Content

Content 1

1.Introduction 3

1.1 Purpose 3

1.2 Document Convention 3

1.3 Project Scope 3

2. Design Overview 4

2.1 Introduction 4

2.2 Background 4

2.3 Overview 4

2.4 References 4

3. Use Case Diagram 5

3.1 U1 : Login 6

3.2 U2 : Initiate Session 6

3.3 U3 : Join Session 7

3.4 U4 : Register Course 7

3.5 U5 : Add Course 7

3.6 U6 : View 2D Map 8

3.7 U7 : View Past Record 8

4. Sequence Diagram 9

4.1 Use Case: Login 9

4.2 Use Case : Initiate Session 10

4.3 Use Case : Join Session 11

4.4 Use Case : Register Course 12

4.5 Use Case :Add Course 13

4.6 Use Case :View 2D map 14

4.7 Use Case :View Past Report 15

5. Object Oriented Model 16

5.1 Domain Modelling 16

Boundary Object 16

Controller Object 16

Entity Object 16

5.2. Class Description 17

5.2.1 LoginUI 17

5.2.2 ProfesserMainMenuUI 17
5.2.3 StudentMainMenuUI 19
5.2.4 2DViewUI 20
5.2.5 LoginManager 21
5.2.6 Manage_Session 22
5.2.7 CourseRegistrationManager 23
5.2.8 CourseAdditionManager 24
5.2.9 Socket : IO.socket-client 25
5.2.10 PastReportManager 26
5.2.11 Socket : server 26
5.2.12 Professor 28
5.2.13 Student 29
5.2.14 Course 30
5.2.15 Lecture 32
5.2.16 Classroom 33
5.2.17 StudentStateDb 33

6. Class Diagram 35

7 . FlowChart for the System 36

1.Introduction

This Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within this Software Design Document are narrative and graphical documentation graphical documentation of the software design for the project including use case models, sequence diagrams, class diagrams and other supporting requirement information.

1.1 Purpose

The purpose of this document is to give a detailed description of the design for the Lecture Monitoring 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 Convention

Term	Meaning
Instructor/Professor	Person who shall be using the software for tracking state
Student	Person who shall be monitored by the instructor
Users	Collectively refers to the students and instructors
Android	Android is a mobile operating system developed by Google
Google	Google is an American multinational technology company specializing in Internet-related services and products.

1.3 Project Scope

This software is meant to be deployed in an IT-enabled large classroom environment where in the lecture delivered by the instructor is via an android device fulfilling the basic requirements as mentioned in the SRS document.This software shall allow the instructor to conveniently monitor the state of their students in real time.This software has 3 major components one is the server running on the device of the instructor, next is the client side application running on the students' devices which monitors the sensors and the third is an online portal that helps the students and instructors review past state tracked data.

2. Design Overview

2.1 Introduction

The Design Overview is section to introduce and give a brief overview of the design.This allows for the reader and user of the document to orient themselves to the design and see a summary before proceeding into the details of the design.

2.2 Background

The existing system for monitoring the lecture requires a lot of manual work and leads to wastage of important time which could have been devoted to lecture . The class participation of student is monitored by the instructor manually which may lead to error. Our app attacks these problems by shifting the manual labour required from users to software drastically reducing the wastage time and allowing the instructor an easier and better way of monitoring students present in the class and also identifying the absentee.

2.3 Overview

The Software Design Document is divided into sections with various subsections.
The sections of the Software Design Document are:
1 Introduction
2 Design Overview
3 Use Cases
4 Sequence Diagram
5 Object Oriented Model
6 Class Diagram
7 Flow Chart Diagram

2.4 References

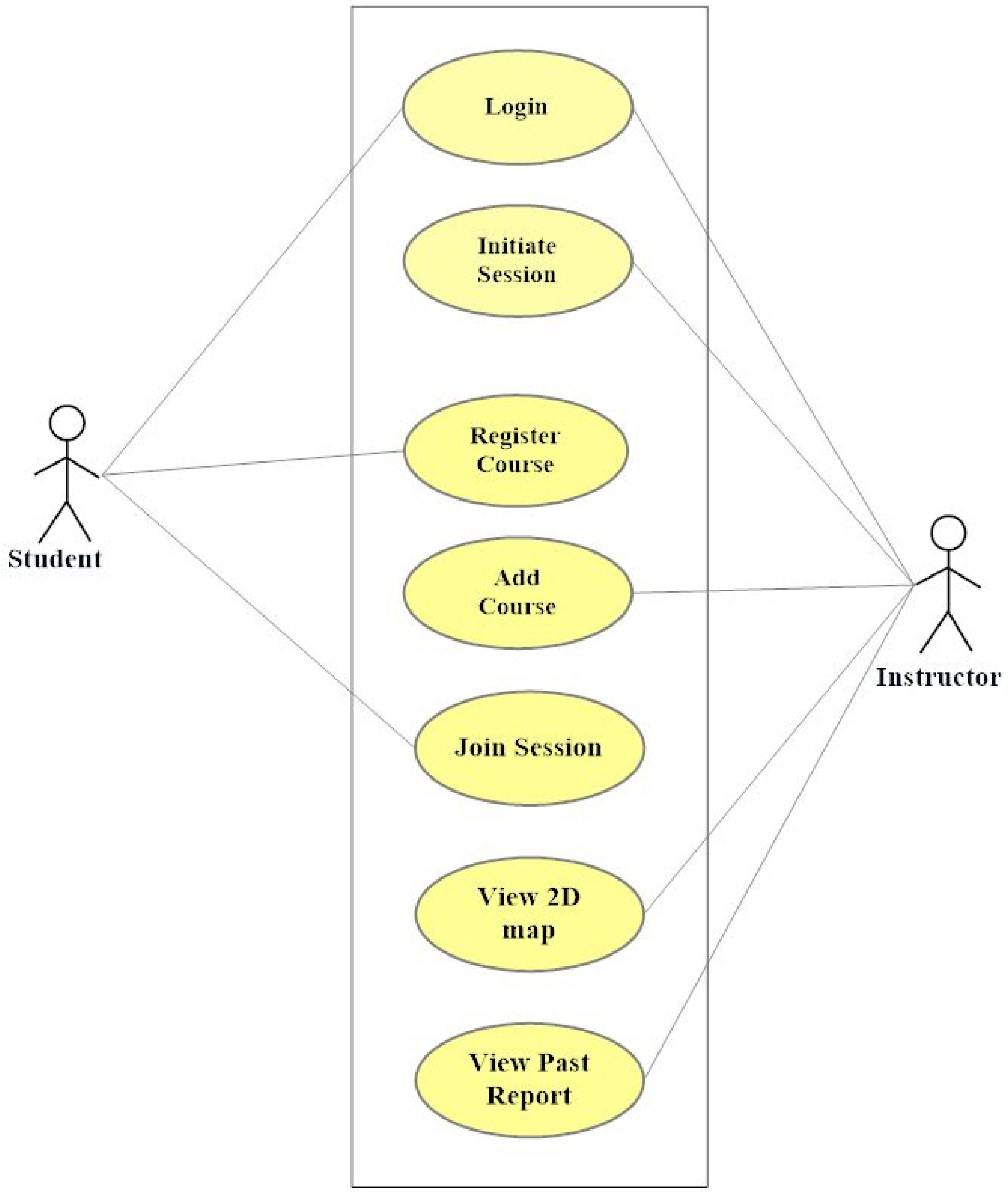
Tools For Creating UML diagram

1. Smart Draw
2. https://www.draw.io/

Other Sources of References

1. Stack Overflow Forum https://stackoverflow.com/
2. Socket Server API Reference https://socket.io/docs/server-api/
3. R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th Ed., McGraw Hill,

3. Use Case Diagram



3.1 U1 : Login

Actors : Student , Professor (both collectively referred to as user)

Scenario 1 : Mainline Sequence :

1. User : Requests the login page.
2. System : Prompts to enter details.
3. User : Enter the details i.e username and password.
4. System : Displays Login acknowledgement and redirects to main menu.

Scenario 2: At step 4 in mainline:

4. System : Indicates error in case of incorrect and/or empty username and password Combination.

3.2 U2 : Initiate Session

Actors : Professor
Precondition : The professor has logged in with his credentials

Scenario 1 : Mainline Sequence :

1. Professor : Selects the initiate session option.
2. System : Display the course list of current professor.
3. Professor : Select course name for which session is to be created.
4. System : Displays success message along with session key \
5. Professor : Selects to terminate session after class.
6. System : Display successful termination message.

