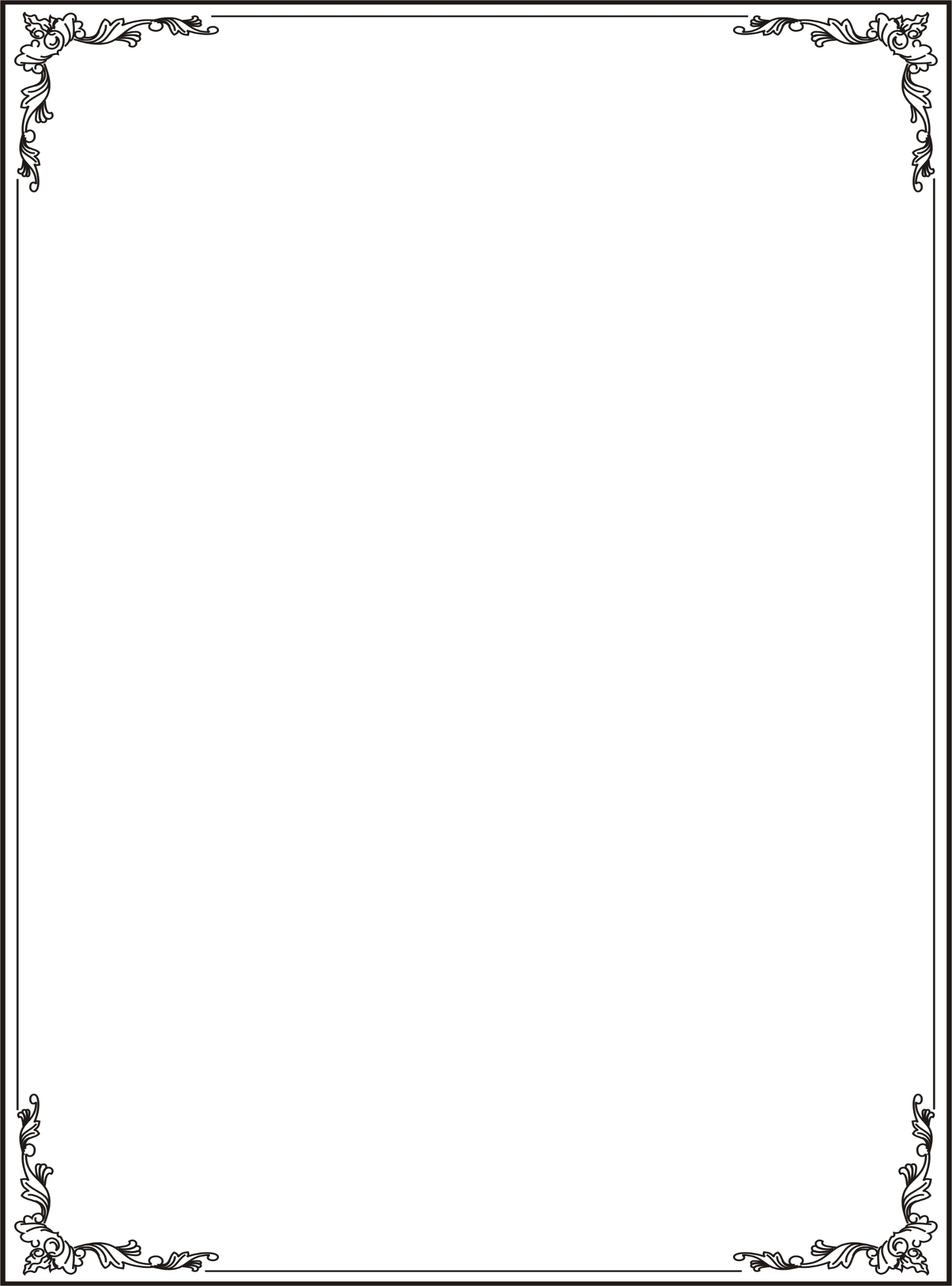
**VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF INFORMATION SYSTEMS**

**CAPSTONE PROJECT FINAL REPORT**

**INFORMATION SYSTEM ANALYSIS AND DESIGN**

**ENGLISH CENTER MANAGEMENT SYSTEM**

Nguyễn Chí Kha – 21522179

Phùng Thiên Phúc – 21521297

Nguyễn Quang Lâm – 21521057

Phạm Mạnh Hùng – 21520901

**MỤC LỤC**

[Chapter 1: INTRODUCTION 5](#_Toc138086269)

[1.1. Introduction 5](#_Toc138086270)

[1.2. Research problems / Problems of the project 5](#_Toc138086271)

[1.3. Research Objectives / Objectives of the project 6](#_Toc138086272)

[1.4. Scope of project / Limitations of the study 6](#_Toc138086273)

[Chapter 2: REQUIREMENT SPECIFICATION 7](#_Toc138086274)

[2.1. Situation survey/ Survey of the current situation 7](#_Toc138086275)

[2.1.1 Interview 7](#_Toc138086276)

[2.1.2. Survey of the organization/ Current organizational situation 11](#_Toc138086277)

[2.1.3. Survey of business processes, major activities 11](#_Toc138086278)

[2.1.3.1: Academic Department Business: Responsible for designing and delivering language courses, managing the curriculum, and providing academic support to students. 11](#_Toc138086279)

[2.1.3.2: Tutors 11](#_Toc138086280)

[2.1.3.3: Marketing Department: 12](#_Toc138086281)

[2.1.3.4: Technology Department: 12](#_Toc138086282)

[2.2. Software requirements 12](#_Toc138086283)

[2.2.1. Functional requirements 12](#_Toc138086284)

[2.2.1.1 Storage function 12](#_Toc138086285)

[2.2.1.2 Feature function 13](#_Toc138086286)

[2.2.2. Non-functional requirements 13](#_Toc138086287)

[Chapter 3: ANALYSIS AND DESIGN 14](#_Toc138086288)

[3.1. Use-case Diagram 14](#_Toc138086289)

[3.1.1 List of actors 18](#_Toc138086290)

[3.1.2 List of Use-case 18](#_Toc138086291)

[3.2. Use-case specification and activity diagram (for administrator) 20](#_Toc138086292)

[3.2.1 Login 20](#_Toc138086293)

[3.2.2 Manage student information 22](#_Toc138086294)

[3.2.3: Manage tutor information: 25](#_Toc138086295)

[3.2.4: Manage class information: 28](#_Toc138086296)

[3.2.6 Manage Lesson Report 34](#_Toc138086297)

[3.2.7 Manage courses 36](#_Toc138086298)

[3.2.8 Manage room schedule 40](#_Toc138086299)

[3.2.9 Manage statistic report 41](#_Toc138086300)

[3.3. Use-case specification and activity diagram (for tutors) 43](#_Toc138086301)

[3.3.1 Login 43](#_Toc138086302)

[3.3.2 Manage tutor information 46](#_Toc138086303)

[3.3.3 Manage lesson report 48](#_Toc138086304)

[3.3.3.1 Check attendance 48](#_Toc138086305)

[3.3.3.2 Study content 50](#_Toc138086306)

[3.3.4: Manage student grades. 52](#_Toc138086307)

[3.3.5 Statistic report 56](#_Toc138086308)

[3.3.6 Tutor’s schedule 57](#_Toc138086309)

[3.3.7 Room schedule 58](#_Toc138086310)

[3.4 Sequence diagram and class diagram (for administrator) 59](#_Toc138086311)

[3.4.1 Login in 59](#_Toc138086312)

[3.4.1.1 Sequence diagram 59](#_Toc138086313)

[3.4.1.2 Class diagram 60](#_Toc138086314)

[3.4.2 Manage student information 60](#_Toc138086315)

[3.4.2.1 Sequence diagram 60](#_Toc138086316)

[3.4.2.2 Class diagram 64](#_Toc138086317)

[3.4.3 Manage tutor information 65](#_Toc138086318)

[3.4.3.1 Sequence diagram 65](#_Toc138086319)

[3.4.3.2 Class diagram 68](#_Toc138086320)

[3.4.4 Manage classes 69](#_Toc138086321)

[3.4.4.1 Sequence diagram 69](#_Toc138086322)

[3.4.4.2 Class diagram 73](#_Toc138086323)

[3.4.5 Manage lesson reports 74](#_Toc138086324)

[3.4.5.1 Sequence diagram 74](#_Toc138086325)

[3.4.5.2 Class diagram 76](#_Toc138086326)

[3.4.6 Manage room schedule 76](#_Toc138086327)

[3.4.6.1 Sequence diagram 76](#_Toc138086328)

[3.4.6.2 Class diagram 77](#_Toc138086329)

[3.4.7 Manage statistic report 77](#_Toc138086330)

[3.4.5.1 Sequence diagram 77](#_Toc138086331)

[3.4.5.2 Class diagram 77](#_Toc138086332)

[3.5 Sequence diagram and class diagram (for tutor) 78](#_Toc138086333)

[3.5.1 Login in 78](#_Toc138086334)

[3.5.1.1 Sequence diagram 78](#_Toc138086335)

[3.5.1.2 Class diagram 79](#_Toc138086336)

[3.5.2 Manage tutor information 81](#_Toc138086337)

[3.5.2.1 Sequence diagram 81](#_Toc138086338)

[3.5.2.2 Class diagram 83](#_Toc138086339)

[3.5.3 Manage lesson reports 86](#_Toc138086340)

[3.5.3.1 Sequence diagram 86](#_Toc138086341)

[3.5.3.2 Class diagram 89](#_Toc138086342)

[3.5.4 Manage grade report 102](#_Toc138086343)

[3.5.4.1 Sequence diagram 102](#_Toc138086344)

[3.5.4.2 Class diagram 107](#_Toc138086345)

[3.5.5 Statistic report 114](#_Toc138086346)

[3.5.5.1 Sequence diagram 114](#_Toc138086347)

[3.5.5.2 Class diagram 115](#_Toc138086348)

[3.5.6 Tutor’s schedule 120](#_Toc138086349)

[3.5.6.1 Sequence diagram 120](#_Toc138086350)

[3.5.6.2 Class diagram 121](#_Toc138086351)

[3.5.7 Room schedule 125](#_Toc138086352)

[3.5.7.1 Sequence diagram 125](#_Toc138086353)

[3.5.7.2 Class diagram 126](#_Toc138086354)

[3.6 State diagram 130](#_Toc138086355)

[3.6.1 Student 130](#_Toc138086356)

[3.6.2 Class 131](#_Toc138086357)

[3.6.3 Classroom 132](#_Toc138086358)

[3.6.5 Tutor 133](#_Toc138086359)

[Chapter 4: Implementation and testing 133](#_Toc138086360)

[4.1 Introduction to the implementation environment. 133](#_Toc138086361)

[4.2 User interface. 134](#_Toc138086362)

[4.2.1 Tutor’s interface. 134](#_Toc138086363)

[4.2.2 admin’s interface. 142](#_Toc138086364)

[Chapter 5: Conclusion 147](#_Toc138086365)

[5.1 Achievements 147](#_Toc138086366)

[5.2 Limitations 148](#_Toc138086367)

# Chapter 1: INTRODUCTION

## 1.1. Introduction

* In today's time, language learning has become an important and indispensable skill for students. In order to meet that huge demand, thousands of startup centers /tutoring classes have been opened and attracted a large number of students, however, in order to be able to operate and manage the English center smoothly and effectively, it is necessary to apply technologies and software in the Information Technology era. Understanding that problem, the group of Sweet Daddies decided to choose the topic “Building English center management software”. In which the development of an information system diagram is an important and essential part for software development, due to lack of experience in practice, the team will inevitably make mistakes. We will do our best to look forward to receiving your comments to improve the topic even more.

## 1.2. Research problems / Problems of the project

* Because the demand for synchronous and efficient management increases, a proposed solution is an application (web-based, mobile-based, desktop-based) for the purpose of managing the business efficiently. It’s not only convenient for students to retrieve information about their classes, time, grade, tuition, etc. but also easy to perform actions like choosing a class, choosing a tutor, evaluating business, etc. In addition, managers can access the mutual information containing lots of classes with variant types of persons (tutor, student, sanitation worker, etc.). The tutor can manage their students such as marks, time in class, and document; managers are also able to watch Student’s information to evaluate and upgrade the quality of the tutor. Nowadays, almost all English centers have applied technology to management independently

## 1.3. Research Objectives / Objectives of the project

* In the scope of this subject, we would like to create an software not only responsible for allowing manager manage their business conveniently and effectively, but also help tutor and student know their status of learning and teaching
* There are some our systems main features:

• Upload notification

• Register for a course

• Store information of student, tutor

• Manage room, tutor, timetable, revenue, quality of business

## 1.4. Scope of project / Limitations of the study

* Because lack of experience, expense, human resources, knowledge. Our team focus on several main features:
* Timekeeping.
* View user information .
* Register for a class.
* Upload notifications .

# Chapter 2: REQUIREMENT SPECIFICATION

## 2.1. Situation survey/ Survey of the current situation

### 2.1.1 Interview

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **General survey**  System: True English Educator, Toeic Dang Ngoc.  Creator: Pham Manh Hung Created date: 20/5/2023 | | | | |
| Ord | Topic | Requirement | Start | End |
| 1 | Class management | To get detailed information about type of class, class output standard, tutor, wholesale | 20/5/2023 | 20/5/2023 |
| 2 | Employee management | To get detailed information about tutor, accountant in order to ensure the quality of teaching as well as revenue | 20/5/2023 | 20/5/2023 |
| 3 | Analysis and report | Process of making report on revenue, teaching progress of classes. Analyze the effectiveness of attracting students | 20/5/2023 | 20/5/2023 |
| 4 | Physical status | Physical equipment such as seats,  rooms, cash registers, ... | 20/5/2023 | 20/5/2023 |

**Table 1. Survey strategy**

|  |  |
| --- | --- |
| **Survey forms**  System: True English Educator | |
| Interviewee: Nguyen Dinh Quoc Trung | Interviewer: Pham Manh Hung |
| Place: VNUHCM Campus  Target: collect and understand working progress of business | Start: 8:00 am 21/5/2023  End: 10:00 am 21/5/2023 |
| Details of the interview:  • Introduction  • Overview about current system  • Survey plan:   * Topic 1: Class management * Topic 2: Employee management * Topic 3: Analysis and report * Topic 4: Physical status | Estimate time:   * 5 minutes * 15 minutes * 30 minutes |
| General observe |  |
| Unexpected occurrence |  |

**Table 2. Survey form**

|  |  |  |
| --- | --- | --- |
| Interviewee: Nguyen Dinh Quoc Trung | | 21/5/2023 |
| *question* | | *records* |
| Topic 1: class management | There are how many type of classes is opening ? | There are two types of classes, one is training, the first one is specializes in toeic exam training, the second one is specializes in ielts exam training |
| When do the opening classes for student enrollment take place ? | Classes will be open every 3 month for toeic classes, the other one is every 6 month for ielts classes. But it sometimes Admissions right after a classes end |
| What would happen if number of registered students is not enough ? | Enhance marketing and attracting students, if not as expected, that class may be cancel |
| How is the quality of the class assessed ? | By following the result of the test, by the way, they often make a form to get evaluate from students |
| Topic 2: employee management | How can you evaluate the level of tutors ? | Every tutor applying to business needs to show the degree or must pass an exam. In addition, tutors will be evaluated again after class that they teach end |
| How can you count the number of time tutor teaching ? | Every tutor needs to timekeeping everytime they on work |
| What do you will do if one of tutors is low evaluated by students ? | There are a interview from that tutor and manager to solve the problem which is that tutor is encountered |
| Topic 3:  Analysis and report | How long does it take you to calculate the revenue of your business every month ? | It maybe 1-2 hour to calculate the revenue of business |
|  | What the statistic does the problem need ? | We need:   * automatic revenue calculation * notification display feature |
|  | How can you know the progress teaching of tutor ? | The lesson teaching in the class will be stored in database |
| Topic 4:  Physical status | Do you need to store the time of purchase of furniture? | This is not really necessary |
|  | Do you need to remember number of tables, chairs, boards ? | This helps me to manage the number of them |

**Table 3. Questions and reports**

### 2.1.2. Survey of the organization/ Current organizational situation

**A picture containing text, rectangle, screenshot, diagram

Description automatically generated**

### 2.1.3. Survey of business processes, major activities

#### 2.1.3.1: Academic Department Business: Responsible for designing and delivering language courses, managing the curriculum, and providing academic support to students.

* *Design course content process:*
* Design course content.
* Preprocess data and store into our database for future query process.
* Some kinds of data: course name, level, curriculum,…
* Manage curriculum process:
* Design curriculums for courses.
* Make adjustments to curriculums for certain desires.

#### 2.1.3.2: Tutors

Teaching in the center, including viewing teaching schedule, student attendance, grade management, teaching report.

* *Teaching schedule tracking process:*
* Keep track of the teaching schedule, including assigned class information, class schedule.
* *Class management process:*
* Manage student grade, attendance.
* Manage lesson content.

#### 2.1.3.3: Marketing Department:

Responsible for promoting the center and attracting new students.

* *Advertising process:*
* Propose various strategies to attract students.
* Inform the technical department for implementation.

#### 2.1.3.4: Technology Department:

Responsible for managing the center’s technology infrastructure, including hardware, software and online learning platform.

* *Information management:*
* *Manage tutor information.*
* *Manage student information.*
* *Manage class data.*
* *Manage classroom data.*
* *Operating system process:*
* *Hardware, software maintenance.*
* *Ensure system stability.*

## 2.2. Software requirements

### 2.2.1. Functional requirements

#### 2.2.1.1 Storage function

* The storage feature is one of the most important features of the system, which is the basis for implementing system's features. To ensure effective management, the system needs to store the following information:

• Tutor information: tutor ID, name, phone number, email address, address, date of birth

* Student information: student ID,name, phone number,email, address, date of birth

• Room information: room ID, room name, room number

• Class information: class ID, class name , tutor ID, room number, course name, time, start date, end date.

• Grade report: student ID, class ID, progress mark, midterm mark, term end mark, total

• Lesson report: class ID, lesson number, study content, date

#### 2.2.1.2 Feature function

* In addition, the system must also meet the following requirements:

• Query function: tutor can search to view their teaching progress, document, their students’ informationManagerser can also search for tutors’ information, students’ information, revenue.

• Statistical, analysis, reporting functions: help the manager monitor the general situation of the business.

• Update function: the system must ensure that the update function is correct and avoid errors due to the user's negligence.

• Delete function: the system abort the class which is not enough student or lack of tutor. There also abort students who cancel course or not complete the tuition responsibility.

### 2.2.2. Non-functional requirements

• Tutor GUI requirements: multilingual, friendly interface, highly interactive.

• The management GUI: simple, close as report as possible, easily to query.

Quality requirements:

• Evolution:

o Easy to upgrade or increase utility modules.

• Convenience:

o The system has an intuitive, user-friendly and easy-to-use interface.

o Clear specifications and instructions for use.

• Efficiency:

o The website shows the information of tutor and dashboard of manager well.

o The website works stably and reliably.

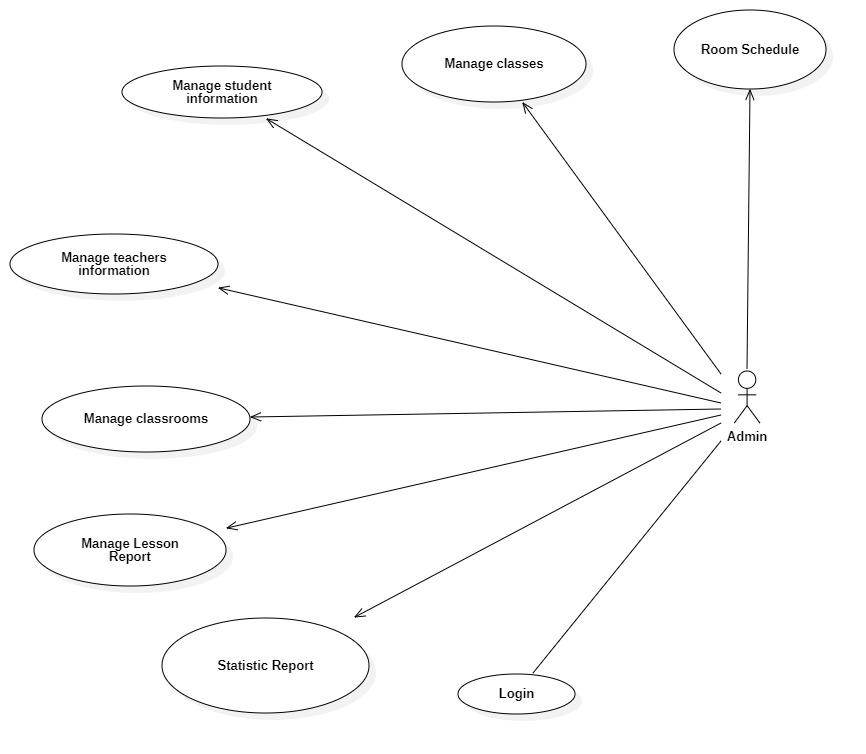
• Compatibility:

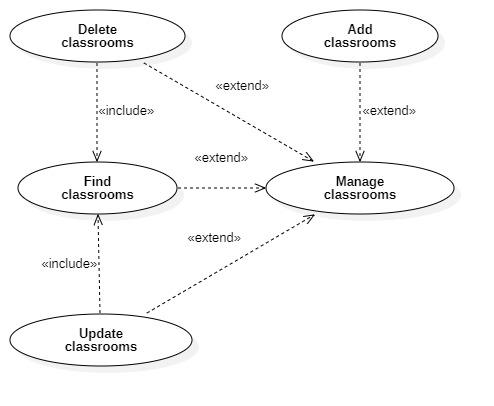
o Multi-adaption with popular web browsers (Internet Explorer, Fire Fox, Chrome).

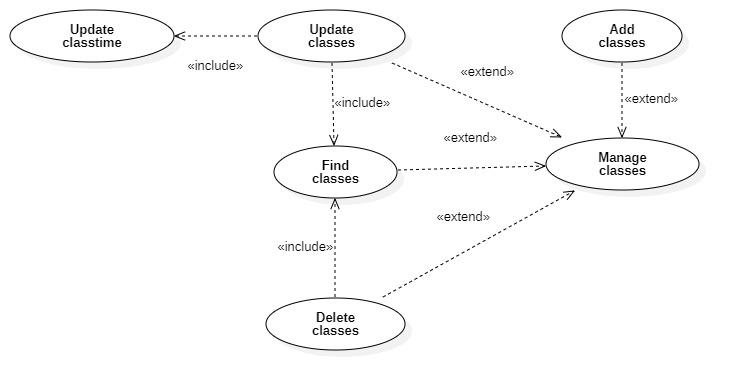
# Chapter 3: ANALYSIS AND DESIGN

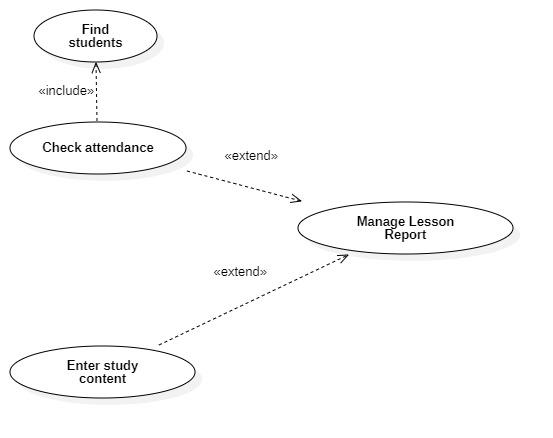
## 3.1. Use-case Diagram

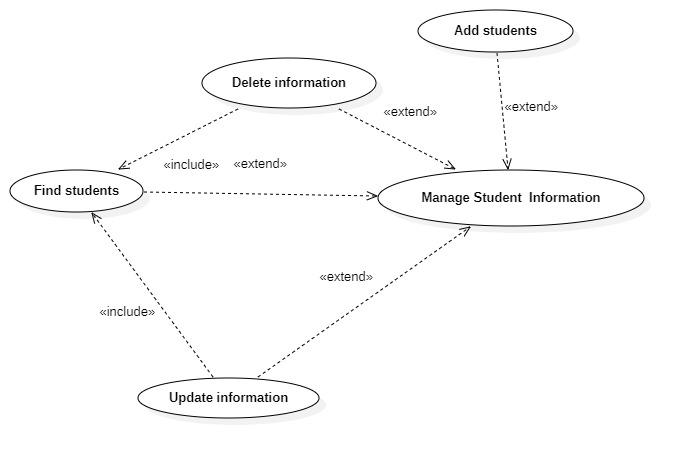
**Use-case for admin**

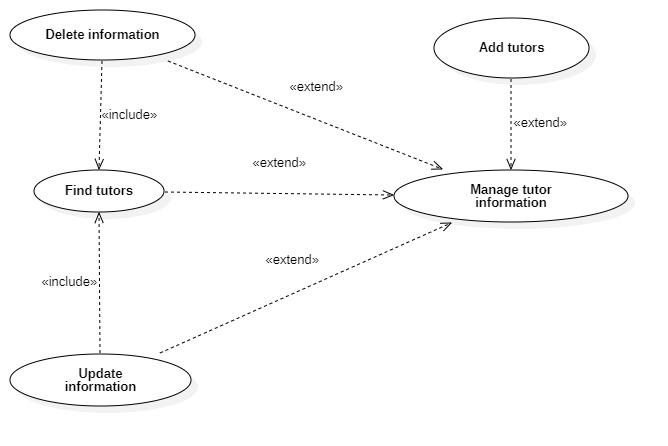
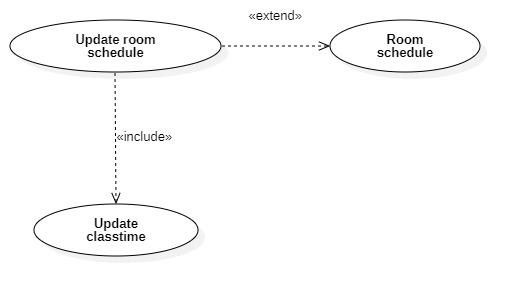




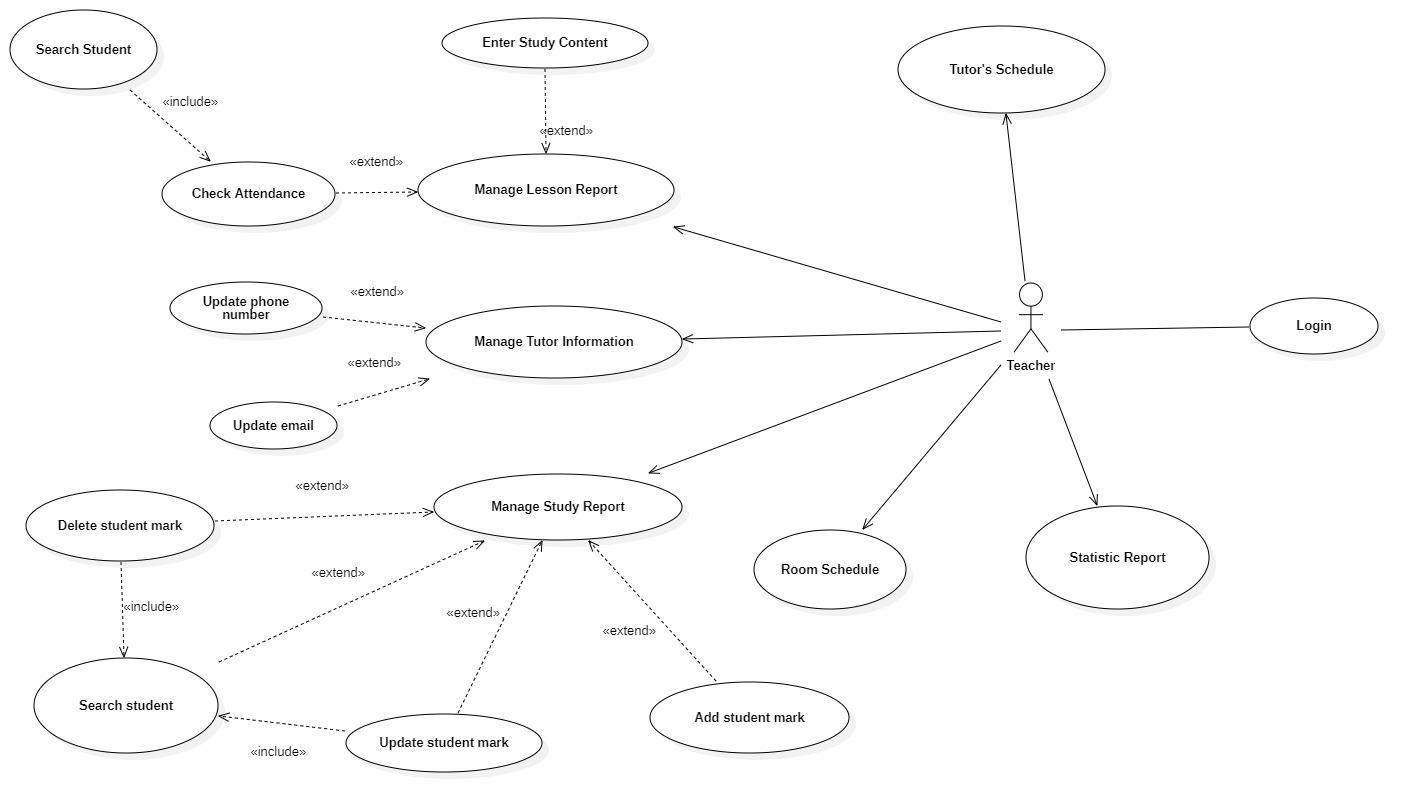






**Use case for tutor**



### 3.1.1 List of actors

|  |  |  |
| --- | --- | --- |
| **Index** | **Actors** | **Meaning** |
| **1** | Admin | People manage the system. |
| **2** | Tutor | Main workforce, manage academic-related tasks. |

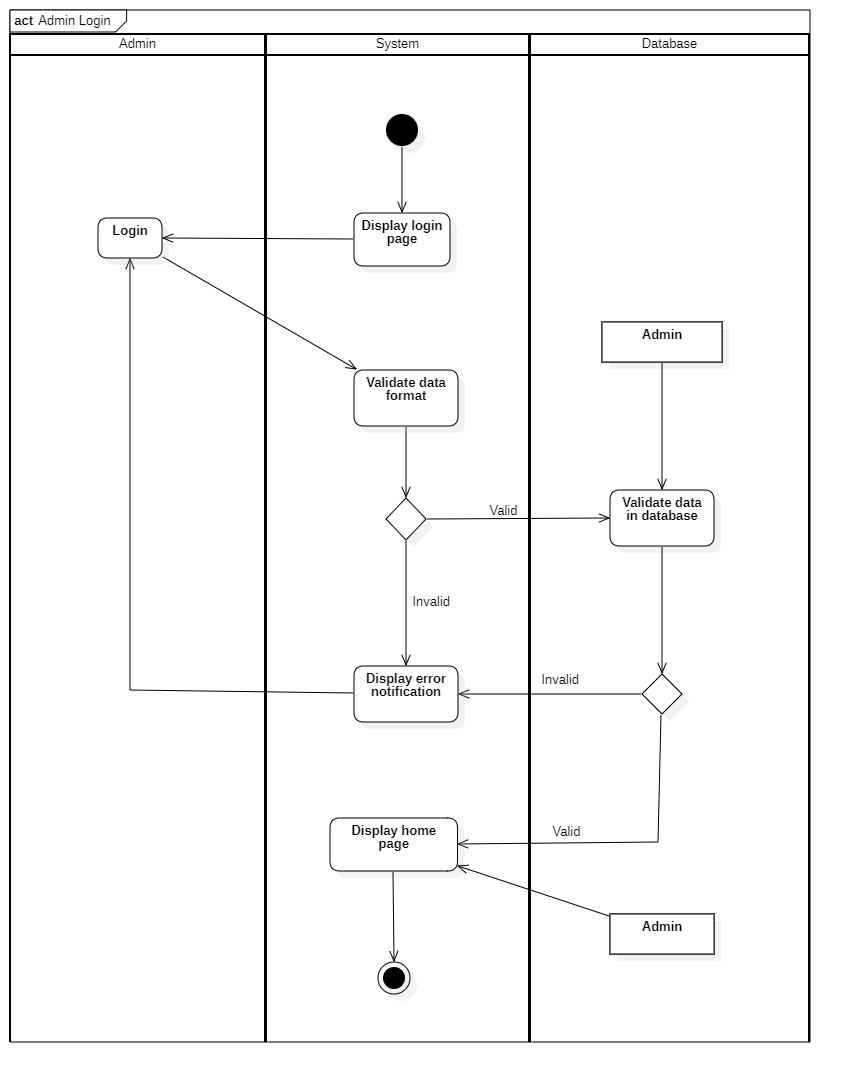
### 3.1.2 List of Use-case

|  |  |  |
| --- | --- | --- |
| **Role** | **Use case** | **Meaning** |
| **Admin’s Role** | **Login** | Login to use services |
| **Manage student information** | Insert/update/delete/search student information |
| **Manage tutor information** | Insert/update/delete/search tutor information |
| **Manage classes** | Insert/update/delete/search classes |
| **Manage classrooms** | Insert/update/delete/search clasrooms |
| **Manage statistic report** | To receive a general view about the tutor’s teaching status |
| **Manage lesson report** | Analysis student’s status based on taking attendance and updating study content |
| **Manage room schedule** | Arrange the room for teaching |
| **Tutor’s Role** | **Login** | Login to use services |
| **Manage tutor information** | Update personal information |
| **Manage lesson report** | Store study content and check attendance and prinft report |
| **Manage grade report** | Insert/update/delete/ student marks and to search students |
| **Statistic report** | View overall statistic |
| **Tutor’s schedule** | View personal schedule |
| **Room schedule** | View room schedule for classes tutors teach |

