

  6. System : Prompts user to capture 20 photos of the student.

  7. Professor : Captures the required number of images.

  8. System : Displays acknowledgement message for successful editing of student's data.

3.5 U5: Delete Student Record

- **Actors :** Professor
- **Precondition :** The user must have logged in with his credentials.
- **Scenario 1 :** Mainline Sequence:

  1. Professor : Selects 'Delete Student' option.

  2. System : Displays prompt to enter the roll number of student to be deleted.

  3. Professor : Enters the roll number of the student to be deleted.

  4. System : Displays acknowledgement message regarding deletion of the student.

- **Scenario 2 :** At step 4 of mainline sequence:

  4. System : Displays error if the student's roll number is absent in the database or invalid.

3.6 U6: View Student Record

- **Actors :** Professor
- **Precondition :** The user must have logged in with his credentials.
- **Scenario 1 :** Mainline Sequence:

  1. Professor : Selects 'View Student Record' option.

  2. System : Displays prompt to enter course ID.

  3. Professor : Enters the required value.

  4. System : Displays list of students enrolled in the given course ID along with their roll numbers.

- **Scenario 2 :** At step 4 of mainline sequence:

  4. System : Displays error if the requested 'Course ID' is invalid/absent in database.

3.7 U7: Initiate Camera Session

- **Actors :** Professor
- **Precondition :** The user must have logged in with his credentials.
- **Scenario 1 :** Mainline Sequence:

  1. Professor : Selects 'Camera View' option.

  2. System : Initiates the rear camera view.

  3. Professor : Selects the 'Start' option.

  4. System : Displays bounding box (coloured according to the allocated state) around the recognized faces.

4 Object Oriented Model

4.1 Domain Modeling

4.1.1 U1 SignUp:

4.1.1.1 Entity Objects

1. UserDB

4.1.1.2 Controller Objects

1. SignUpController

4.1.1.3 Boundary Objects

1. SignUpBoundary

4.1.2 U2 Login:

4.1.2.1 Entity Objects

1. UserDB

4.1.2.2 Controller Objects

1. LogInController

4.1.2.3 Boundary Objects

1. LogInBoundary

4.1.3 U3 Add Student Records:

4.1.3.1 Entity Objects

1. StudentRecords

4.1.3.2 Controller Objects

1. AddStudentController

4.1.3.3 Boundary Objects

1. AddStudentBoundary

4.1.4 U4 Edit Student Records:

4.1.4.1 Entity Objects

1. StudentRecords

4.1.4.2 Controller Objects

1. EditStudentController

4.1.4.3 Boundary Objects

1. EditStudentBoundary

4.1.5 U5 Delete Student Records

4.1.5.1 Entity Objects

1. StudentRecords

4.1.5.2 Controller Objects

1. DeleteStudentController

4.1.5.3 Boundary Objects

1. DeleteStudentBoundary

4.1.6 U7 View Student Records

4.1.6.1 Entity Objects

1. StudentRecords

4.1.6.2 Controller Objects

1. ViewStudentController

4.1.6.3 Boundary Objects

1. ViewStudentBoundary

4.1.7 U7 Inititate Camera Session

4.1.7.1 Entity Objects

1. StudentRecords

4.1.7.2 Controller Objects

1. CameraViewController

4.1.7.3 Boundary Objects

1. CameraView

4.2 Class Description

4.2.1 SignUpBoundary:

Description: This class handles the user interface for SignUp form.

Attributes:

- Username: string
- EmailID: string
- Password: string
- isValid: integer

Method:

- **Constructor()**

  Parameters: Void

  Return Value: Form window instance

  Description: This method loads the signup window onto screen.

  Called by: This method is called by ClickEventListener() of Signup button.

  Calls: void

- **ClickEventListener()**

  Parameters: buttonAction, functionName

  Return Value: void

  Description: This method calls the function provided in the parameter's list.

  Called by: This method is called on click of a button.

  Calls: Calls the function provided in the parameter's list.

- **FormSubmit()**

  Parameters: Username, EmailID, Password

  Return Value: void

  Description: This method accepts the input from the signup window and passes on for validation process.

  Called by: Method is called by 'Submit' ClickEventListener() Handler.

  Calls: Method calls ValidateFormData() from SignUpController object and then calls DisplayMessage() method.

- **DisplayMessage()**

  Parameters: Truth value of isValid parameter.

  Return Value: Displays the Confirmation message or Error message depending upon value of isValid.

  Description: This method checks the value of isValid parameter and displays corresponding message onto the screen. If isValid = 1, then confirmation message is displayed. If isValid = 0, then "Incomplete Form" error will be displayed. If isValid = -1, then "Data not valid" is displayed

  Called by: Method is called by FormSubmit().

  Calls: This function makes no further calls.

4.2.2 SignUpController

Description: This class checks the validity of data entered by the Professor in the Sign Up Form and creates a new user in the database with the input data.

Methods:

- **ValidateFormData()**

  Parameters: Username, EmailID, Password

  Return Value: isValid value

  Description: This method checks if all the input parameters are non-empty and all data are as per required format (e.g. email-id should be valid).

  Called by: Method called be FormSubmit() method of SignUpBoundary object.

  Calls: It calls CreateUser() of UserDB object if input data is valid.

4.2.3 LoginBoundary:

Description: This class handles the user interface for Login.

Attributes:

- Username: string
- Password: string

Method:

- **Constructor()**

  Parameters: Void

  Return Value: Form window instance

  Description: This method loads the Add Student window on the screen.

  Called by: This method is called by ClickEventListener() of Login button.

  Calls: No further calls.

- **ClickEventListener()**

  Parameters: buttonAction, functionName

  Return Value: void

  Description: This method calls the function provided in the parameter's list.

  Called by: This method is called on click of a button.

  Calls: Calls the function provided in the parameter's list.

- **FormSubmit()**

  Parameters: Username, Password

  Return Value: void

  Description: This method accepts the input from the login window and passes on for validation process.

  Called by: Method is called by 'Submit' ClickEventListener() Handler.

  Calls: Method calls ValidateFormData() from LogInController object and then calls DisplayMessage() method.

- **DisplayMessage()**

  Parameters: Truth value of isValid parameter.

  Return Value: Displays the Confirmation message or Error message depending upon value of isValid.

  Description: This method checks the value of isValid parameter and displays corresponding message onto the screen.

  Called by: Method is called by FormSubmit()

  Calls: No further calls.

4.2.4 LoginController:

Description: This class checks the validity of the credentials entered by the user and allows access to the main portal if they are valid.

Method:

- **ValidateFormData()**

  Parameters: Username, Password

  Return Value: IsValid

  Description: This method accepts the username and password entered by the user and checks if the details are valid.

  Called by: It is called after Form Submission FormSubmit().

  Calls: Verify() function of UserDB

4.2.5 UserDB:

Description: This class helps to verify the credentials entered by the professor and create a new student entry in database.

Attributes:

- Username: string
- Password: string
- Email: string

Methods:

- **CreateUser()**

  Parameters: Username, Password, Email

  Return Value: void

  Description: This method stores the values of user credentials.

  Called by: Method is called by ValidateFormData() of SignUp class if data provided is valid

  Calls: This method has no further calls.