Scenario 2: At step 4 in mainline:

4. System : Displays error if a problem is encountered while creating the session.

Scenario 3: At step 2 in mainline:

2. System : Displays Add course option if there are no courses for the professor.

3.3 U3 : Join Session

Actors : Student
Precondition : Student is logged into the system and a session has been initiated by the professor.

Scenario 1 : Mainline Sequence :

1. Student : Select Join Session option.
2. System : Display Course list to select an option.
3. Student : Select a course to join session from course list.
4. System : Prompt for session key
5. Student : Enter session key as instructed by professor
6. System : Shows a message acknowledging joining the session.

Scenario 2: At step 6 in mainline:

6. System : Displays error when student enters wrong session key for course and prompts to re enter session key.

3.4 U4 : Register Course

Actors : Student
Precondition : Student is logged in

Scenario 1 : Mainline Sequence :

1. Student : Select Register Course option.
2. System : Displays all available course list.
3. Student : Select a course from list to register.
4. System : Displays message “You have been registered for this course”.

3.5 U5 : Add Course

Actors : Professor
Precondition : Professor is logged in

Scenario 1 : Mainline Sequence :

1. Professor : Selects the add course option.
2. System :Prompt for course details
3. Professor : Enters details for new course
4. System : Displays success message course added successfully.

Scenario 2: At step 4 in mainline:

4. System : Displays error if there is an existing course with same details.

3.6 U6 : View 2D Map

Actors : Professor
Precondition : The professor is logged in and initiated a session and at least one student has joined the session.

Scenario 1 : Mainline Sequence :

1. Professor : Selects the view 2D map option.
2. System : Display the class map showing students at their seat location and their state augmented on their position .
3. Professor : Select a specific student to view student details
4. System : Displays student details i. e. state history during lecture, name, roll no etc.

Scenario 2: At step 2 in mainline:

2. System : Displays error if a problem is encountered while creating class view.

3.7 U7 : View Past Record

Actors : Professor
Precondition : Professor is logged in

Scenario 1 : Mainline Sequence :

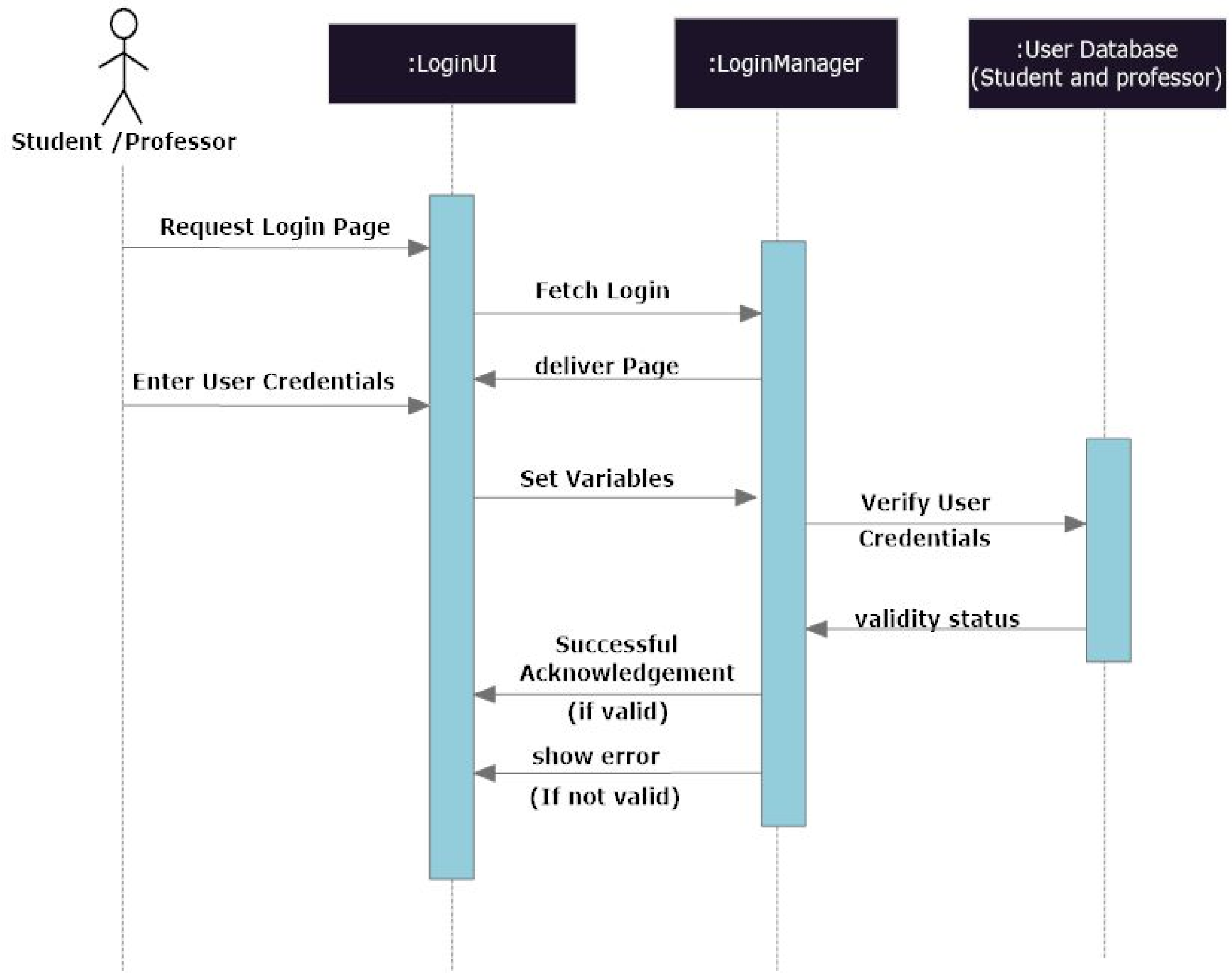
1. Professor : Select option of viewing past record.
2. System : Displays a prompt to select the course and enter session details like date etc
3. Professor : Selects a course and enter session details
4. System : Displays the student state statistics for that lecture.

Scenario 2 : At step 4 of mainline sequence:

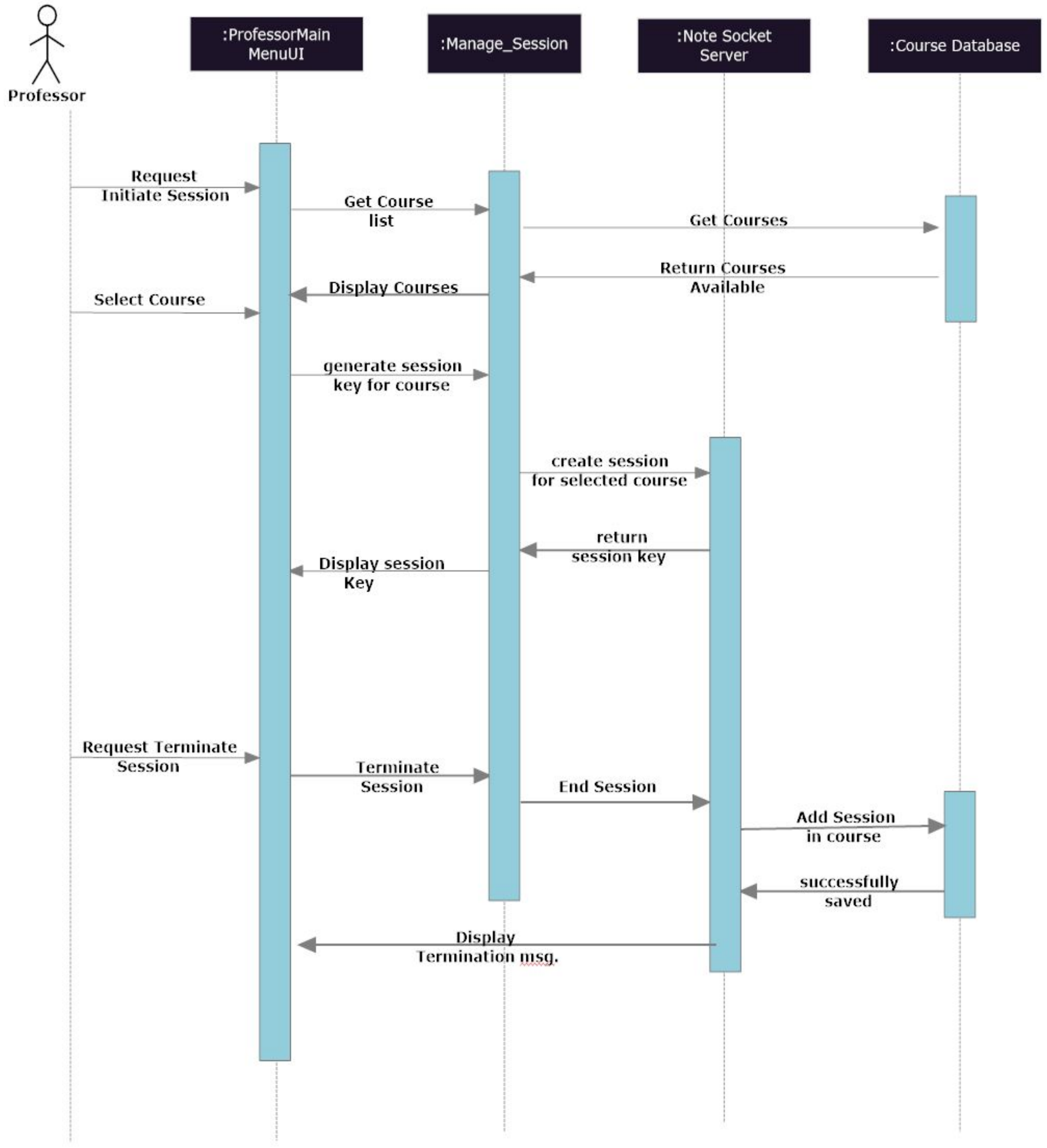
4. System : Displays an error message that the particular session details are incomplete or incompatible(record not exists) and prompts to enter the correct details