## 3.2. Use-case specification and activity diagram (for administrator)

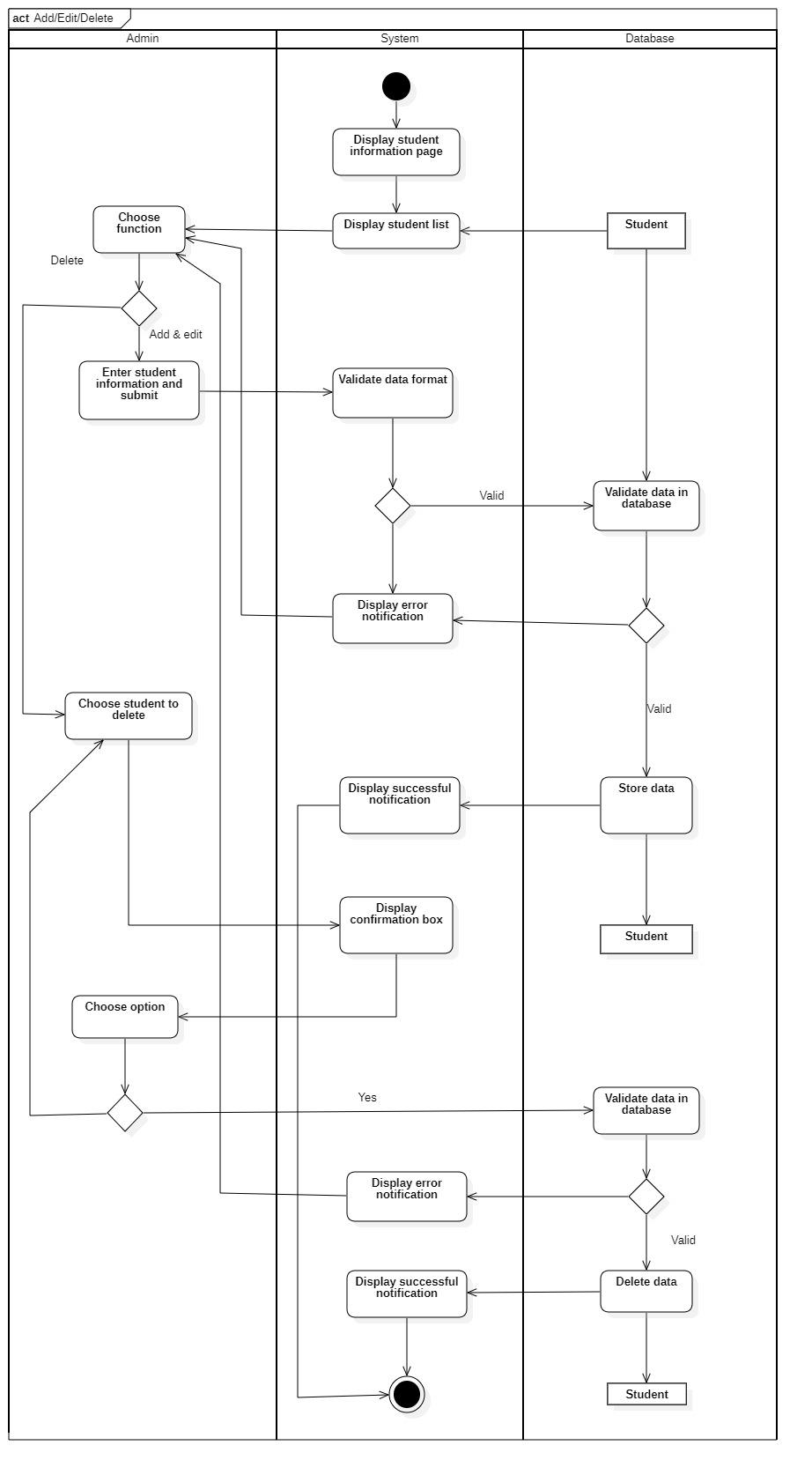
### 3.2.1 Login

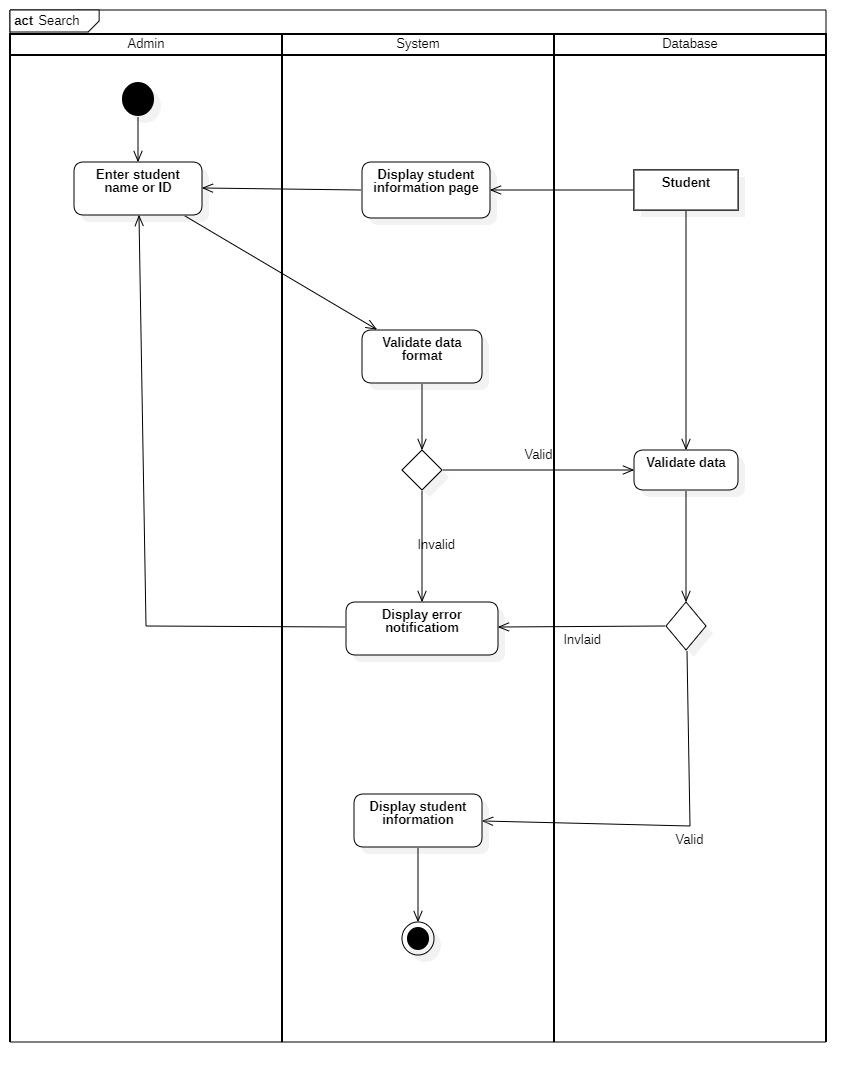
|  |  |
| --- | --- |
| **Use-case name** | Login (for administrator). |
| **Description** | Administrator use exist management account to login to system for all monitoring. |
| **Trigger** | Administrators clicks to button “Login” at management page |
| **Pre - condition** | Administrators’ device must connect to the Internet.  Administrators’ account existed. |
| **Post - condition** | Administrators get a notification about signed-in successful.  Administrators is redirected to home page. |
| **Basic flow** | 1. Administrators clicks “login” button at management page 2. System display management sign-in page 3. Administrators enter username & password 4. System validate administrators entered username & password in correct format 5. System validates administrators’ data with existed administrator account in database. 6. System store signed-in status of administrators using sessions 7. System displays successful signed-in and redirect to home page. |
| **Alternative flow** | * 4a.System display error due to wrong format. * 5a.System display error due to administrators’ account information does not match with existed account data in database. |

****

### 3.2.2 Manage student information

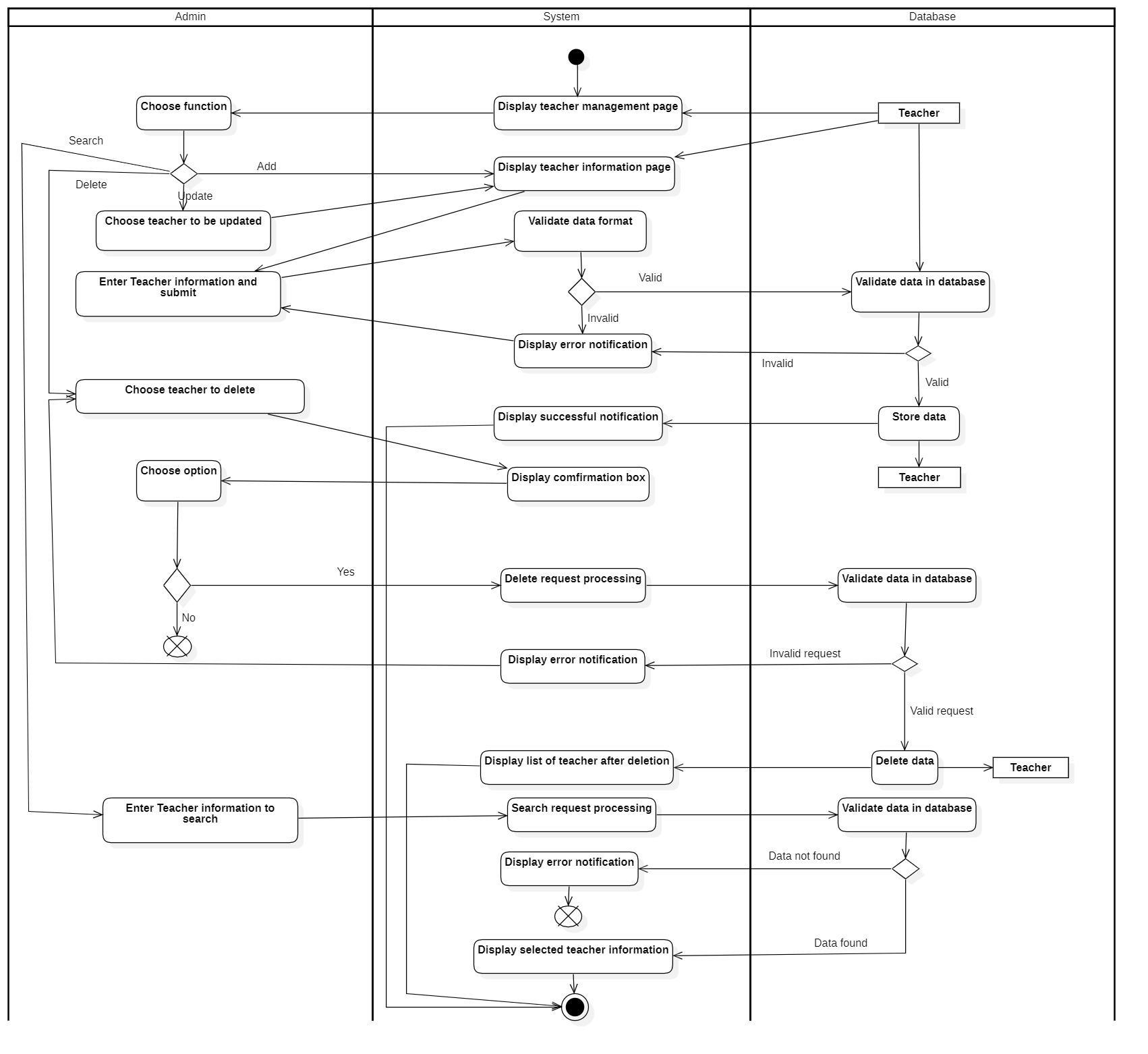
|  |  |
| --- | --- |
| **Use-case name** | Manage student information |
| **Description** | A page for admin to manage student information |
| **Trigger** | Administrators clicks to button “Student management” at homepage |
| **Pre - condition** | Administrators’ device must connect to the Internet.  Administrators’ account existed. |
| **Post - condition** | Student’s information is successfully inserted/updated to database or deleted from database  Data at management page is updated. |
| **Basic flow** | 1. System display student management page. 2. Admin select function: search, add, update or delete student information.   **Add/Edit**:   1. System display information page. 2. Admin enters student’s name or ID then submits. 3. System validates if the information is valid. 4. System validates student’s information with existing data in the database. 5. System saves student’s information in the database. 6. System displays success notification and displays the student information page.   **Delete**:   1. Admin chooses student to be deleted. 2. System displays the student information page. 3. System displays confirm delete box, admin chooses delete option. 4. System processes delete request. 5. System validates if the request is valid. 6. System deletes student’s information in database. 7. System displays successful notification and displays the student management page.   **Search**:   1. Admin enters keywords. 2. System validate if data format is valid. 3. System validate if data is valid. 4. System queries data related to keywords. 5. System filters data at student management page. |
| **Alternative flow** | * 4a. System display error due to wrong format. * 5a. System display error due to student’s data does not match or existed in database. * 11a. User choose no and return to student management page. * 13a. System display invalid data due to student’s data does not match or having constrainst in database. * 17a. System display error due to wrong format. * 18a. System display error due to student’s data does not match or existing in database. |

****

****

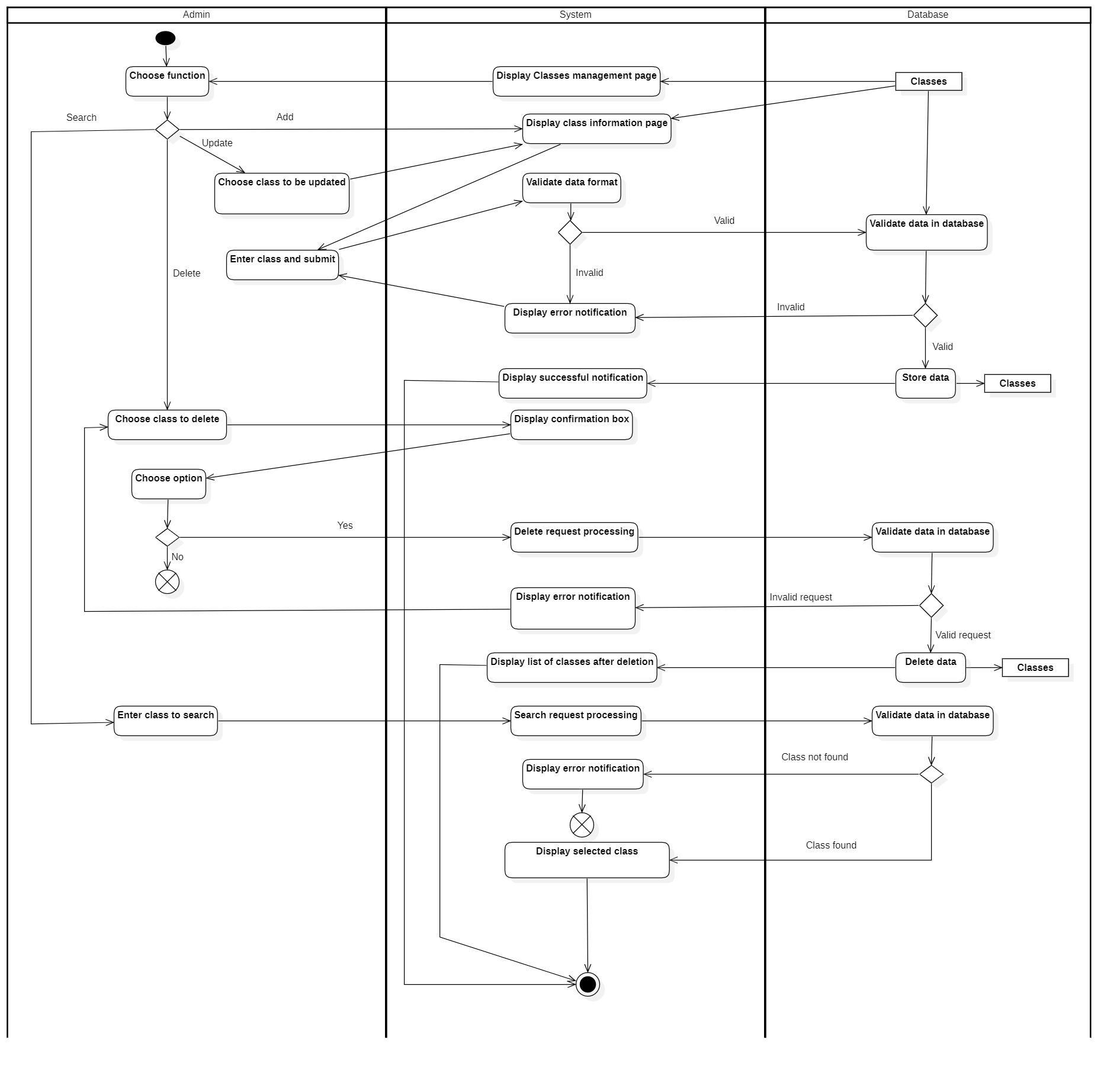
### 3.2.3: Manage tutor information:

|  |  |
| --- | --- |
| **Use-case name** | Manage Tutor Information |
| **Description** | A page for admin to manage tutor information |
| **Trigger** | Administrators clicks to button “Tutor management” at homepage |
| **Pre - condition** | Administrators’ device must connect to the Internet.  Administrators’ account existed.  Administrator’s status must be signed-in |
| **Post - condition** | Tutor’s information is successfully inserted/updated to database or deleted from database  Data at management page is updated. |
| **Basic flow** | 1. System display tutor management page. 2. Admin select function: search, add, update or delete tutor information.   **Add/Edit**:   1. System display information page. 2. Admin enters tutor information and submit. 3. System validates if the data is valid. 4. System validates tutor information with existed data in database. 5. System saves tutor’s information in database. 6. System displays successful notification and displays the tutor information page.   **Delete**:   1. Admin chooses tutor to be deleted. 2. System displays the tutor information page. 3. System displays confirm delete box, employee chooses delete option. 4. System processes delete request. 5. System validates if the request is valid. 6. System deletes tutor information in database. 7. System displays successful notification and displays the tutor management page.   **Search**:   1. Admin enters keywords. 2. System processes search request. 3. System queries data related to keywords. 4. System filters data at tutor management page. |
| **Alternative flow** | * 5a. System displays invalid information due to wrong format. Use-case continues at step 4. * 6a. System displayed invalid information because data had been existed in the database. Use-case continues at step 4. * 12a. System displays invalid information due to wrong format. Use-case continues at step 11. * 13a. System display invalid information because data had not been existed in the database or not be able to modify the existing data. Use-case continues at step 11. * 20a. System displays error notification because data can’t be deleted. Stop delete process. * 24a. System displays empty result notification due to not able to query any result. |

****

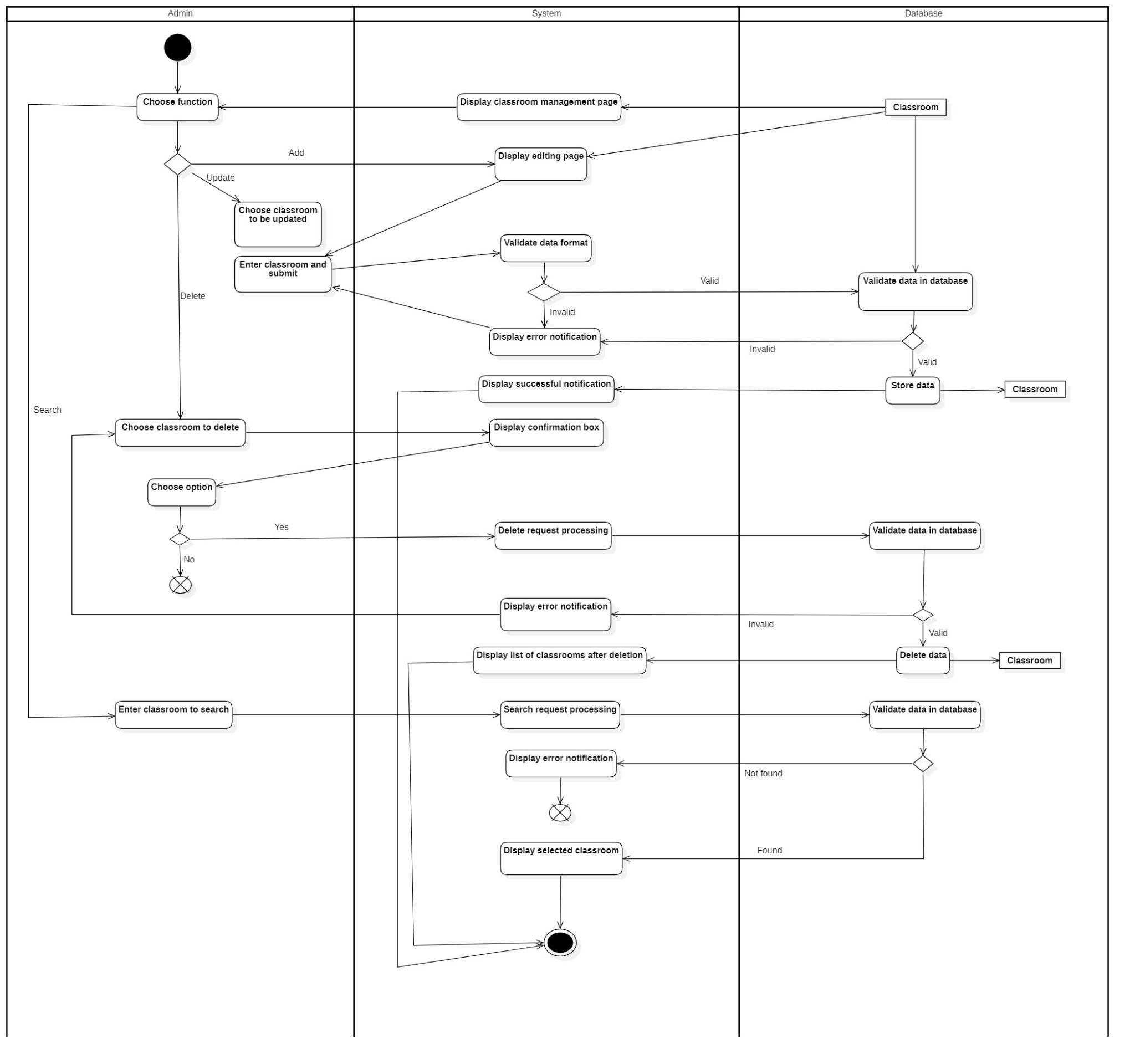
### 3.2.4: Manage class information:

|  |  |
| --- | --- |
| **Use-case name** | Manage Class Information |
| **Description** | A page for admin to manage class information |
| **Trigger** | Administrators clicks to button “class management” at homepage |
| **Pre - condition** | Administrators’ device must connect to the Internet.  Administrators’ account existed.  Administrator’s status must be signed-in |
| **Post - condition** | Class information is successfully inserted/updated to database or deleted from database  Data at the management page is updated. |
| **Basic flow** | 1. System display class management page. 2. Admin select function: search, add, update or delete class information.  * **Add**:  1. System display class information page. 2. Admin enters class information and submit. 3. System validates if the information is valid. 4. System validates class information with existed data in database. 5. System saves class information in database. 6. System displays successful notification and displays the class information page.   - **Update**:   1. Admin chooses class to be updated. 2. System displays the class information page. 3. Admin enters class information and click button “submit”. 4. System validates if the information is valid. 5. System validates class information with existed data in database. 6. System saves class information in database. 7. System displays successful notification and displays the class management page.   - **Delete**:   1. Admin chooses class to be deleted. 2. System displays the class information page. 3. System displays confirm delete box, employee chooses delete option. 4. System processes delete request. 5. System validates if the request is valid. 6. System deletes class information in database. 7. System displays successful notification and displays the class management page.  * **Search**:  1. Admin enters keywords. 2. System processes search request. 3. System queries data related to keywords. 4. System filters data at class management page. |
| **Alternative flow** | * 5a. System displays invalid information due to wrong format. Use-case continues at step 4. * 6a. System displayed invalid information because data had been existed in the database. Use-case continues at step 4. * 12a. System displays invalid information due to wrong format. Use-case continues at step 11. * 13a. System display invalid information because data had not been existed in the database or not be able to modify the existing data. Use-case continues at step 11. * 20a. System displays error notification because data can’t be deleted. Stop delete process. * 24a. System displays empty result notification due to not able to query any result. |

****

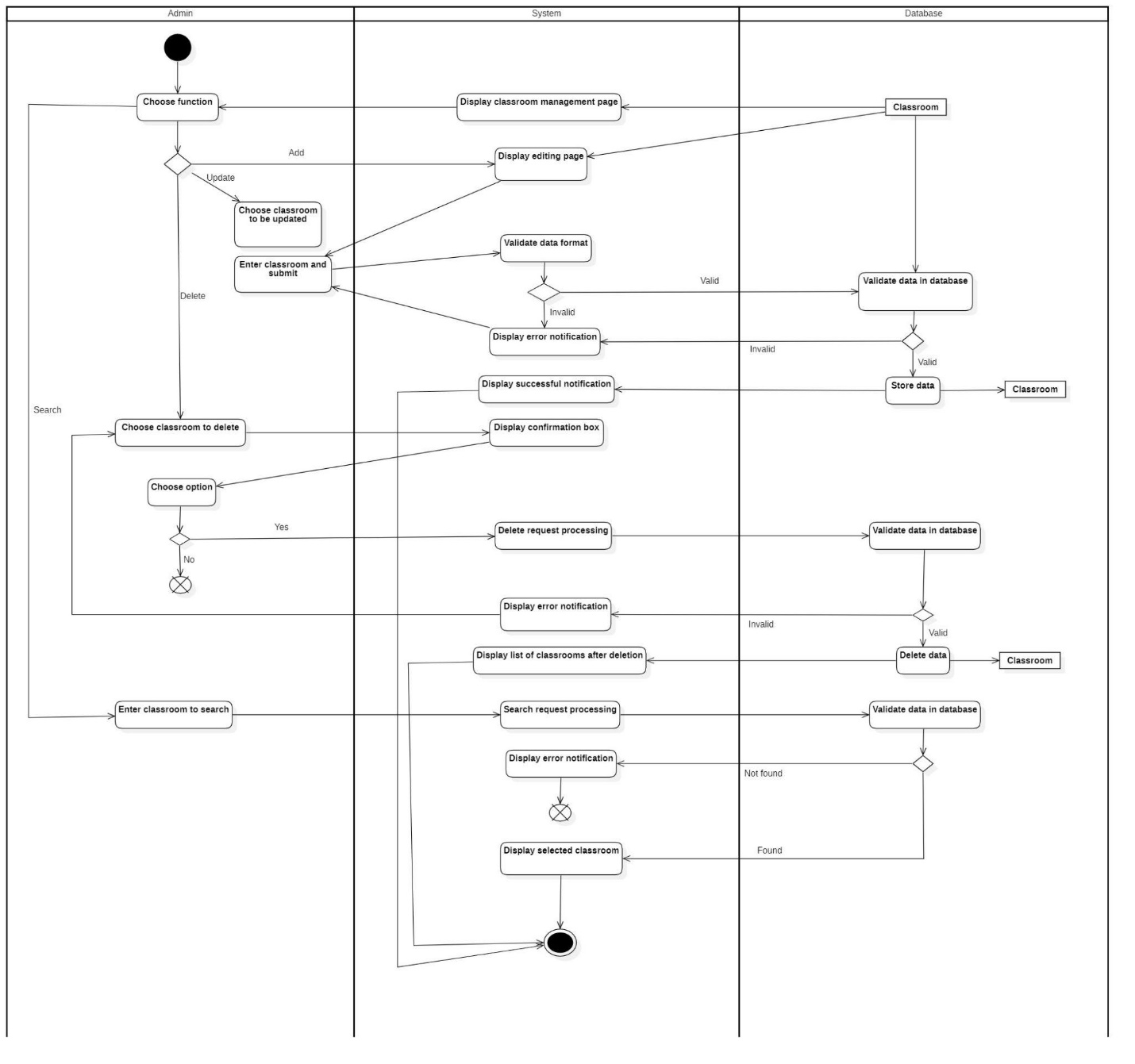
**3.2.5 Manage classroom**

|  |  |
| --- | --- |
| **Use-case name** | Manage classroom information |
| **Description** | A page for admin to manage classroom information |
| **Trigger** | Administrators clicks to button “Classroom management” at homepage |
| **Pre - condition** | Administrators’ device must connect to the Internet.  Administrators’ account existed.  Administrator’s status must be signed-in |
| **Post - condition** | Classroom information is successfully inserted/updated to database or deleted from database  Data at management page is updated. |
| **Basic flow** | 1. System display classroom management page. 2. Admin select function: search, add, update or delete classroom information.   **Add/Edit**:   1. System display information page. 2. Admin enters classroom information and submit. 3. System validates if the information is valid. 4. System validates classroom information with existed data in database. 5. System saves classroom information in database. 6. System displays successful notification and displays the classroom information page.   **Delete**:   1. Admin chooses classroom to be deleted. 2. System displays the classroom information page. 3. System displays confirm delete box, admin chooses delete option. 4. System processes delete request. 5. System validates if the request is valid. 6. System deletes classroom information in database. 7. System displays successful notification and displays the classroom management page.   **Search**:   1. Admin enters keywords. 2. System processes search request. 3. System queries data related to keywords. 4. System filters data at classroom management page. |
| **Alternative flow** | * 5a. System displays invalid information due to wrong format. Use-case continues at step 4. * 6a. System displayed invalid information because data had been existed in the database. Use-case continues at step 4. * 12a. System displays invalid information due to wrong format. Use-case continues at step 11. * 13a. System display invalid information because data had not been existed in the database or not be able to modify the existing data. Use-case continues at step 11. * 20a. System displays error notification because data can’t be deleted. Stop delete process. * 24a. System displays empty result notification due to not able to query any result. |

****

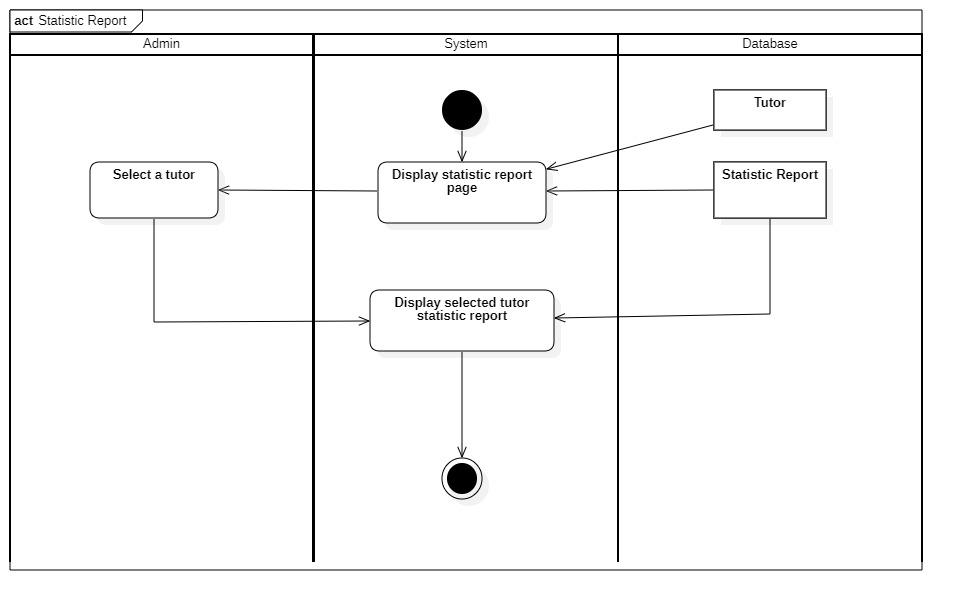
### 3.2.5 Manage classroom

|  |  |
| --- | --- |
| **Use-case name** | Manage classroom information |
| **Description** | A page for admin to manage classroom information |
| **Trigger** | Administrators clicks to button “Classroom management” at homepage |
| **Pre - condition** | Administrators’ device must connect to the Internet.  Administrators’ account existed.  Administrator’s status must be signed-in |
| **Post - condition** | Classroom information is successfully inserted/updated to database or deleted from database  Data at management page is updated. |
| **Basic flow** | 1. System display classroom management page. 2. Admin select function: search, add, update or delete classroom information.    1. **Add**: 3. System display information page. 4. Admin enters classroom information and submit. 5. System validates if the information is valid. 6. System validates classroom information with existed data in database. 7. System saves classroom information in database. 8. System displays successful notification and displays the classroom information page.    1. - **Update**: 9. Admin chooses classroom to be updated. 10. System displays the classroom information page. 11. Admin enters classroom information and click button “submit”. 12. System validates if the information is valid. 13. System validates classroom information with existed data in database. 14. System saves classroom information in database. 15. System displays successful notification and displays the classroom management page.     1. - **Delete**: 16. Admin chooses classroom to be deleted. 17. System displays the classroom information page. 18. System displays confirm delete box, admin chooses delete option. 19. System processes delete request. 20. System validates if the request is valid. 21. System deletes classroom information in database. 22. System displays successful notification and displays the classroom management page.     1. **Search**: 23. Admin enters keywords. 24. System processes search request. 25. System queries data related to keywords. 26. System filters data at classroom management page. |
| **Alternative flow** | * 5a. System displays invalid information due to wrong format. Use-case continues at step 4. * 6a. System displayed invalid information because data had been existed in the database. Use-case continues at step 4. * 12a. System displays invalid information due to wrong format. Use-case continues at step 11. * 13a. System display invalid information because data had not been existed in the database or not be able to modify the existing data. Use-case continues at step 11. * 20a. System displays error notification because data can’t be deleted. Stop delete process. * 24a. System displays empty result notification due to not able to query any result. |

