CS-242/243

Project - #9

Smart Warning System

Software Design Document

Course Instructor -
Prof. Samit Bhattacharya

Team - #23
Jatin Goyal - 160101036
Namit Kumar - 160101046
Nitin Kedia - 160101048

Table of Contents:

1. Introduction

1.1 Purpose

The objective of this document is to present a detailed design description of the mobile app *Smart Warning System.* It follows the guidelines of *Object Oriented Design* and thus includes use cases, sequence diagrams, class diagrams and class description. 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 Scope

The application will be a mobile app, on the Android platform which implements an intelligent and robust warning system to enhance student engagement during lectures by continuous monitoring and feedback. It will facilitate more meaningful and interactive teaching sessions.
The app is being developed as a project to gain practical experience in the practices of software engineering, as a part of the course of same name offered by Department of CSE, IIT Guwahati.

1.3 Document Conventions

Terms	**Definitions**
Professor	Person who shall be using the software for monitoring.
Student	Person who shall be monitored and receive appropriate alerts
[ ]	This symbol represents an Array of data
Blacklist	Contains those students who do not respond to an Alert Notification.

1.4 References

1. Software Requirement Specification Document for *Smart Warning System,* Group 23.
2. https://www.creately.com for creating UML diagrams.
3. R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th Ed., McGraw Hill, 2010.
4. **IEEE 1016-2009**, *IEEE Standard for Information Technology—Systems Design—Software Design Descriptions.*

2. Use Cases

2.1 Use Case Diagram

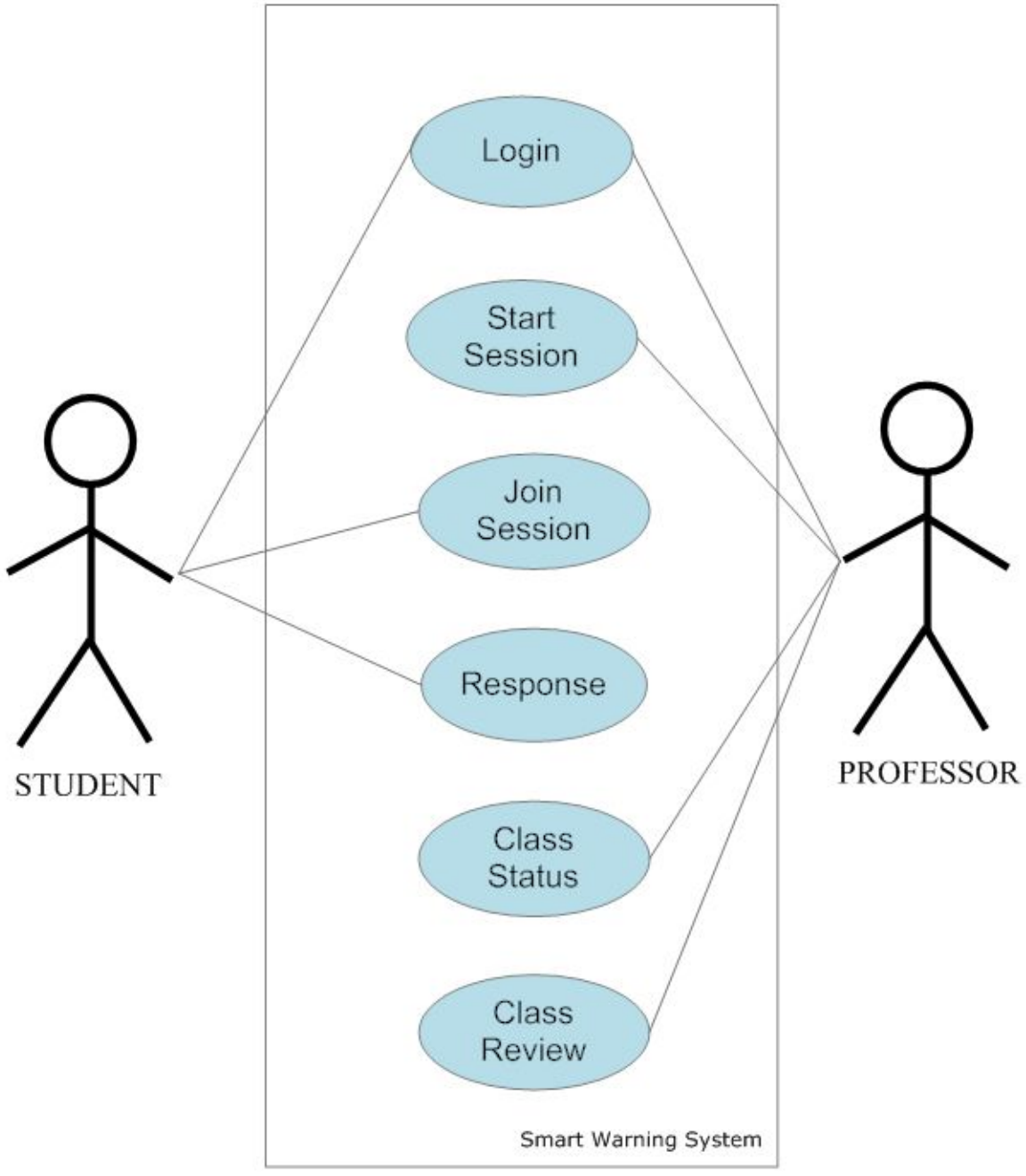


Figure 3.1: Use Case Diagram

2.2 Use Case Dialogues

2.2.1 Login

Actors: Student, Professor (collectively called User)

Scenario 1: Mainline Sequence:

1. System: Shows login screen asking for username and password.
2. User: Enters “kedia.nitin” as username and “*******” as password, taps “Login” button.
3. System: Changes screen to main menu and shows acknowledgement “Successfully Logged in”.

Scenario 2: At step 4 in mainline sequence:

3. System: Displays one of the following errors and reverts to login screen.:
   a. “One or both fields are missing.”
   b. “Invalid credentials” in case of mismatch.

2.2.2 Start Session

Actor: Professor

Precondition: Professor is already logged into his account.