4. Sequence Diagram

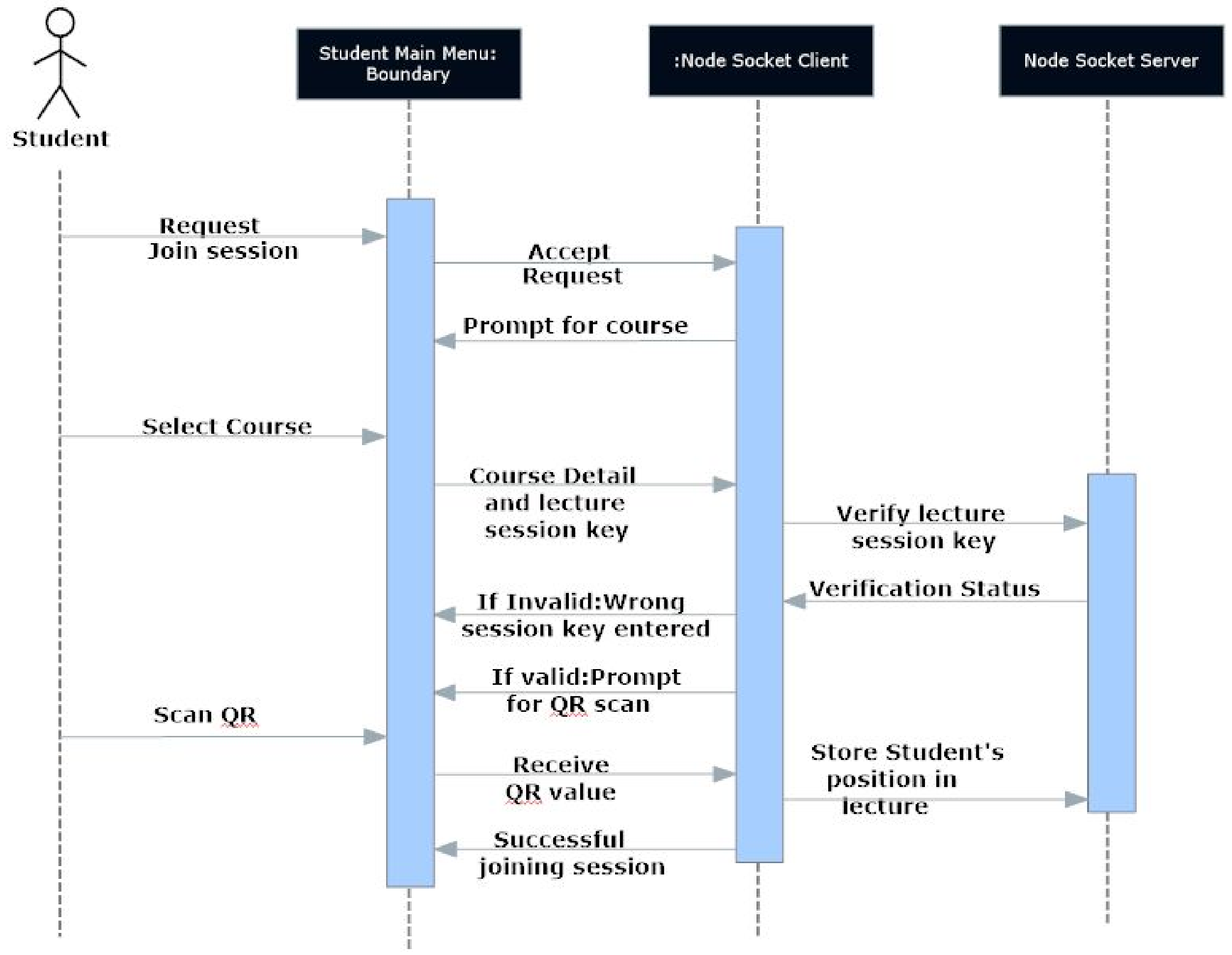
4.1 Use Case: Login



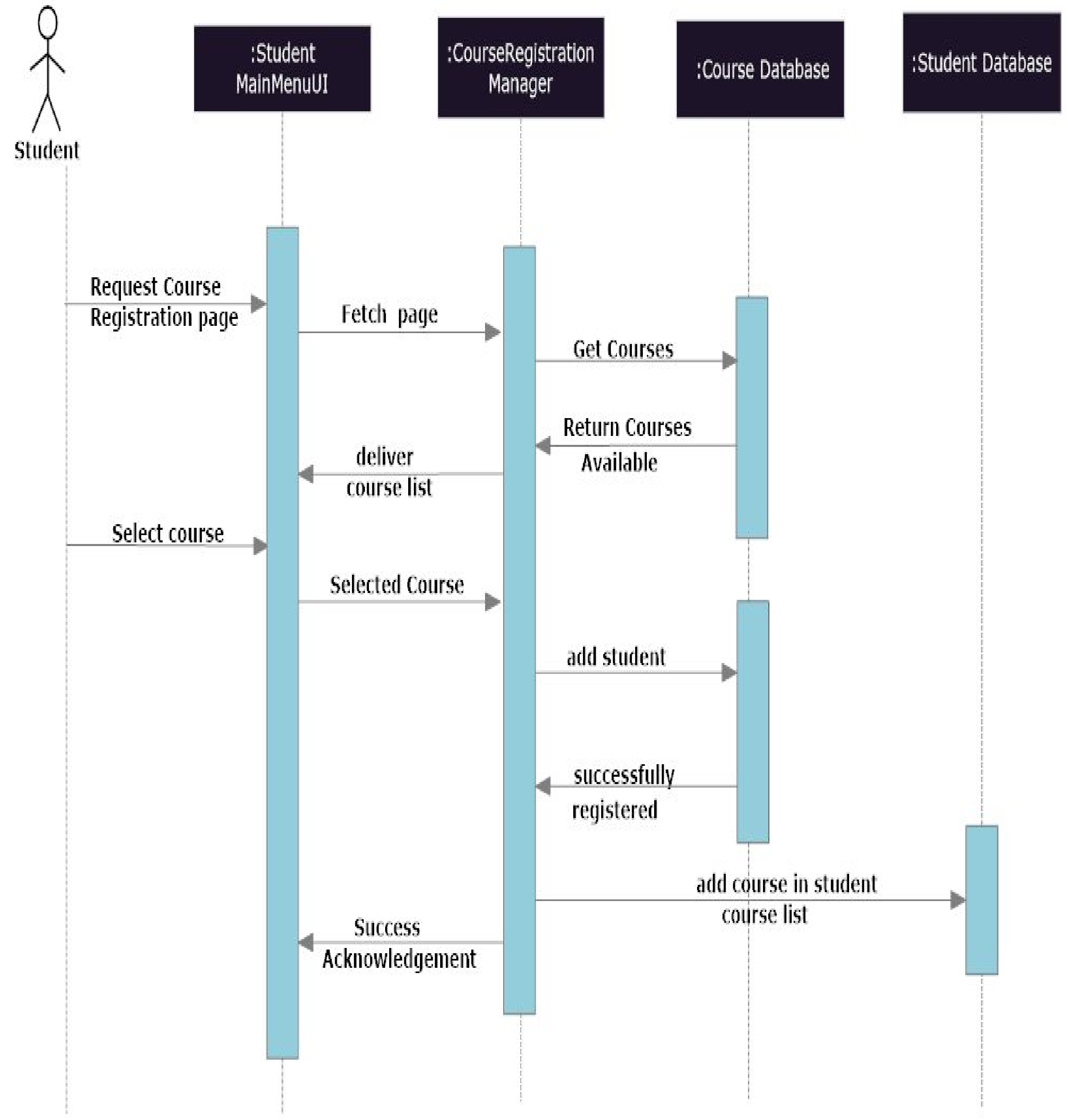
4.2 Use Case : Initiate Session



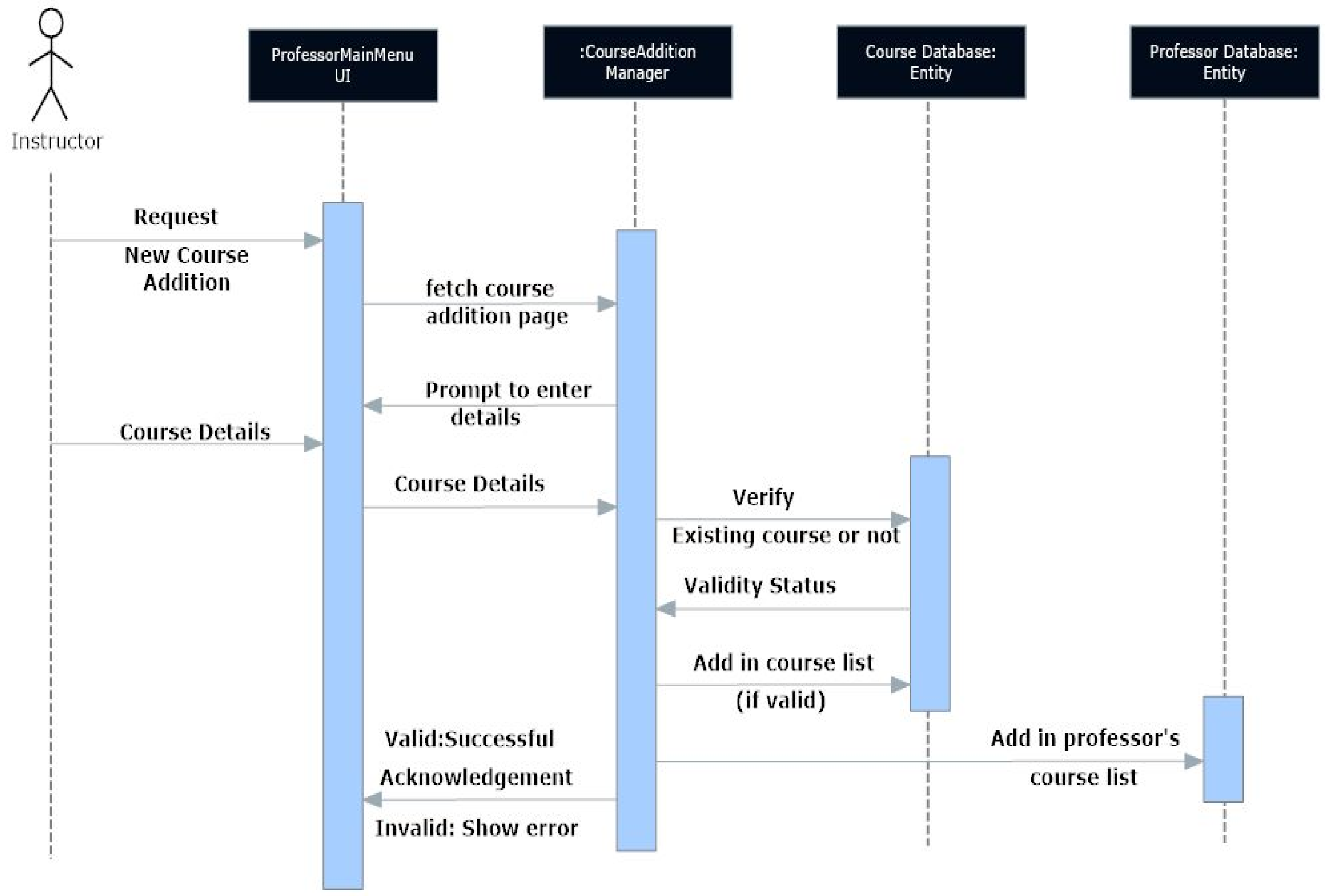
4.3 Use Case : Join Session


Student
Student Main Menu: Boundary
:Node Socket Client
Node Socket Server
Request Join session
Accept Request
Prompt for course
Select Course
Course Detail and lecture session key
Verify lecture session key
Verification Status
If Invalid:Wrong session key entered
If valid:Prompt for QR scan
Scan QR
Receive QR value
Store Student's position in lecture
Successful joining session