- **Verify()**

  Parameters: Username, Password.

  Return Value: Boolean value that is true if user credentials matches with the correct user credentials, else displays false.

  Description: This method checks the parameter password with the correct password of the username.

  Called by: Method is called by ValidateFormData() of Login class

  Calls:

4.2.6 AddStudentBoundary:

Description: This class handles add student form and interaction with user.

Attributes:

- StudentName: string
- StudentRollNo: string
- StudentCourse: string

Methods:

- **Constructor()**

  Parameters: Void

  Return Value: Form window instance

  Description: This method loads the Add Student window on the screen.

  Called by: This method is called by ClickEventListener() of Login button.

  Calls: Method calls FormSubmit() method

- **ClickEventListener()**

  Parameters: buttonAction, functionName

  Return Value: void

  Description: This method calls the function provided in the parameter's list.

  Called by: This method is called on click of a button.

  Calls: Calls the function provided in the parameter's list.

- **FormSubmit()**

  Parameters: StudentName, StudentRollNo, StudentCourse

  Return Value: Void

  Description: This method accepts the input from the login window and passes on for validation process.

  Called by: Method is called by ClickEventListener() on clicking form submit button.

  Calls: Method calls ValidateFormData() method and then calls DisplayMessage().

- **DisplayMessage()**

  Parameters: Truth value of isValid parameter.

  Return Value: Displays the Confirmation message or Error message depending upon value of isValid.

Description: This method checks the value of isValid parameter and displays corresponding message onto the screen.

Called by: Method is called by FormSubmit()

Calls: No further calls.

4.2.7 AddStudentController:

Description: This class is responsible for the control of AddStudent process.

Methods:

- **ValidateFormData()**

  Parameters: AddStudentFormData

  Return Value: IsValid

  Description: This method checks if the data entered by the user is valid.

  Called by: FormSubmit()

  Calls: OpenCameraView() if the form is valid(i.e. IsValid=1).

- **OpenCameraView()**

  Parameters: SwitchSignal(i.e. camera on/off command)

  Return Value: CameraView

  Description: This method switches on the camera view to take photographs depending on the SwitchSignal

  Called by: Method is called by ValidateFormData()

  Calls: SwitchCamera()

4.2.8 CameraView:

Description: This class controls the Camera and returns boolean truth value when user has taken sufficient(20) images.

Attributes:

- SwitchSignal: Boolean
- ImageCounter: Integer

Methods:

- **SwitchCamera()**

  Parameters: SwitchSignal

  Return Value: ImageNum

  Description: This method switches on the camera whenever the SwitchSignal is one and counts number of images taken by the user

Called by: OpenCameraView()

Calls: CheckImageNum()

- **CheckImageNum()**

  Parameters: ImageNum

  Return Value: IsValid

  Description: This method takes number of images clicked by the user as input and checks if it satisfies the minimum requirement of 20 images.

  Called by: Method is called by SwitchCamera()

  Calls: AddStudentList() when sufficient number of images(20) has ben captured

4.2.9 StudentRecords:

Description: This class maintains the database of students, their details and images.

Attributes:

- StudentName: String
- StudentRollNumber: Integer
- CourseID: string
- ImageList[]: Images
- StudentList[] : [StudentName:string, StudentRollNumber:integer, CourseID:string , ImageList[]:images]

Methods:

- **AddStudentList()**

  Parameters: StudentData i.e StudentName, StudentRollNumber and CourseID.

  Return Value: boolean True if data is successfully addded, else it is False.

  Description: This method checks if the input RollNo is already present in the list of added students. If not, it creates a new student in StudentList and saves his data in StudentRecord. Status of success is returned.

  Called by: CheckImageNum()

  Calls: CheckRollNo()

- **CheckRollNo()**

  Parameters: RollNo

Return Value: boolean True if roll number is present in the list, else it is False.

Description: This method checks if the input RollNo is already present in the list of added students.

Called by: AddStudentList() from AddStudentController and ValidateRollNumber() from EditStudentController and DeleteStudentController

Calls: No further calls

- **EditDetails()**

  Parameters: StudentRollNumber, StudentName and CourseID

  Return Value: Confirmation message

  Description: This method updates the values of the student corresponding to the given roll number.

  Called by: ValidateData()

  Calls: No further calls

- **StoreImage()**

  Parameters: StudentRollNumber and list of images

  Return Value: Confirmation message

  Description: This method updates the training images of the student corresponding to the given roll number.

  Called by: CheckImgNum()

  Calls: No further calls

- **DeleteRecord()**

  Parameters: StudentRollNumber

  Return Value: Confirmation message

  Description: This method deletes the values of the student corresponding to the given roll number.

  Called by: ValidateRollNumber()

  Calls: No further calls

- **GetStudentsList()**

  Parameters: CourseID

  Return Value: List of students registered in the course.

  Description: This method returns the list of students registered in the given course.

  Called by: RequestStudentRecords()

  Calls: No further calls

- **MatchFaceFeatures()**

  Parameters: FeatureMatrix[]

  Return Value: list of students matched with the features.

  Description: This method checks the StudentsList and returns the list of students matched with the features.

  Called by: FeatureExtraction()

  Calls: No further calls

- **GetState()**

  Parameters: StudentRollNumber

  Return Value: StateValue

  Description: This method checks the StudentsList and returns the StateValue corresponding to given StudentRollNumber.

  Called by: AugmentBoundingBox()

  Calls: No further calls

- **SetState()**

  Parameters: List containing student roll number and the corresponding state value. StudentState[] : array of [StudentRollNumber, StateValue].

  Return Value: void

  Description: This method updates the states of the student according to the input list.

  Called by: Method is called by Generate().

  Calls: No further calls

4.2.10 EditStudentBoundary:

Description: This class helps to edit student details such as student name and course name.

Attribute:

- isValidRollNumber : integer
- isValidInput : boolean

Methods:

- **Constructor()**

  Parameters: void

  Return Value: Form window instance.

  Description: This method loads the edit student windown onto screen.

Called by: Method is called by ClickEventListener() Handler of Edit Student button.

Calls: This method doesn't make any further calls.

- **ClickEventListener()**

  Parameters: buttonAction, functionName

  Return Value: void

  Description: This method calls the function provided in the parameter's list.

  Called by: This method is called on click of a button.

  Calls: Calls the function provided in the parameter's list.

- **InputRollNumber()**

  Parameters: Roll number of the student.

  Return Value: void

  Description: This method takes the roll number as input whose details are to be editted.

  Called by: Method is called by ClickEventListener() Handler of Submit button.

  Calls: It calls ValidateRollNumber() of EditStudentController object to validate the data and then calls either DisplayError() or EditDetailsInput() based on the value of isValidRollNumber.

- **EditDetailsInput()**

  Parameters: Student name and course ID.

  Return Value: void

  Description: This method takes the new student name and course id as input and makes changes corresponding to the roll number in database.

  Called by: Method is called by InputRollNumber().

  Calls: This method calls the ValidateData() of EditStudentController object. It then calls DisplayConfirmation() or DisplayError() function based on the value of isValidRollNumber.

- **DisplayError()**

  Parameters: void

  Return Value: Error message based on the value of isValidRollNumber and isValidInput.