****

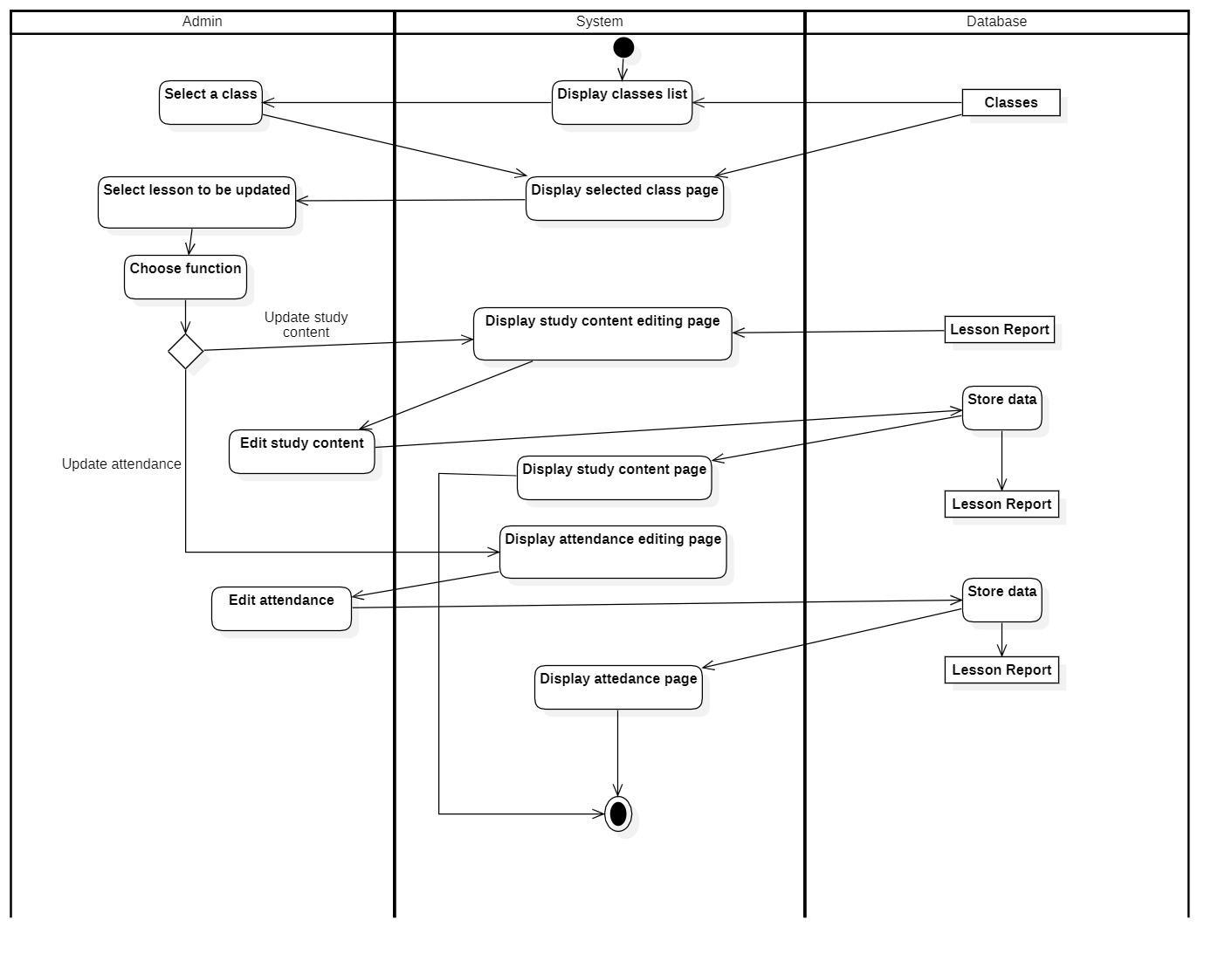
### 3.2.6 Manage statistic report

|  |  |
| --- | --- |
| **Use-case name** | Manage statistic report |
| **Description** | A page for admin to manage statistic report |
| **Trigger** | Administrators clicks to button “Statistic report management” at homepage |
| **Pre - condition** | Administrators’ device must connect to the Internet.  Administrators’ account existed.  Administrator’s status must be signed-in |
| **Post - condition** | Room schedule information is successfully updated to database.  Data at management page is updated. |
| **Basic flow** | 1. The system will display statistic report page 2. Admin selects a tutor 3. The system will display the selected tutor’s statistic report. |
| **Alternative flow** |  |

****

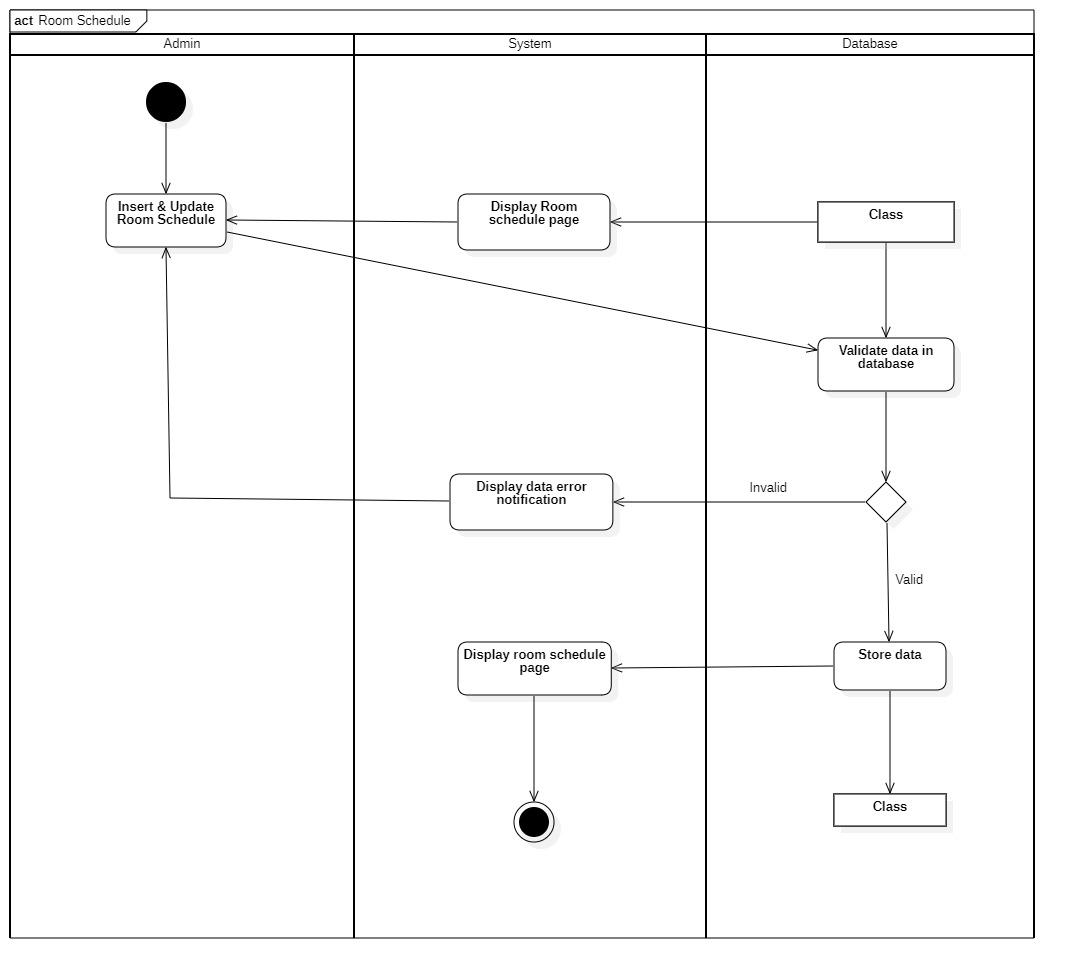
### 3.2.7 Manage Lesson Report

|  |  |
| --- | --- |
| **Use-case name** | Manage Lesson report |
| **Description** | A page for admin to manage lesson report |
| **Trigger** | * Administrators clicks on button “classes list ” at Class management page * Administrator selects a class and clicks on button “ Lesson report ” from a selected class page. |
| **Pre - condition** | * Administrators’ device must connect to the Internet. * Administrators’ account existed. * Administrator’s status must be signed-in |
| **Post - condition** | * Lesson report information is successfully store to database * Data at the management page is updated. |
| **Basic flow** | 1. System display classes list 2. Admin selects a class to be updated 3. System display a selected class 4. Admin select lesson to be updated. 5. Admin choose a function:   **Study content**   1. System display editing page for study content. 2. Admin update study content 3. The data is store into the database 4. The system displays a success notification.   **Attendance**   1. System display editing page for editing attendance 2. Admin edits attendance 3. The data is store into the database |
| **Alternative flow** | * System displays invalid information due to the wrong format. * System displays invalid information due to classroom information that does not match or exists in the database. |

****

### 3.2.8 Manage room schedule

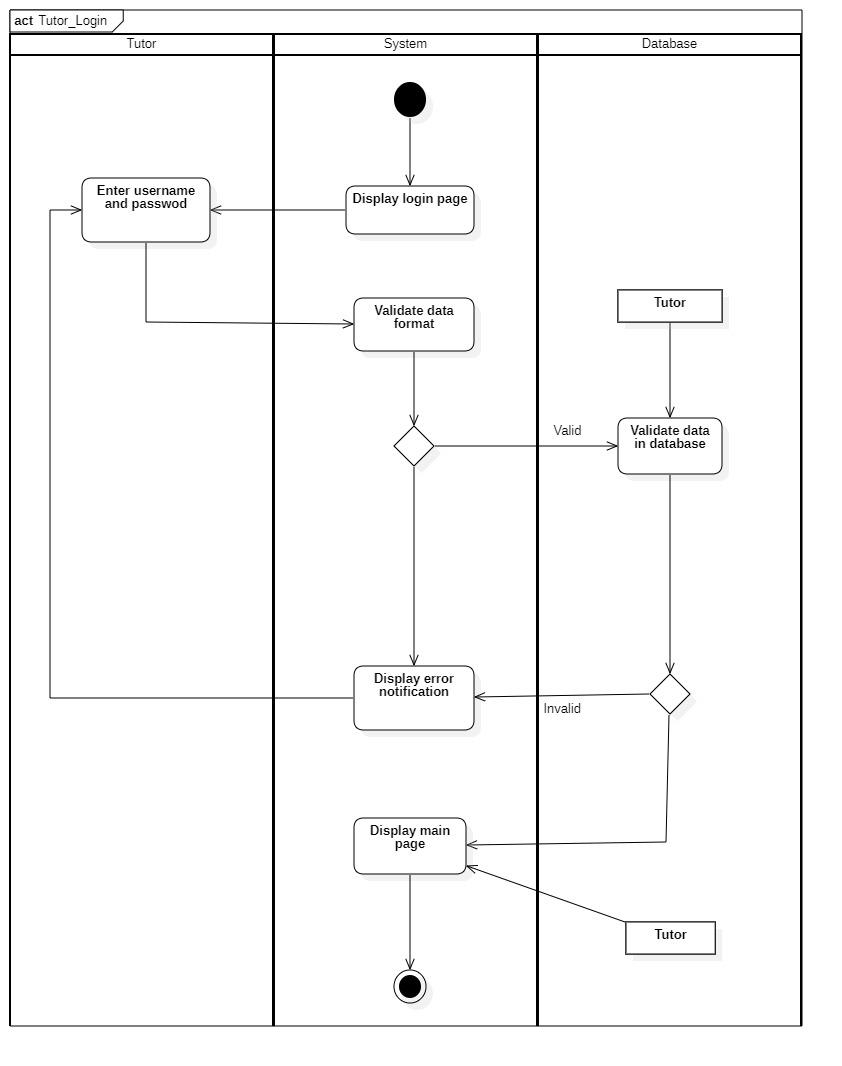
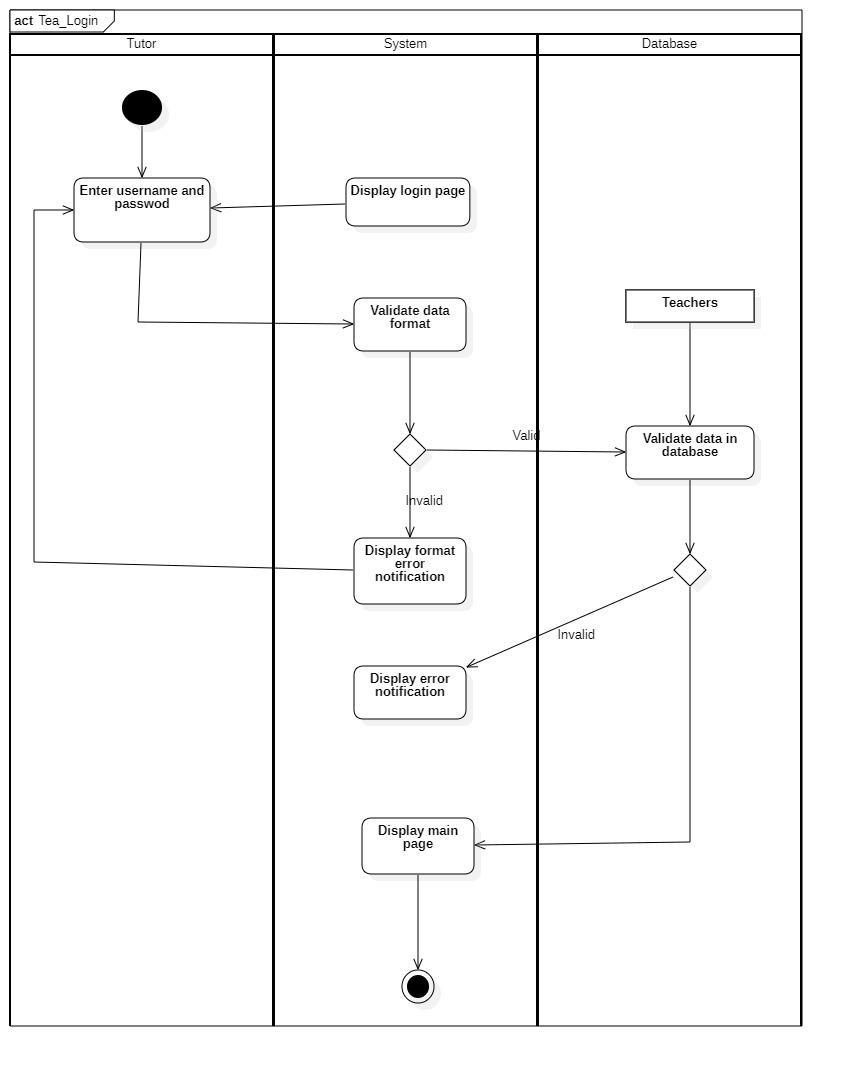
|  |  |
| --- | --- |
| **Use-case name** | Manage room schedule |
| **Description** | A page for admin to manage room schedule |
| **Trigger** | Administrators clicks to button “Room schedule management” at homepage |
| **Pre - condition** | Administrators’ device must connect to the Internet.  Administrators’ account existed.  Administrator’s status must be signed-in |
| **Post - condition** | Room schedule information is successfully updated to database.  Data at management page is updated. |
| **Basic flow** | 1. The system will display room schedule page 2. The admin choose a function to update the room schedule 3. The system will display editing page for room schedule 4. The admin insert room schedule 5. The system validate data 6. The data is stored in the database 7. The system will display room schedule page . |
| **Alternative flow** | * System will display error notification due to invalid data in database |

****

## 3.3. Use-case specification and activity diagram (for tutors)

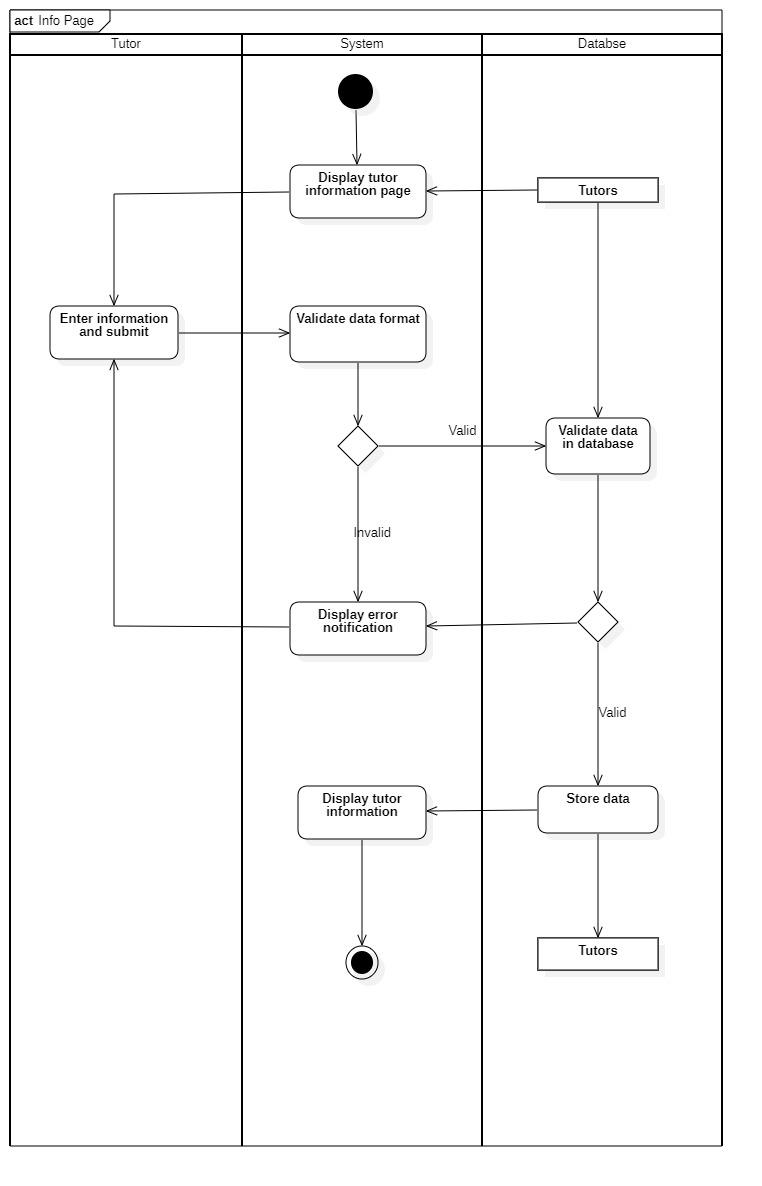
### 3.3.1 Login

|  |  |
| --- | --- |
| **Use-case name** | Login |
| **Description** | Page to login into app to access services |
| **Trigger** | Tutor click on login on front page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account existed. |
| **Post - condition** | Account is logged in |
| **Basic flow** | 1. User click on “login” button 2. User enter username and password 3. Login information format is validated by the system 4. Login information is validate with database to see if it matched 5. User is directed to main page |
| **Alternative flow** | * System displays invalid information due to wrong format. * System display invalid information due to either wrong username or password |

****

### 3.3.2 Manage tutor information

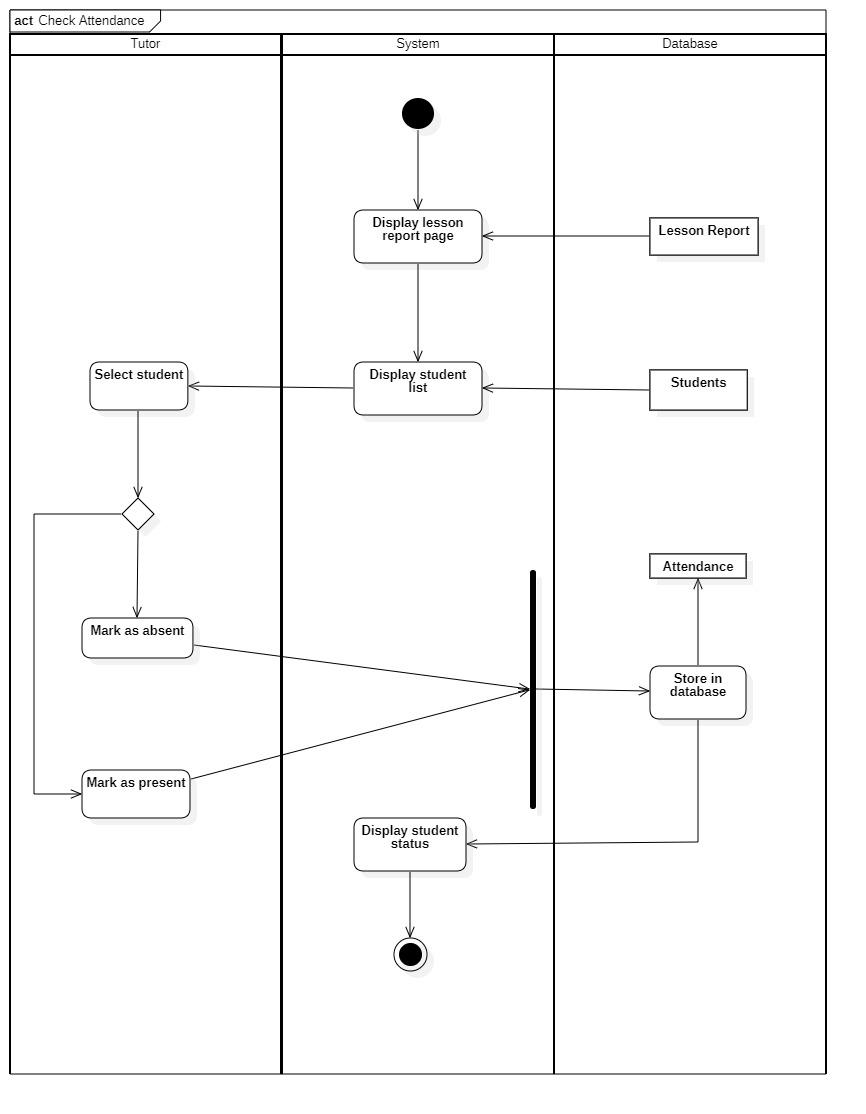
|  |  |
| --- | --- |
| **Use-case name** | **Manage tutor information** |
| **Description** | Allow tutor to update their personal information |
| **Trigger** | Tutor clicks on “Account” on main page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account already logged in |
| **Post - condition** | New information is stored in database |
| **Basic flow** | 1. User choose to update personal information 2. User enter new information 3. Data format is validated in system 4. Data is validated in database 5. Data is store in “Teacher ” table 6. Display success notification 7. Return to “Acount” page |
| **Alternative flow** | * System displays invalid information due to wrong format. * System displays invalid information due to either wrong username or password |

****

### 3.3.3 Manage lesson report

#### 3.3.3.1 Check attendance

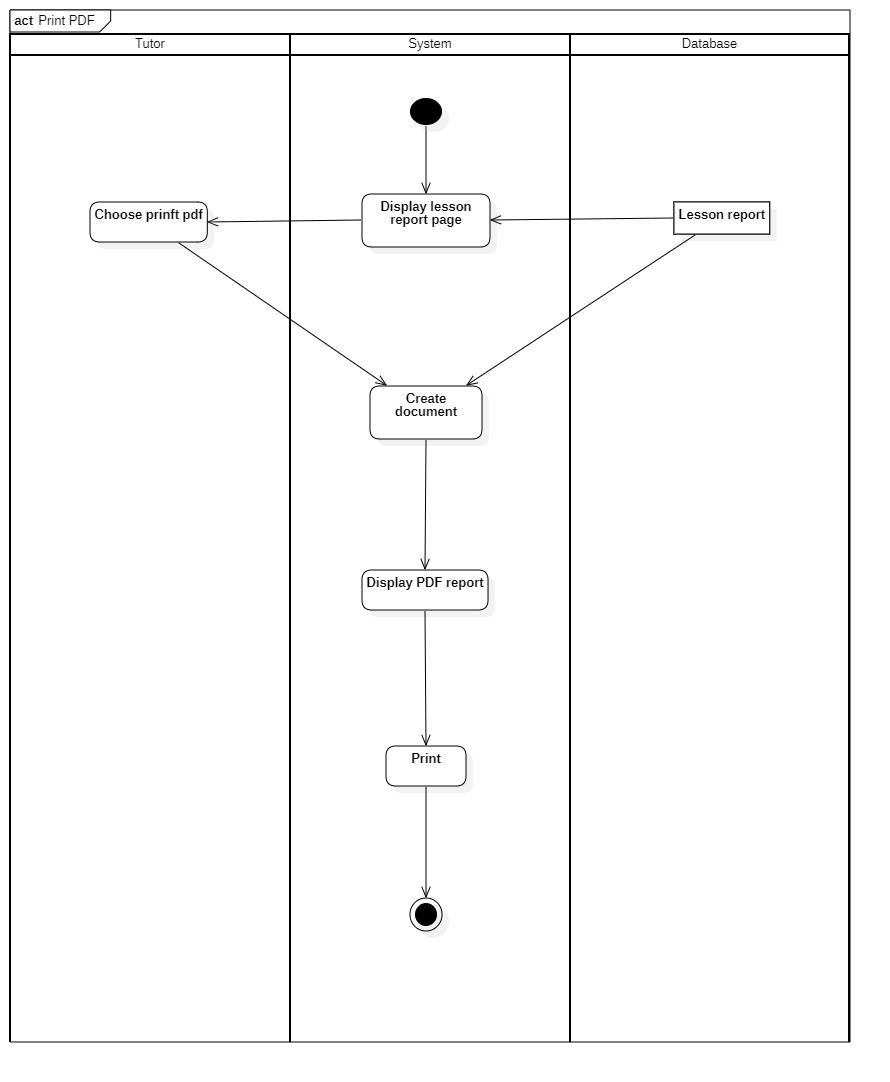
|  |  |
| --- | --- |
| **Use-case name** | **Check attendance** |
| **Description** | Allow tutor to check student attendance each day |
| **Trigger** | Tutor click on “Study content” on “Lesson report” page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account already logged in |
| **Post - condition** | Student attendance is stored in database |
| **Basic flow** | 1. User enter select student 2. Student is marked as absence 3. System process request 4. Data is stored in “Lesson report” table 5. Student status is display on screen |
| **Alternative flow** | * Student is present, is not marked in system |



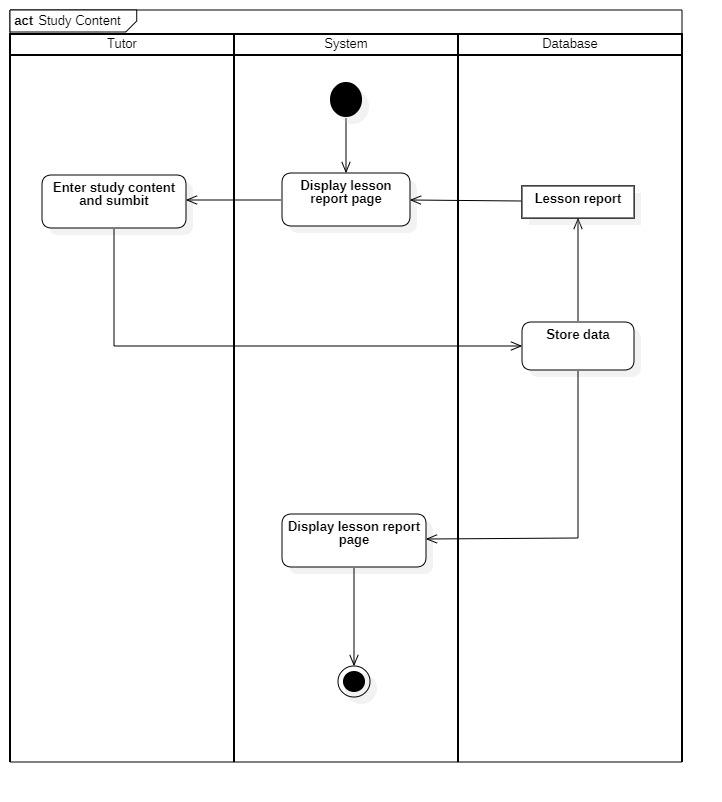
#### 3.3.3.2 Study content

|  |  |
| --- | --- |
| **Use-case name** | **Enter study content** |
| **Description** | Allow tutor to submit content of the lesson each day |
| **Trigger** | Tutor click on “Study content” on “Lesson report” page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account already logged in |
| **Post - condition** | Study content is stored in database |
| **Basic flow** | 1. User enter today’s report 2. Data is stored in “Lesson report” table 3. Display “Lesson report” page |
| **Alternative flow** |  |

#### 3.3.3.3 Prinf PDF

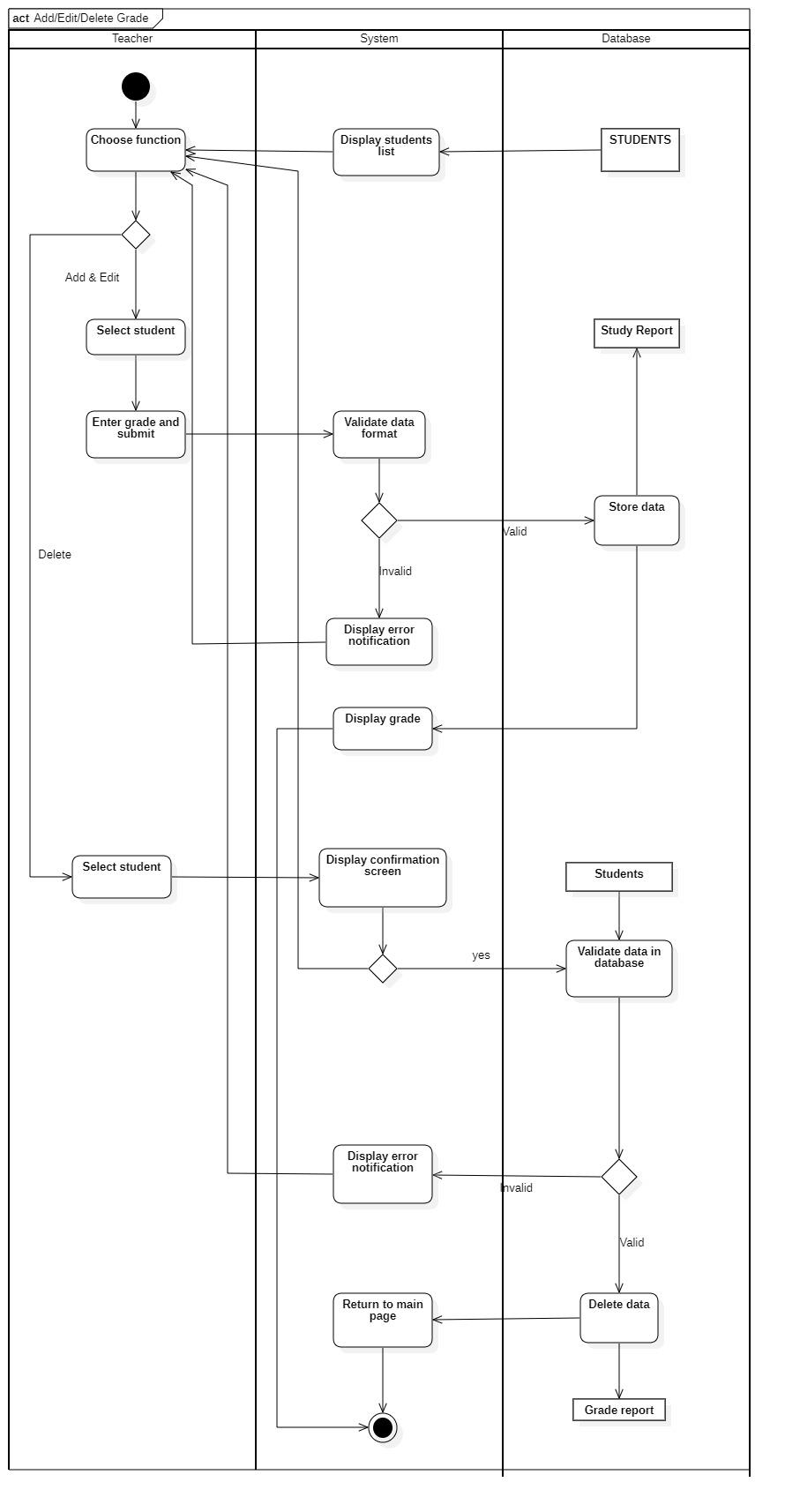


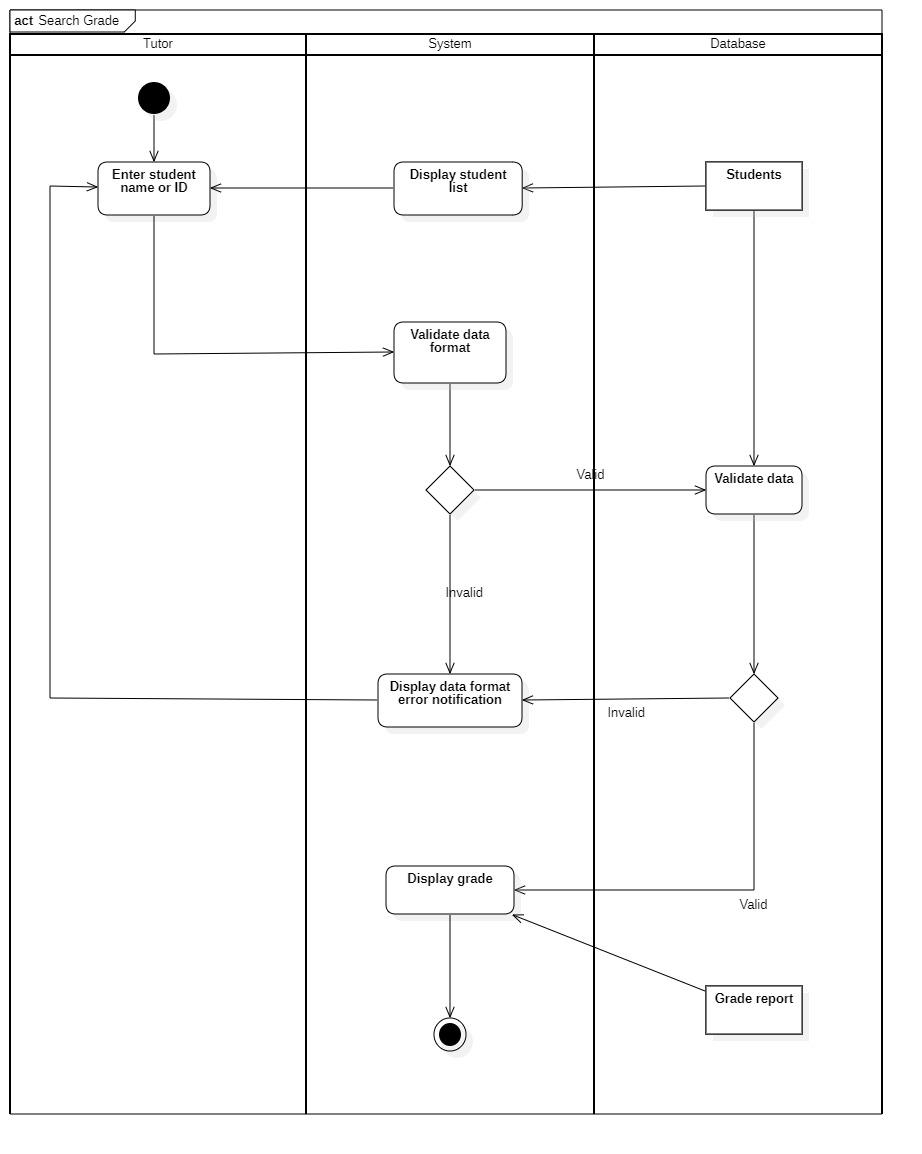
|  |  |
| --- | --- |
| **Use-case name** | **Print PDF** |
| **Description** | Allow tutor to print the lesson report |
| **Trigger** | Tutor click on “Print PDF” on “Lesson report” page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account already logged in |
| **Post - condition** | Report is printed out by printer |
| **Basic flow** | 1. User click on print PDF 2. System create report 3. System display PDF report 4. System print out report |
| **Alternative flow** |  |