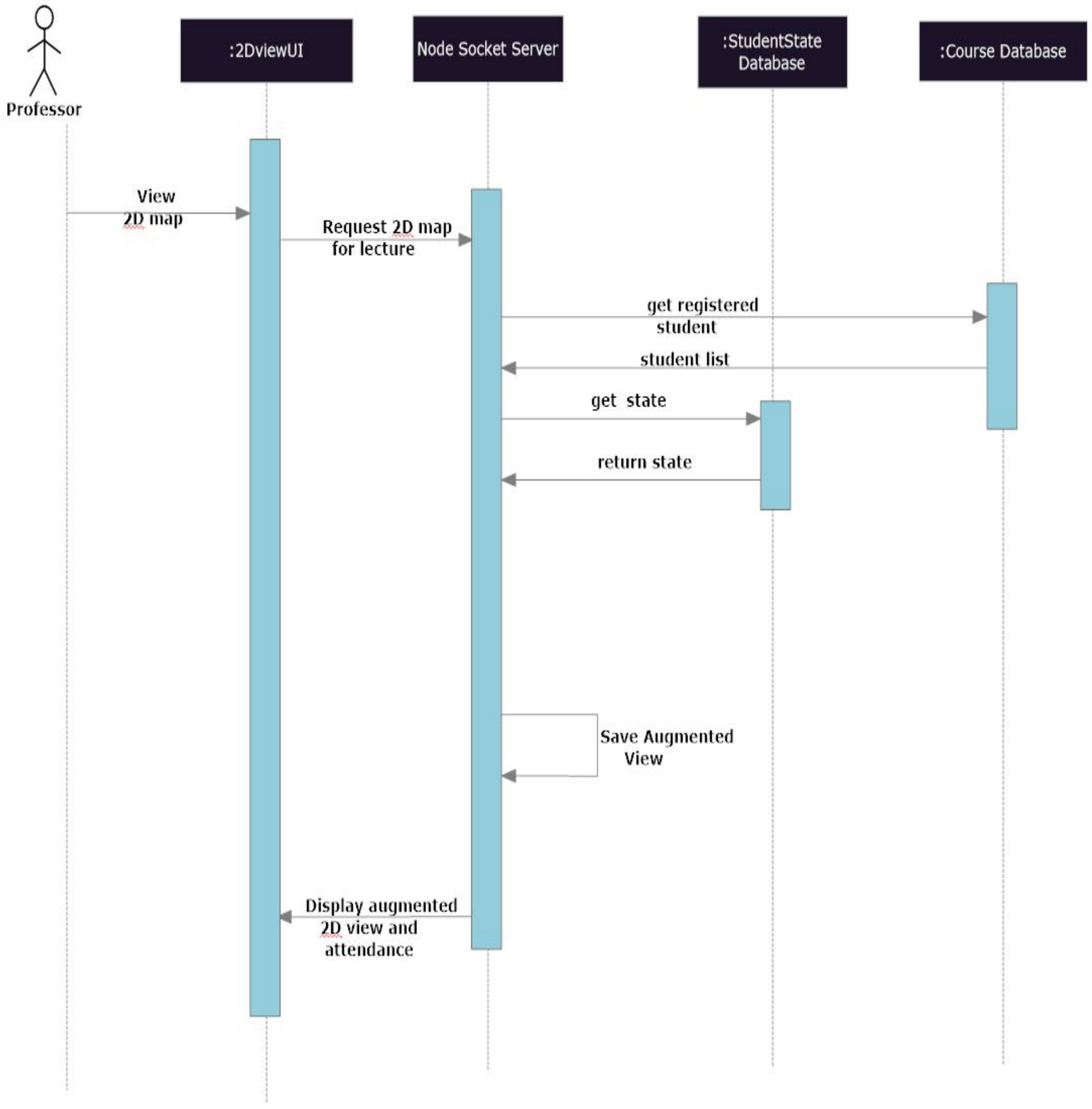
4.4 Use Case : Register Course



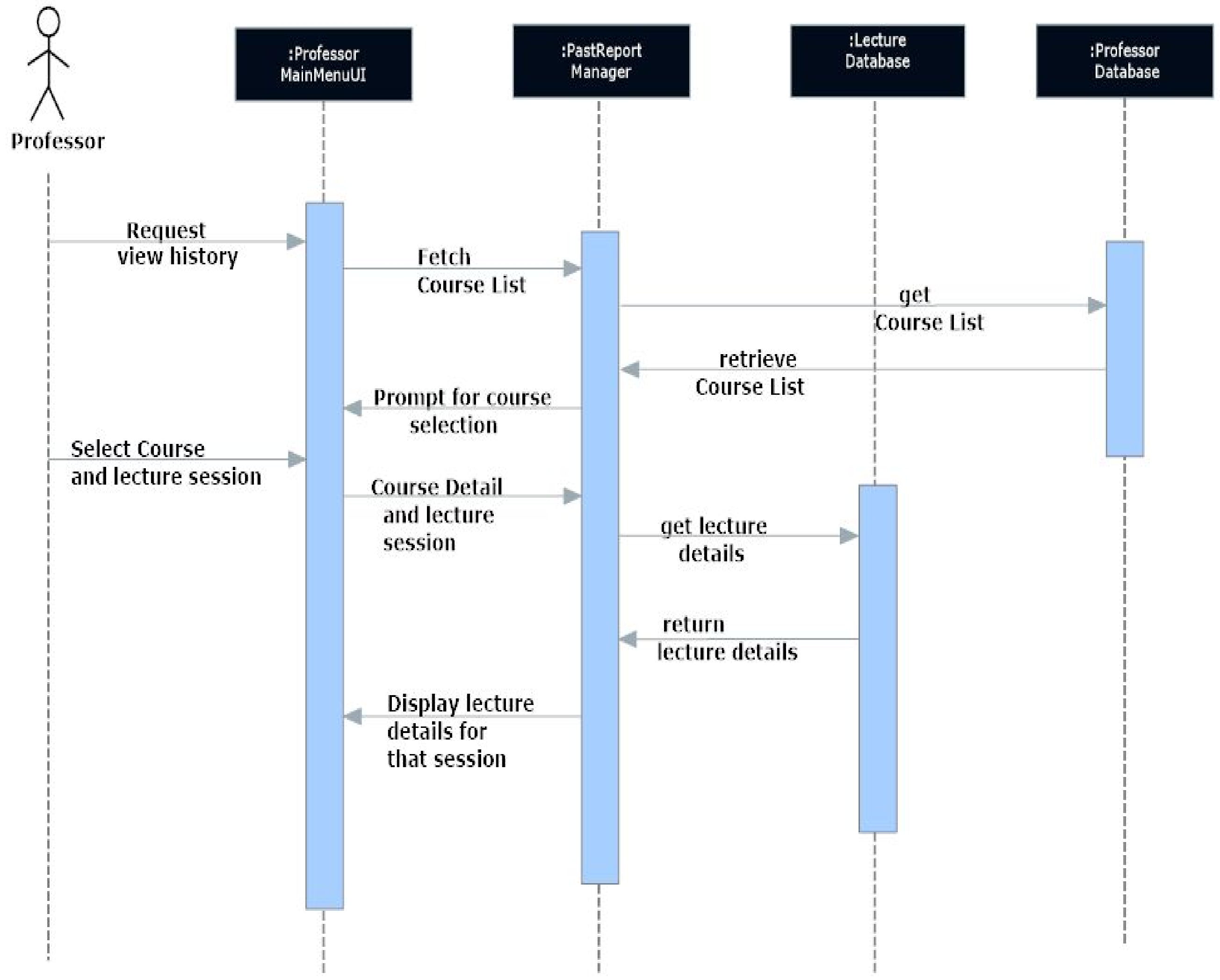
4.5 Use Case :Add Course


Instructor
ProfessorMainMenu UI
:CourseAddition Manager
Course Database: Entity
Professor Database: Entity
Request New Course Addition
fetch course addition page
Prompt to enter details
Course Details
Course Details
Verify Existing course or not
Validity Status
Add in course list (if valid)
Valid:Successful Acknowledgement
Invalid: Show error
Add in professor's course list

4.6 Use Case :View 2D map



4.7 Use Case :View Past Report



5. Object Oriented Model

5.1 Domain Modelling

1. Boundary Object

- LoginUI
- ProfessorMainMenuUI
- StudentMainMenuUI
- 2DviewUI

2. Controller Object

- LoginManager
- Manage_Session
- CourseRegistrationManager
- CourseAdditionManager
- Socket : IO.socket-client
- Past ReportManager
- Socket : server

3. Entity Object

- Professor
- Student
- Course
- Lecture
- Classroom
- StudentStateDb

5.2. Class Description

5.2.1 LoginUI

Description : Interacts with the user and take user's credential and send it for verification.

Attributes

1. <String> profile
2. <String> userName
3. <String> password

Methods

1. set_credentials()
    - Parameters : <String> (profile),<String>(userName),<String>(password)
    - Return Value : void
    - Description : This method sets the credentials of user
    - Called By : method is called in the main program/activity to get input from user and store it

2. call_VerifyLogin()
    - Parameters : void
    - Return Value : boolean
    - Description : This method calls the controller to verify credentials and returns success of failure (0 or 1)
    - Called By : method is called in the main program/activity to get input from user
    - Calls : this function calls the controller to verify the credentials

5.2.2 ProfesserMainMenuUI

Description : It interacts with professor and provides a gateway to add new course to his course list or select course to initiate session.

Attributes

1. <Professor-object> currentProfessor
2. <Course-object> selectedCourse
3. <Course-object> newCourse

Methods

1. Constructor : ProfessorMainMenu()
    - Parameters : <Professor-object>
    - Return Value : void
    - Description : This constructor initialises currentProfessor
    - Called By : LoginManager

2. identify_Response()
    - Parameters : User Response
    - Return Value : void
    - Description : This method let user(here professor) choose initiate session or add new course option. Based on user selection it calls relevant function.
    - Called By : method is called in the main program/activity to get input from user
    - Calls : select_Course() or add_newCourse()

3. select_Course()
    - Parameters : void
    - Return Value : <Course-object>
    - Description : This method let the user select a course for which session is to be initiated.
    - Called By : identify_Response()

4. add_newCourse()
    - Parameters : <Course-object>
    - Return Value : boolean
    - Description : This method receives details for the new course to be created.
    - Called By : identify_Response()

5. call_Controller()
    - Parameters : User response corresponding to select course or add new course
    - Return Value : boolean
    - Description : This method calls controller object based on response by user and passes parameters to it and the return value denotes whether the operation was successful or not.
    - Called By:main function after the type of response and its corresponding variable have been initialised
    - Calls : calls the controller object according to user response

6. view_PastRecord()
    - Parameters : <course-object>,<session details>
    - Return Value : boolean
    - Description : This method display past session details
    - Called By:method is called in the main program/activity to get input from user
    - Calls : calls the controller object to bring lecture details.

7. terminate_Session()
    - Parameters : void
    - Return Value : boolean
    - Description : This method calls terminates the socket server connection for the currently active session ,at a time only one session is active so there is no ambiguity about the course whose session has to be terminated.
    - Called By : method is called in the main program/activity to get input from user
    - Calls:calls the controller object Manage_Session to inform about user’s request for termination

5.2.3 StudentMainMenuUI

Description : It creates a main menu UI for student through which student can register in a new course or join session.