Scenario 1: Mainline Sequence:

1. Professor: Taps “Start Session” button on the main menu.
2. System: Changes screen asking to enter session details.
3. Professor: Enters “CS242” as course name and “24” as Lecture number.
4. System: Displays acknowledgement - “Your session has been started” and returns session ID - “CS242_24”.

Scenario 2: At step 4 in mainline sequence:

4. System: Displays error - “Your session could not be created” if there is problem faced while creating a session.

Scenario 3: At step 4 in mainline sequence:

4. System: Displays message - “One or both fields are missing.” if some required information is not provided by the professor and returns to screen for creating session.

2.2.3 Join Session

Actor : Student

Precondition : Student has been logged into the system and the professor has already started a session.

Scenario 1: Mainline Sequence:

1. Student: Taps on “Join Session” on the android screen.
2. System: Opens the “Join Session” screen and displays a text box for session ID and “Submit” button.
3. Student: Types “CS242_24” and taps “Submit” button.

4. System: Displays a message- “You have successfully joined the session”. User is returned to main menu.

Scenario 2: At step 4 of mainline sequence:

4. System: Displays a message “You have entered an incorrect session ID”. Then the student is returned to step 3.

2.2.4 Response

Actor: Student

Precondition: Both student and professor are logged in and the professor has started a session. At Least one student has joined the session.

Scenario 1: Mainline Sequence:

1. System: Based on state values, sends an alert notification - “Your attentiveness has been low for the last 5 minutes” to the student’s mobile device.
2. Student: Taps on the notification.
3. System: Displays acknowledgement - “Your response has been recorded” and removes the notification.

Scenario 2: At step 2 of mainline sequence:

2. Student: Does not tap on the alert.
3. System: After an internal timer for responding to the alert is finished, displays a message - “You did not respond to the last alert” and does not remove the notification.

2.2.5 Class Status

Actors: Professor

Precondition: Data collected in present session is being simultaneously saved and Professor is logged in.

Scenario 1: Mainline Sequence:

1. Professor: Taps the “See Class Status” button in main menu.
2. System: Changes screen where it shows the student list with corresponding states. It also has a “Go back” button.
3. Professor: After seeing the summary, taps the “Go back” button.
4. System: Reverts back to the main menu.

Scenario 2: At step 2 of main menu:

2. System: Displays message “No ongoing session found”.

2.2.6 Class Review

Actors: Professor

Precondition: The professor has already started a session and at least one student has joined the session.

Scenario 1: Mainline Sequence

1. Professor: Taps “Class Review” button on the main menu.
2. System: Displays all students cards as shown below with a remark text box and two buttons “Submit”.

   Name: Yash BT
   Alert State: 3
   No Response Time: 10 min
   Professor Remark: _____

   Name: Mohit Kumar
   Alert State: 5
   No Response Time: 15 min
   Professor Remark: _____

   Submit

3. Professor: Enters the remarks like “5 marks deducted”, “10 marks deducted” and taps on “Submit”. Remarks are allowed to be left blank.
4. System: Message is displayed ”Performance review Done”. The professor is returned to main menu.

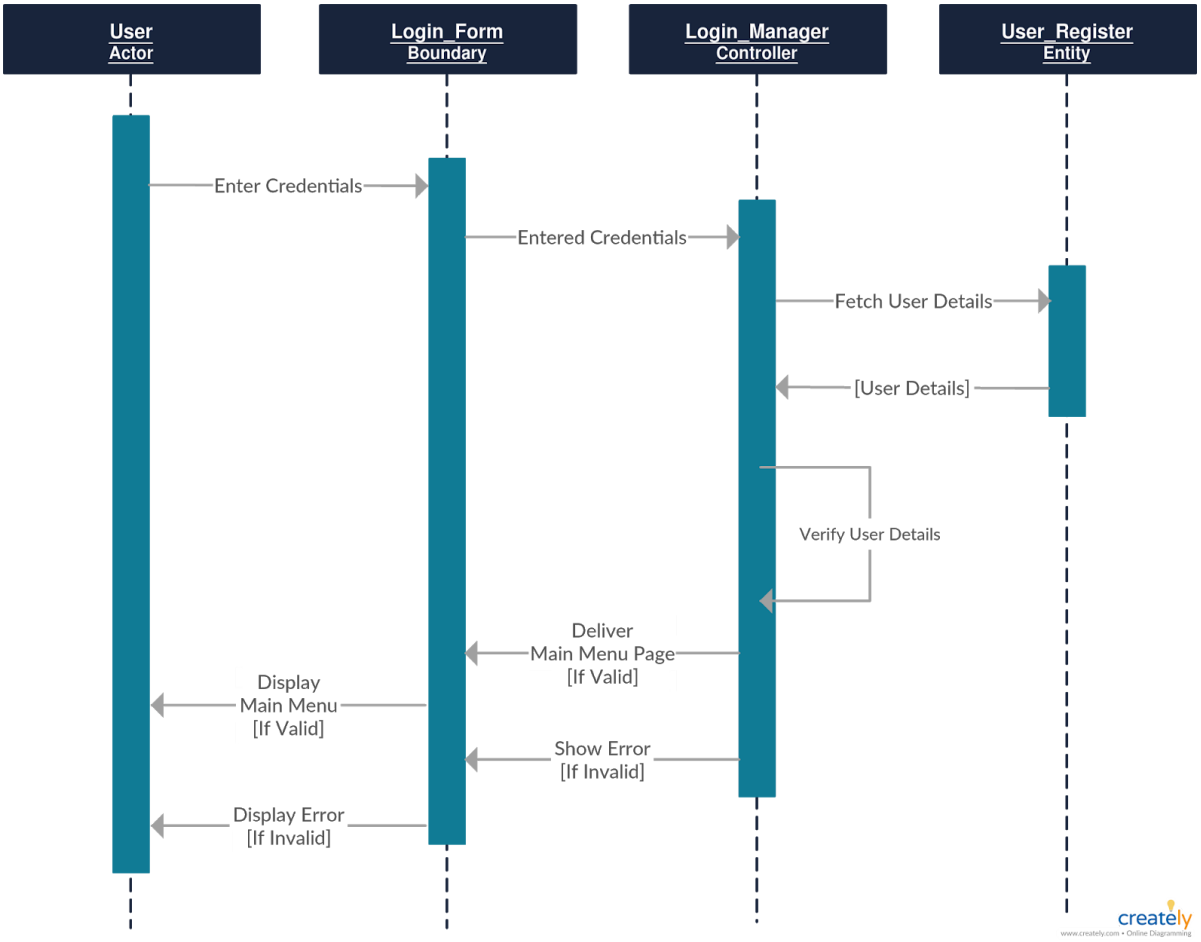
Scenario 2: At step 2 of mainline sequence:

2. System: Displays a message - “No defaulter students” and stays in the main menu.

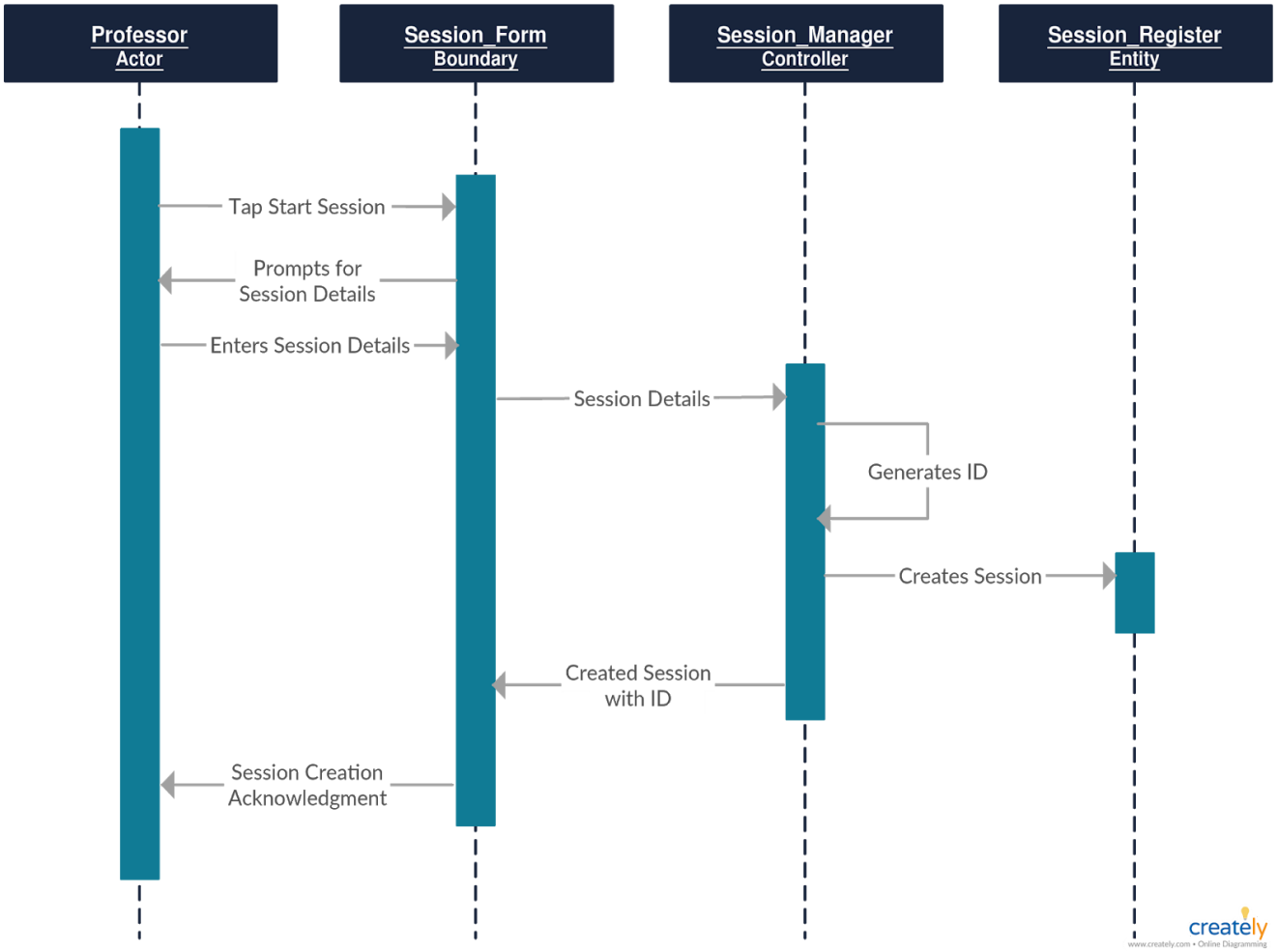
3. Sequence Diagram

Corresponding to each use case specified in previous section, we make a sequence diagram.