****

### 3.3.4: Manage student grades.

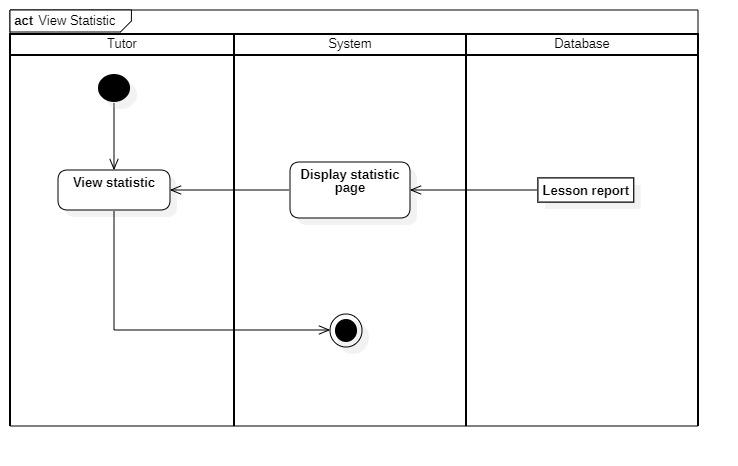
|  |  |
| --- | --- |
| **Use-case name** | **Manage student grade** |
| **Description** | Allow tutor to update students’ grades |
| **Trigger** | Tutor clicks on “Grade report” on class page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account already logged in |
| **Post - condition** | Grade is successfully inserted/updated to database or deleted from database  Data at the management page is updated |
| **Basic flow** | 1. System display “Grade report ” page 2. Tutor select function: search, add, update or delete student grade   **Add/Edit:**   1. Tutor select student to add/edit grade 2. System check if grade data format is valid 3. System store grade in Grade report database 4. System display grade on screen   **Delete:**   1. Tutor select student to delete grade from 2. System display confirmation box 3. Database check if the data has any constrainst 4. Data is deleted from database 5. System return to main page   **Search:**   1. Tutor enter student name or ID to search 2. System check if data format is valid 3. Database check if data is valid 4. System display student grade |
| **Alternative flow** | * 4a.System displays error information due to wrong format. * 8a.User choose no and return to main page * 13. System displays error information due to wrong format. |

****

****

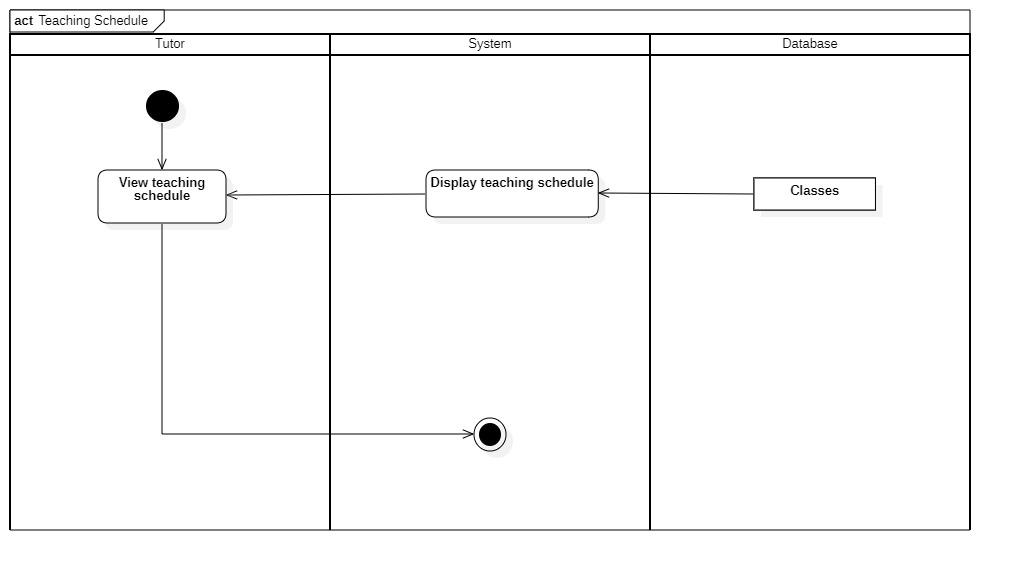
### 3.3.5 Statistic report

|  |  |
| --- | --- |
| **Use-case name** | **Statistic report** |
| **Description** | Allow tutor to view personal statistic report such as time taught,nunber of classes ,… |
| **Trigger** | Tutor clicks on “Report” on main page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account already logged in |
| **Post - condition** | Report is updated and display on “Report” page |
| **Basic flow** | System display statistic page |
| **Alternative flow** |  |

****

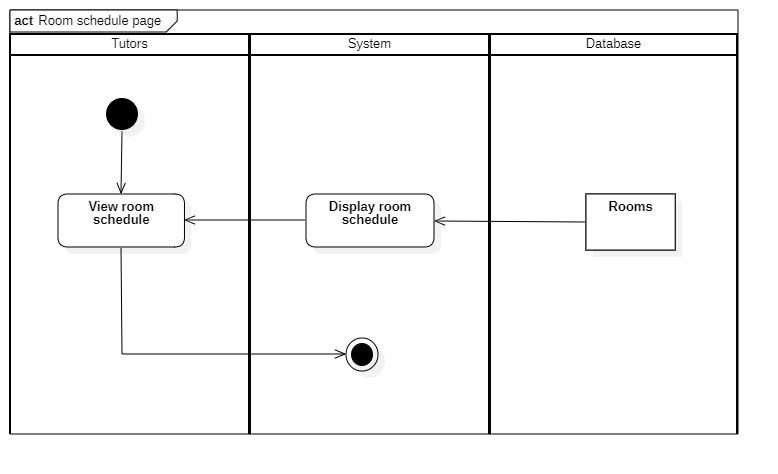
### 3.3.6 Tutor’s schedule

|  |  |
| --- | --- |
| **Use-case name** | **Tutor’s schedule** |
| **Description** | Allow tutor to view personal teaching schedule |
| **Trigger** | Tutor clicks on “Timetable” on main page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account already logged in |
| **Post - condition** | Schedule is updated and display on “Timetable” page |
| **Basic flow** | System display teaching schedule |
| **Alternative flow** |  |

****

### 3.3.7 Room schedule

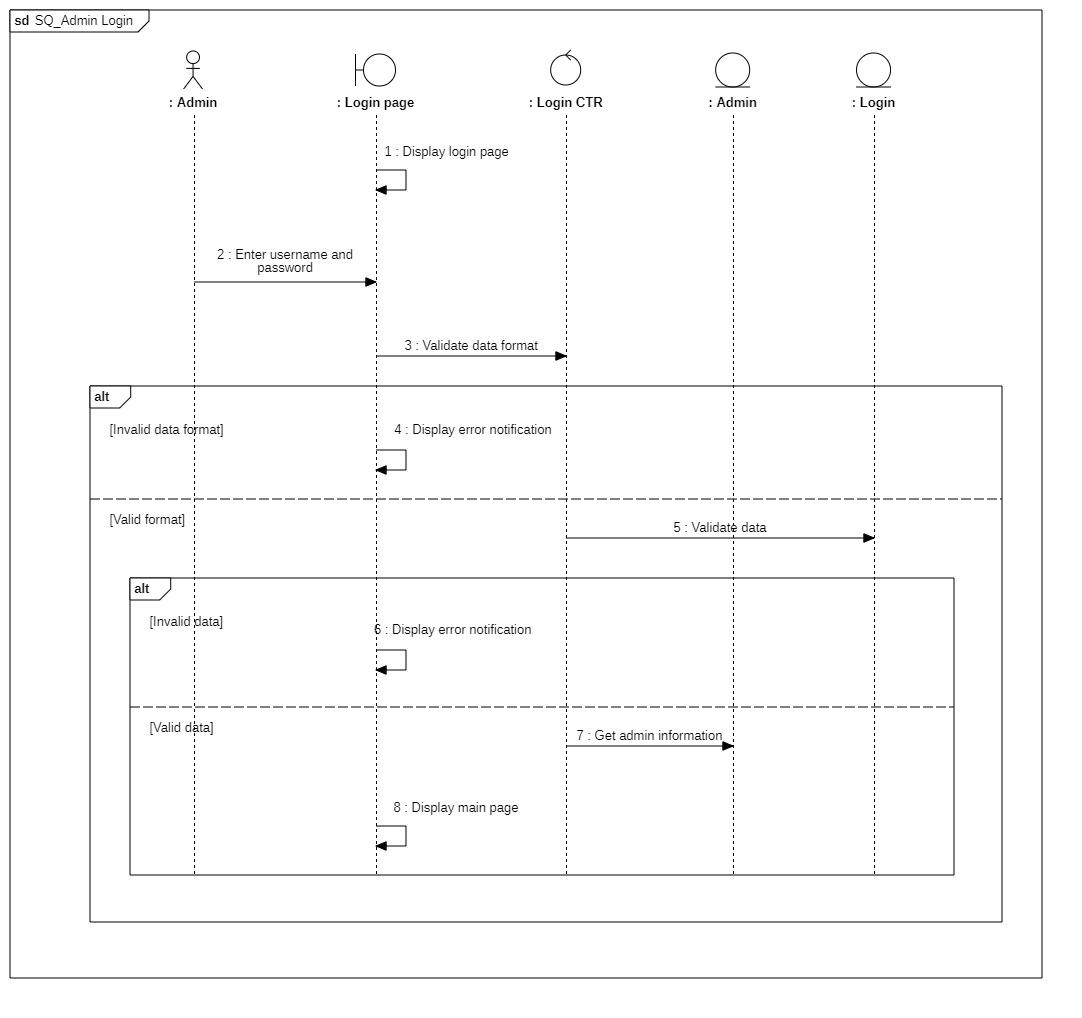
|  |  |
| --- | --- |
| **Use-case name** | **Room schedule** |
| **Description** | Allow tutor view room in which they have classes in |
| **Trigger** | Tutor clicks on “Timetable” on class page |
| **Pre - condition** | Tutor device must connect to the Internet.  Tutor’s account already logged in |
| **Post - condition** | Room schedule is updated and display on timetable page |
| **Basic flow** | System display room schedule |
| **Alternative flow** |  |

****

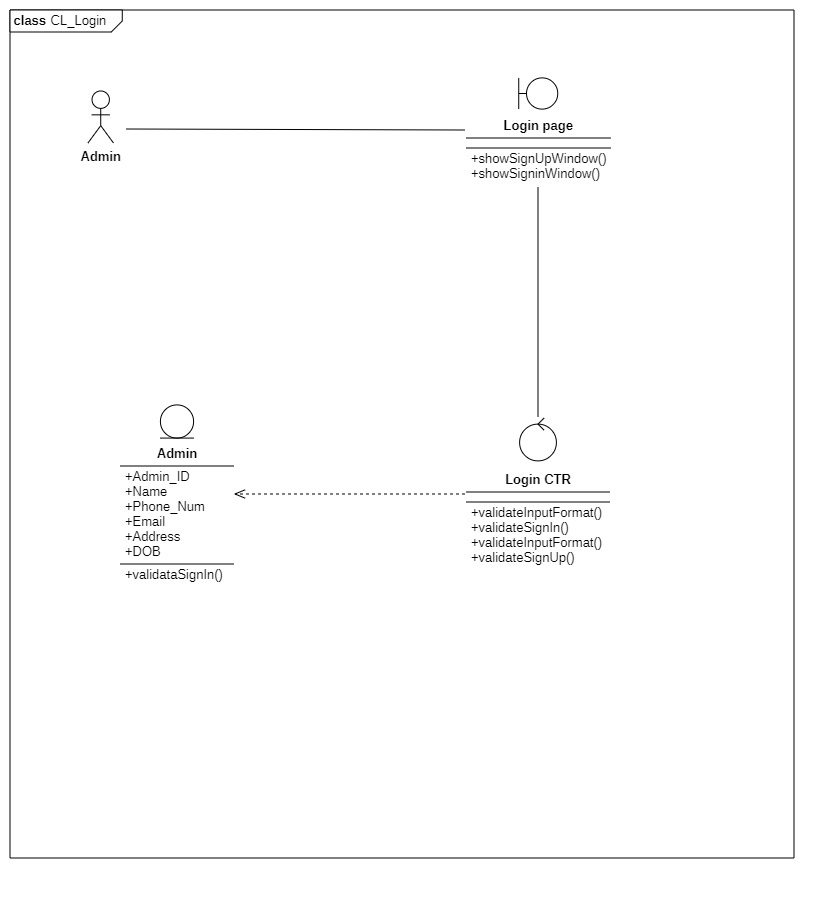
## 3.4 Sequence diagram and class diagram (for administrator)

### 3.4.1 Login in

#### 3.4.1.1 Sequence diagram

****

#### 3.4.1.2 Class diagram

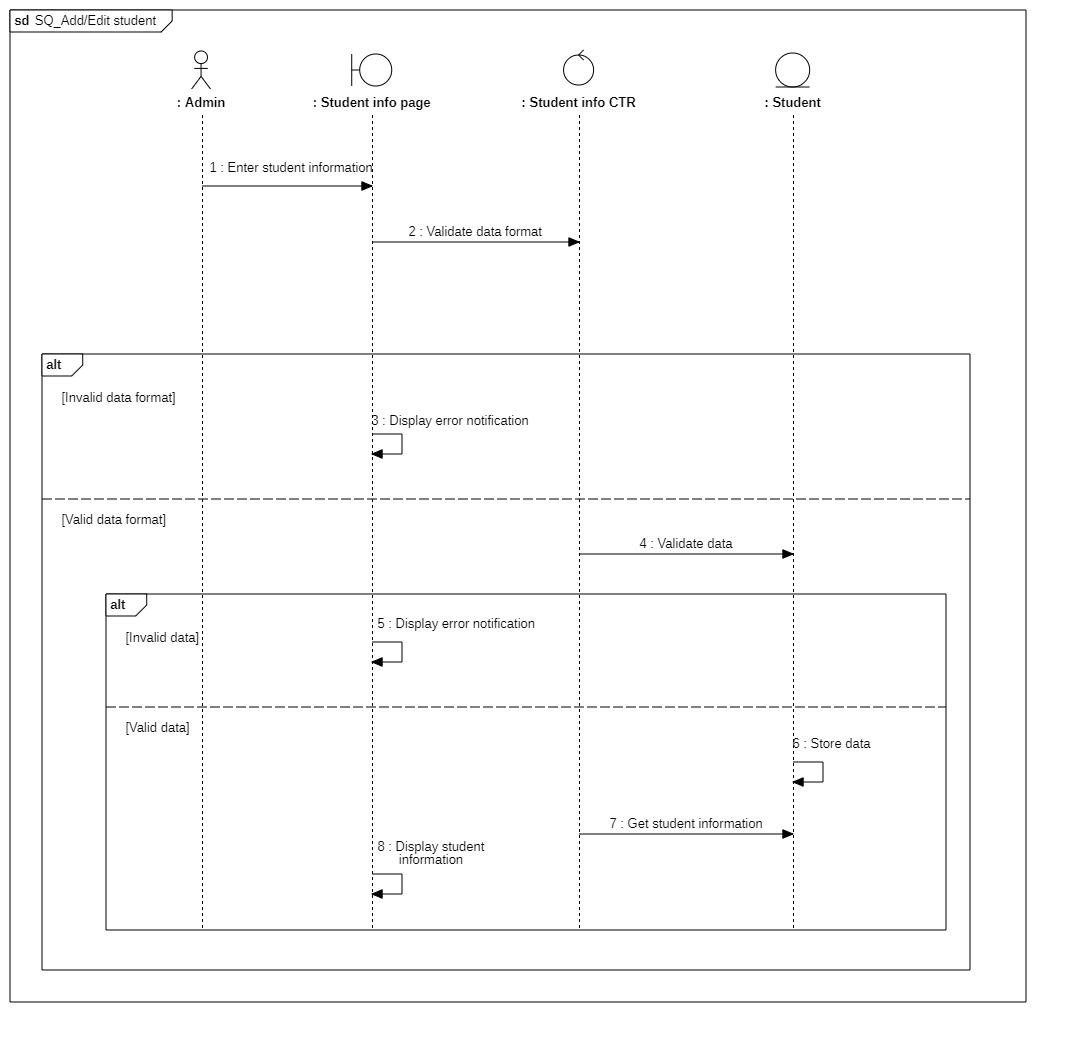
****

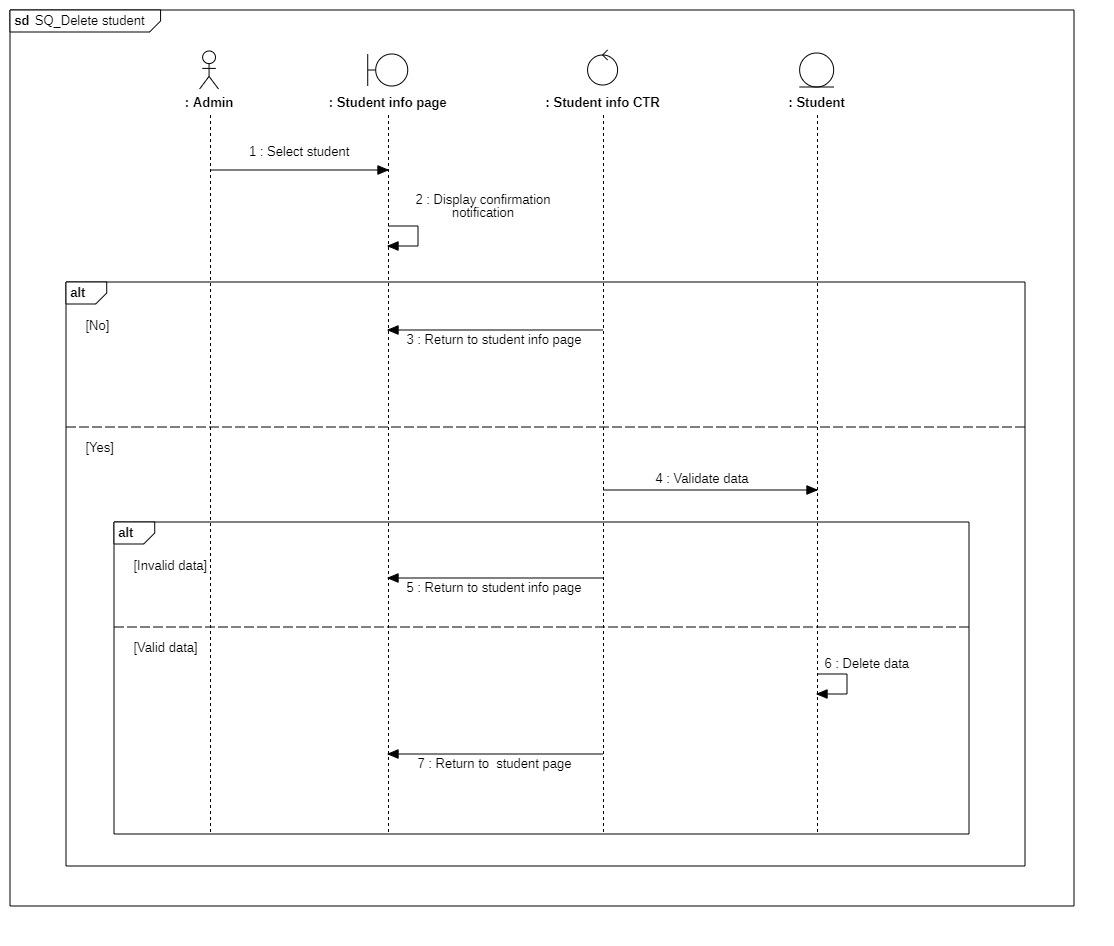
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Login page** |  | validateInputFormat() bool | Check if input format is valid |
|  | validateSignIn() bool | Check if sign-in is valid |
| **Login CTR** |  | validateInputFormat() bool | Check if input format is valid |
|  | validateSignIn() bool | Confirm sign-in |
|  | validateInputFormat() bool | Check if input format is valid |
|  | validateSignUp() bool | Confirm sign-up |
| **Admin** | Admin\_ID |  | Admin’s main identification |
| Name |  | Admin’s name |
| Phone\_Num |  | Admin’s phone number |
| Email |  | Admin’s email |
| Address |  | Admin’s address |
| DOB |  | Admin’s date of birth |
|  |  | ValidateSignIn() | Confirm sign-in |

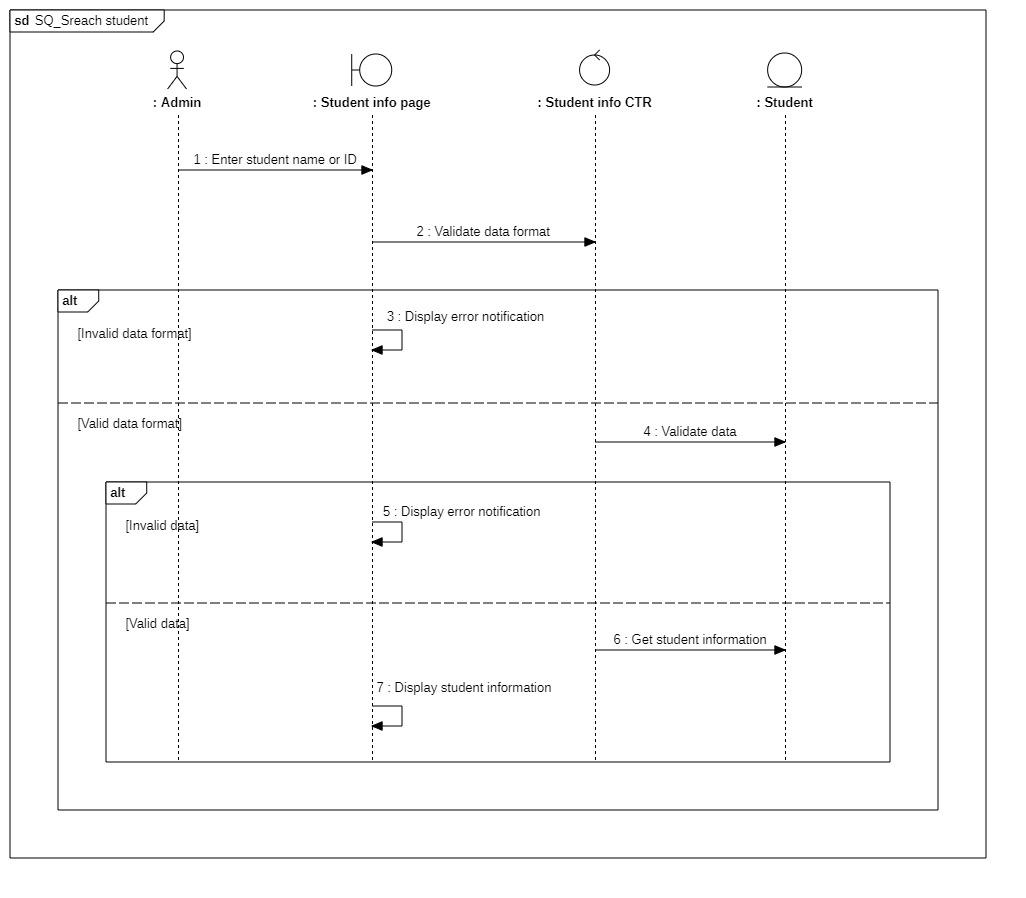
### 3.4.2 Manage student information

#### 3.4.2.1 Sequence diagram

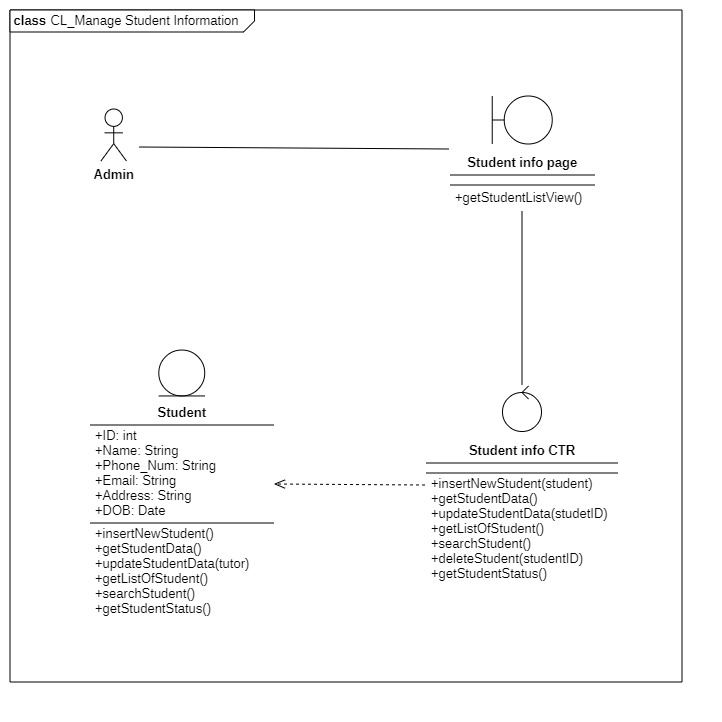
****

****

****

****

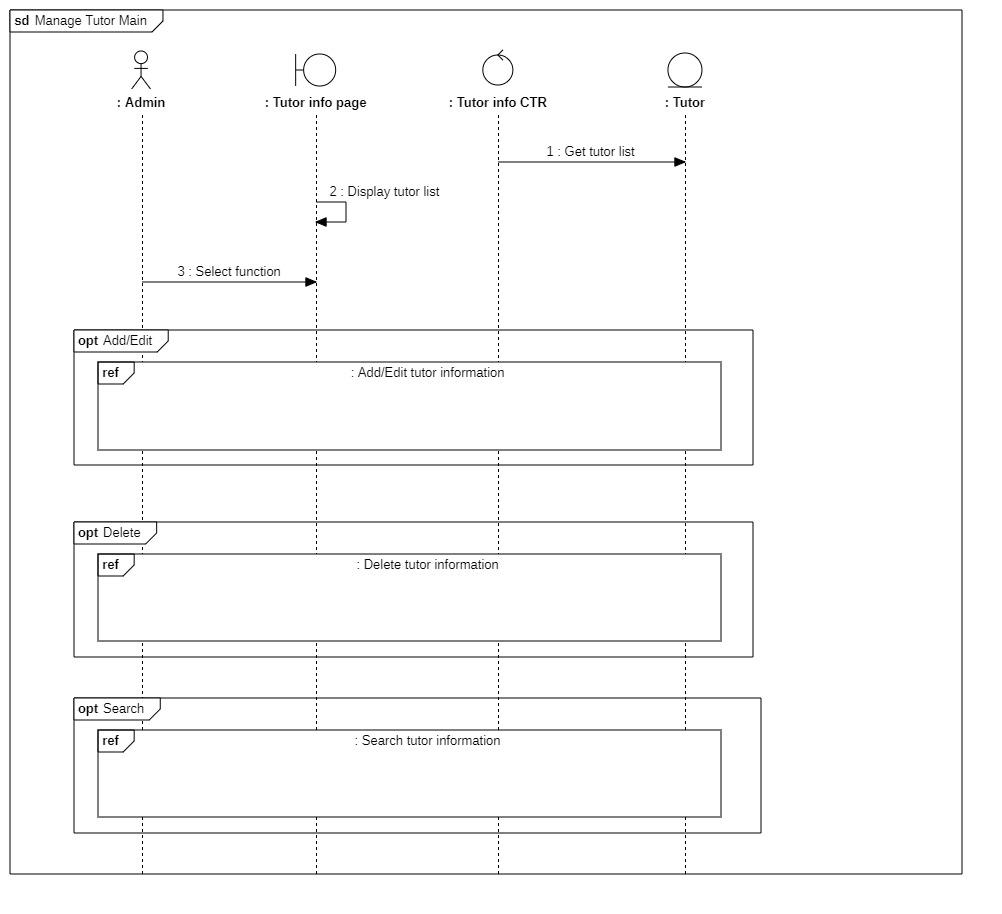
#### 3.4.2.2 Class diagram

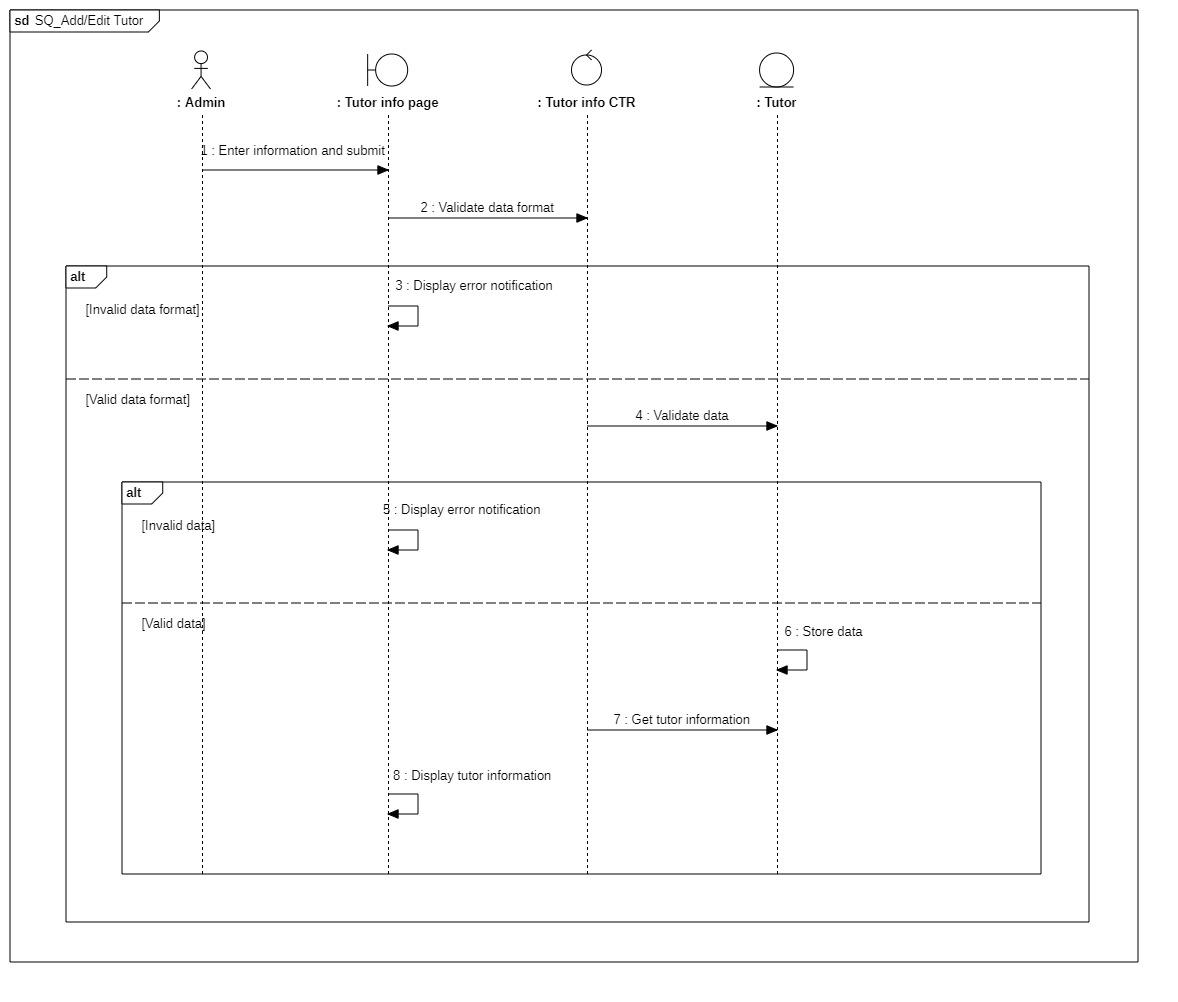
****

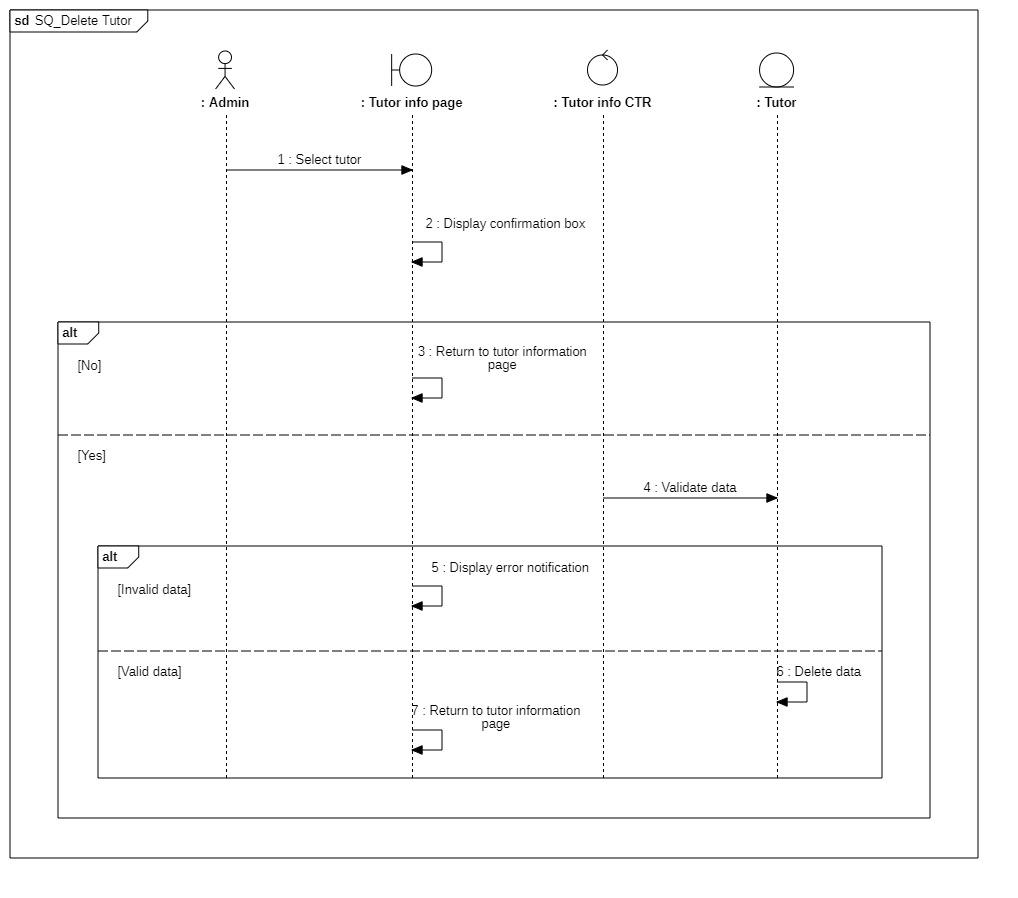
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Student info page** |  | getStudentListView | View list of student |
| **Student info CTR** |  | insertNewStudent(student) void | Add new student |
|  | getStudentData() student | Get student data based on text field |
|  | updateStudentData(stu\_ID) void | Update student data |
|  | getListOfStudent() array | Get list of student |
|  | searchStudent() array | Search for student based on ID or name |
|  | deleteStudent(studentID) void |  |
|  | getStudentStatus() bool |  |
| **Student** | Stu\_ID |  | Student’s main identification |
| Name |  | Student’s name |
| Phone\_Num |  | Student’s phone number |
| Email |  | Student’s email |
| Address |  | Student’s address |
| DOB |  | Student’s date of birth |
|  | insertNewStudent() void | Add new student |
|  | getStudentData() student | Get student data from text field |
|  | updateStudentData(stu\_ID) void | Get student data |
|  | getListOfStudent() array | Get list of student from a class from text field |
|  | searchStudent() array | Search for student from name or ID from text field |
|  | getStudentStatus() bool | Check if Student is active or not |