Description: This methods displays error message based on the value of isValidRollNumber. If isValidRollNumber = -1, then 'Roll number doesn't exist' message is displayed. If isValidRollNumber = 0, then 'Incomplete form is displayed'. If isValidInput = 0, then 'Invalid Form' is displayed.

Called by: Called either by inputRollNumber() or EditDetailsInput() based on various scenarios as discussed.

Calls: No further calls.

- **DisplayConfirmation()**

  Parameters: void

  Return Value: Confirmation message

  Description: This methods displays confirmation message indicating successful completion of editing.

  Called by: Method is called by EditDetailsInput()

  Calls: No further calls.

4.2.11 EditStudentController:

Description : This class helps to validate roll number and helps to edit student details such as student name and course name.

Attribute :

- rollNumber : integer
- isDataValid : boolean
- isFormValid : boolean

Methods:

- **ValidateRollNumber()**

  Parameters : rollNumber

  Return Value : isDataValid

  Description : This function checks the validity of entered roll number and asks for the presence of roll number in the students' list. If roll number is present, then isDataValid = 1, else isDataValid = 0.

  Called by : Method is called by InputRollNumber()

  Calls : It calls CheckRollNo() method of StudentList object.

- **ValidateData()**

  Parameters : StudentName and CourseID

  Return Value : isFormValid

Description : This function checks the validity of entered details of the student. If all entries are valid (i.e. course ID belongs to what the professor teaches,etc) , then isFormValid = 1, else isFormValid = 0.

Called by : Method is called by EditDetailsInput().

Calls : If isFormValid = 1, then it calls SwitchSignal() asking if the user wants to add new training images. It then calls EditDetails() of StudentList object.

4.2.12 DeleteStudentBoundary

Description: This class helps to delete student details given the roll number.

Attribute:

- isValidRollNumber : integer
- isValidInput : boolean

Methods:

- **Constructor()**

  Parameters: Void

  Return Value: Form window instance.

  Description: This method loads the delete student windown onto screen.

  Called by: Method is called by ClickEventListener() Handler of Edit Student button.

  Calls: This method doesn't make any further calls.

- **InputRollNumber()**

  Parameters: Roll number of the student.

  Return Value: void

  Description: This method takes the roll number as input whose details are to be editted.

  Called by: Method is called by ClickEventListener() Handler of Submit button.

  Calls: It calls ValidateRollNumber() of DeleteStudentController object to validate the data and then calls either DisplayError() or DisplayConfirmation() based on the value of isValidRollNumber and isValidInput.

- **ClickEventListener()**

  Parameters: buttonAction, functionName

  Return Value: void

Description: This method calls the function provided in the parameter's list.

Called by: This method is called on click of a button.

Calls: Calls the function provided in the parameter's list.

- **DisplayError()**

  Parameters: void

  Return Value: Error message based on the value of isValidRollNumber and isValidInput.

  Description: This methods displays error message based on the value of isValidRollNumber. If isValidRollNumber = 0, then 'Roll number doesn't exist' message is displayed. If isValidInput = 0, then 'Invalid Form' is displayed.

  Called by: Called by InputRollNumber() depending upon isValidInput and isValidRollNumber values.

  Calls: Display module of window screen.

- **DisplayConfirmation()**

  Parameters: void

  Return Value: Confirmation message

  Description: This methods displays confirmation message indicating successful completion of editing.

  Called by: Method is called by InputRollNumber() depending upon isValidInput and isValidRollNumber values.

  Calls: No further calls.

4.2.13 DeleteStudentController

Description : This class helps to validate roll number and helps to delete student details having given roll number.

Attribute :

- isDataValid : boolean
- isFormValid : boolean

Methods:

- **ValidateRollNumber()**

  Parameters : rollNumber

  Return Value : isDataValid

Description : This function checks the validity of entered roll number and asks for the presence of roll number in the students' list. If roll number is present, then isDataValid = 1, else isDataValid = 0.

Called by : Method is called by InputRollNumber()

Calls : It calls CheckRollNo() method of StudentList object and if roll number is present it further calls DeleteRecord() of StudentRecords class.

4.2.14 ViewRecordBoundary

Description : This shows the requested student records from database

Attribute :

- listOfStudent[Name, RollNumber, lastState] : [string, integer, integer]

Methods:

- **Constuctor()**

  Parameters : CourseID

  Return Value : void

  Description : This function builds the instantiates the screen for viewing student records.

  Called by : clickEventListener() of button clicked by user for viewing records.

  Calls : Calls RequestStudentRecords() from ViewRecordController.

- **ClickEventListener()**

  Parameters: buttonAction, functionName

  Return Value: void

  Description: This method calls the function provided in the parameter's list.

  Called by: This method is called on click of a button.

  Calls: Calls the function provided in the parameter's list.

4.2.15 ViewRecordController

Description : Gets all student records from database.

Methods:

- **RequestStudentRecords()**

  Parameters : void

  Return Value : Student Records

Description : Request student records from studentList.

Called by : Constructor() calls just after initiation of screen.

Calls : calls GetStudentList() from studentList.

4.2.16 CameraSessionBoundary:

Description: This class handles the view from camera and augmentation of bounding box to identified students.

Attribute

- rearCameraView : mp4

Method:

- **Constructor()**

  Parameters: Void

  Return Value: Camera Instance

  Description: This function starts rear camera instance.

  Called by: This is called by ClickEventListener() handler of the button clicked by user.

  Calls: ConvertVideoType() method of CameraSessionController.

- **ClickEventListener()**

  Parameters: buttonAction, functionName

  Return Value: void

  Description: This method calls the function provided in the parameter's list.

  Called by: This method is called on click of a button.

  Calls: Calls the function provided in the parameter's list.

- **AugmentBoundingBox()**

  Parameters: Coordinates of identified students, their identity, states.

  Return Value: Video with augmented bounding box.

  Description: This method augments bounding box to video according to the states to the face of identified students.

  Called by: This is called in sequence after CameraSessionController class after states are obtained from databse.

  Calls: GetState().

Term	Definition
Red	1 to 4
Blue	5 to 7
Green	8 to 10

Table 4.1: Color Code

4.2.17 CameraSessionController:

Description: This class controls camera session, and converts format of video.

Attribute

- imageFramesList[] : 2d Array of integers representing RGB values.

Method:

- **ConvertVideoType()**

  Parameters: Rear Camera Video

  Return Value: Converted video in OpenCV format.

  Description: This function converts video into format processble by OpenCV.

  Called by: This is called by camera session after it is initiated.

  Calls: This further calls FrameGenerator() to obtain discrete image frames for processing.

4.2.18 FrameGenerator:

Description: This function generates discrete image frames from video.

Method:

- **GenerateFrames()**

  Parameters: Video in OpenCV format.

  Return Value: Discrete Image Frames.

  Description: This function generates discrete image frames.

  Called by: This is called by ConvertVideoType() after generation of video.

  Calls: calls IdentifyStudents().

4.2.19 FaceRecognizer:

Description: This recognizes the students present in image frames.