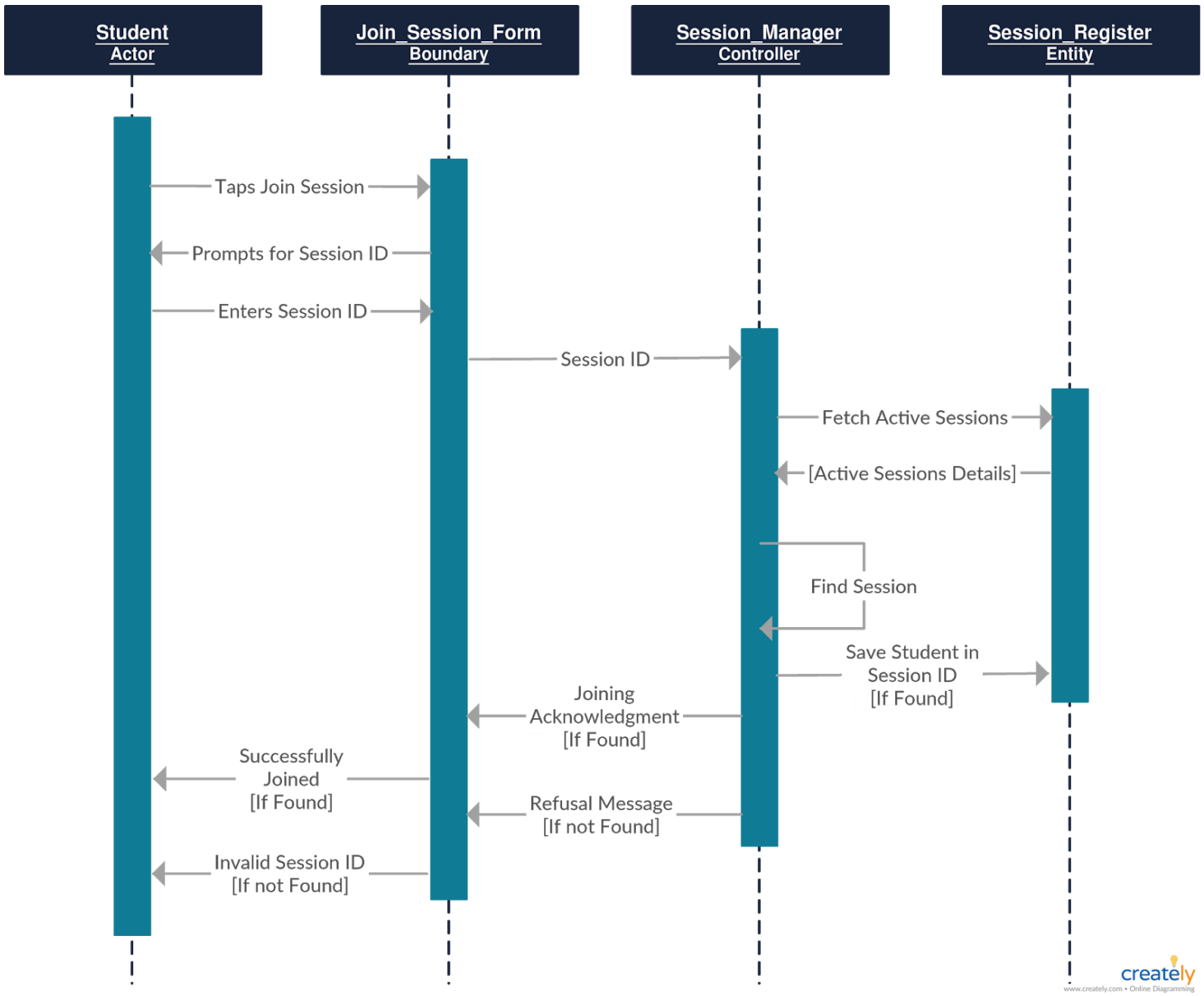
3.1 Login



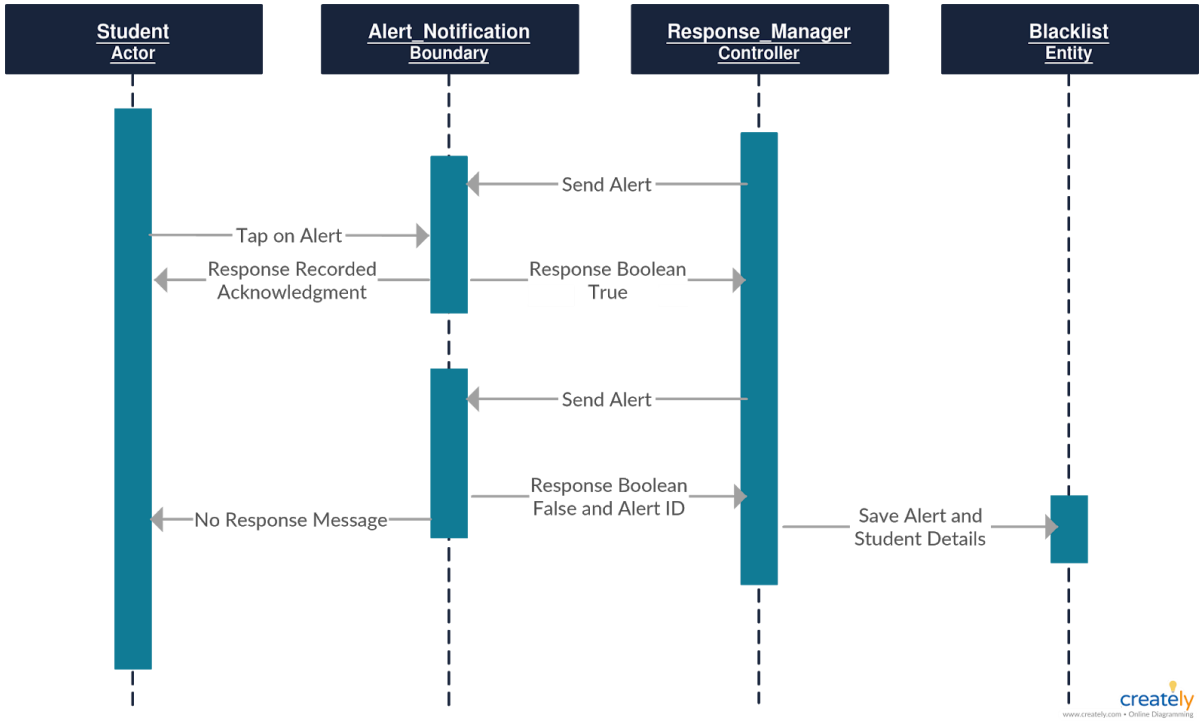
3.2 Start Session



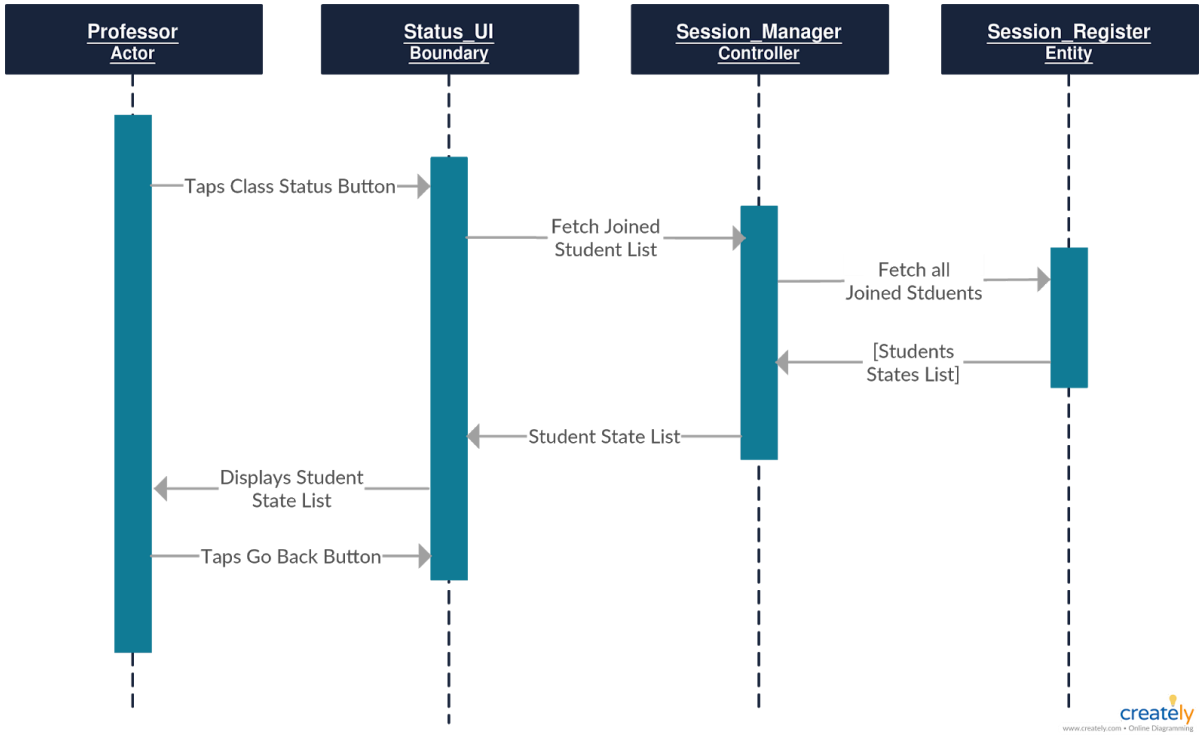
3.3 Join Session



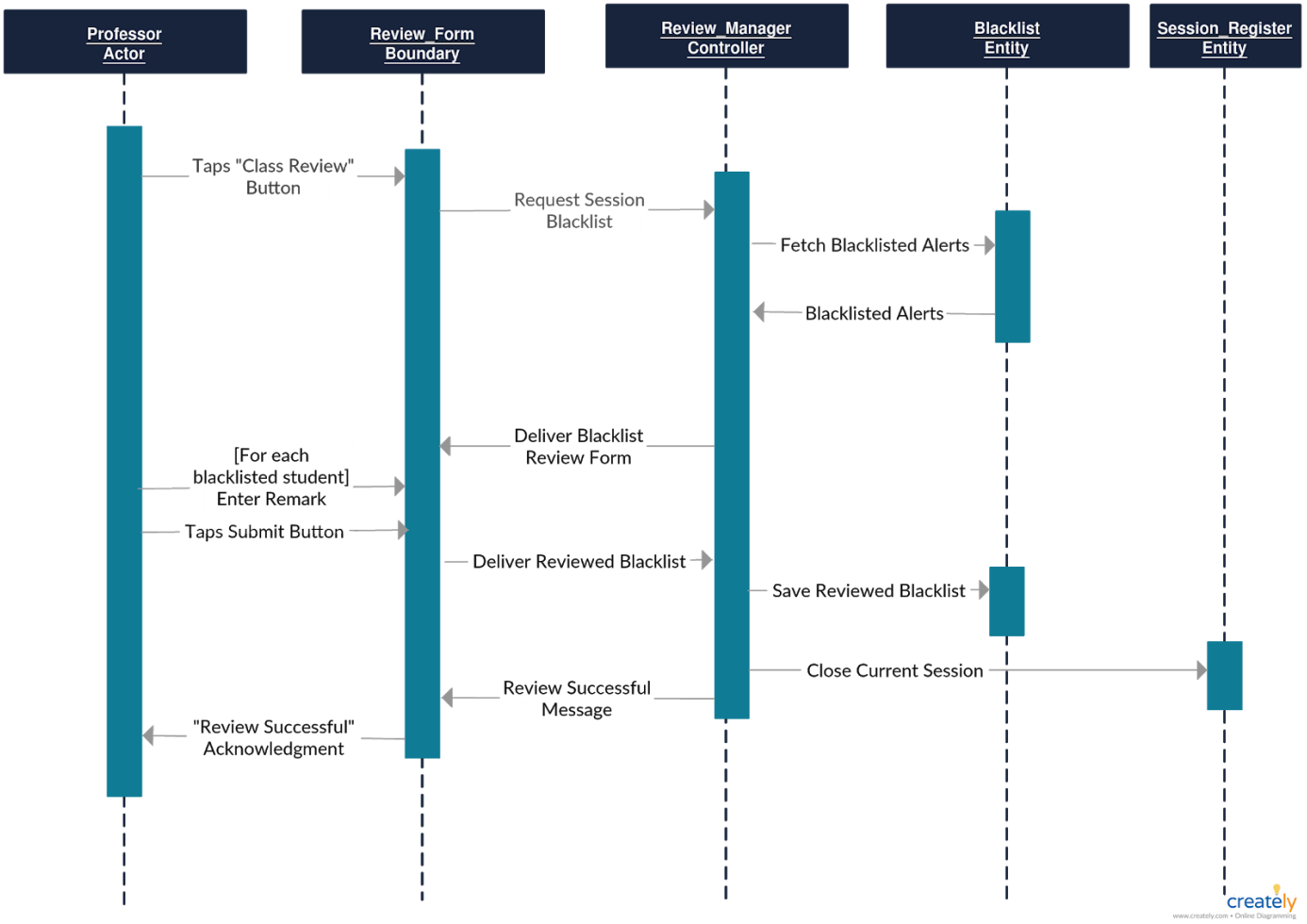
3.4 Response



3.5 Class Status



3.6 Class Review



4. Object Oriented Model

4.1 Domain Modelling

4.1.1 Entity Objects

1. User_Register
2. Session_register
3. Blacklist

4.1.2 Controller Objects

1. Login_Manager
2. Session_Manager
3. Response_manager
4. Review_Manager

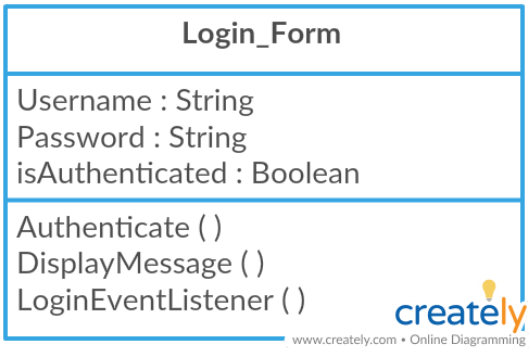
4.1.3 Boundary Objects

1. Login_Form
2. Session_Form
3. Join_Session_Form
4. Alert Notification
5. Status_UI
6. Review_Form

The instances of above objects were analyzed and it was found that the Session_Manager, Response_manager and Review Manager can be implemented through one combined class of Session Manager.

4.2 Class Descriptions

1. **Login_Form:**



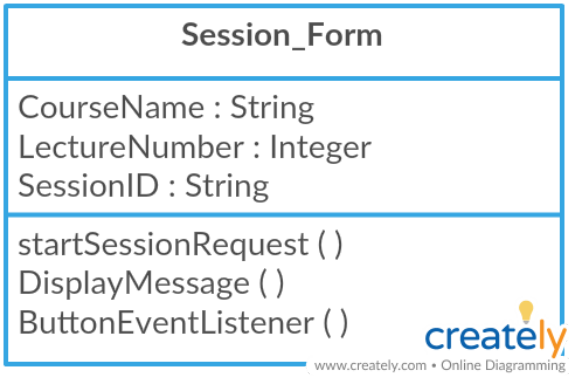
   a. Attributes:
      i. Username [String]
      ii. Password [String]
      iii. isAuthenticated [Boolean]

b. Methods:
    i. LoginEventListener()