Attributes

1. <Student-object> currentStudent
2. <Course-object> selectedCourse
3. <Course-object> list[ ]
4. <String> QR value
5. <Course-object> newCourse
6. <String> sessionKey

Methods

1. Constructor : StudentMainMenu( )
    - Parameters : <Student-object>
    - Return Value : void
    - Description : This constructor initialises currentStudent
    - Called By : LoginManager

2. identify_Response()
    - Parameters : User Response
    - Return Value :void
    - Description : This method let user choose join session or register new course option. Based on user selection it calls relevant function.
    - Called By : method is called in the main program/activity to get input from user
    - Calls : select_Course() or register_newCourse()

3. select_Course()
    - Parameters :void
    - Return Value : <course-object>
    - Description : This method let the student select a course for joining a session which has been initiated from his/her course list.
    - Called By : identify_Response()

4. register_newCourse()
    - Parameters : <Course-object>
    - Return Value : boolean
    - Description : This method identifies the course which student wants to register .
    - Called By : identify_Response()

5. call_Controller()
    - Parameters : User response corresponding to select course or register_newcourse
    - Return Value : boolean
    - Description : This method calls controller object based on response by user and

passes parameters to it returning successful or unsuccessful.
- Calls : calls the controller object according to user response

6. set_Qr()
    - Parameters : scanned QR value
    - Return Value : void
    - Description : This method let user scan its QR code and initialises QR attribute
    - Called By : method is called in the main program/activity to get input from user

5.2.4 2DViewUI

Description : 2DviewUI is a boundary class that displays a 2D graphical view of classroom with augmented states of students who have joined session.

Attributes

1. <Lecture-object> lecture
2. <2D-map> classMap

Methods

1. Constructor : StudentMainMenu()
    - Parameters : <lecture-object>,<2d-map>
    - Return Value : void
    - Description : This constructor initialises the attribute lecture
    - Called By : This constructor is called when the professor clicks on view 2D map button to initialise the current lecture

2. display_2Dmap()
    - Parameters : classMap
    - Return Value : void
    - Description : This function displays map on the screen
    - Called By : method is called in the main program/activity and refresh_View()

3. refresh_View()
    - Parameters : void
    - Return Value : void
    - Description : This function refreshes the class map every 10 minute.
    - Called By : method is called in the main program/activity
    - Calls : calls the controller object to refresh class map and then display_2Dmap()

4. call_Details()
    - Parameters : <student-object>
    - Return Value : void
    - Description : When a prof selects a specific student in class view , this function bring details about a student and displays it on screen.
    - Called By : This function is invoked when a specific student is selected.

- Calls : popup_Details()

5. popup_Details()
    - Parameters : <student details>
    - Return value : void
    - Description : This function display student details on the screen such as student state throughout the class ,name, roll number etc.
    - Called By : call_Details()

5.2.5 LoginManager

Description : It verify the user credentials .

Attributes

1. <Professor-object> ProfessorList[]
2. <Student-object> StudentList[]
3. <String> profile
4. <String> userName
5. <String> password

Methods

1. Constructor: set_Credentials()
    - Parameters : <String>(profile),<String>(username) ,<String>(password)
    - Return Value : void
    - Description : This method initialises the attributes to verify credentials and initialises the StudentList[] and ProfessorList[ ]
    - Called By : call_verifyLogin() in the LoginUI

2. verifyLogin()
    - Parameters : void
    - Return Value : boolean
    - Description : This function matches the credential from professor list or student list depending on the userprofile and returns whether user credentials have matched or not and then redirects to the user’s main page as per profile if a there has been a match.
    - Called By : call_verifyLogin() in the LoginUI

5.2.6 Manage_Session

Description : This class communicate with the node socket server and initiate or end session .

Attributes

1. <Professor-object> currentProfessor
2. <Course-object> selectedCourse
3. <String> sessionKey
4. <Lecture-object> currentlecture

Methods

1. Constructor: set_professor()
    - Parameters :<Professor-object>
    - Return Value : void
    - Description : This method initialises the currentProfessor to identify which professor is the user
    - Called By : function in ProfessorMainMenuUI

2. get_Courselist()
    - Parameters : void
    - Return Value : <Course-object> courselist[]
    - Description : This function retrieves the course list of the professor returning it to the boundary to show as options available to choose from.
    - Called By : select_course() function in ProfessorMainMenuUI

3. set_selectedCourse()
    - Parameters : <Course-object>
    - Return Value : void
    - Description : This function sets the value of selectedCourse attribute of the object denoting the course for which session has to be initiated .
    - Called By : function in ProfessorMainMenuUI

4. assign_sessionkey()
    - Parameters : void
    - Return Value : <String> sessionKey
    - Description : This function sets unique sessionkey for this session
    - Called By : function in ProfessorMainMenuUI

5. request_seversession()
    - Parameters : void
    - Return Value : boolean
    - Description : This function is responsible for requesting the main node server to

create a new session interacting with which students join session
- Calls:constructor of the node server giving information about the lecture session.

5.2.7 CourseRegistrationManager

Description : This is a controller class that register student in course.

Attributes

1. <Student-object> currentStudent
2. <Course-object> courseList[]
3. <Course-object> newCourse

Methods

1. Constructor: set_Student()
   - Parameters : <Student-object>
   - Return Value : void
   - Description : This method initialises the currentStudent identifying the student who is the current user.
   - Called By : call in the StudentMainMenuUI

2. return_courselist()
   - Parameters : void
   - Return Value : <Course-object> courselist[]
   - Description : This function provides the available courses for the student to register from the course database.
   - Called By : Enroll_newcourse() in the StudentMainMenuUI

3. Register_Student()
   - Parameters : <Course-object>
   - Return Value : void
   - Description : This function adds student in course's student list and adds course in the student's course list
   - Called By : Call_EnrollCourse() in the StudentMainMenuUI

5.2.8 CourseAdditionManager

Description : Adds the new course to course database and update the course list in professor course list.

Attributes

1. <Professor-object> currentProfessor
2. <Course-object> courseList[]
3. <Course-object> newCourse

Methods

1. Constructor: set_Professor()
   - Parameters : <Professor-object>
   - Return Value : void
   - Description : This method initialises the currentProfessor identifying the professor who is the current user and courseList[] from course database.
   - Called By : when the object of this class is created

2. verify_courselist()
   - Parameters : <Course-object>
   - Return Value : <boolean>
   - Description : This function verifies whether a course with same details exists or not and returns the boolean corresponding to the value
   - Called By : function in the ProfessorMainMenuUI

3. Add_Course()
   - Parameters : <Course-object>
   - Return Value : <boolean>
   - Description : This function adds Course in professor's course list and adds course in the course database
   - Called By : function in the ProfessorMainMenuUI

5.2.9 Socket : IO.socket-client

Inherited from : IO

Description: socket-client is a library class that allows establishment and management of the connection between the student and Professor devices. It will be instantiated on the student device and manage connection from students point of view.

Attributes

1. <String> sessionKey
2. <String> socket.id
3. <Student-object> student
4. <Course-object> selectedCourse
5. <String> QRvalue

Methods

1. Constructor: socket()
   - Parameters : <Student-object>
   - Return Value : void
   - Description : initialises the socket.id when connection is created and identifies the student
   - Called By : Called when student joins a session and connected to professor server.

2. set_coursesession()
   - Parameters : <Course-object>,<String>
   - Return Value : void
   - Description : This function identifies the course of which student wants to attend the session and the session key
   - Called By : Triggered when the details about course and session key is entered by the student after connection is established

3. join_coursesession()
   - Parameters : void
   - Return Value : boolean
   - Description : This function calls node server to check whether the selectedcourse session is undergoing and the session key entered is correct or not
   - Called By : function in controller itself after the attributes are initialised
   - Calls : Socket:server object

4. set _QRvalue()
   - Parameters : <int>
   - Return Value : void
   - Description : This function calls node server to render the sitting position of the student in the session undergoing if the session key entered is correct
   - Called By : function in controller itself after the join_coursesession() validate the

credentials
- Calls : Socket:server object

5. onDisconnectEventListener()
    - Parameters : <boolean>
    - Return Value : void
    - Description : Informs students about connection problem
    - Called By : Triggered when the connection is destroyed

5.2.10 PastReportManager

Description: Retrieve past records of lecture.

Attributes:
1. <lecture-object> selectedSession
2. <professor-object> currentLecturer

Methods:
1. Constructor ()
    - Parameters : <professor-object>
    - Return Value : void
    - Description :initialises the current Professor
    - Called By :Called by boundary object

2. get_lecture()
    - Parameters : <lecture-object>
    - Return Value : void
    - Description :When user selects a lecture of a course for which he wants to view the record , this function returns that lecture detail.
    - Called By :whenever a user needs to see past lecture details .