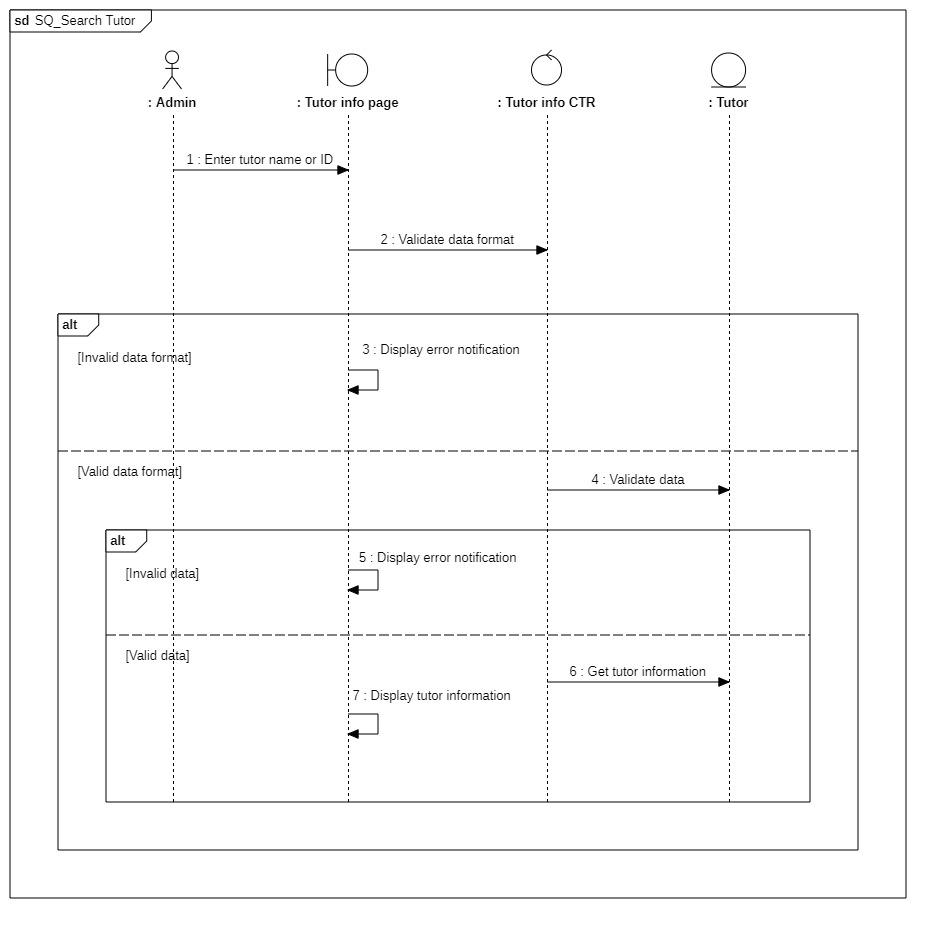
### 3.4.3 Manage tutor information

#### 3.4.3.1 Sequence diagram

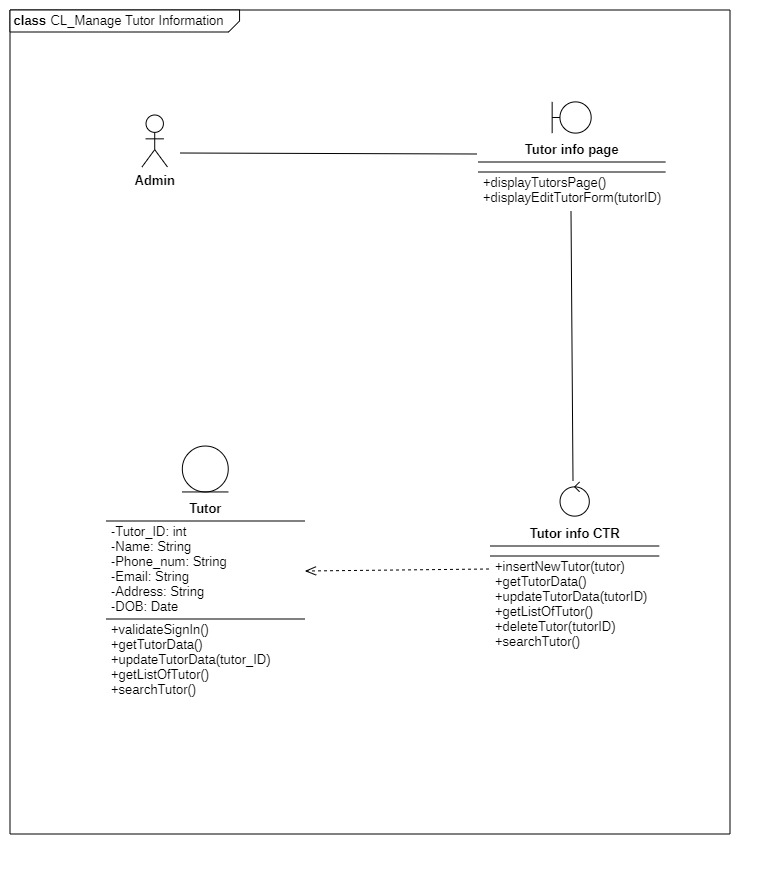
****

****

****

****

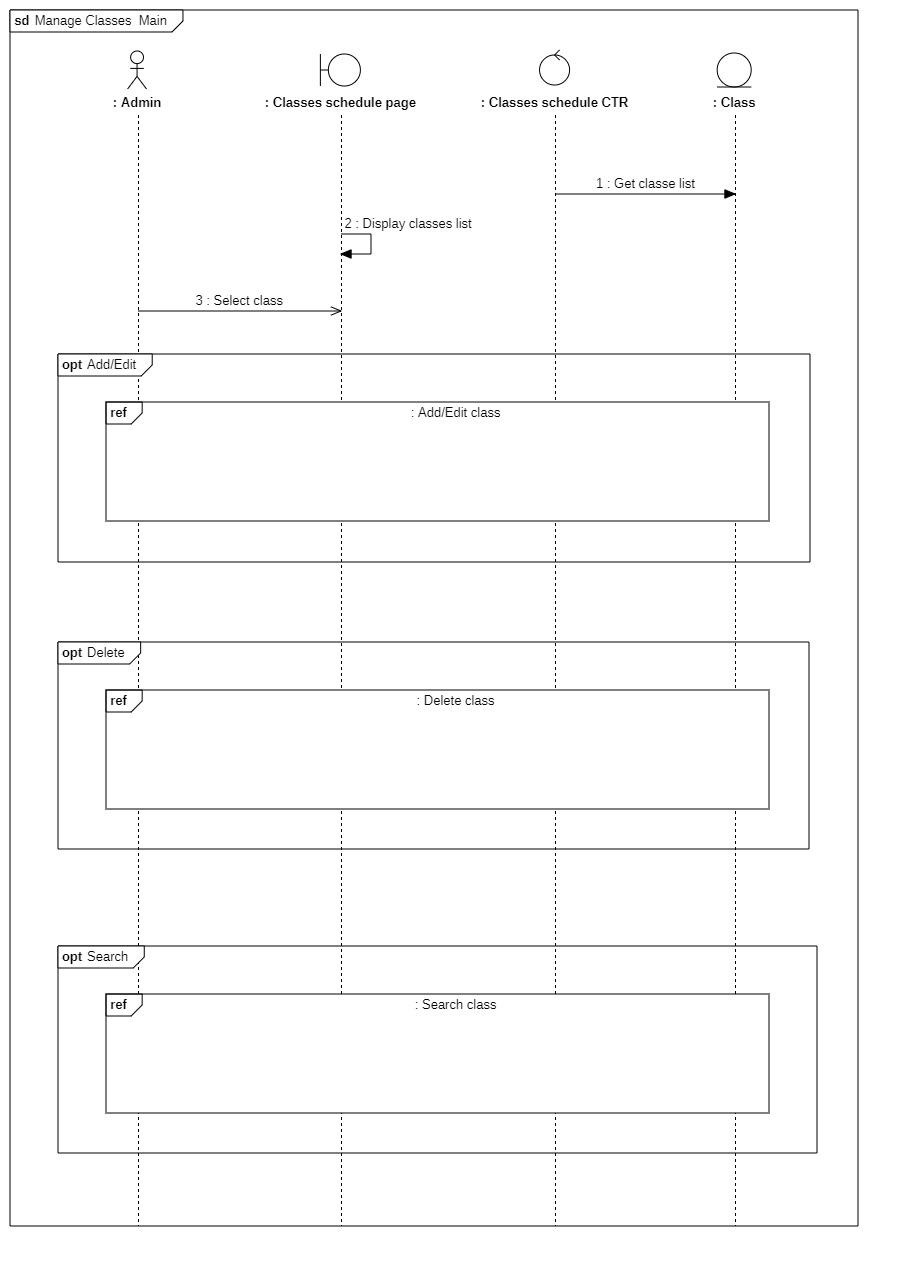
#### 3.4.3.2 Class diagram

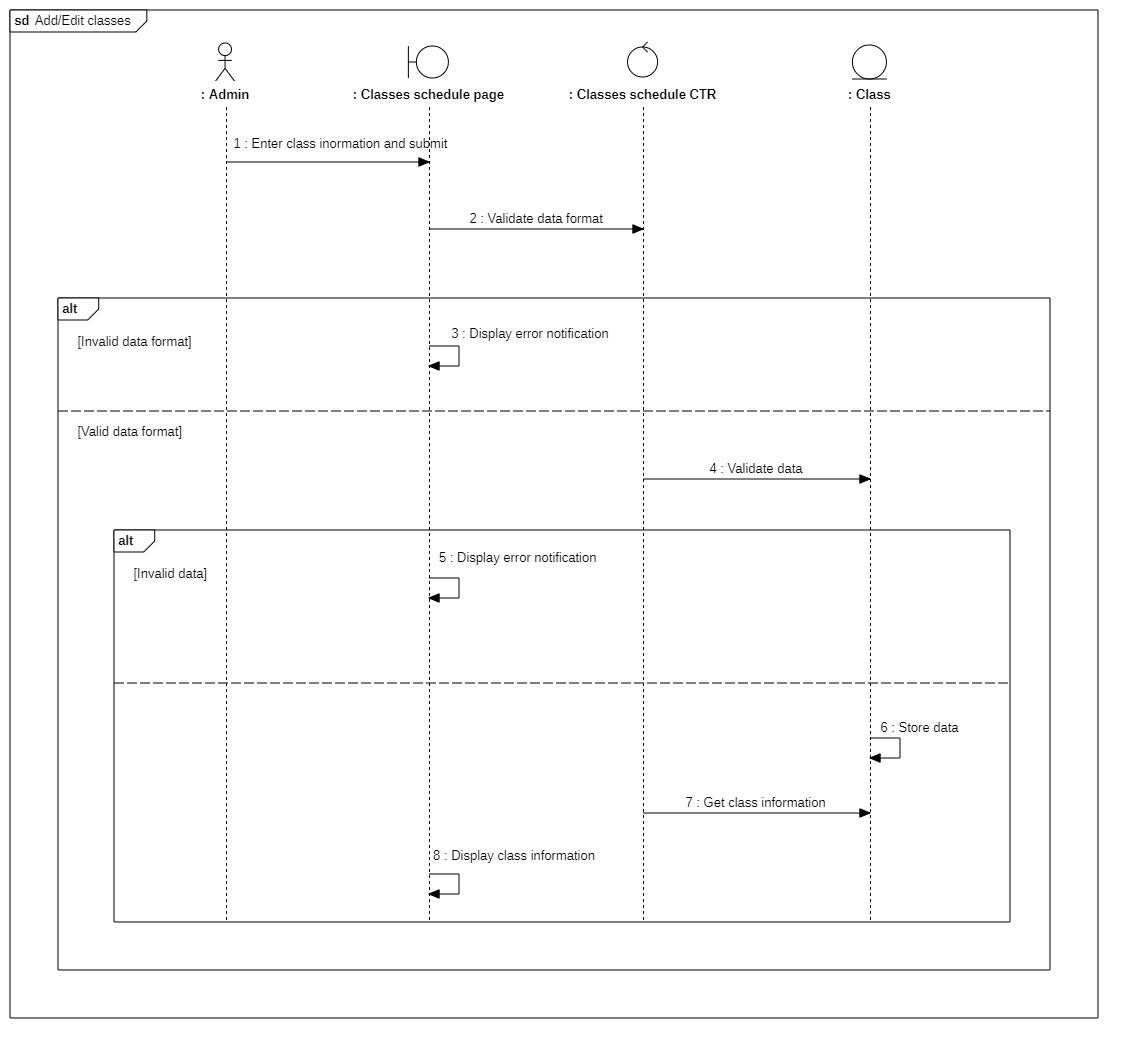
****

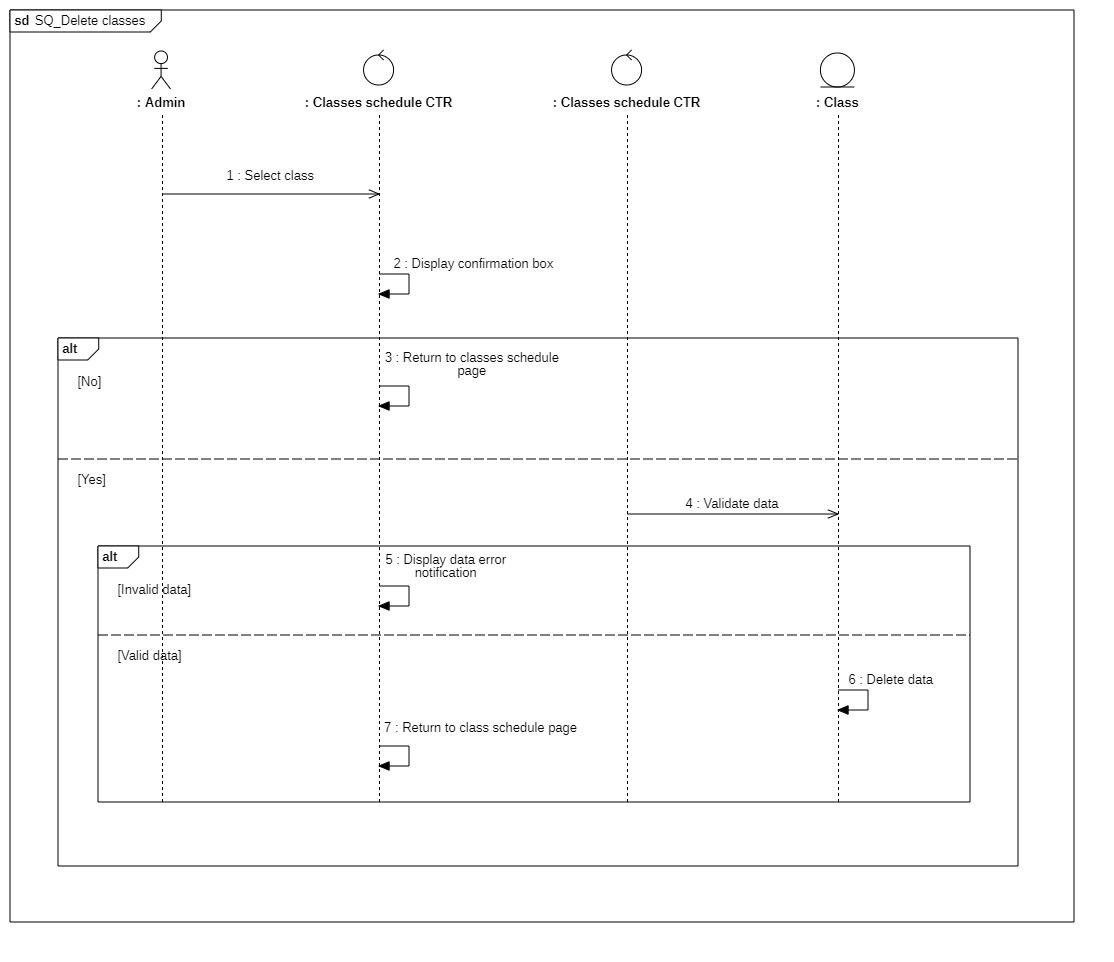
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Tutor info page** |  | displayTutorsPage() void | Display tutor information page |
|  |  | displayEditTutorForm(tutorID) void | Display form to edit tutor information |
| **Tutor info CTR** |  | insertNewTutor(tutor) void | Add new tutor |
|  | getTutorData() tutor | Get tutor data |
|  | updateTutorData(tutorID) | Update tutor data with tutor\_ID |
|  | getListOfTutor() array | Get list of all tutors |
|  | deleteTutor(tutorID) void | Delete tutor from database |
|  | searchTutor() array | Search tutor based on name or ID |
| **Tutor** | Tutor\_ID |  | Tutor’ s main identification |
| Name |  | Tutor’s name |
| Phone\_num |  | Tutor’s phone number |
| Email |  | Tutor’s email |
| Address |  | Tutor’s home address |
| DOB |  | Tutor’s date of birth |
|  | validateSignIn() bool | Check if sign-in is valid |
|  | getTutorData() tutor | Get tutor’s data |
|  | updateTutorData(tutor\_ID) void | Update the inserted tutor’s information |
|  | getListOfTutor() array | Return a list of all tutors |
|  | searchTutor() array | Search for information about the tutor from the text field |