    Parameter: TouchScreenData
    Returns: None
    Desctription: It is called when the user presses the “Login” button after entering ccredentials. It inturn invokes the Authenticate() function.

    ii. Authenticate()
    Parameters: Username, Password
    Returns: isAuthenticated
    Description: Sends the entered credentials to Login_Manager controller and sets the isAuthenticated attribute to [True/False]. It in turn invokes the DisplayMessage() function.

    iii. DisplayMessage()
    Parameter: isAuthenticated
    Returns: None
    Description: Displays Login acknowledgement or an error message based on isAuthenticated boolean.

2. **Session_Form:**

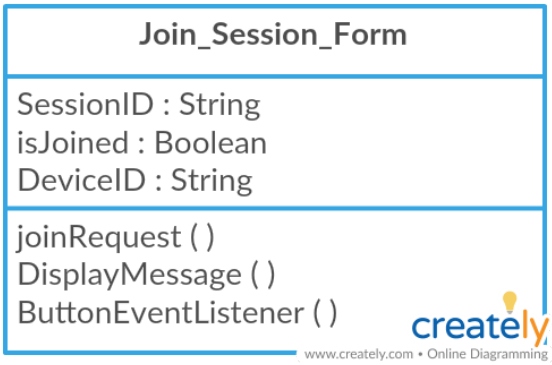


a. Attribute:
    i. CourseName [string]
    ii. LectureNumber [integer]
    iii. SessionID [string]
b. Methods:
    i. ButtonEventListener()
    Parameter: TouchScreenData
    Returns: None
    Description: It is called when the user presses the “Start Session” button. It invokes the startSessionRequest() function.

    ii. startSessionRequest()
    Parameter: Course Name, Lecture Number
    Returns: SessionID