5.2.11 Socket : server

Inherited from : IO
Description: A Socket is the fundamental class for interacting with browser clients. A Socket uses an underlying Client to communicate which is the student in this case

Attributes
1. <lecture-object> currentlecture
2. <Student-object> presentstudentlist[]
3. <Student-object> totalstudentlist[]
4. <Student_State_Db-object> states[]
5. <int> positionlist[]

Methods

1. Constructor: socket()
    - Parameters : <lecture-object>
    - Return Value : <String>(sessionkey)
    - Description : initialises the currentlecture when the professor initiates a session returning a session key using which students can join the session and also initialises the totalstudentlist[] using the course database and the course for which lecture session is undergoing
    - Called By : everytime a professor initiates a session for any of the his/her courses

2. add_student()
    - Parameters : <Student-object>,<int>(QRvalue)
    - Return Value : boolean
    - Description : This function appends the student in the presentstudentlist[] signifying marking of the student's attendance and sitting position in the classroom.This function is bounded by time limit all of the student must join session in fixed time interval after that joining is not valid
    - Called By : set_QRvalue() in the socketClient object

3. get_states()
    - Parameters : <int>(no of students present)
    - Return Value : <int> list[]
    - Description : This function initialises the state database attribute which holds the value of each student's state as the number of students increase the size of list returned by the function also increases matching the number of students.This function is called periodically many times as the session continues refreshing the state of the students.
    - Called By : the object itself periodically after fix time interval from the start of session till its termination
    - Calls:random function that returns value between 0 to 9

4. onTerminateEventListener()
    - Parameters : <boolean>
    - Return Value : void
    - Description : this function terminates the session and updates lecture database.
    - Called By : Triggered when a professor wants to end session
    - Calls:add_lecture() in the course object to add the lecture in its lecture list as history.

5. show_2Dview()
    - Parameters : void
    - Return Value : generated 2D model of class
    - Description : this function is used by the professor to graphically view the class

attendance ,the sitting position of each student augmented with symbols corresponding to their state value.
- Called by: main activity whenever professor demands for the 2D view

5.2.12 Professor

Description: Encapsulation of professor data.

Attributes

1. <string> name
2. <string> designation
3. <string> department
4. <string> user ID
5. <string> password
6. <Course-Object> Course_list[ ]

Methods

1. Constructor :professor( )
   - Parameters : <Course-object>list [],<String>,<String>,<String>,<String>,<String>
   - Return Value : void
   - Description : Constructor to initialise attributes
   - Called By : it is called when a new object of lecture is created.

2. get_Name( )
   - Parameters : void
   - Return Value :<String>
   - Description : This function returns the Professor’s name
   - Called By : whenever a function needs prof name this function is called.

3. get_Designation( )
   - Parameters : void
   - Return Value : <String>
   - Description : This function returns the Professor’s designation .
   - Called By : whenever a function needs professor designation this function is called.

4. get_Department( )
   - Parameters : void
   - Return Value : <String>
   - Description : This function returns the Professor’s department.
   - Called By : whenever a function needs professor department this function is called.

5. get_UserName( )

- Parameters : void
- Return Value : <String>
- Description : This function returns the Professor's username.
- Called By : whenever a function needs professor username this function is called.

6. get_CourseList( )
   - Parameters : void
   - Return Value : Course_list[ ]
   - Description : This function returns the list of courses taught by professor .
   - Called By : whenever a function needs courses taught by professor this function is called.

7. add_course ( )
   - Parameters : course
   - Return Value : updated course_list[ ]
   - Description : This function appends the new course in the course list .
   - Called By : whenever professor adds a new course.

5.2.13 Student

Description: Encapsulation of student data.

Attributes

1. <String> name
2. <String> rollNo
3. <String> department
4. <String> userName
5. <String> password
6. <Course-object> courseList [ ]

Methods

1. student(CONSTRUCTOR)
   - Parameters : <Course-object> [ ], <String>,<String>,<String>,<String>
   - Return Value : void
   - Description : Constructor to initialise attributes
   - Called By : it is called when a new object of student is created.

2. get_name()
   - Parameters : void
   - Return Value : <String>
   - Description : This function returns the student's name
   - Called By : whenever a function needs student's name this function is called.

3. get_RollNo( )
   - Parameters : void

- Return Value : <String>
- Description : This function returns Roll Number of student.
- Called By : whenever a function needs student roll number this function is called.

4. get_Department()
   - Parameters : void
   - Return Value : <String>
   - Description : This function returns the Student's department.
   - Called By : whenever a function needs student's department this function is called.

5. get_UserName()
   - Parameters : void
   - Return Value : <String>
   - Description : This function returns username of student.
   - Called By : whenever a function needs student's username this function is called.

6. get_CourseList()
   - Parameters : void
   - Return Value : Course_list[ ]
   - Description : This function returns the courses in which student is registered.
   - Called By : Whenever a function needs course list in which student is registered function is called.
7. add_course()
   - Parameters : void
   - Return Value : updated Course_list[ ]
   - Description : This function appends the new course in students course list.
   - Called By : Whenever students register in a new course.

5.2.14 Course

Description: Encapsulates the course data.

Attributes

1. <String> name
2. <String> courseID
3. <Professor-object>lecturer
4. <Classroom-object>room
5. <Student-object>studentList [ ]
6. <Lecture-object>lectureList[ ]

Methods

1. Constructor : course()
   - Parameters : <String>,<String>,<String>,<String>, studentList[ ], lectureList[ ]
   - Return Value : void
   - Description : Constructor to initialise attributes
   - Called By : it is called when a new object of lecture is created.

2. get_Name()
   - Parameters :void
   - Return Value : <String>
   - Description : Returns Course name
   - Called By : it is called when courseName need to be accessed.

3. get_CourseID()
   - Parameters :void
   - Return Value : <String>
   - Description : Returns CourseID
   - Called By : it is called when courseID attribute of this class need to be accessed.

4. get_Lecturer()
   - Parameters :void
   - Return Value : <Professor-object>
   - Description : Returns Professor of this course
   - Called By : it is called when Lecturer attribute of this class need to be accessed.

5. get_Room()
   - Parameters :void
   - Return Value : <Classroom-object>
   - Description : Returns room for this course
   - Called By : it is called when room attribute of this class need to be accessed.

6. get_SutudentList()
   - Parameters :void
   - Return Value : <Student-object> list
   - Description : Returns students registered in this course
   - Called By : it is called when student list attribute of this class need to be accessed.

7. get_LectureList()
   - Parameters :void
   - Return Value : <Lecture-object> list
   - Description : Returns lectureList for this course
   - Called By : it is called when lecture list attribute of this class need to be accessed.

5.2.15 Lecture

Description: Encapsulation of Lecture data.

Attributes

7. <Course-object>courseName
8. <Student-object>presentStudentList [ ]
9. <String> sessionKey
10. <2D array>position

Methods

1. Constructor : lecture()
    - Parameters : <Course-object>,<String>, position
    - Return Value : void
    - Description : Constructor to initialise attributes
    - Called By : it is called when a new object of lecture is created.

2. get_CourseName()
    - Parameters :void
    - Return Value : <Course-object>
    - Description : Returns Course corresponding to this lecture
    - Called By : it is called when courseName attribute of this class need to be accessed.

3. get_PresentStudentList()
    - Parameters :void
    - Return Value : <Student-object>
    - Description : Returns students who are present in lecture
    - Called By : it is called when studentList attribute of this class need to be accessed.

4. get_Position()
    - Parameters :void
    - Return Value : <2D array>
    - Description : Returns position of students
    - Called By : it is called when position attribute of this class need to be accessed.

5. get_SessionKey()
    - Parameters :void
    - Return Value : <String>
    - Description : Returns Course corresponding to this lecture
    - Called By : it is called when sessionKey attribute of this class need to be accessed.
6. add_Student()
    - Parameters :<Student-object>
    - Return Value : void

- Description :Add a student to presentStudentList[ ]
- Called By : it is called when a student need to be added in this lecture

5.2.16 Classroom

Description: Encapsulates classroom data.

Attributes

1. <int> row
2. <int> column

Methods

1. Constructor : classroom()
    - Parameters :row, column
    - Return Value : void
    - Description : Constructor to initialise attributes
    - Called By : it is called when a new object of lecture is created.

2. get_Row()
    - Parameters :void
    - Return Value : <int>
    - Description : Returns row in class
    - Called By : called when row attribute of the class needs to be accessed

3. get_Column()
    - Parameters :void
    - Return Value : <int>
    - Description : Returns column in class
    - Called By : called when column attribute of the class needs to be accessed

5.2.17 StudentStateDb

Description: Encapsulates student state database.