Attribute:

- identifiedStudentsList[Name, RollNumber]:[string, integer]

Method:

- **IdentifyStudents()**

  Parameters: ImageFrame

  Return Value: List of identified students

  Description: This function calls to oter functions which preprocess the image frames and identify features.

  Called by: Called by GenerateFrames().

  Calls: This calls FaceDetection() function, FeatureExtraction function and matches faces from database.

- **FaceDetection()**

  Parameters: ImageFrame

  Return Value: Image frame and coordinates of detected faces based on trained models.

  Description: This uses trained models to detects faces.

  Called by: IdentifyStudents() method

  Calls: No further calls.

- **FeatureExtraction()**

  Parameters: Coordinates of detected faces and image frames

  Return Value: Coordinates and information about extracted face features.

  Description: This function calls to other functions which preprocess the image frames

  Called by: Called by IdentifyStudentMethod.

  Calls: MatchFaceFeatures()

4.2.20 RandomStateGenerator:

Description: This function generates randomstates for given set of identified student list.

Method:

- **Generate()**

  Parameters: List of identified students.

  Return Value: Random Numbers

  Description: This function generates discrete image frames.

Called by: This is called by CameraSessionController().

Calls: This function calls SetState() function in student List to further set states in database for further augmentation purposes.

5 Sequence Diagram

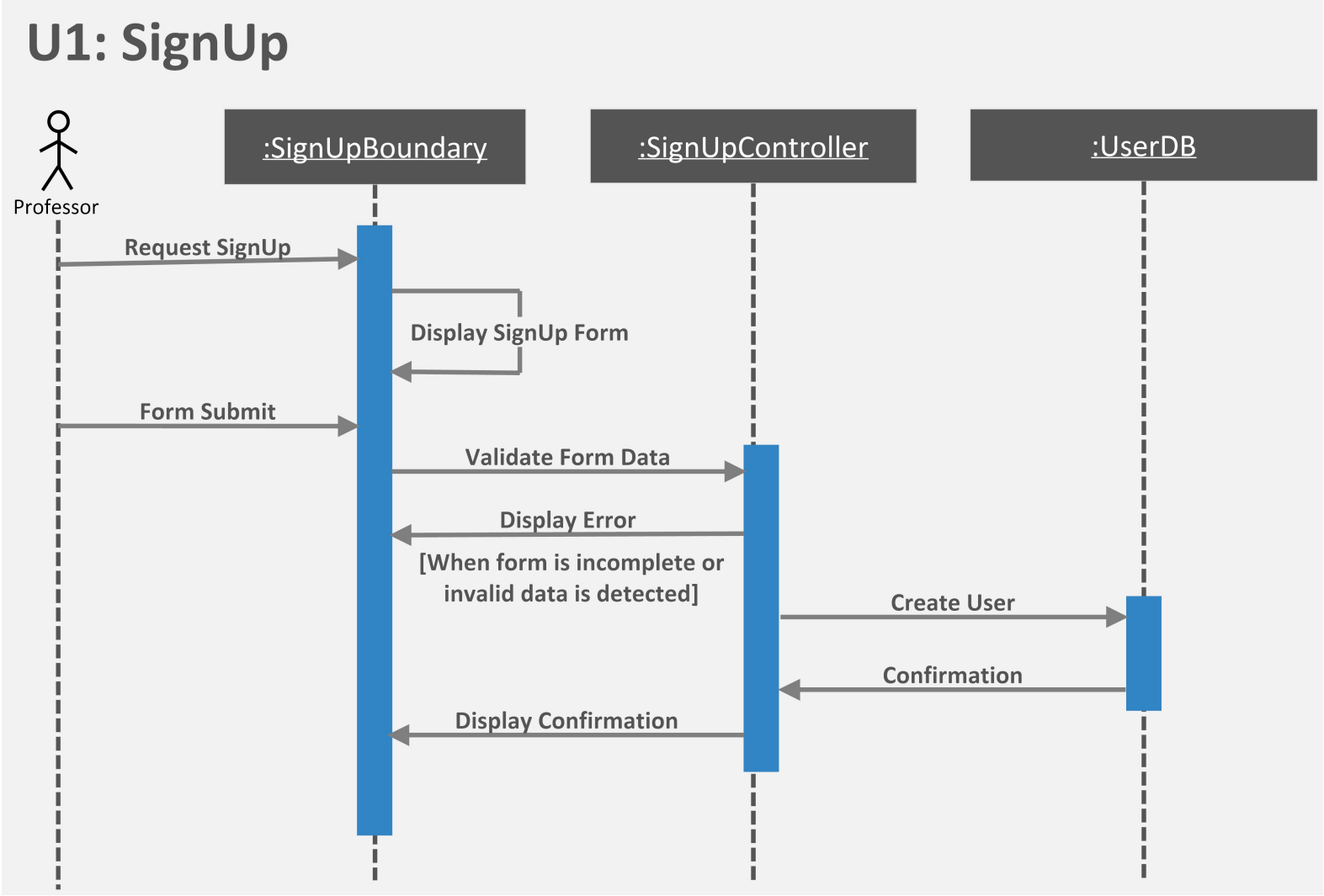

Figure 5.1: Sequence Diagram : SignUp

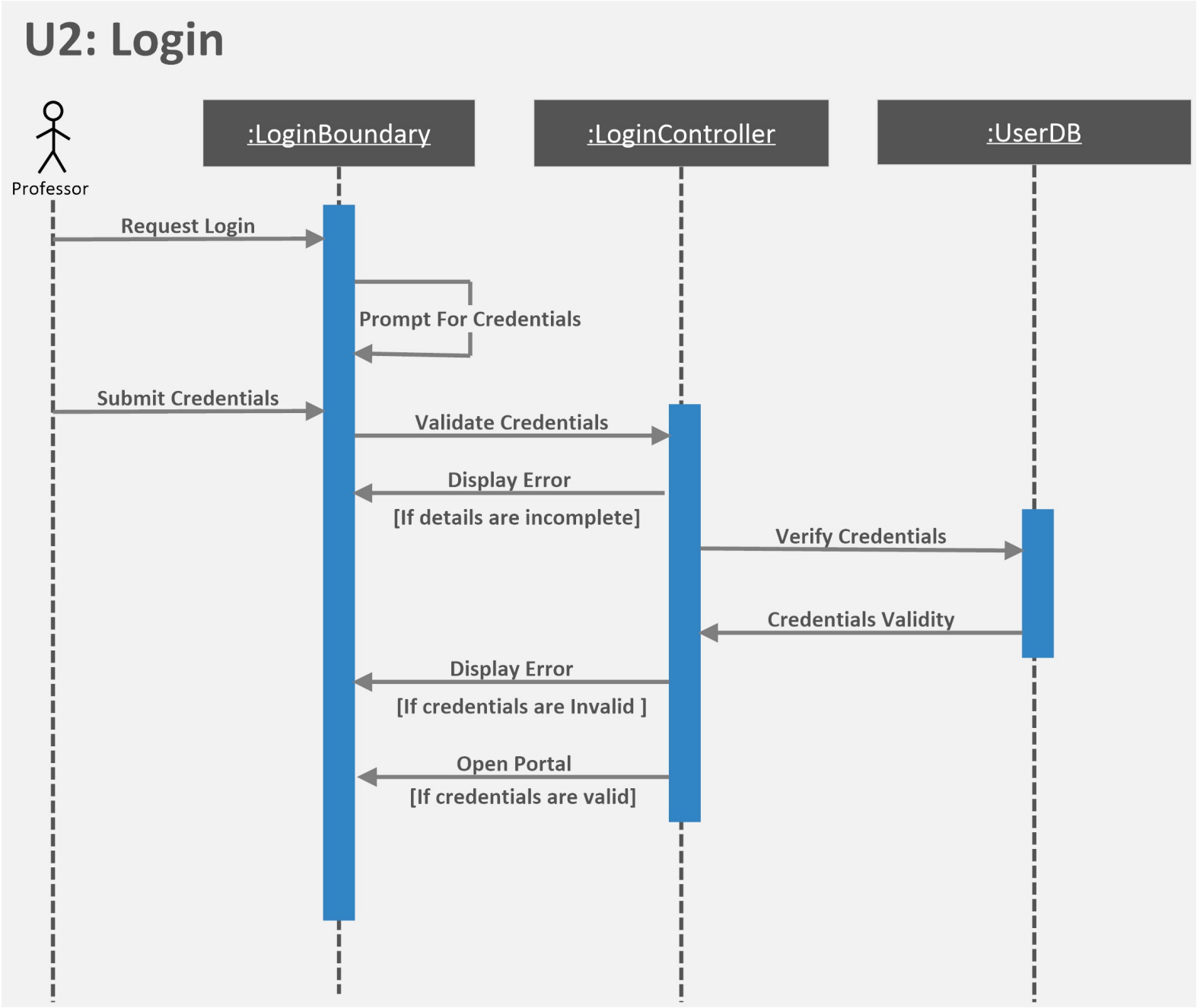

Figure 5.2: Sequence Diagram : Login

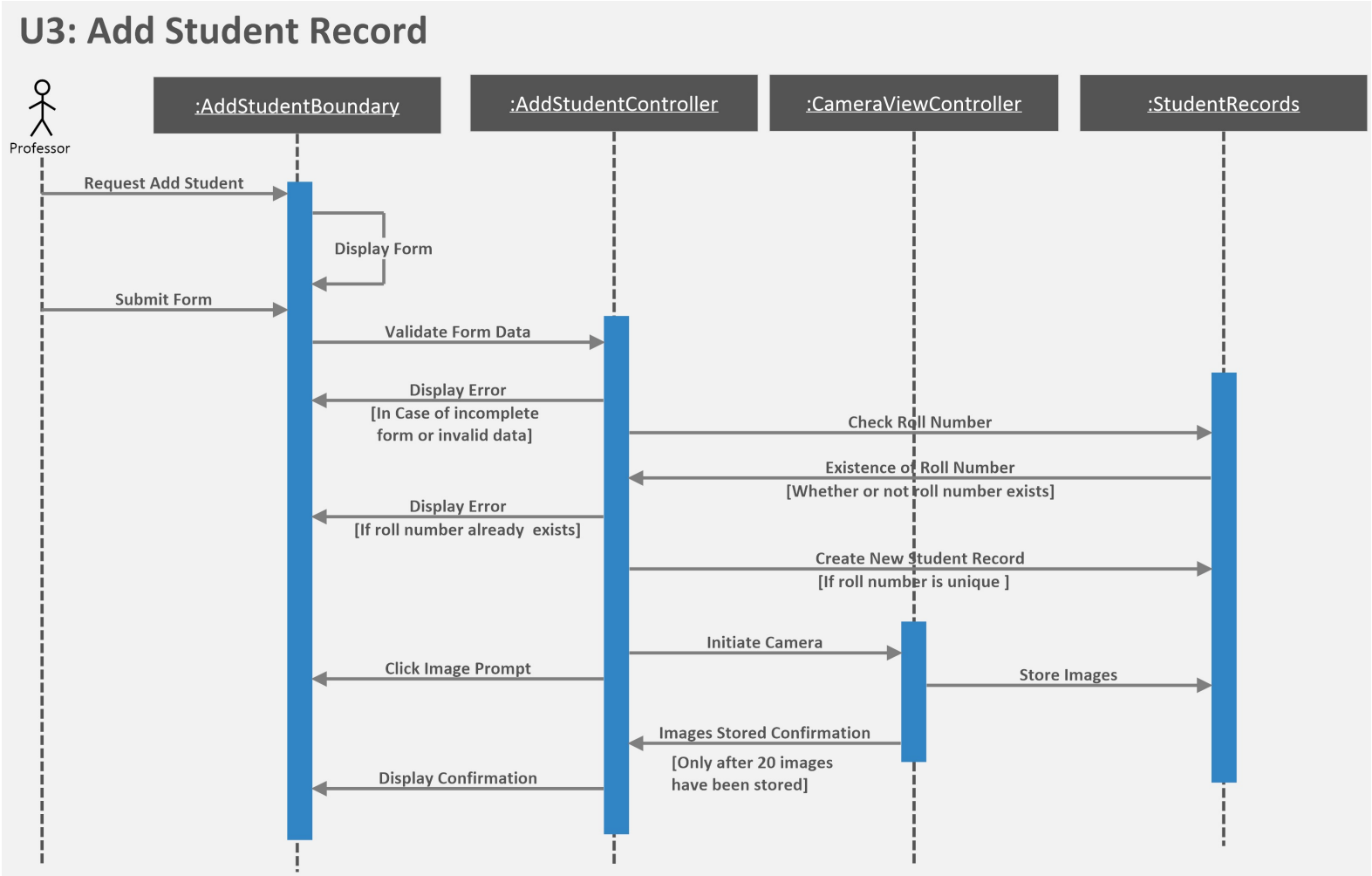

Figure 5.3: Sequence Diagram : Add Student Record

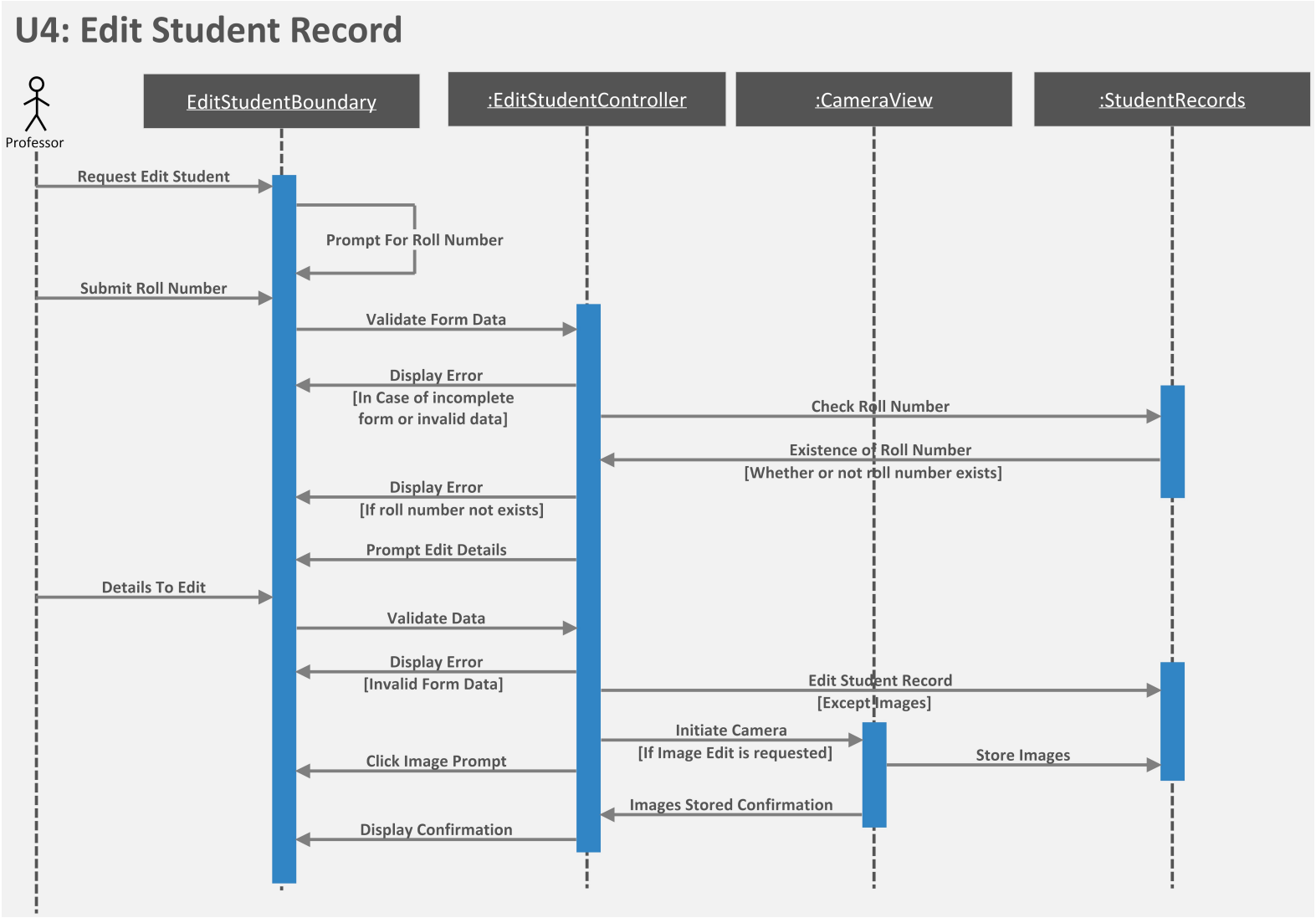

Figure 5.4: Sequence Diagram : Edit Student Record