    Description: Sends the entered details to Session_Manager controller and sets the SessionID attribute based on received ID. It invokes the displayMessage() function.

iii. DisplayMessage()
Parameter: SessionID
Returns: None
Description: Displays the session creation acknowledgement or an error message if SessionID is NULL.

3. **Join_Session_Form:**

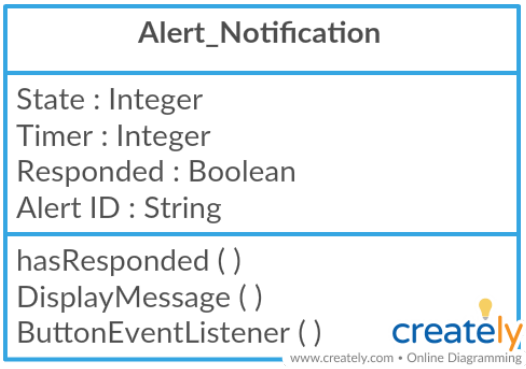


a. Attribute:
   i. SessionID [string]
   ii. isJoined [boolean]
   iii. DeviceID [string]
b. Methods:
   i. ButtonEventListener()
   Parameter: TouchScreenData
   Returns: None
   Description: It is called when the user presses the “Join Session” button. It invokes the joinRequest() function.

   ii. joinRequest()
   Parameter: SessionID, Device ID
   Returns: Boolean isJoined
   Description: Sends the Session ID, Device ID to Session_Manager and sets the isJoined attribute. It invokes the DisplayMessage() function.

   iii. DisplayMessage()
   Parameter: isJoined Boolean
   Returns: Void
   Description: Displays the session join acknowledgement or an error message if isJoined is FALSE.