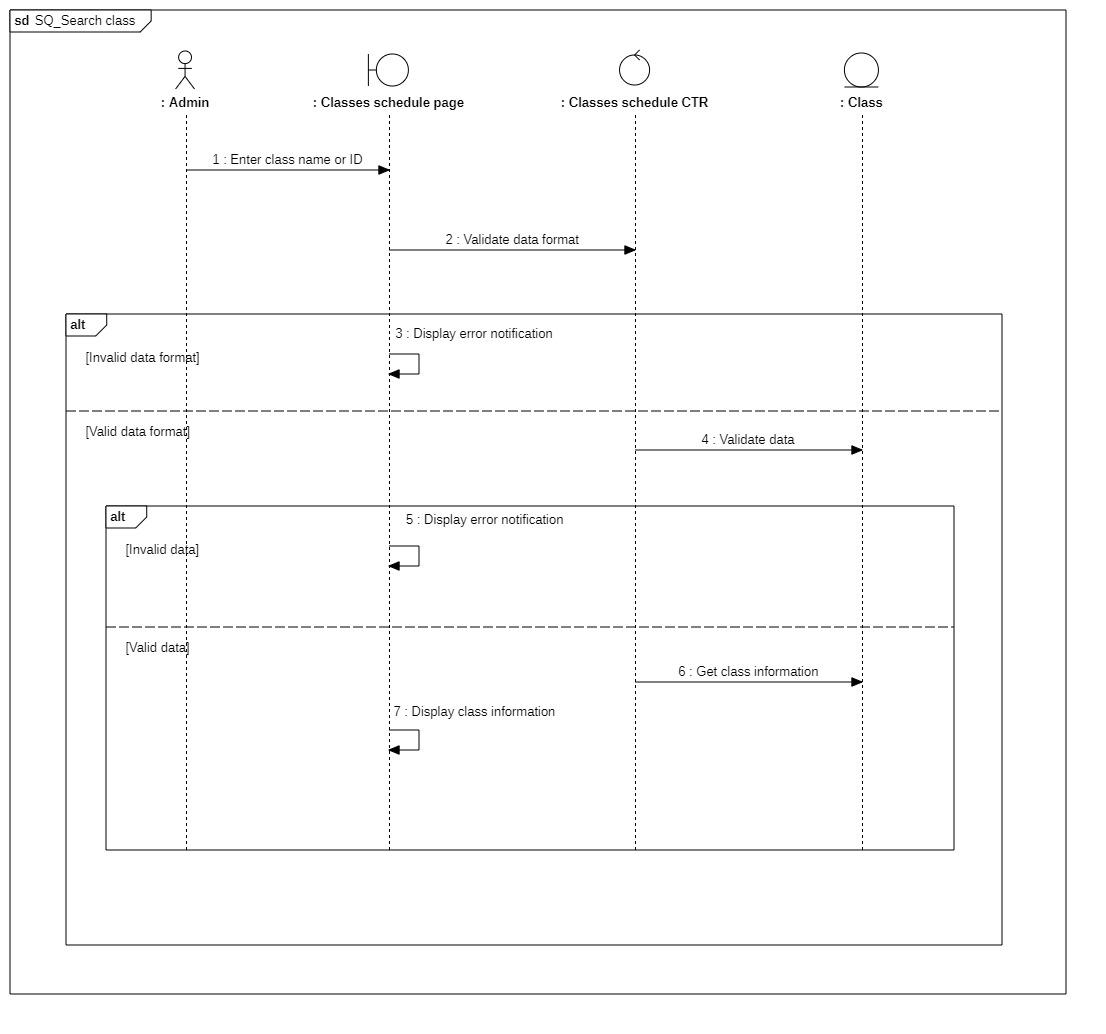
### 3.4.4 Manage classes

#### 3.4.4.1 Sequence diagram

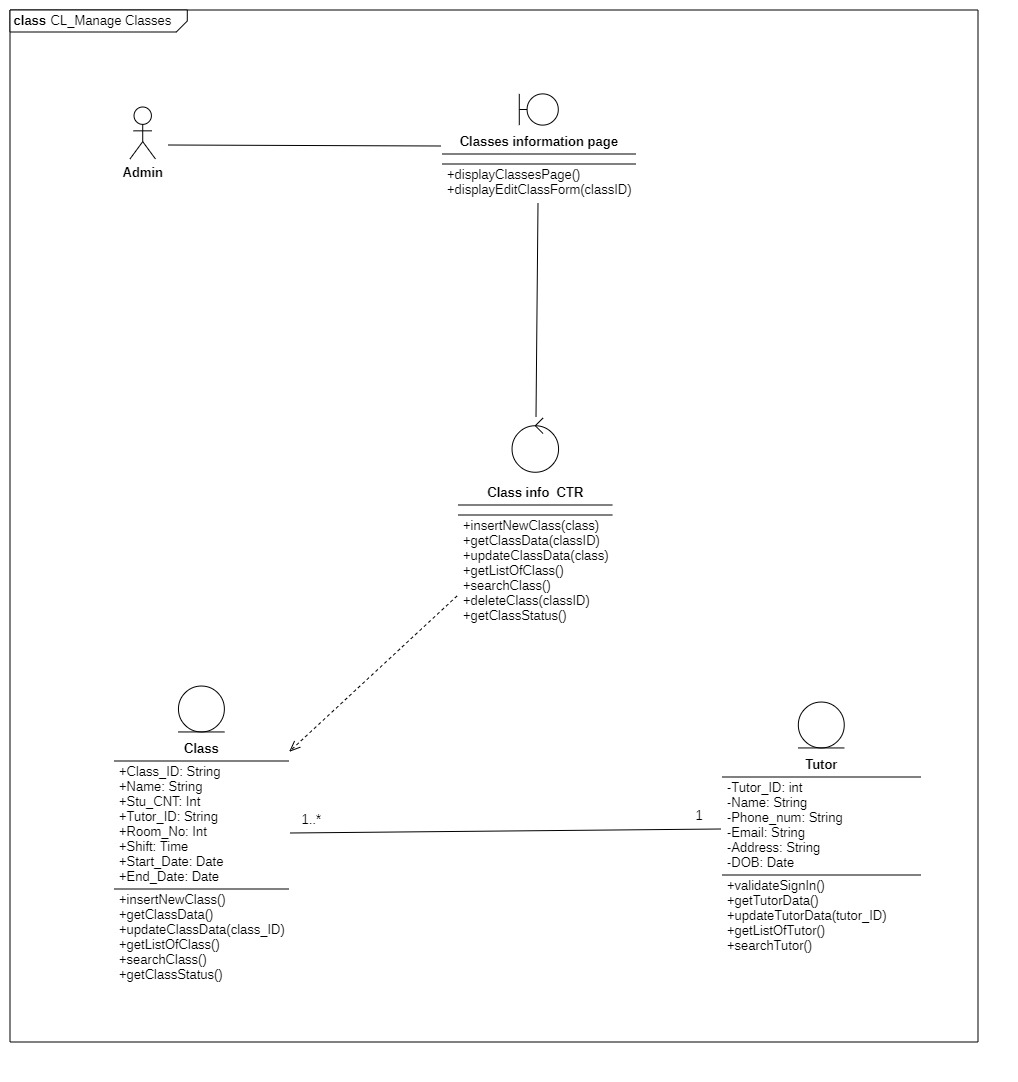
****

****

****

****

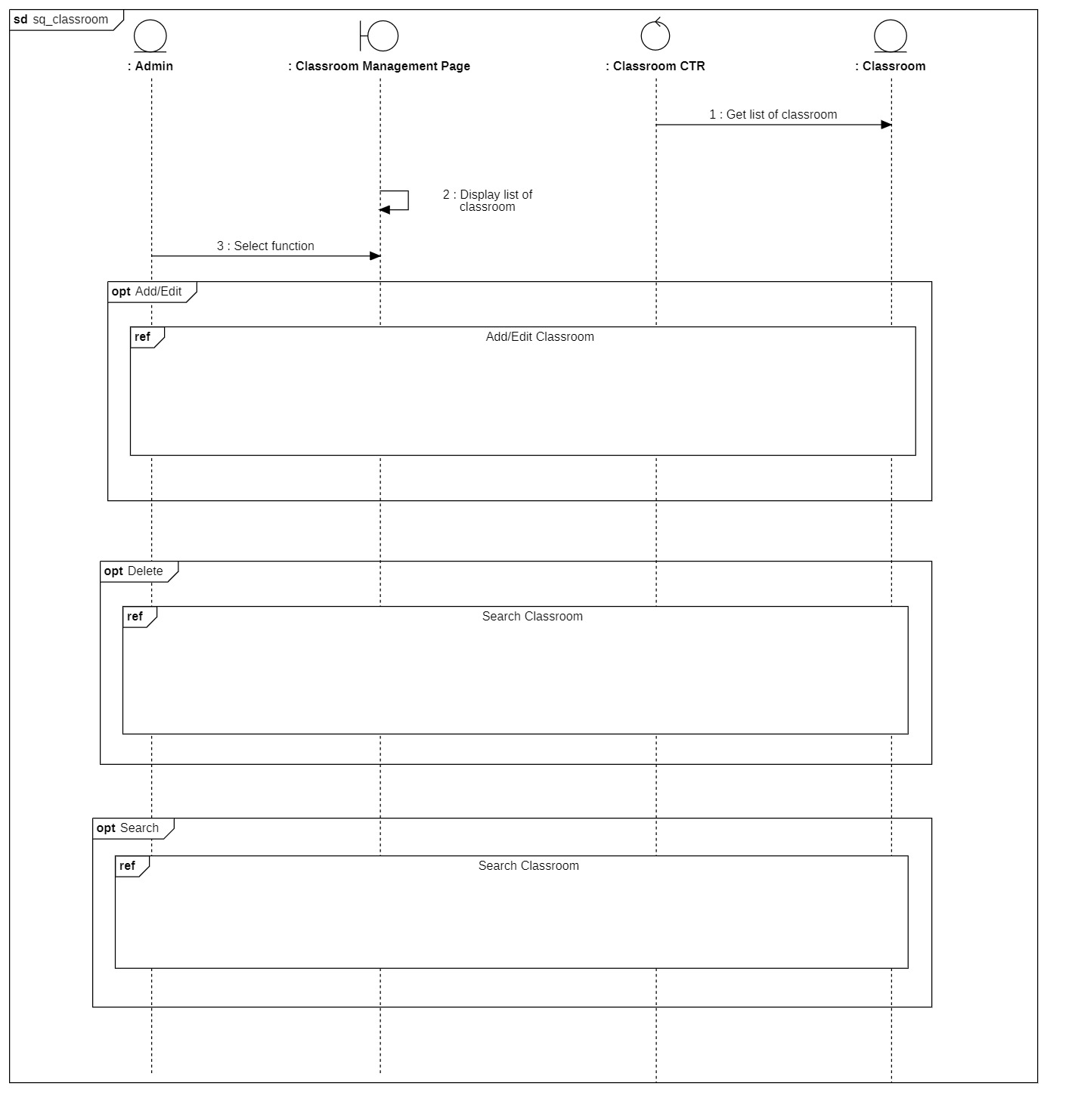
#### 3.4.4.2 Class diagram

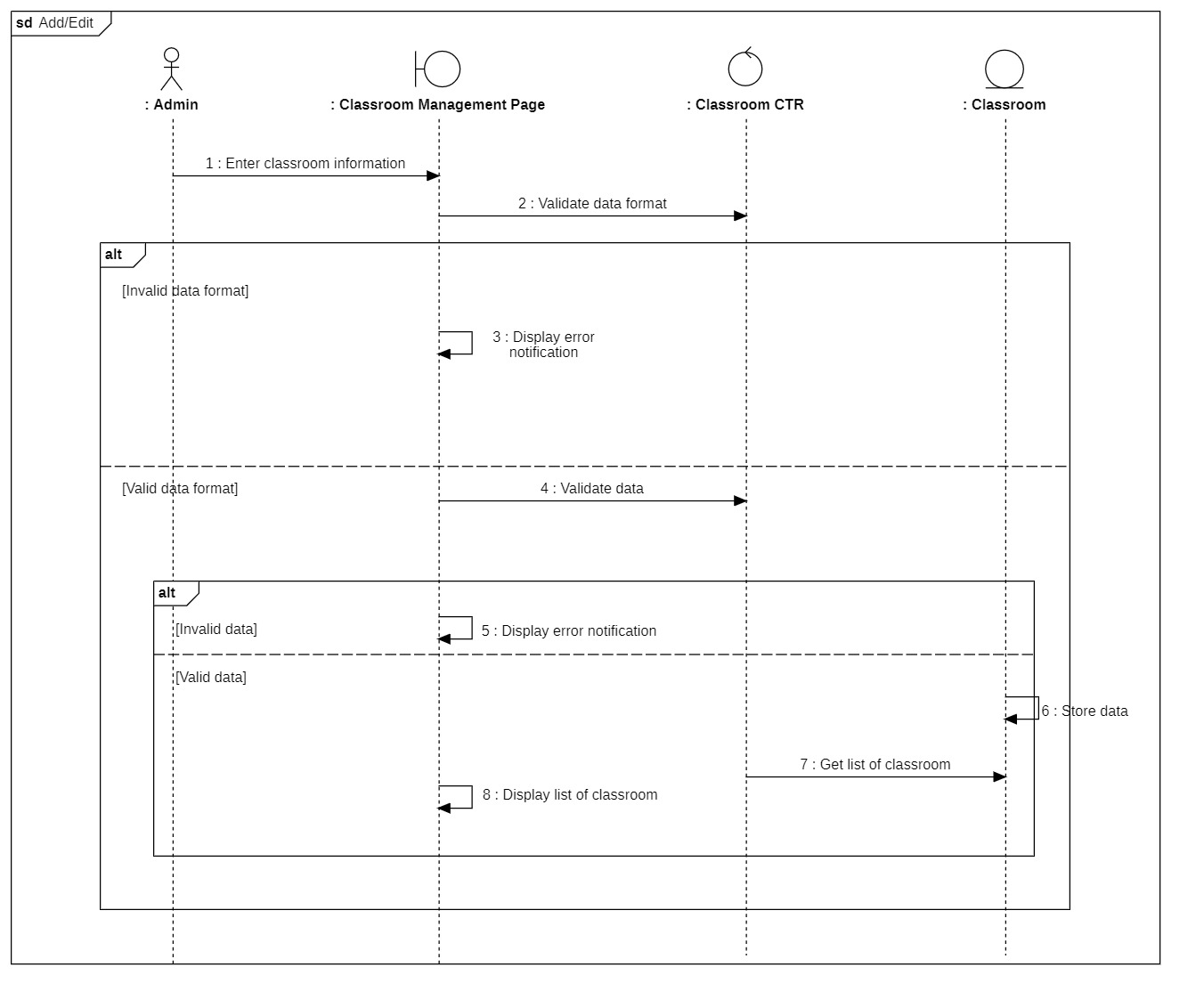
****

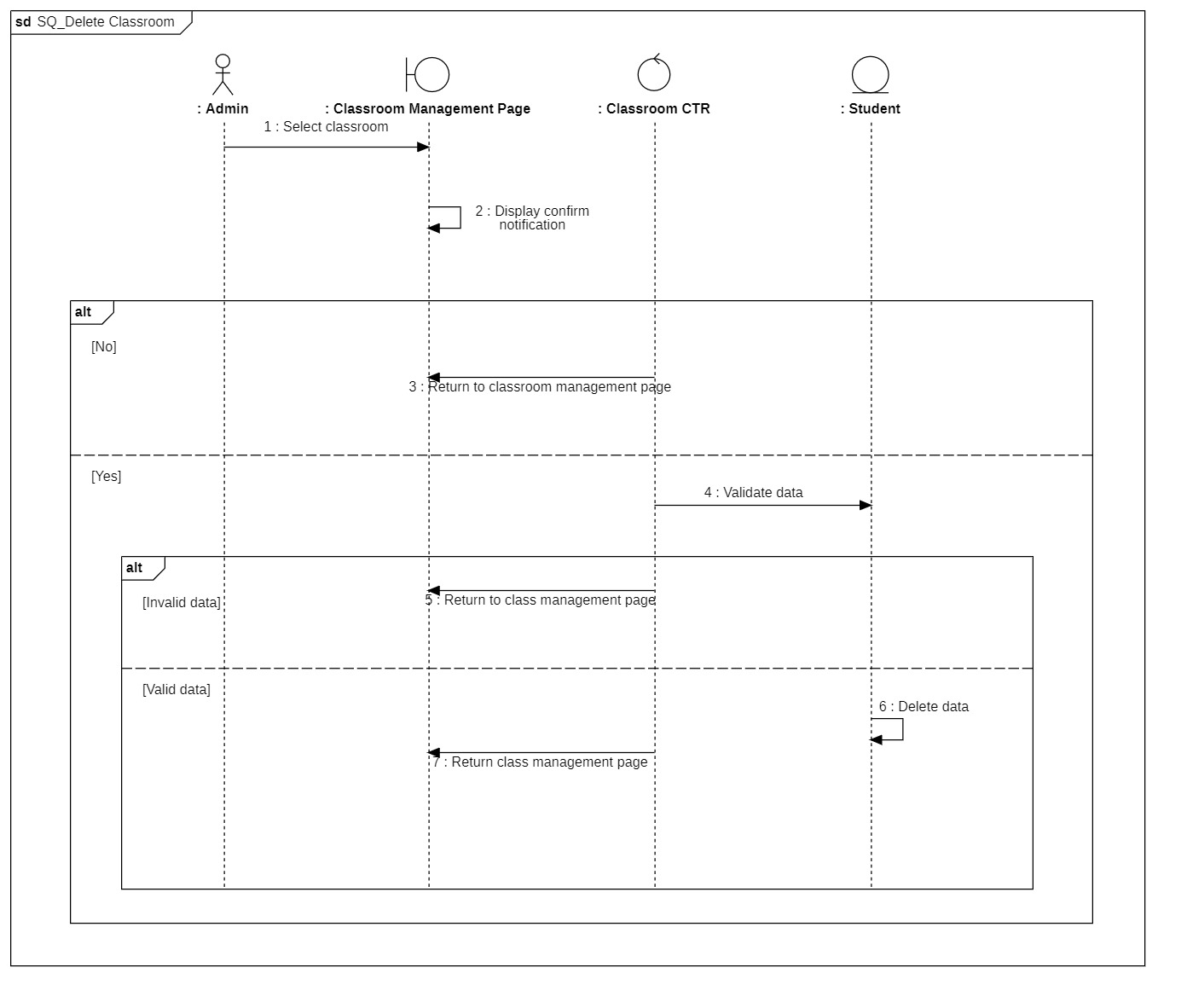
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Classes information page** |  | displayClassesPage() void | Display class information page |
|  |  | displayEditClassForm(classID) void | Display form to edit class information |
| **classes info CTR** |  | insertNewTutor(tutor) void | Add new tutor |
|  | getTutorData() tutor | Get tutor data |
|  | updateTutorData(tutorID) | Update tutor data with tutor\_ID |
|  | getListOfTutor() array | Get list of all tutors |
|  | deleteTutor(tutorID) void | Delete tutor from database |
|  | searchTutor() array | Search tutor based on name or ID |
| **Tutor** | Tutor\_ID |  | Tutor’ s main identification |
| Name |  | Tutor’s name |
| Phone\_num |  | Tutor’s phone number |
| Email |  | Tutor’s email |
| Address |  | Tutor’s home address |
| DOB |  | Tutor’s date of birth |
|  | validateSignIn() bool | Check if sign-in is valid |
|  | getTutorData() tutor | Get tutor’s data |
|  | updateTutorData(tutor\_ID) void | Update the inserted tutor’s information |
|  | getListOfTutor() array | Return a list of all tutors |
|  | searchTutor() array | Search for information about the tutor from the text field |
| **Class** | Class\_ID |  | Class’s main identification |
| Name |  | Class’s name |
| Stu\_CNT |  | Number of student in class |
| Tutor\_ID |  | ID of tutor in charge of class |
| Room\_No |  | Room number |
| Time |  | Time of study |
| Start\_Date |  | Start date for class |
| End\_Date |  | End date for class |
|  | insertNewClass() void | Create new class |
|  | getClassData() class | Get class data |
|  | updateClassData(class\_ID) void | Update class data w for selected class with class\_ID |
|  | getListOfClass() array | Get list of current classes |
|  | searchClass() array | Search for a class based on text field |
|  | getClassStatus() bool | Check if class is still active or not |

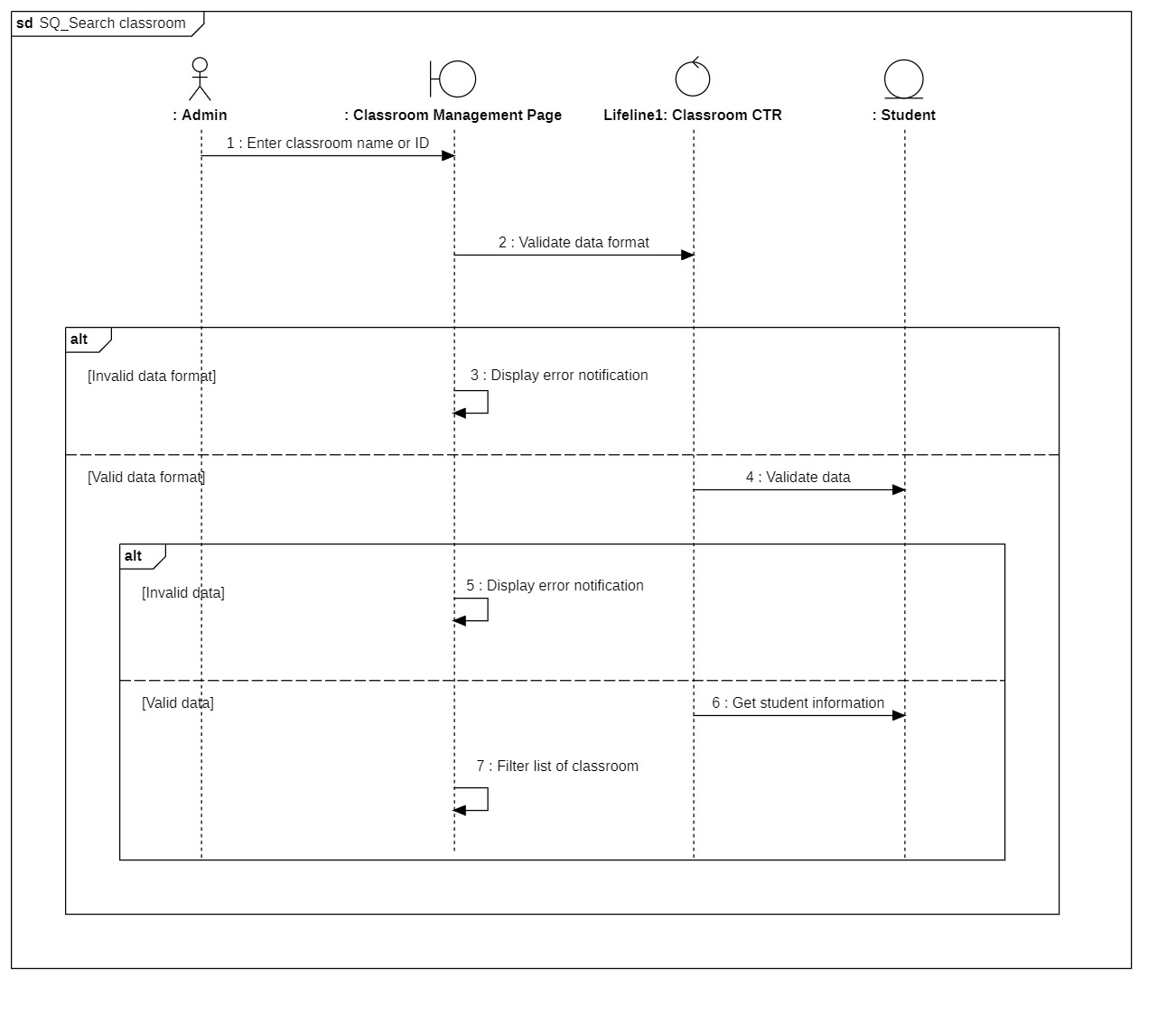
### 3.4.5 Manage classrooms

#### 3.4.5.1 Sequence diagram

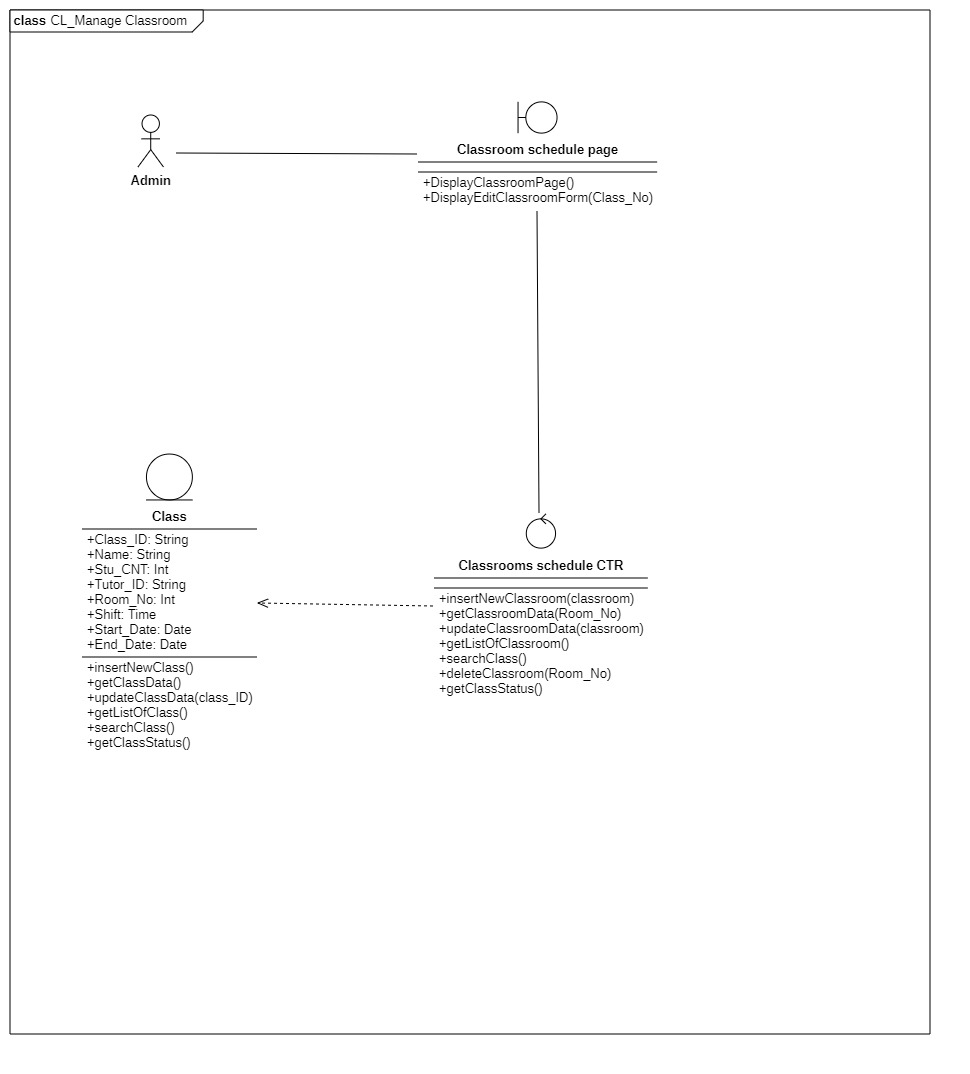








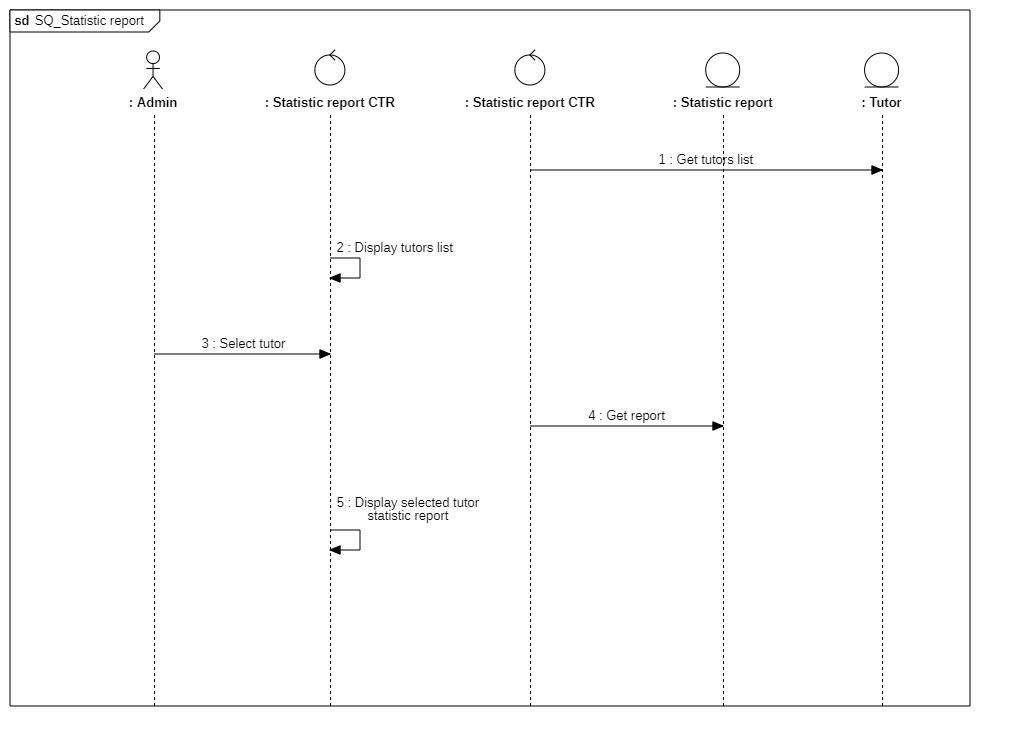
#### 3.4.5.2 Class diagram

****

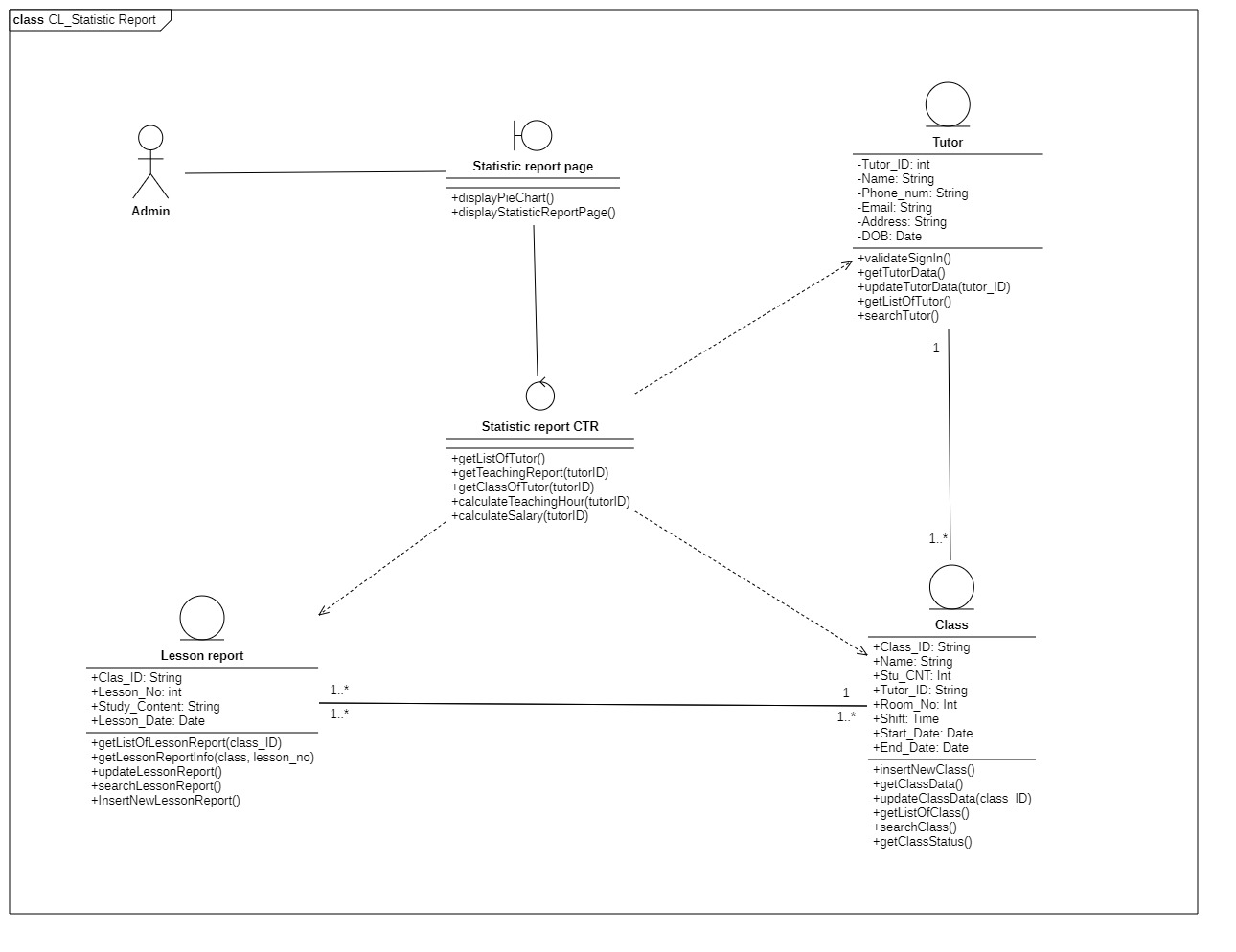
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Classroom information page** |  | displayClassesPage() void | Display classrooms information page |
|  |  | displayEditClassForm(classID) void | Display form to edit classroom information |
| **classroom info CTR** |  | insertNewTutor(tutor) void | Add new tutor |
|  | getTutorData() tutor | Get tutor data |
|  | updateTutorData(tutorID) | Update tutor data with tutor\_ID |
|  | getListOfTutor() array | Get list of all tutors |
|  | deleteTutor(tutorID) void | Delete tutor from database |
|  | searchTutor() array | Search tutor based on name or ID |
| **Class** | Class\_ID |  | Class’s main identification |
| Name |  | Class’s name |
| Stu\_CNT |  | Number of student in class |
| Tutor\_ID |  | ID of tutor in charge of class |
| Room\_No |  | Room number |
| Time |  | Time of study |
| Start\_Date |  | Start date for class |
| End\_Date |  | End date for class |
|  | insertNewClass() void | Create new class |
|  | getClassData() class | Get class data |
|  | updateClassData(class\_ID) void | Update class data w for selected class with class\_ID |
|  | getListOfClass() array | Get list of current classes |
|  | searchClass() array | Search for a class based on text field |
|  | getClassStatus() bool | Check if class is still active or not |

### 3.4.6 Manage statistic report

#### 3.4.6.1 Sequence diagram

****

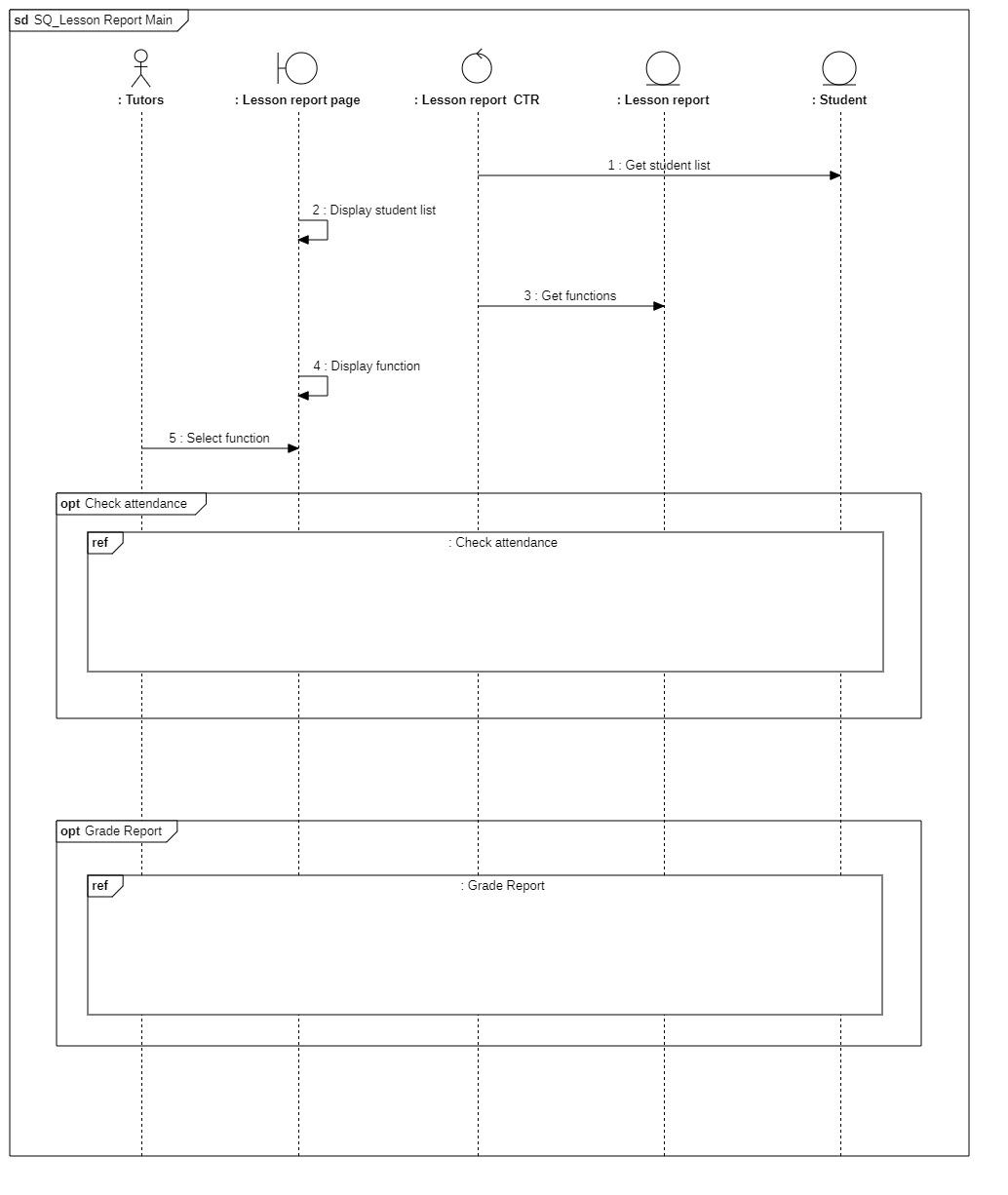
#### 3.4.6.2 Class diagram

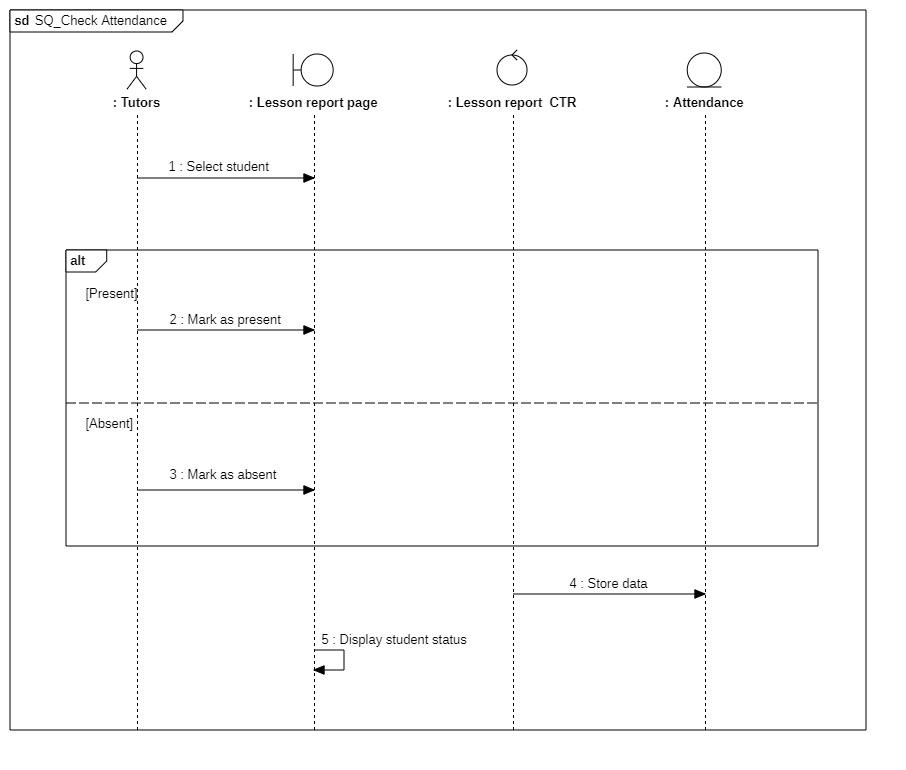
****

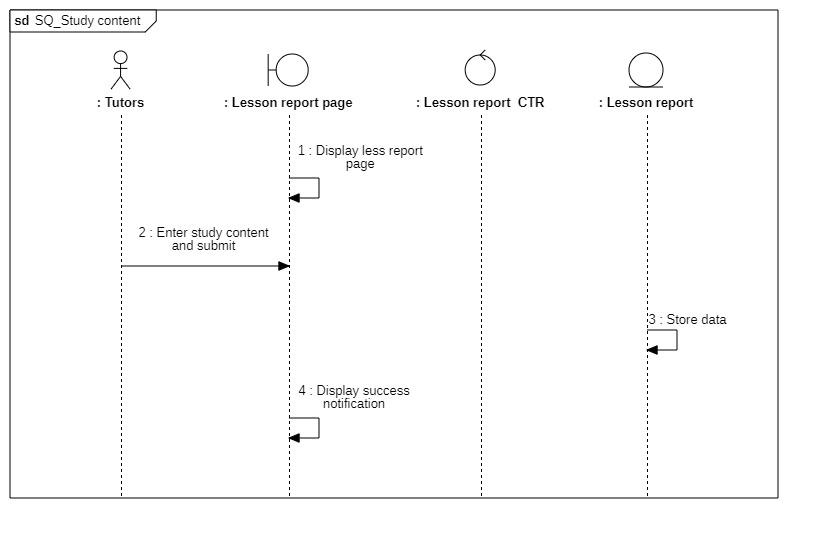
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Statistic** |  | getTeachingReport(tutorID, classID) | get teaching report for tutor based on classes taught |
|  |  | calculateTeachingWork(tutorID, classID) | Calculate the number of lesson for each class tutor taught |
| **Lesson Report** | Class\_ID |  | Class’s ID |
| Lesson\_No |  | Lesson’s cardinal number |
| Study\_Content |  | Report on the content of the lesson |
| Lesson\_Date |  | The date of the lesson |
|  | getListOfLessonReport(class\_ID) array | Get list of lesson report based on the class\_ID |
|  | getLessonReportInfo(class\_ID, lesson\_no) lessonreport | Return the lesson report info based on the lesson\_no and class\_ID |
|  | updateLessonReport() void | Update the lesson report of the current day |
|  | searchLessonReport() array | Search for lesson report based on text field |
|  | InsertNewLessonReport() lessonreport | Insert a new lesson report for the current day |
| **Class** | Class\_ID |  | Class’s main identification |
| Name |  | Class’s name |
| Stu\_CNT |  | Number of student in class |
| Tutor\_ID |  | ID of tutor in charge of class |
| Room\_No |  | Room number |
| Time |  | Time of study |
| Start\_Date |  | Start date for class |
| End\_Date |  | End date for class |
|  | insertNewClass() void | Create new class |
|  | getClassData() class | Get class data |
|  | updateClassData(class\_ID) void | Update class data w for selected class with class\_ID |
|  | getListOfClass() array | Get list of current classes |
|  | searchClass() array | Search for a class based on text field |
|  | getClassStatus() bool | Check if class is still active or not |
| **Tutor** | Tutor\_ID |  | Tutor’ s main identification |
| Name |  | Tutor’s name |
| Phone\_num |  | Tutor’s phone number |
| Email |  | Tutor’s email |
| Address |  | Tutor’s home address |
| DOB |  | Tutor’s date of birth |
|  | validateSignIn() bool | Check if sign-in is valid |
|  | getTutorData() tutor | Get tutor’s data |
|  | updateTutorData(tutor\_ID) void | Update the inserted tutor’s information |
|  | getListOfTutor() array | Return a list of all tutors |
|  | searchTutor() array | Search for information about the tutor from the text field |