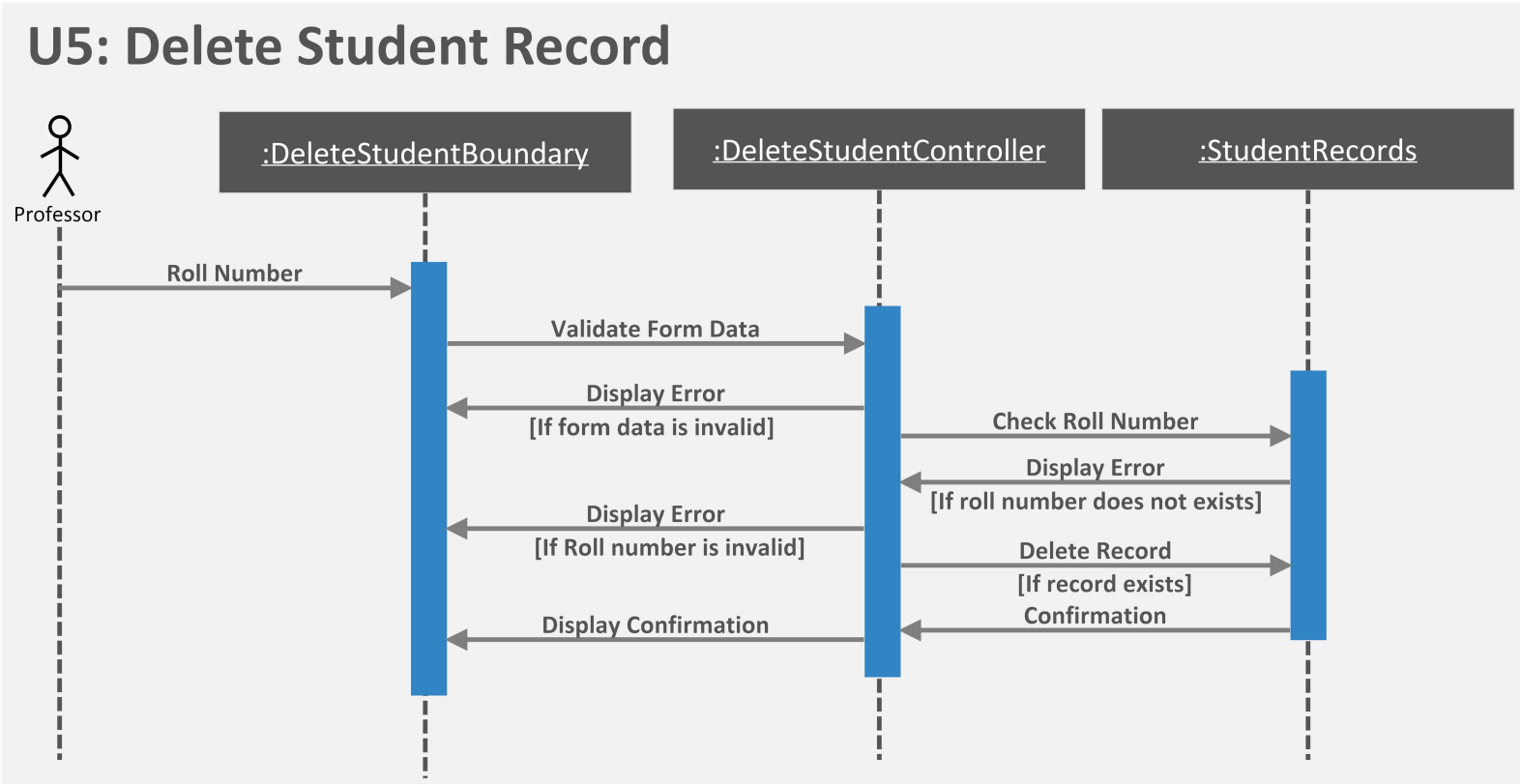

Figure 5.5: Sequence Diagram : Delete Student Record

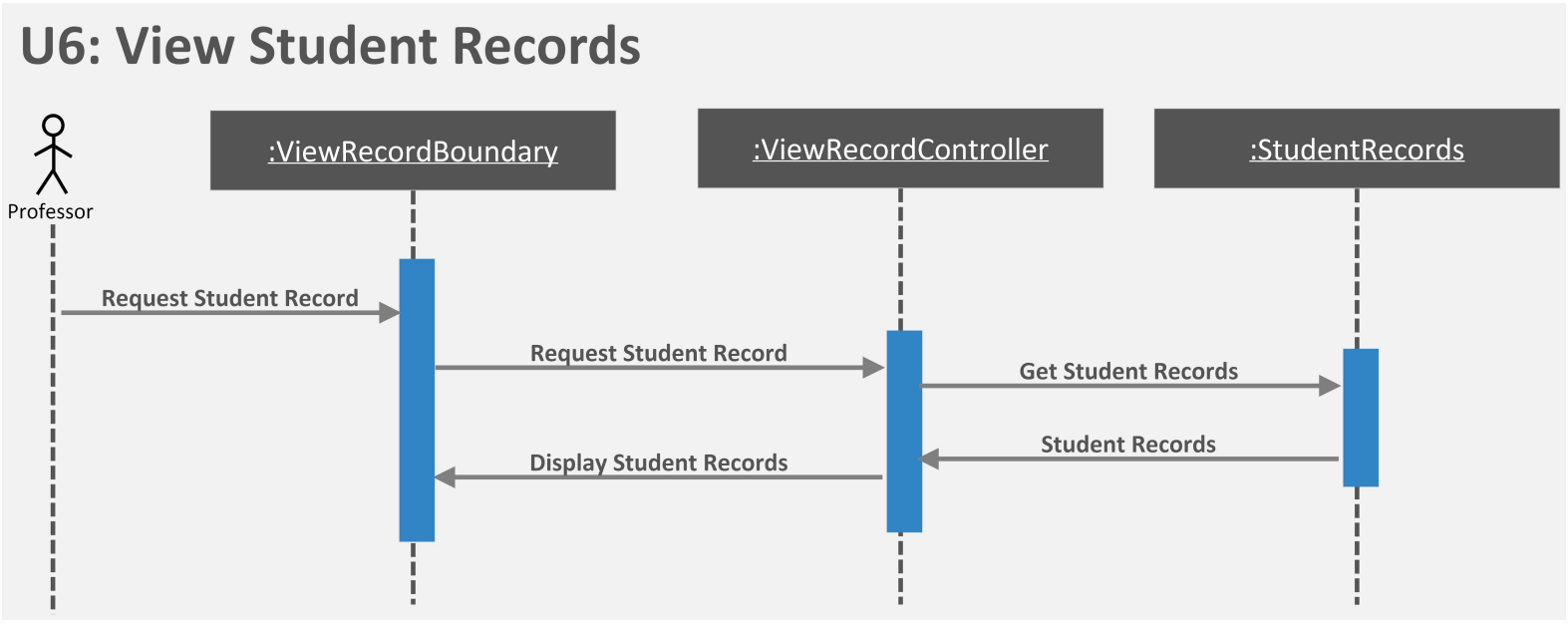

Figure 5.6: Sequence Diagram : View Student Record

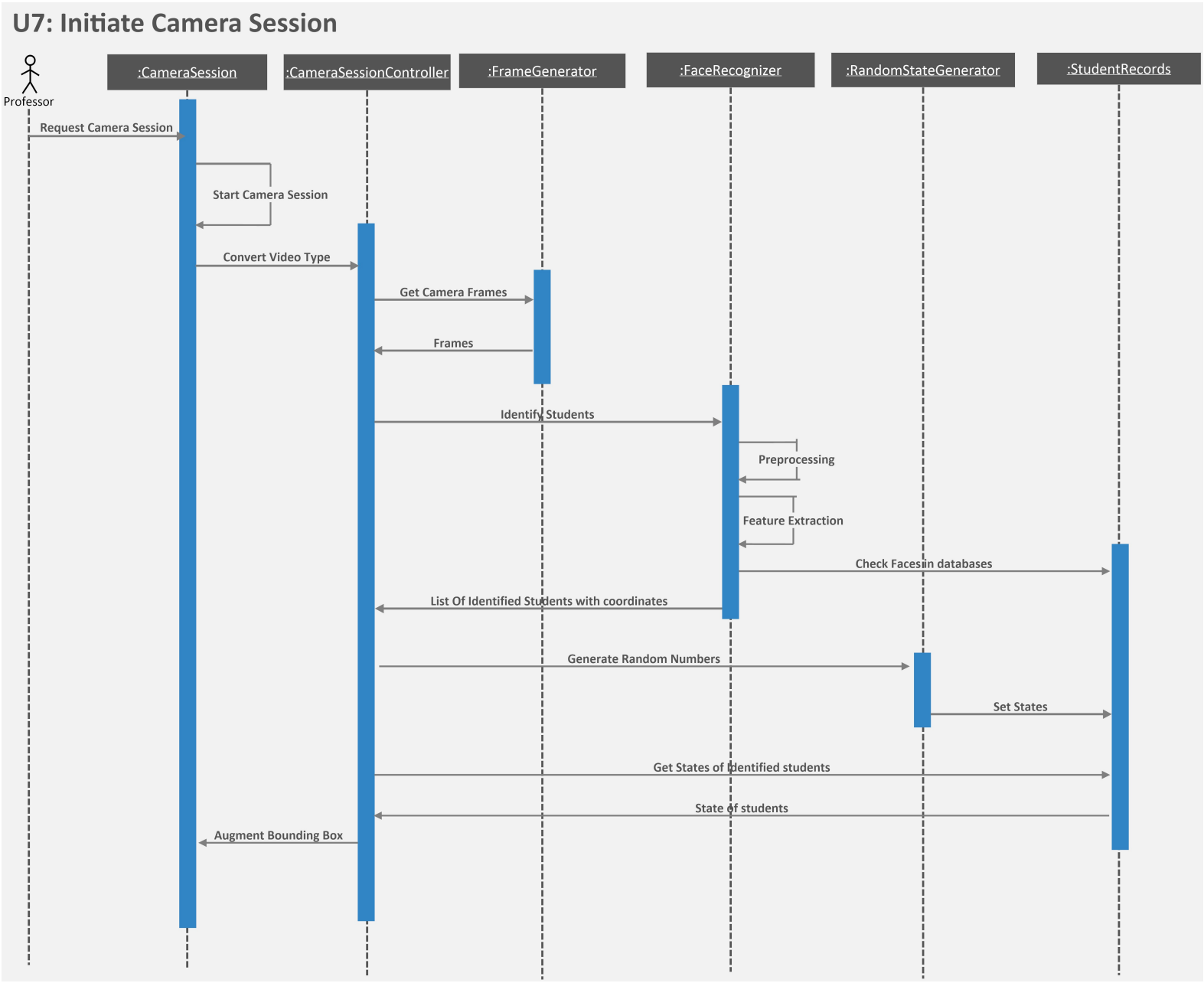

Figure 5.7: Sequence Diagram : Initiate Camera Session

6 Class Diagram

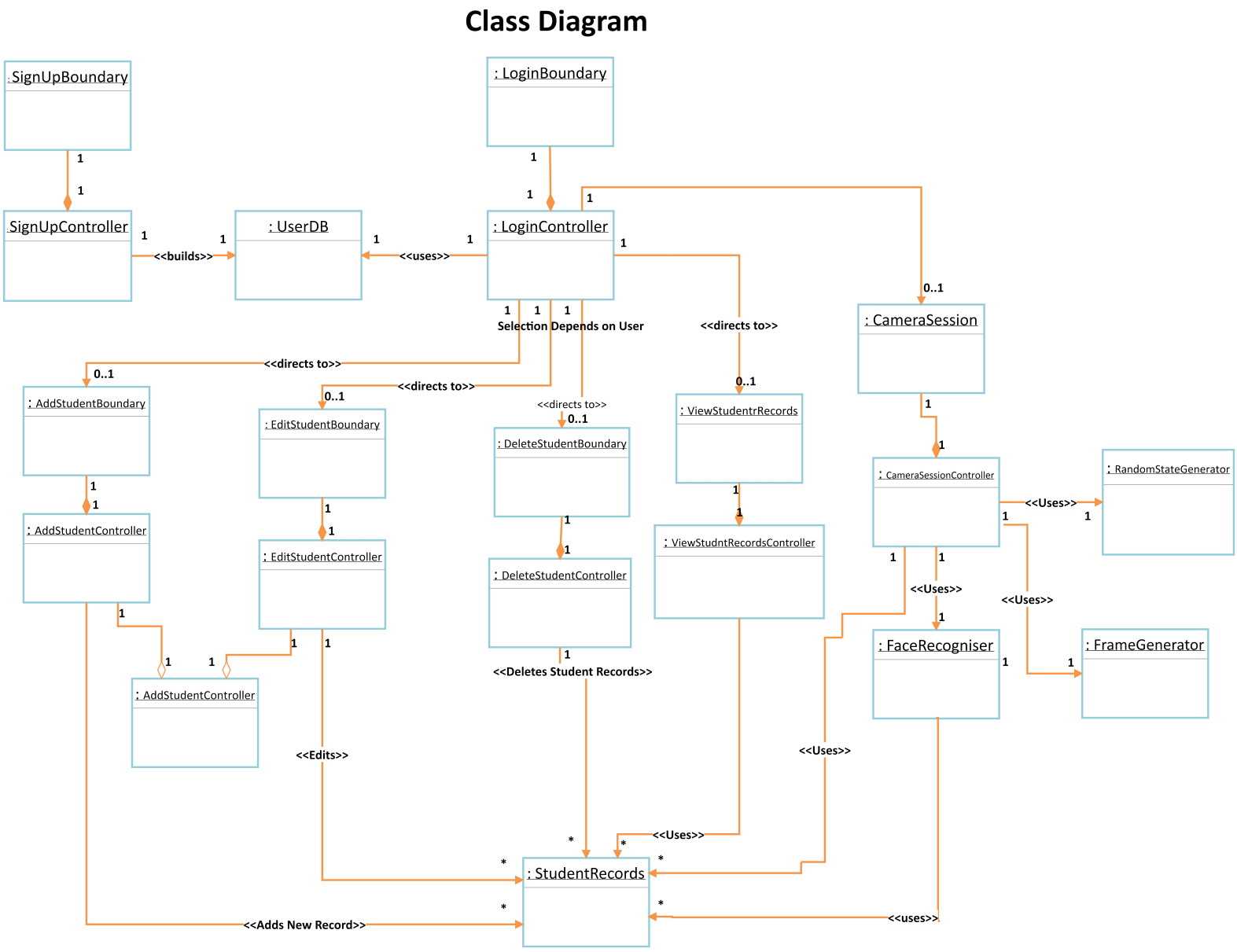

Figure 6.1: Class Diagram

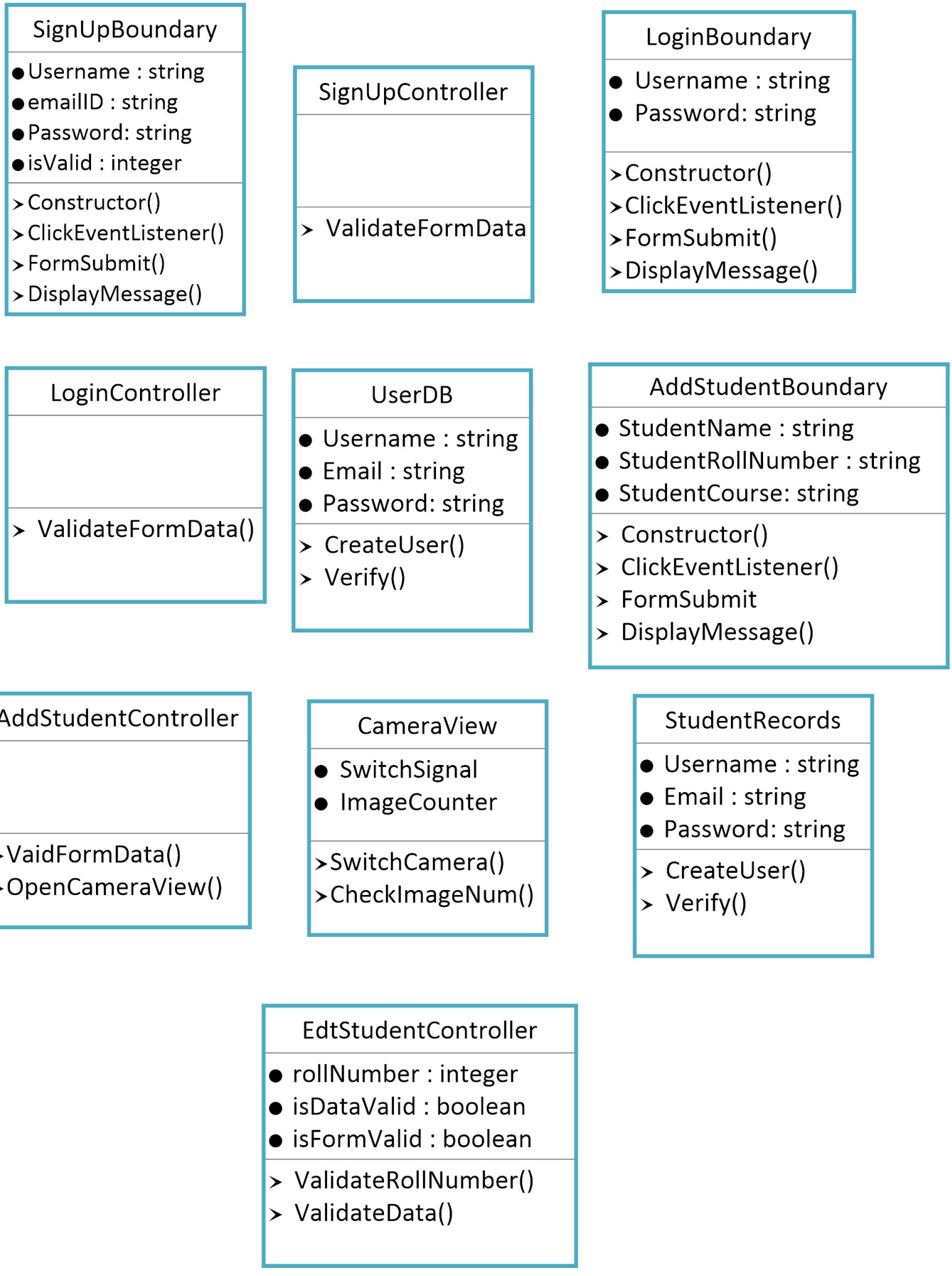

Figure 6.2: Class Summary

Figure 6.3: Class Summary