4. **Alert_Notification:**



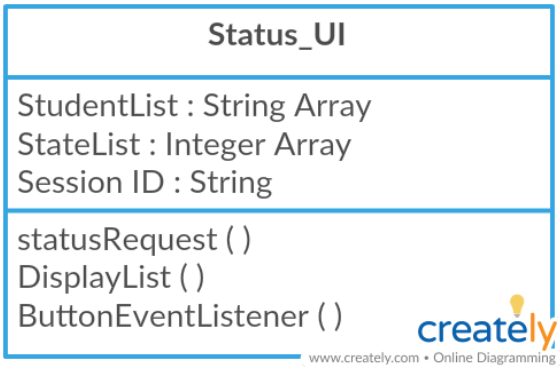
a. Attribute:
   i. State [Integer]
   ii. Timer [Integer]
   iii. Responded [Boolean]
   iv. Alert ID [string]

b. Methods:
   i. ButtonEventListener()
   Parameter: TouchScreenData
   Returns: None
   Description: It is called when the user taps the Alert Notification and sets the Responded boolean to TRUE, else it remains FALSE.

   ii. hasResponded()
   Parameter: Timer
   Returns: Alert ID, Responded
   Description: It starts when the Alert_Notification object is created. It periodically decrements the timer and checks the Responded boolean. If timer reaches zero and Responded is still False then it sends FALSE to Session_Manager else it sends TRUE. It then invokes DisplayMessage().

   iii. DisplayMessage()
   Parameter: Responded Boolean
   Returns: None
   Description: Displays the response received acknowledgement or no response message.

5. **Status_UI:**



a. Attribute:
   i. Student List [ ] [String Array]
   ii. State List [ ] [Integer Array]
   iii. SessionID

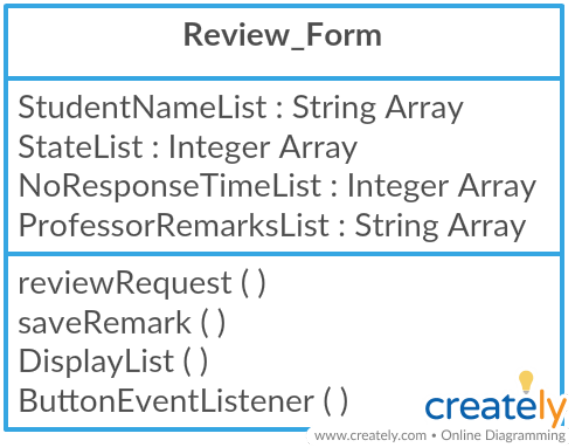
b. Methods:
   i. ButtonEventListener()
   Parameter: TouchScreenData
   Returns: None
   Description: If the user taps the “See Class Status” button then it calls

statusRequest(). If the user taps the “Go Back” button then it terminates DisplayList() and shows the main menu.

ii. statusRequest()
Parameter: Session ID
Returns: None
Description: Calls the session Manager with status request and sets the internal attributes based on data received. It then invokes DisplayList() function.

iii. DisplayList()
Parameter: Student [ ], States [ ]
Returns: Void
Description: Displays the class status list.

6. **Review_Form:**



a. Attribute:
   i. StudentNameList [ ] [String Array]
   ii. StateList [ ] [Integer Array]
   iii. NoResponseTimeList [ ] [Integer Array]
   iv. ProfessorRemarkList [ ] [String Array]

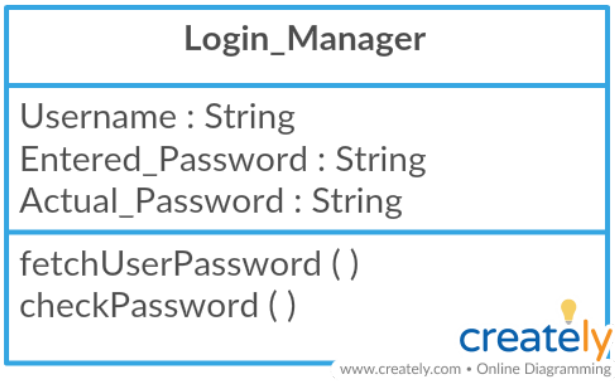
b. Methods:
   i. ButtonEventListener()
   Parameter: TouchScreenData
   Returns: None
   Description: When the user presses the “Class Review” button, it invokes the reviewRequest(). If “Submit” button is tapped, it invokes saveRemark() function.

   ii. reviewRequest()
   Parameter: None
   Returns: None
   Description: Calls the session Manager with review request and sets the internal variables based on data received. It then invokes DisplayList().

iii. DisplayList()
Parameter: Names [ ], States [ ], NoResponseTime [ ]
Returns: Void
Description: Displays the class status list along with Review field where the Professor can enter individual remarks after meeting the student.

iv. saveRemark():
Parameter: Session ID, ProfessorRemark [ ]
Returns: None
Description: Sends the remark list to Session_Manager. Then displays reviewed acknowledgement.

7. **Login_Manager:**

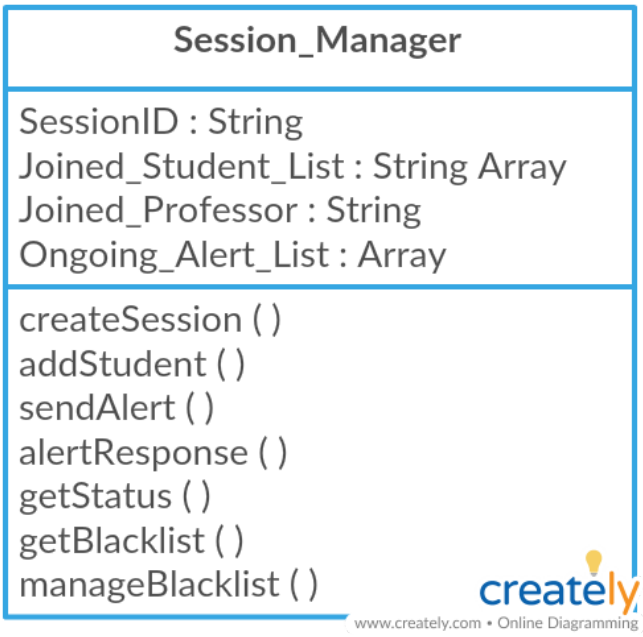


a. Attributes:
   i. Username [string]
   ii. Entered_Password [string]
   iii. Actual_Password [string]

b. Methods:
   i. fetchUserPassword()
   Parameter: Username
   Returns: Actual_Password
   Description: Access the User Database and saves the password of matched user.

   ii. checkPassword()
   Parameter: Entered_password, Actual_Password
   Returns: boolean
   Description: Compares the entered and actual password and return true if they are same, else returns False.