### 3.4.7 Manage lesson reports

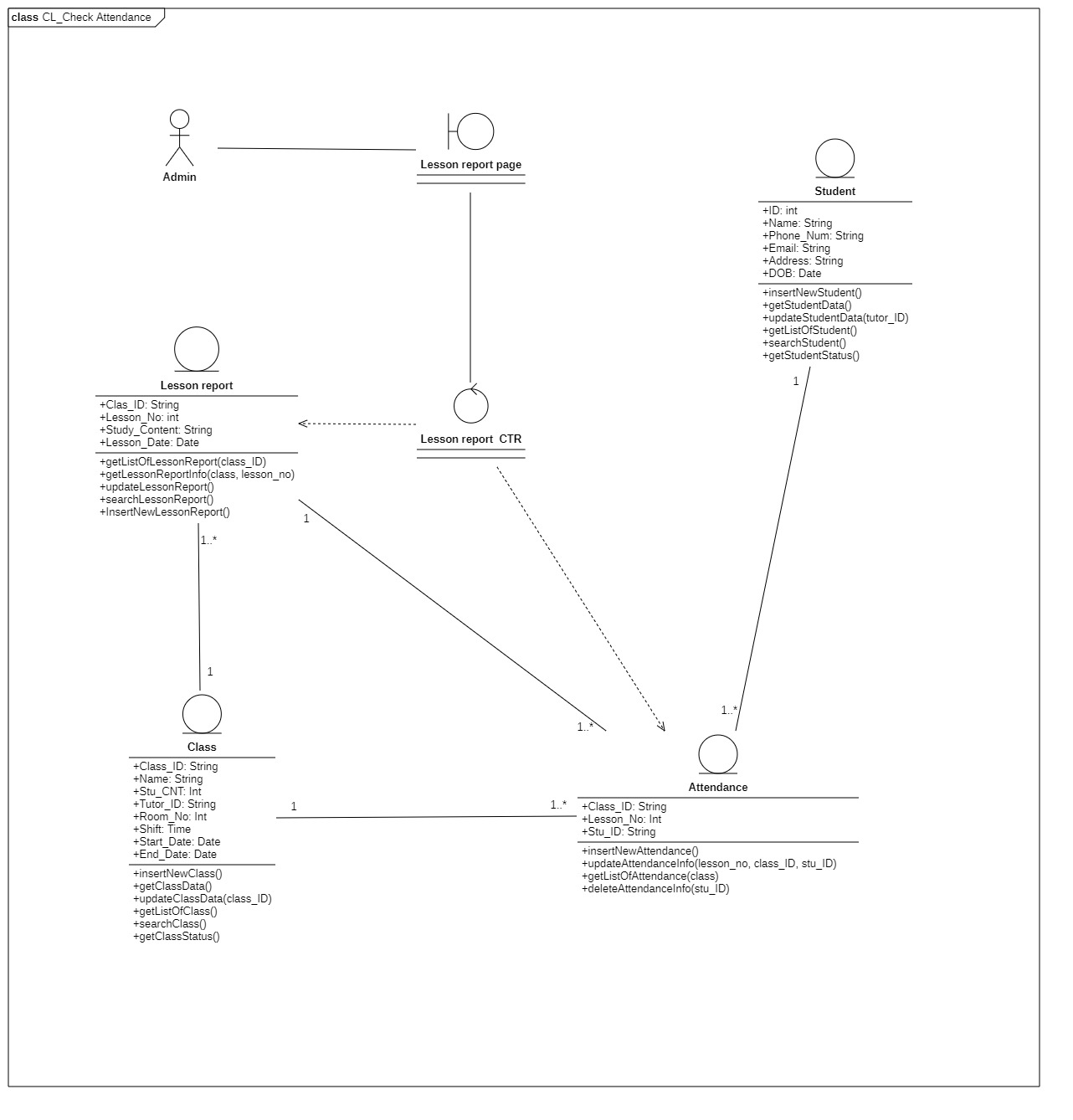
#### 3.4.7.1 Sequence diagram

****

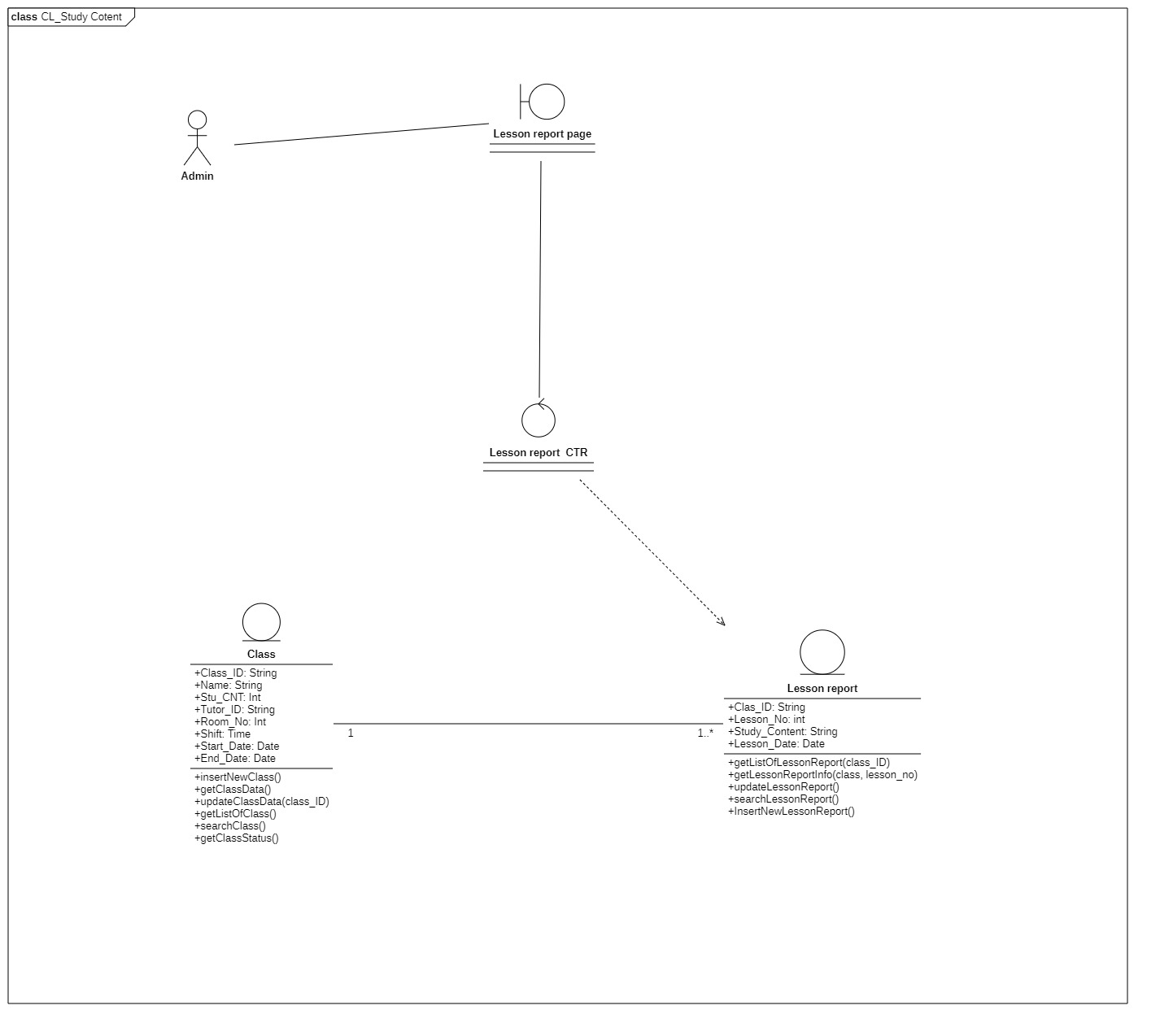
****

****

#### 3.4.7.2 Class diagram

****

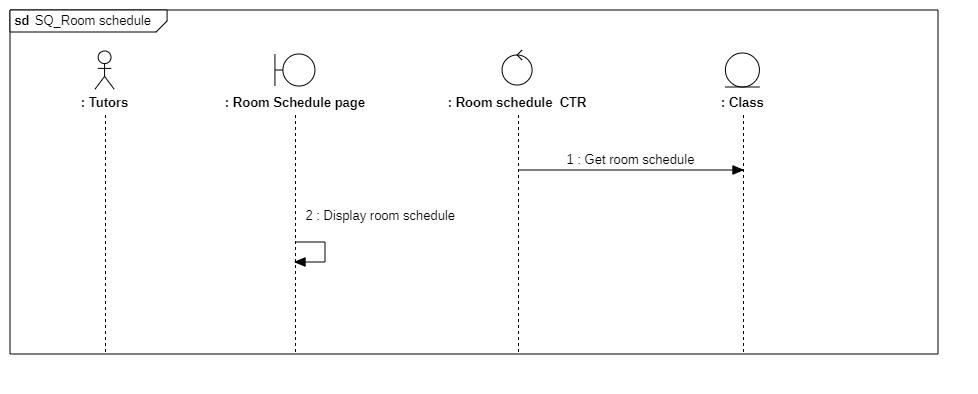
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Lesson Report CTR** |  | getListOfStudent(class\_ID) array | get list of students belonging to class\_ID |
|  | getListOfLessonReport (class\_ID) array | Get list of lesson report belonging to class\_ID |
|  | getListOfAttendance(class\_ID) array | Get list of attendance belonging to class\_ID |
|  | editAttendance(stu\_ID, class\_ID) void | Edit the attendance of |
|  | refreshPage() void | Reload page and display updated data (if any) |
|  | searchStudent(stu\_ID, name) void | Search for student based on their name or stu\_ID |
| **Lesson Report** | Class\_ID |  | Class’s ID |
| Lesson\_No |  | Lesson’s cardinal number |
| Study\_Content |  | Report on the content of the lesson |
| Lesson\_Date |  | The date of the lesson |
|  | getListOfLessonReport(class\_ID) array | Get list of lesson report based on the class\_ID |
|  | getLessonReportInfo(class\_ID, lesson\_no) lessonreport | Return the lesson report info based on the lesson\_no and class\_ID |
|  | updateLessonReport() void | Update the lesson report of the current day |
|  | searchLessonReport() array | Search for lesson report based on text field |
|  | InsertNewLessonReport() lessonreport | Insert a new lesson report for the current day |
| **Attendance** | Class\_ID |  | Class’s main identification |
| Lesson\_No |  | Lesson cardinal numebr |
| Stu\_ID |  | Student’ s main identification |
|  | insertNewAttendance(stu\_ID) attendance | Add new attendance for student for today lesson |
|  | getListOfAtendance(class\_ID) array | Get a list of attendance belonging to class\_ID |
|  | updateAttendanceInfo(lesson\_no, class\_id, student\_id) void | Update the attendance of student stu\_ID from class\_ID from lesson\_No |
|  | getAttendanceInfo(student) attendance |  |
|  | deleteAttendanceInfo(student) void | Delete attendance info from student |
| **Class** | Class\_ID |  | Class’s main identification |
| Name |  | Class’s name |
| Stu\_CNT |  | Number of student in class |
| Tutor\_ID |  | ID of tutor in charge of class |
| Room\_No |  | Room number |
| Time |  | Time of study |
| Start\_Date |  | Start date for class |
| End\_Date |  | End date for class |
|  | insertNewClass() void | Create new class |
|  | getClassData() class | Get class data |
|  | updateClassData(class\_ID) void | Update class data w for selected class with class\_ID |
|  | getListOfClass() array | Get list of current classes |
|  | searchClass() array | Search for a class based on text field |
|  | getClassStatus() bool | Check if class is still active or not |
| **Student** | Stu\_ID |  | Student’s main identification |
| Name |  | Student’s name |
| Phone\_Num |  | Student’s phone number |
| Email |  | Student’s email |
| Address |  | Student’s address |
| DOB |  | Student’s date of birth |
|  | insertNewStudent() void | Add new student |
|  | getStudentData() student | Get student data from text field |
|  | updateStudentData(stu\_ID) void | Get student data |
|  | getListOfStudent() array | Get list of student from a class from text field |
|  | searchStudent() array | Search for student from name or ID from text field |
|  | getStudentStatus() bool | Check if Student is active or not |

****

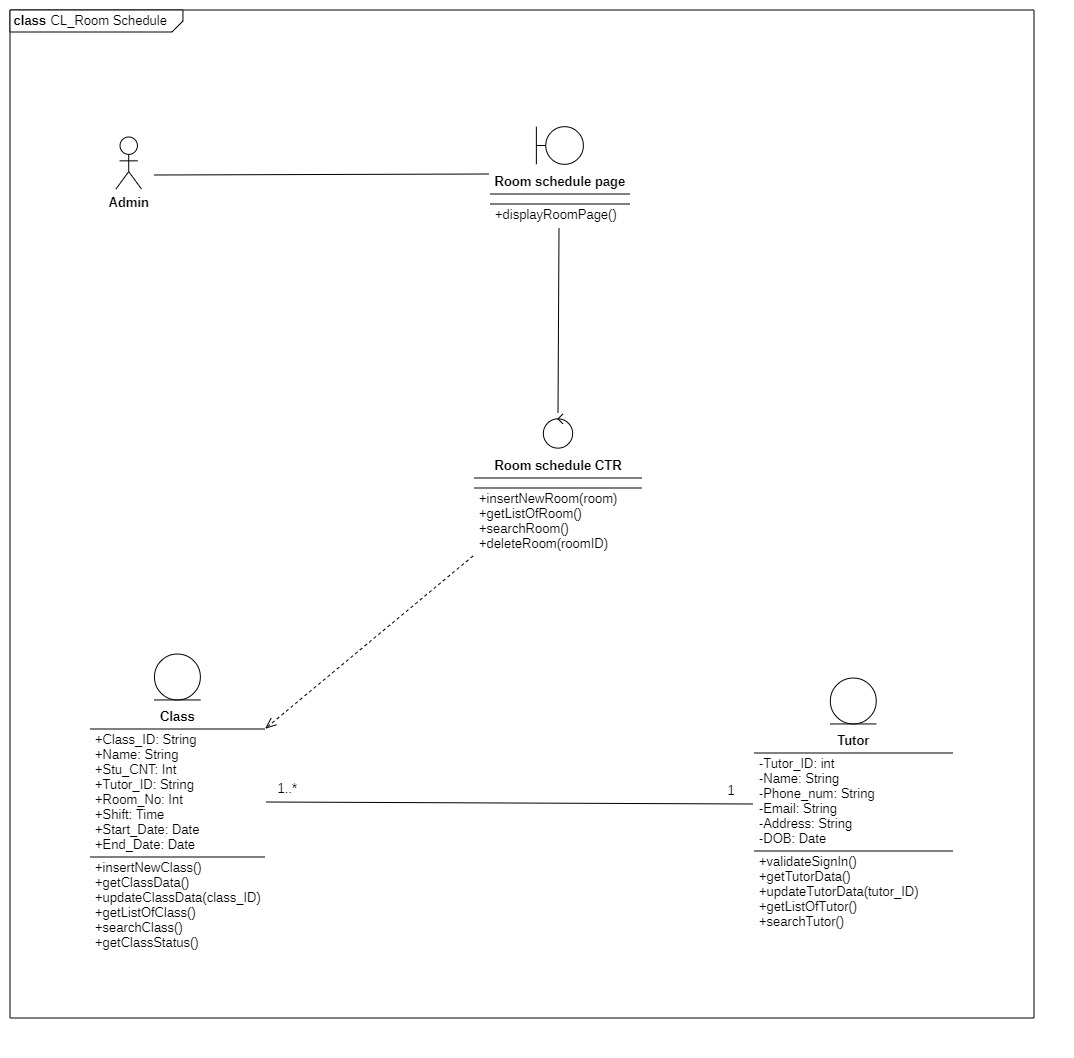
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Lesson Report CTR** |  | getListOfStudent(classID) array | get list of students belonging to class\_ID |
|  | getListOfLessonReport (classID) array | Get list of lesson report belonging to class\_ID |
|  | getListOfAttendance(classID) array | Get list of attendance belonging to class\_ID |
|  | editAttendance(stu\_ID, classID) void | Edit the attendance of |
|  | refreshPage() void | Reload page and display updated data (if any) |
|  | searchStudent(stu\_ID, name) void | Search for student based on their name or stu\_ID |
| **Lesson Report** | Class\_ID |  | Class’s ID |
| Lesson\_No |  | Lesson’s cardinal number |
| Study\_Content |  | Report on the content of the lesson |
| Lesson\_Date |  | The date of the lesson |
|  | getListOfLessonReport(class\_ID) array | Get list of lesson report based on the class\_ID |
|  | getLessonReportInfo(class\_ID, lesson\_no) lessonreport | Return the lesson report info based on the lesson\_no and class\_ID |
|  | updateLessonReport() void | Update the lesson report of the current day |
|  | searchLessonReport() array | Search for lesson report based on text field |
|  | InsertNewLessonReport() lessonreport | Insert a new lesson report for the current day |
| **Class** | Class\_ID |  | Class’s main identification |
| Name |  | Class’s name |
| Stu\_CNT |  | Number of student in class |
| Tutor\_ID |  | ID of tutor in charge of class |
| Room\_No |  | Room number |
| Time |  | Time of study |
| Start\_Date |  | Start date for class |
| End\_Date |  | End date for class |
|  | insertNewClass() void | Create new class |
|  | getClassData() class | Get class data |
|  | updateClassData(class\_ID) void | Update class data w for selected class with class\_ID |
|  | getListOfClass() array | Get list of current classes |
|  | searchClass() array | Search for a class based on text field |
|  | getClassStatus() bool | Check if class is still active or not |

### 3.4.8 Manage room schedule

#### 3.4.8.1 Sequence diagram

****

#### 3.4.8.2 Class diagram

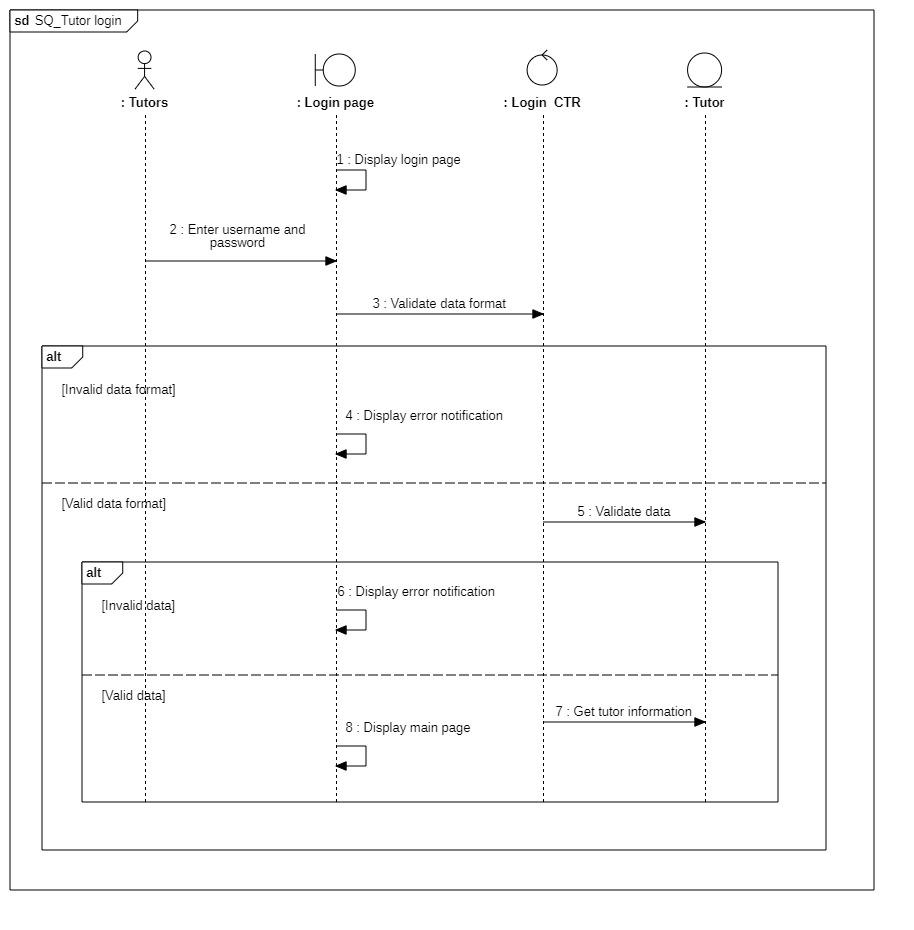
****

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Teaching schedule CTR** |  | getListOfClass(tutorID) | get list of class tutor have to teach |
| **Class** | Class\_ID |  | Class’s main identification |
| Name |  | Class’s name |
| Stu\_CNT |  | Number of student in class |
| Tutor\_ID |  | ID of tutor in charge of class |
| Room\_No |  | Room number |
| Time |  | Time of study |
| Start\_Date |  | Start date for class |
| End\_Date |  | End date for class |
|  | insertNewClass() void | Create new class |
|  | getClassData() class | Get class data |
|  | updateClassData(class\_ID) void | Update class data w for selected class with class\_ID |
|  | getListOfClass() array | Get list of current classes |
|  | searchClass() array | Search for a class based on text field |
|  | getClassStatus() bool | Check if class is still active or not |
| **Tutor** | Tutor\_ID |  | Tutor’ s main identification |
| Name |  | Tutor’s name |
| Phone\_num |  | Tutor’s phone number |
| Email |  | Tutor’s email |
| Address |  | Tutor’s home address |
| DOB |  | Tutor’s date of birth |
|  | validateSignIn() bool | Check if sign-in is valid |
|  | getTutorData() tutor | Get tutor’s data |
|  | updateTutorData(tutor\_ID) void | Update the inserted tutor’s information |
|  | getListOfTutor() array | Return a list of all tutors |
|  | searchTutor() array | Search for information about the tutor from the text field |

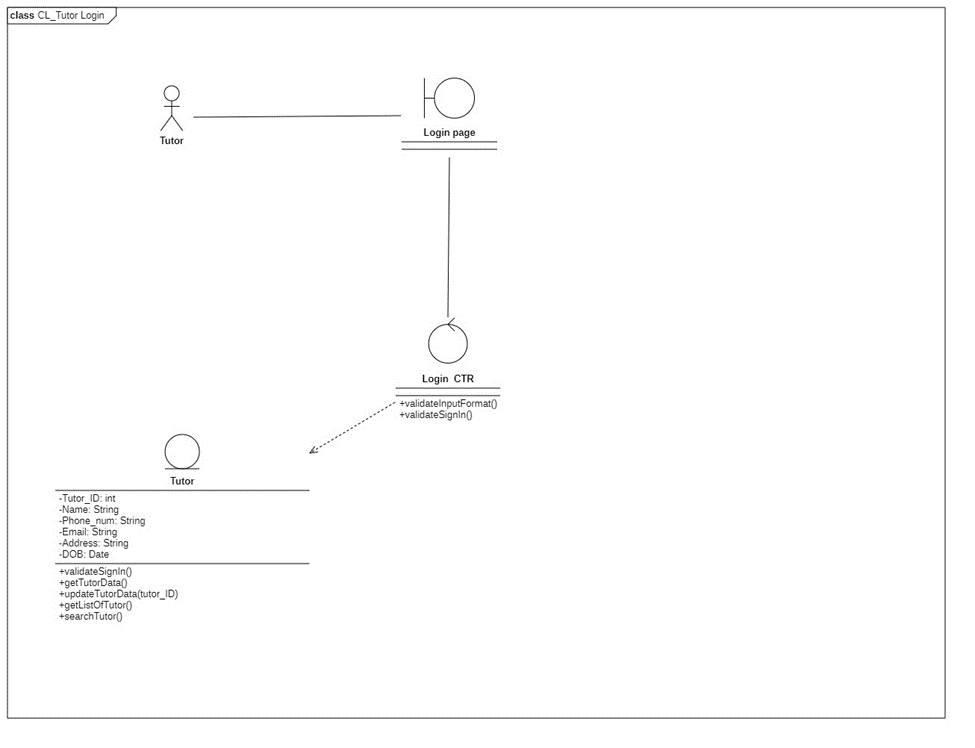
## 3.5 Sequence diagram and class diagram (for tutor)

### 3.5.1 Login in

#### 3.5.1.1 Sequence diagram

****

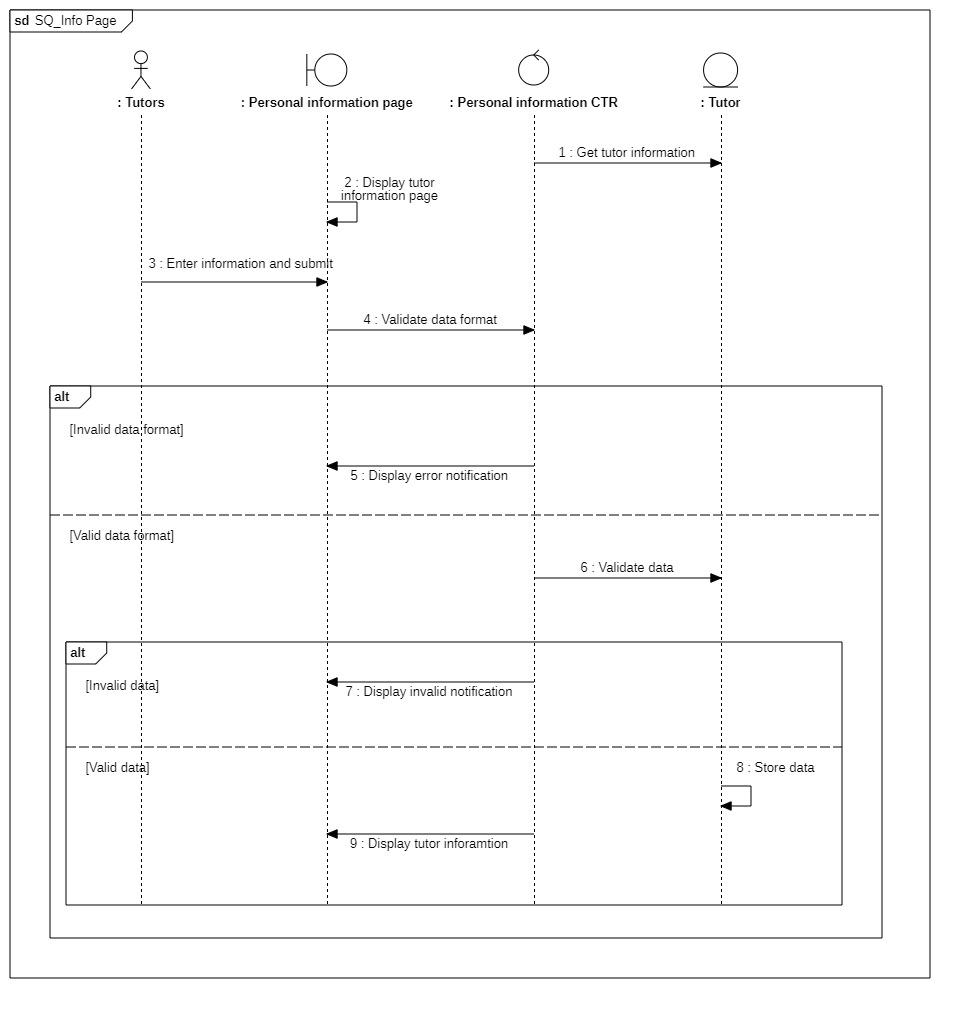
#### 3.5.1.2 Class diagram

****

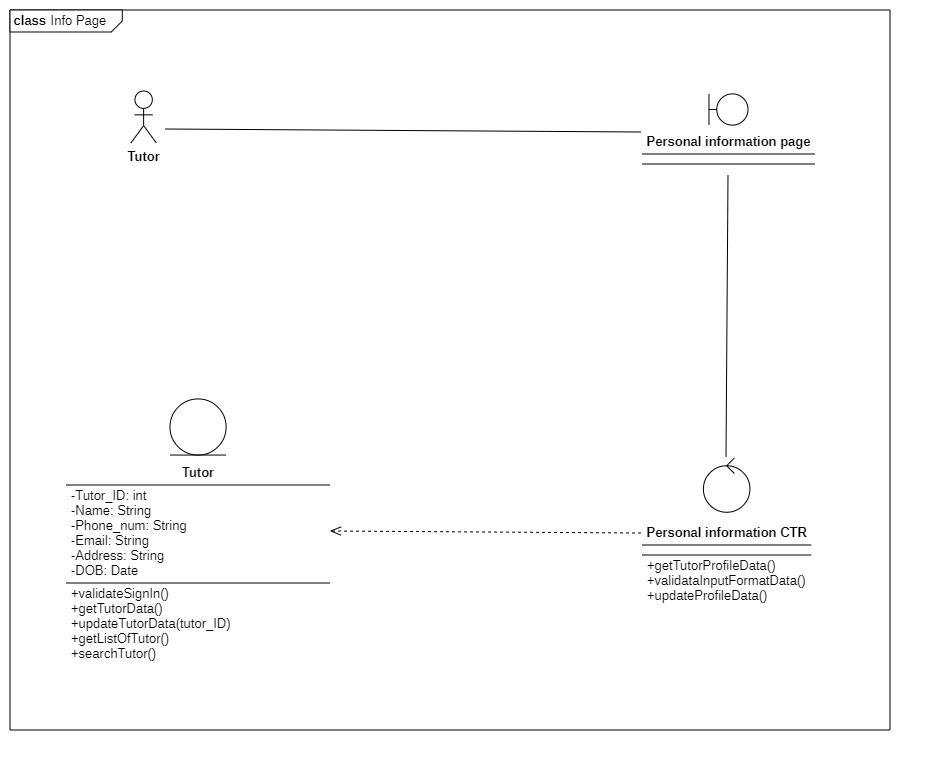
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Login CTR** |  | **validateInputFormat** | **Check if input data format is vaid** |
|  | **validateSignIn** | **Check if sign-in is valid** |
| **Tutor** | **Tutor\_ID** |  | **Tutor’ s main identification** |
| **Name** |  | **Tutor’s name** |
| **Phone\_num** |  | **Tutor’s phone number** |
| **Email** |  | **Tutor’s email** |
| **Address** |  | **Tutor’s home address** |
| **DOB** |  | **Tutor’s date of birth** |
|  | **validateSignIn() bool** | **Check if sign-in is valid** |
|  | **getTutorData() tutor** | **Get tutor’s data** |
|  | **updateTutorData(tutor\_ID) void** | **Update the inserted tutor’s information** |
|  | **getListOfTutor() array** | **Return a list of all tutors** |
|  | **searchTutor() array** | **Search for information about the tutor from the text field** |

### 3.5.2 Manage tutor information

#### 3.5.2.1 Sequence diagram

****

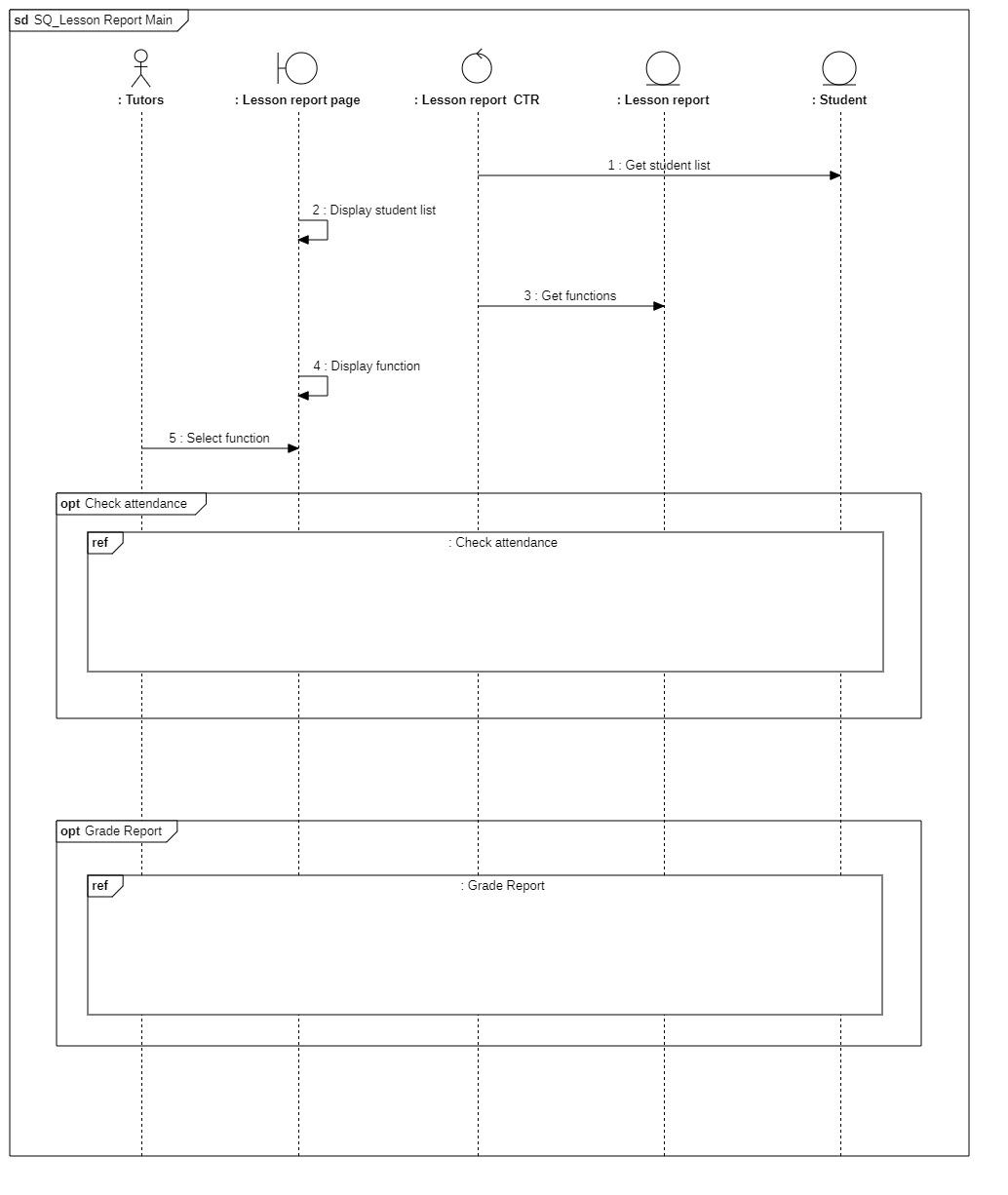
#### 3.5.2.2 Class diagram

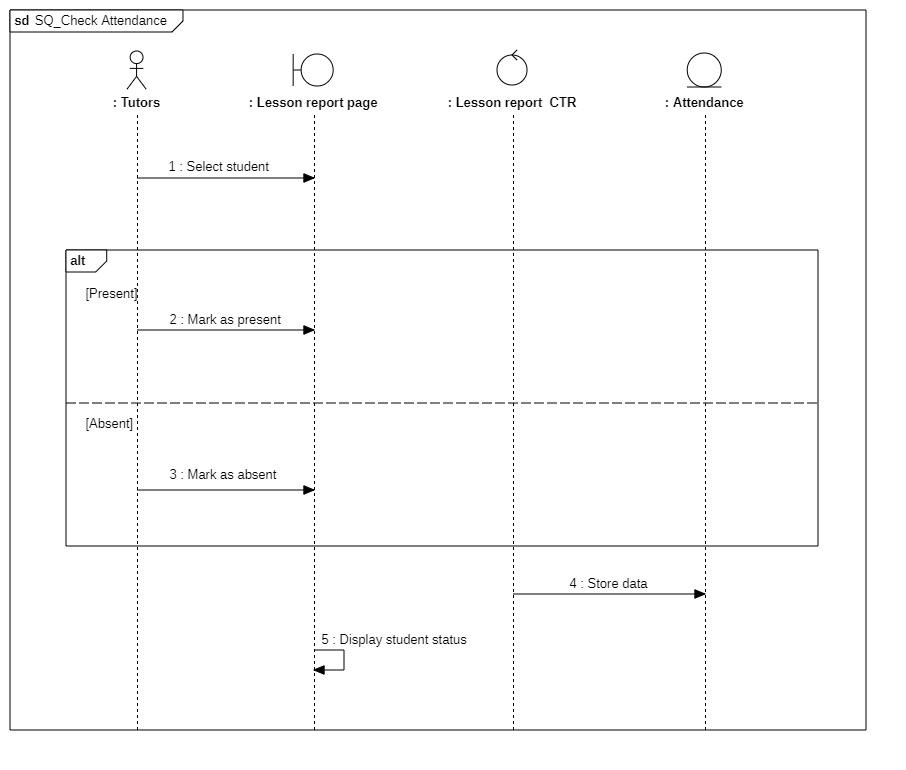
****

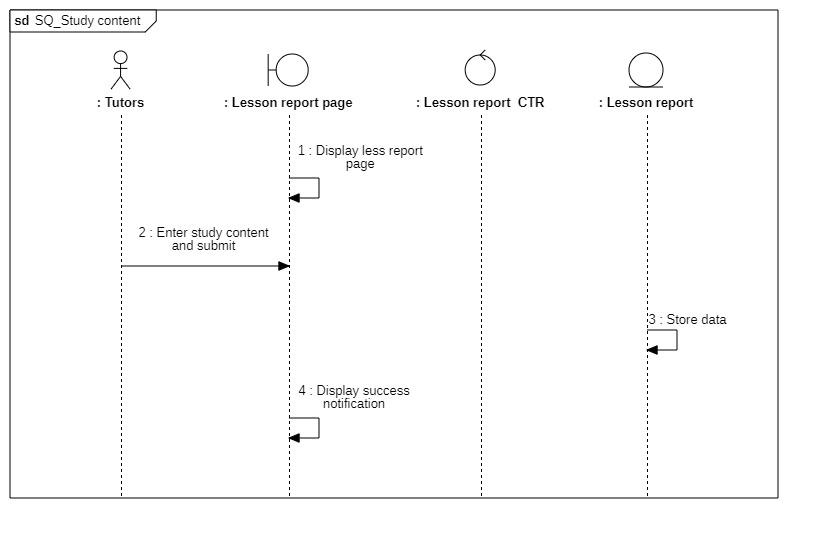
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **functions** | **Meaning** |
| **Personal information CTR** |  | **getTutorProfileData()** | **Get tutor profile** |
|  | **validataInputFormatData()** | **Check input data format is valid** |
|  | **updateProfileData()** |  |
| **Tutor** | **Tutor\_ID** |  | **Tutor’ s main identification** |
| **Name** |  | **Tutor’s name** |
| **Phone\_num** |  | **Tutor’s phone number** |
| **Email** |  | **Tutor’s email** |
| **Address** |  | **Tutor’s home address** |
| **DOB** |  | **Tutor’s date of birth** |
|  | **validateSignIn() bool** | **Check if sign-in is valid** |
|  | **getTutorData() tutor** | **Get tutor’s data** |
|  | **updateTutorData(tutor\_ID) void** | **Update the inserted tutor’s information** |
|  | **getListOfTutor() array** | **Return a list of all tutors** |
|  | **searchTutor() array** | **Search for information about the tutor from the text field** |