Attributes

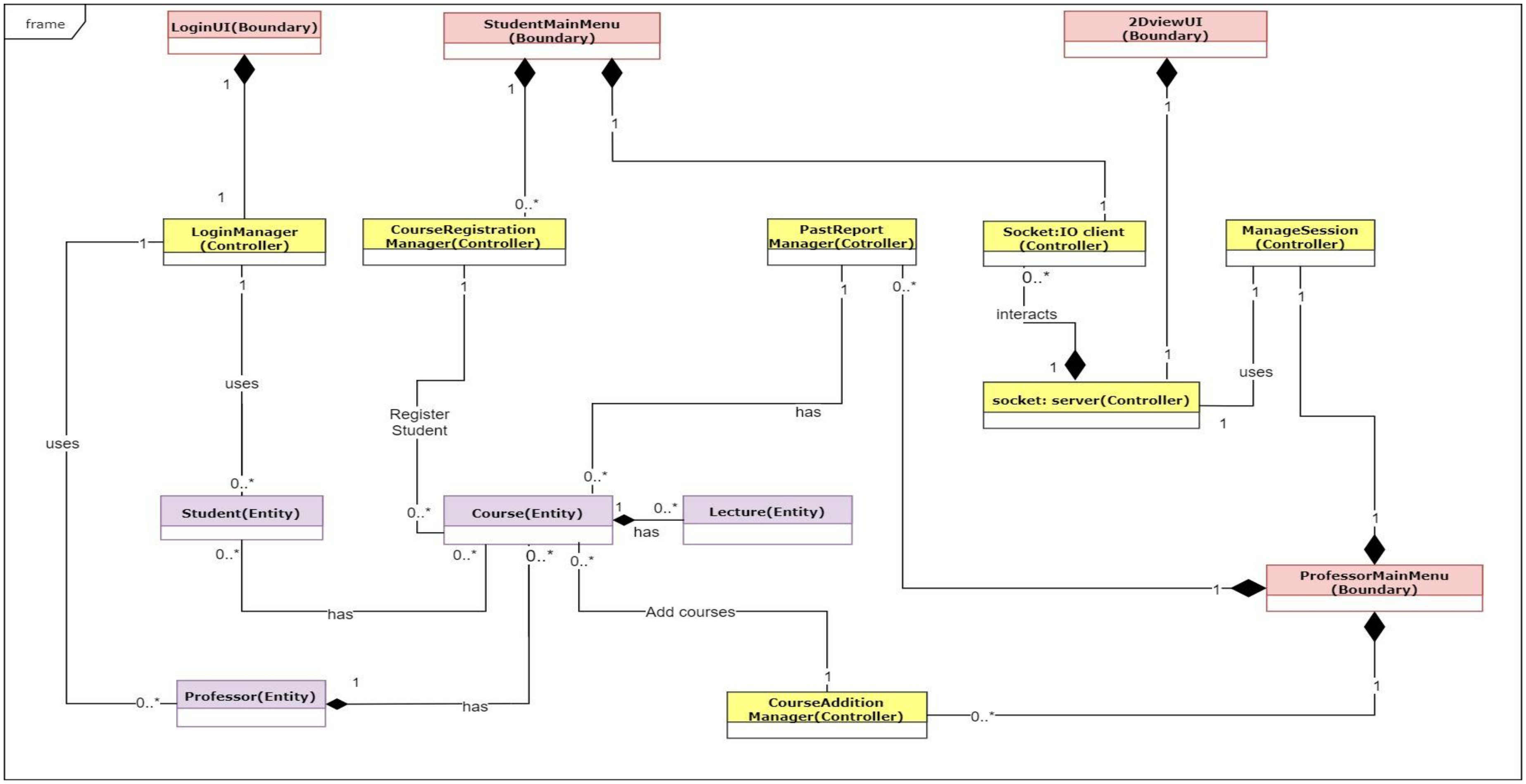
1. <Int> studentStateList[ ]

Methods

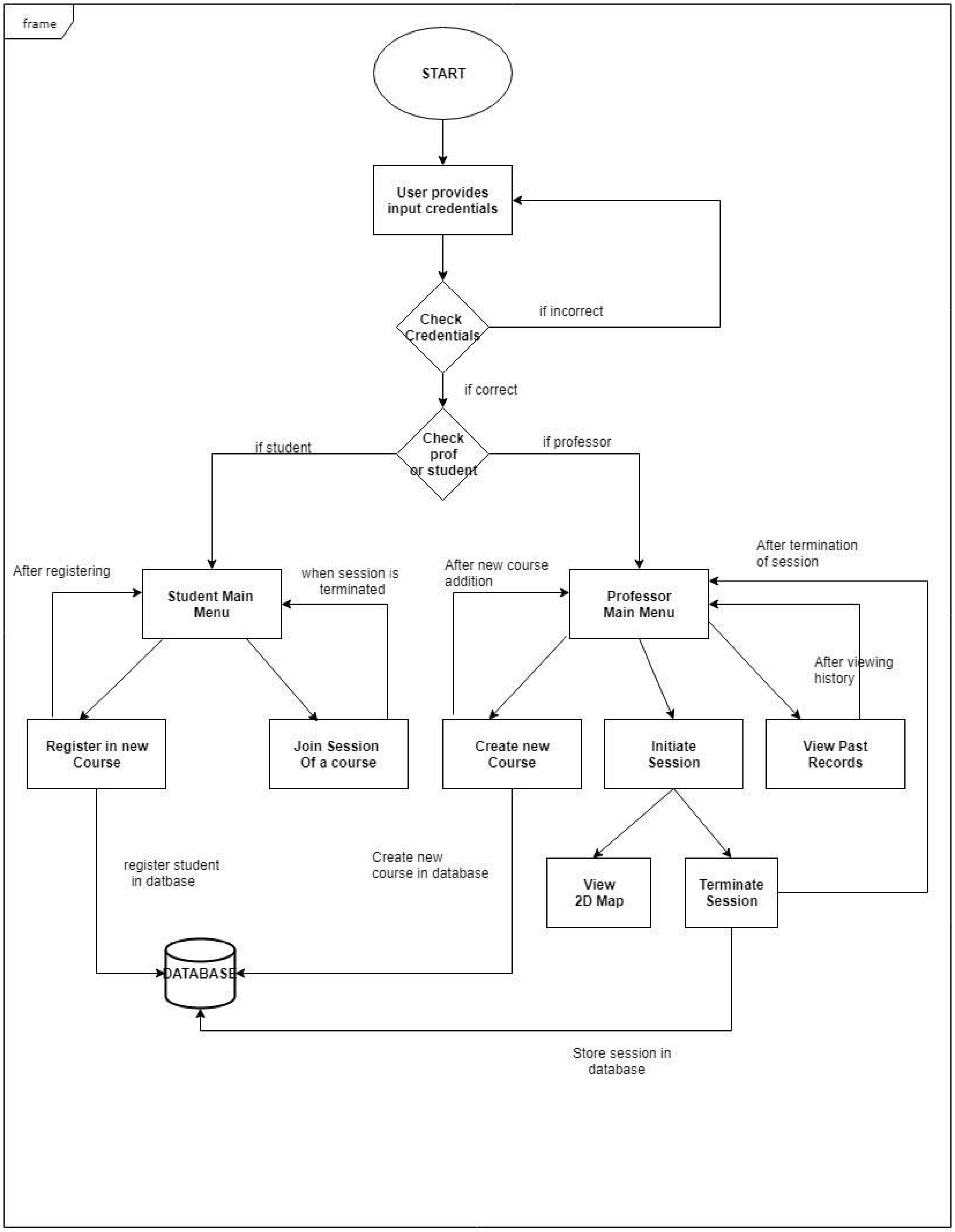
1. Constructor : state_Db()
    - Parameters :void
    - Return Value : void
    - Description : Initialises random values for student state
    - Called By : it is called when a new object of lecture is created.

2. get_StudentState()
    - Parameters : <Student-object>
    - Return Value : <int>
    - Description : Returns student state
    - Called By : called when state of a student is needed

6. Class Diagram


frame
LoginUI(Boundary)
StudentMainMenu (Boundary)
2DviewUI (Boundary)
LoginManager (Controller)
CourseRegistration Manager(Controller)
PastReport Manager(Cotroller)
Socket:IO client (Controller)
ManageSession (Controller)
socket: server(Controller)
Student(Entity)
Course(Entity)
Lecture(Entity)
ProfessorMainMenu (Boundary)
Professor(Entity)
CourseAddition Manager(Controller)
uses
Register Student
has
interacts
Add courses
1
0..*

7 . FlowChart for the System


frame
START
User provides input credentials
Check Credentials
if incorrect
if correct
Check prof or student
if student
if professor
After registering
Student Main Menu
when session is terminated
After new course addition
Professor Main Menu
After termination of session
After viewing history
Register in new Course
Join Session Of a course
Create new Course
Initiate Session
View Past Records
register student in datbase
Create new course in database
View 2D Map
Terminate Session
DATABASE
Store session in database