8. **Session_Manager:**



a. Attributes:
   i. Session ID [string]
   ii. Joined_Student_List [ ] [string Name, string Device ID]
   iii. Joined_Professor [string Name, string Device ID]
   iv. Ongoing_Alert_List [string Alert ID, string StudentName, integer alertState]

b. Methods:
   i. createSession()
   Parameter: Course Name, Lecture Number, Professor Name and Device ID.
   Returns: Session ID
   Description: Creates a new session in Session_Register . Adds the professor to the Joined_Professor attribute. Generates a new session ID, saves it and returns it. It turns on the sendAlert() function.
   Called By: startSessionRequest() in Session_Form.

   ii. addStudent()
   Parameter: Entered Session ID, Student Name and Device ID.
   Returns: Boolean isAdded
   Description: If entered Session ID matches the Session ID stored then adds the Student details to the Joined_Student_List and also adds the student to the Session_Register database. Returns TRUE if added else returns FALSE
   Called By: joinRequest() in Join_Session_Form boundary.

   iii. sendAlert()
   Parameter: Joined_Student_List [ ]
   Returns: Alert Strings
   Description: We assume that we have real-time states of joined

students in Session_Register database. For each student, It periodically checks the state value and sends corresponding alerts for students in low states. Each alert creates a Alert_Notification object in corresponding Device ID and also appends the new alert to the Ongoing_Alert_List attribute.
Called By: createSession()

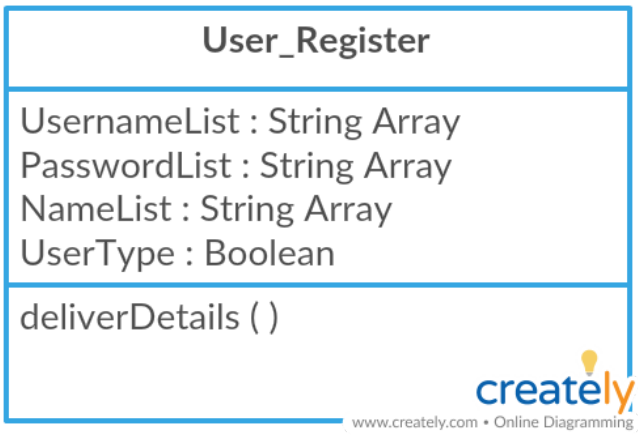
iv. alertResponse()
Parameter: Alert ID, Responded (Boolean)
Returns: StudentName, Alert State
Description: This method runs continuously. When it receives an Alert ID with Responded = TRUE then it deletes the alert from the Ongoing_Alert_List else it forwards the alert details to manageBlacklist().
Called By: hasResponded() in Alert_Notification

v. getStatus()
Parameter: None
Returns: Student State List [ ]
Description: Access the Session_Register database and gets states of all students from current session.
Called By: statusRequest() in Staus_UI boundary.

vi. getBlacklist()
Parameter: None
Returns: Blacklist [ ]
Description: Access the Blacklist entity and fetch the whole Blacklist of current session ID. Forward it to reviewRequest().
Called By: reviewRequest() in Review_Form boundary.

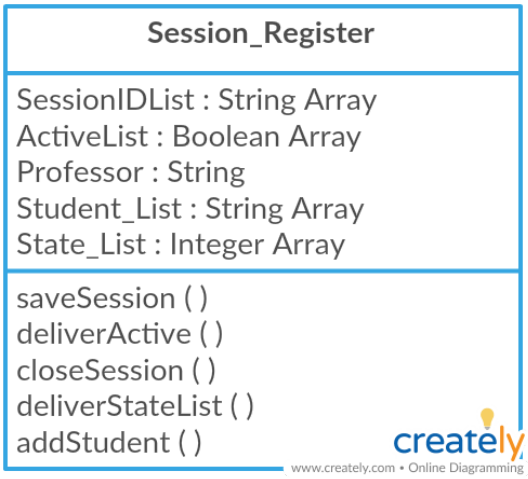
vii. manageBlacklist()
Parameter: Alert(s) to be Blacklisted
Returns: None
Description: Save the input in Blacklist Entity. May add one or more than one entry in the blacklist at a time.
Called By: reviewRequest() in Review_Form boundary, alertResponse().

9. User_Register:



a. Attributes:
    i. User_List [ ] [string Username,
                       string Password,
                       string Name,
                       bool userType]
b. Methods:
    i. deliverDetails():
    Parameters: Username
    Return: Password, Name, Type
    Description: It returns the details of the user with given username.

10. **Session_Register:**



a. Attributes:
    i. SessionID List [string]
    ii. Active List [boolean]
    iii. Professor List [string]
    iv. Student_List [ ] [string array, int state] (for each SessionID)
b. Methods:
    i. saveSession():
    Parameters: Session ID, User
    Return: None
    Description: Creates a new session with given ID, assigns its Active flag to True. Also sets the Professor who created the session.

    ii. deliverActive():
    Parameters: None
    Return: List of Active Sessions
    Description: It returns the session with Active boolean TRUE.

    iii. closeSession():
    Parameters: Session ID
    Returns: None
    Description: Sets the Active flag of given session ID as FALSE.

iv. deliverStateList():
Parameters: Session ID
Returns: Student_List with corresponding states
Description: For given session ID returns the list of users present. Else returns None if no session is found.

v. addStudent():
Parameters: Session ID, username
Returns: None
Description: Adds the username to the Student List of the session with given session ID.

11. **Blacklist:**

a. Attributes:
   i. SessionID List [ ] [string]
   ii. NoResponseList [ ] [string Name, int State, int Time, string Remark] (for each Session ID)
b. Methods:
   i. saveAlert():
   Parameters: Session ID, Name, State, Remark
   Returns: None
   Description: Adds the details of an unresponsive alert to the given session ID.
   ii. deliverBlacklist():
   Parameters: Session ID
   Returns: NoResponseList
   Description: Returns the list of unresponsive alerts of input SessionID for Class Review use case.

4.3 Class Diagram