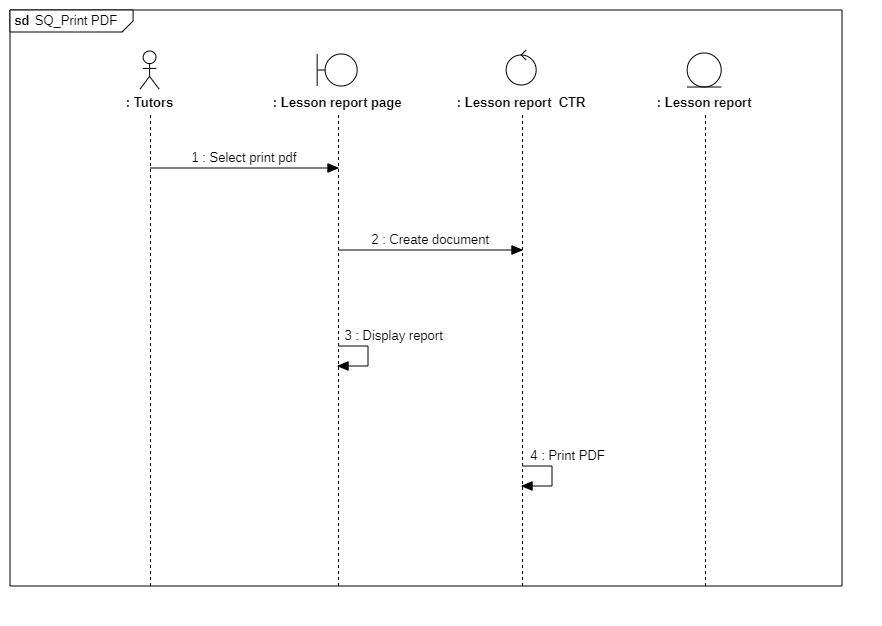
### 3.5.3 Manage lesson reports

#### 3.5.3.1 Sequence diagram

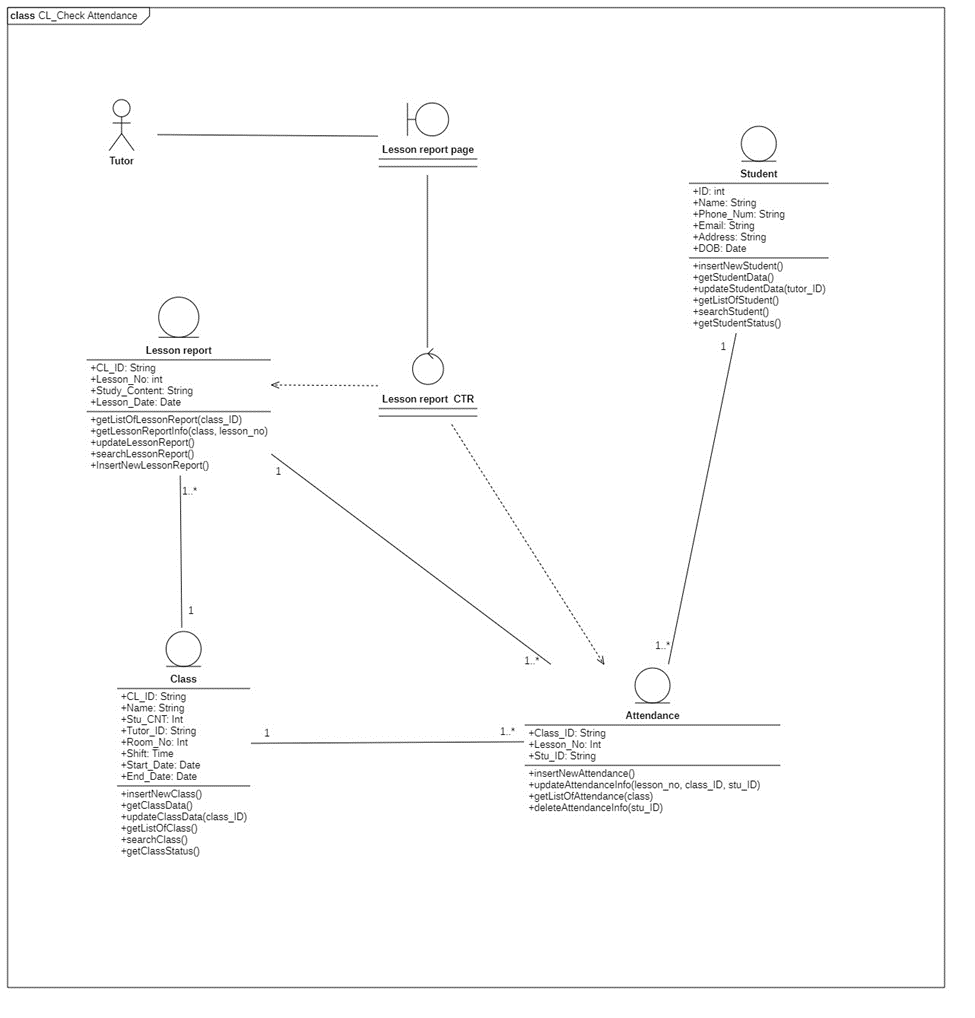
****

****

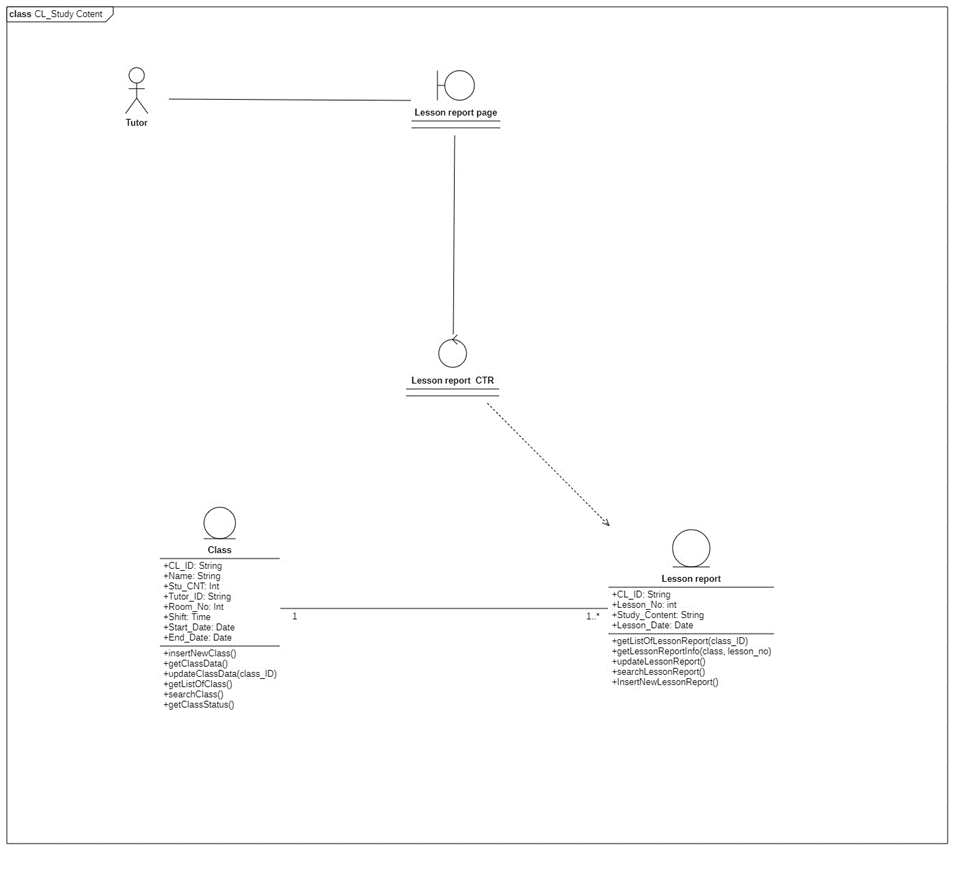
****

****

#### 3.5.3.2 Class diagram

****

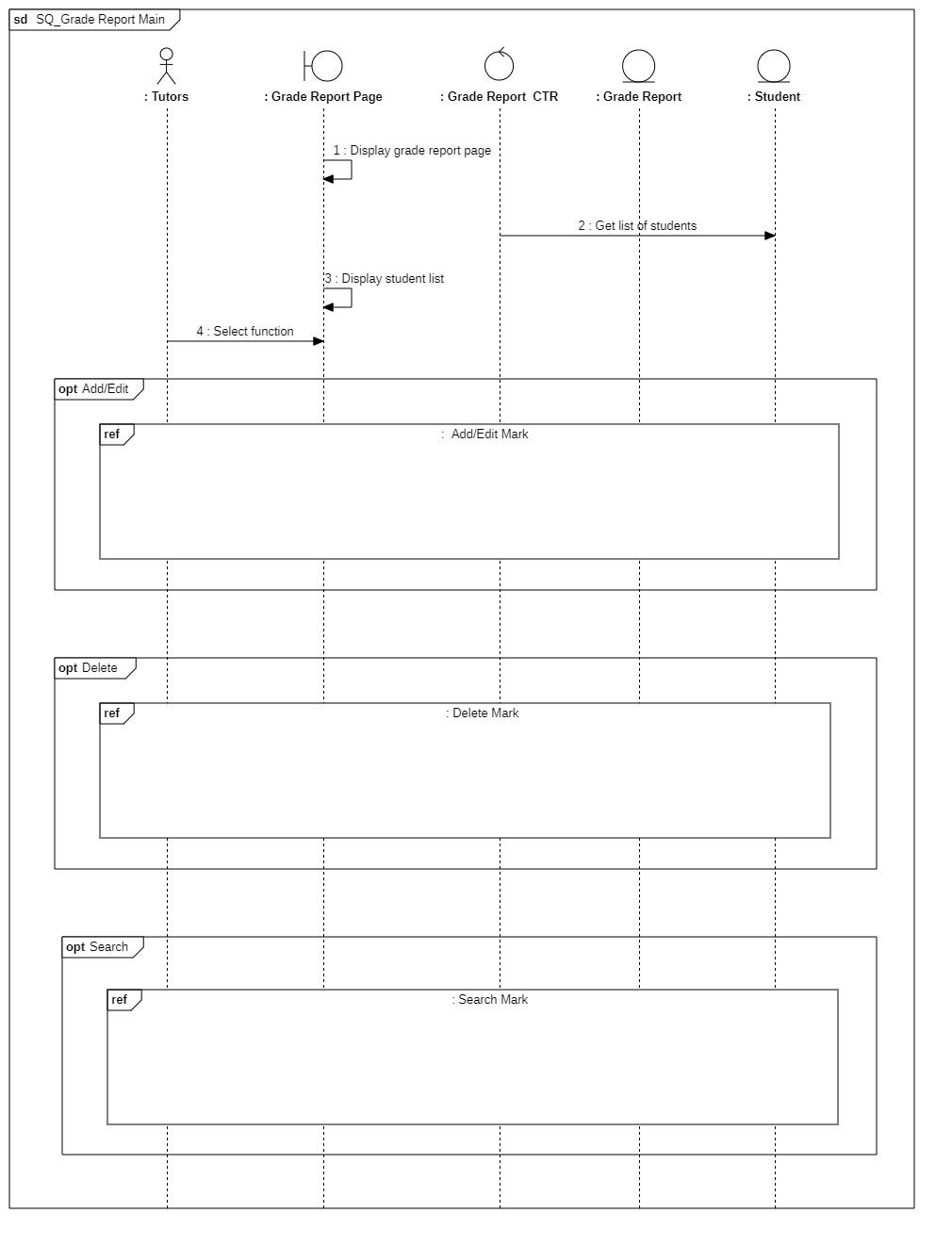
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Lesson Report CTR** |  | **getListOfStudent(classID) array** | **get list of students belonging to class\_ID** |
|  | **getListOfLessonReport (classID) array** | **Get list of lesson report belonging to class\_ID** |
|  | **getListOfAttendance(classID) array** | **Get list of attendance belonging to class\_ID** |
|  | **editAttendance(stu\_ID, classID) void** | **Edit the attendance of** |
|  | **refreshPage() void** | **Reload page and display updated data (if any)** |
|  | **searchStudent(stu\_ID, name) void** | **Search for student based on their name or stu\_ID** |
| **Lesson Report** | **Class\_ID** |  | **Class’s ID** |
| **Lesson\_No** |  | **Lesson’s cardinal number** |
| **Study\_Content** |  | **Report on the content of the lesson** |
| **Lesson\_Date** |  | **The date of the lesson** |
|  | **getListOfLessonReport(class\_ID) array** | **Get list of lesson report based on the class\_ID** |
|  | **getLessonReportInfo(class\_ID, lesson\_no) lessonreport** | **Return the lesson report info based on the lesson\_no and class\_ID** |
|  | **updateLessonReport() void** | **Update the lesson report of the current day** |
|  | **searchLessonReport() array** | **Search for lesson report based on text field** |
|  | **InsertNewLessonReport() lessonreport** | **Insert a new lesson report for the current day** |
| **Attendance** | **Class\_ID** |  | **Class’s main identification** |
| **Lesson\_No** |  | **Lesson cardinal numebr** |
| **Stu\_ID** |  | **Student’ s main identification** |
|  | **insertNewAttendance(stu\_ID) attendance** | **Add new attendance for student for today lesson** |
|  | **getListOfAtendance(class\_ID) array** | **Get a list of attendance belonging to class\_ID** |
|  | **updateAttendanceInfo(lesson\_no, class\_id, student\_id) void** | **Update the attendance of student stu\_ID from class\_ID from lesson\_No** |
|  | **getAttendanceInfo(student) attendance** |  |
|  | **deleteAttendanceInfo(student) void** | **Delete attendance info from student** |
| **Class** | **Class\_ID** |  | **Class’s main identification** |
| **Name** |  | **Class’s name** |
| **Stu\_CNT** |  | **Number of student in class** |
| **Tutor\_ID** |  | **ID of tutor in charge of class** |
| **Room\_No** |  | **Room number** |
| **Time** |  | **Time of study** |
| **Start\_Date** |  | **Start date for class** |
| **End\_Date** |  | **End date for class** |
|  | **insertNewClass() void** | **Create new class** |
|  | **getClassData() class** | **Get class data** |
|  | **updateClassData(class\_ID) void** | **Update class data w for selected class with class\_ID** |
|  | **getListOfClass() array** | **Get list of current classes** |
|  | **searchClass() array** | **Search for a class based on text field** |
|  | **getClassStatus() bool** | **Check if class is still active or not** |
| **Student** | **Stu\_ID** |  | **Student’s main identification** |
| **Name** |  | **Student’s name** |
| **Phone\_Num** |  | **Student’s phone number** |
| **Email** |  | **Student’s email** |
| **Address** |  | **Student’s address** |
| **DOB** |  | **Student’s date of birth** |
|  | **insertNewStudent() void** | **Add new student** |
|  | **getStudentData() student** | **Get student data from text field** |
|  | **updateStudentData(stu\_ID) void** | **Get student data** |
|  | **getListOfStudent() array** | **Get list of student from a class from text field** |
|  | **searchStudent() array** | **Search for student from name or ID from text field** |
|  | **getStudentStatus() bool** | **Check if Student is active or not** |

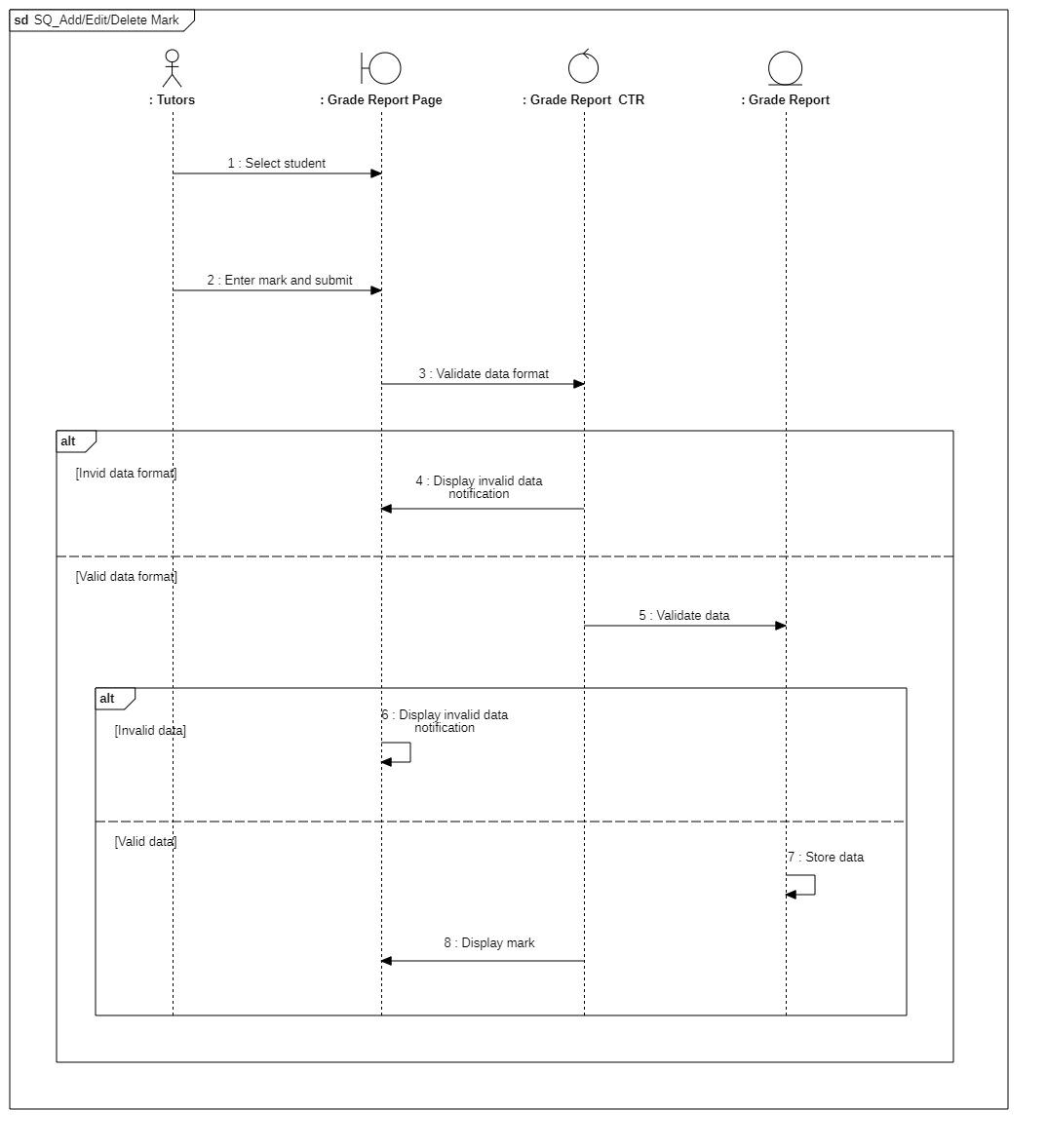
****

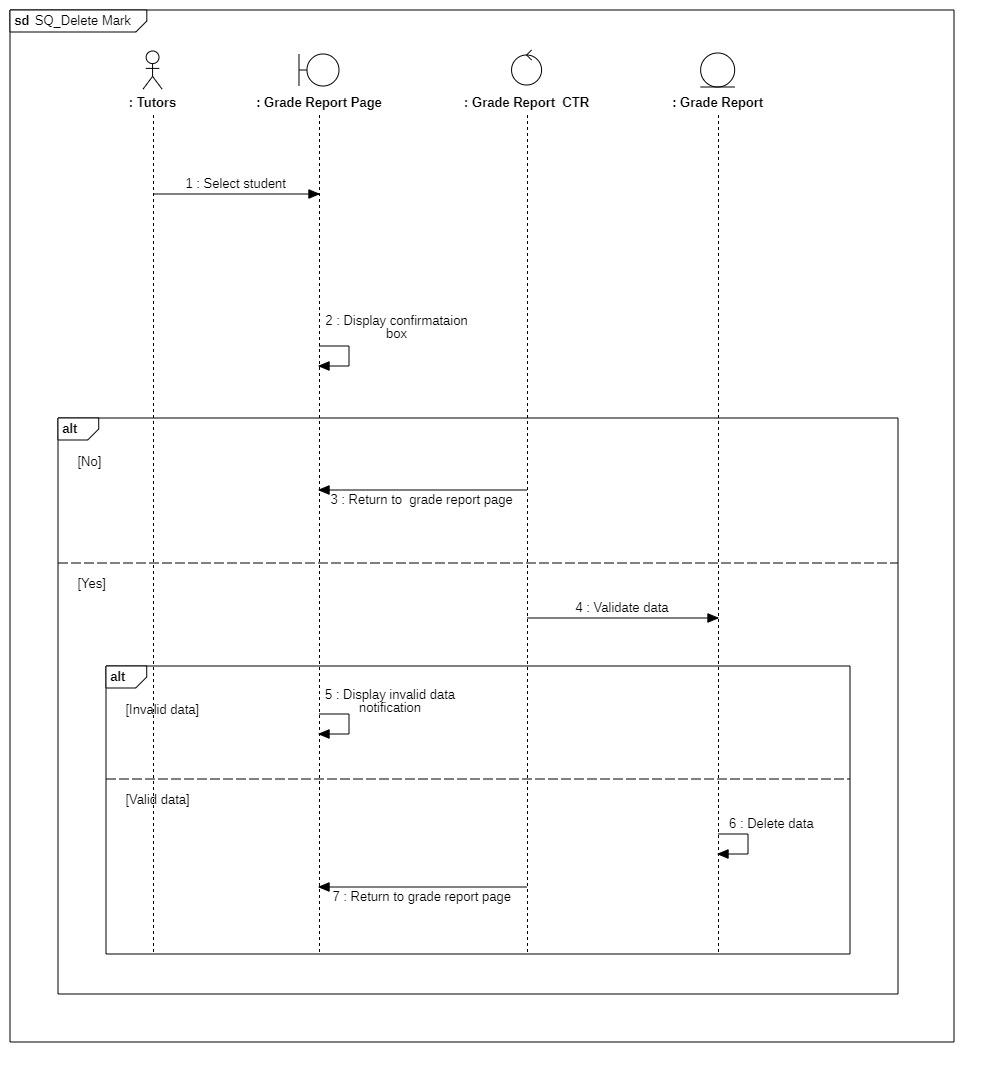
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Lesson Report CTR** |  | **getListOfStudent(classID) array** | **get list of students belonging to class\_ID** |
|  | **getListOfLessonReport (classID) array** | **Get list of lesson report belonging to class\_ID** |
|  | **getListOfAttendance(classID) array** | **Get list of attendance belonging to class\_ID** |
|  | **editAttendance(stu\_ID, classID) void** | **Edit the attendance of** |
|  | **refreshPage() void** | **Reload page and display updated data (if any)** |
|  | **searchStudent(stu\_ID, name) void** | **Search for student based on their name or stu\_ID** |
| **Lesson Report** | **Class\_ID** |  | **Class’s ID** |
| **Lesson\_No** |  | **Lesson’s cardinal number** |
| **Study\_Content** |  | **Report on the content of the lesson** |
| **Lesson\_Date** |  | **The date of the lesson** |
|  | **getListOfLessonReport(class\_ID) array** | **Get list of lesson report based on the class\_ID** |
|  | **getLessonReportInfo(class\_ID, lesson\_no) lessonreport** | **Return the lesson report info based on the lesson\_no and class\_ID** |
|  | **updateLessonReport() void** | **Update the lesson report of the current day** |
|  | **searchLessonReport() array** | **Search for lesson report based on text field** |
|  | **InsertNewLessonReport() lessonreport** | **Insert a new lesson report for the current day** |
| **Class** | **Class\_ID** |  | **Class’s main identification** |
| **Name** |  | **Class’s name** |
| **Stu\_CNT** |  | **Number of student in class** |
| **Tutor\_ID** |  | **ID of tutor in charge of class** |
| **Room\_No** |  | **Room number** |
| **Time** |  | **Time of study** |
| **Start\_Date** |  | **Start date for class** |
| **End\_Date** |  | **End date for class** |
|  | **insertNewClass() void** | **Create new class** |
|  | **getClassData() class** | **Get class data** |
|  | **updateClassData(class\_ID) void** | **Update class data w for selected class with class\_ID** |
|  | **getListOfClass() array** | **Get list of current classes** |
|  | **searchClass() array** | **Search for a class based on text field** |
|  | **getClassStatus() bool** | **Check if class is still active or not** |

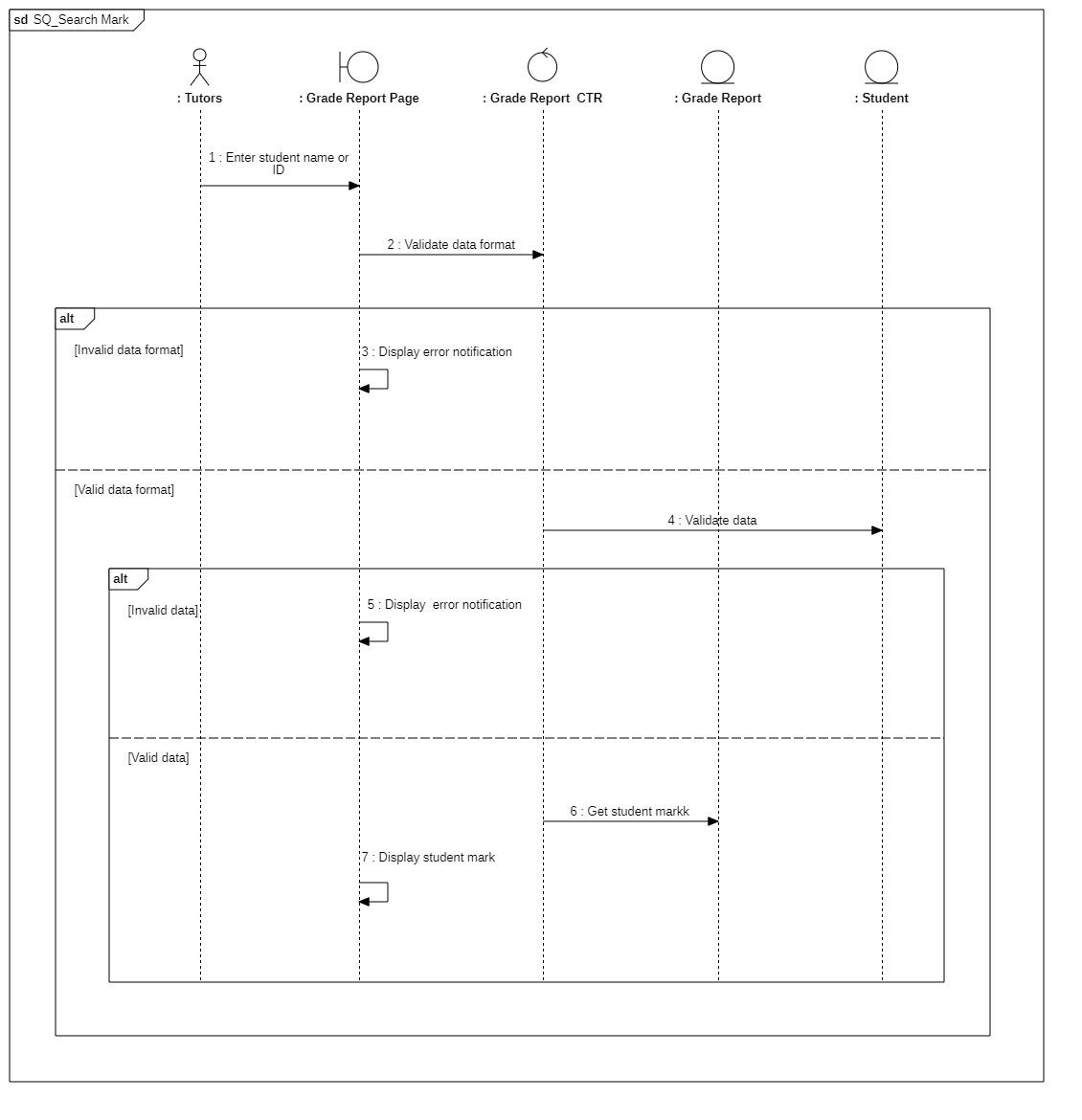
### 3.5.4 Manage grade report

#### 3.5.4.1 Sequence diagram

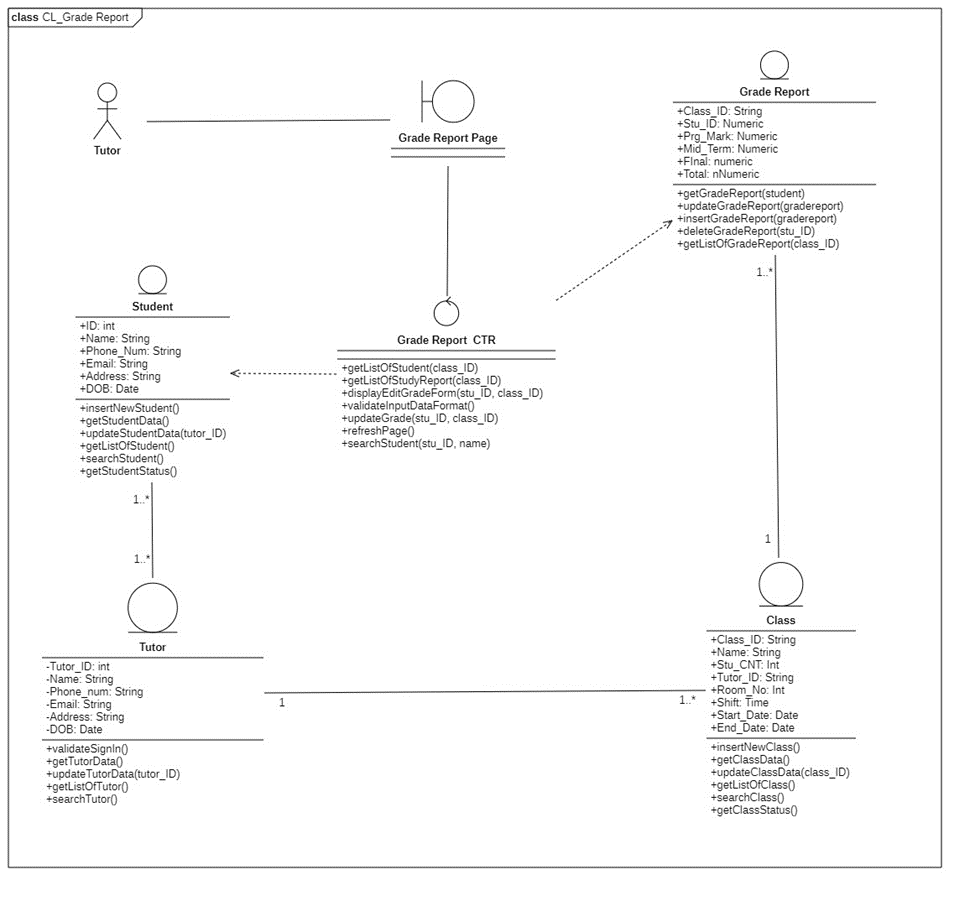
****

****

****

****

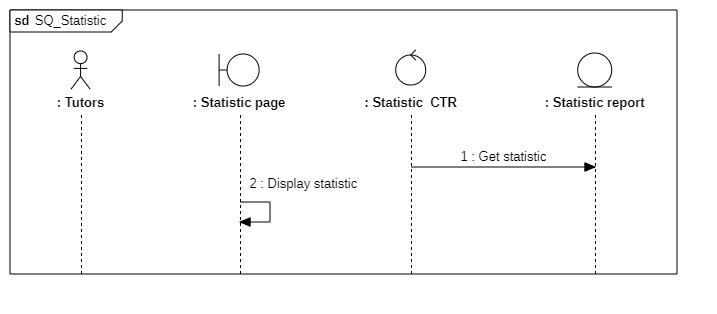
#### 3.5.4.2 Class diagram

****

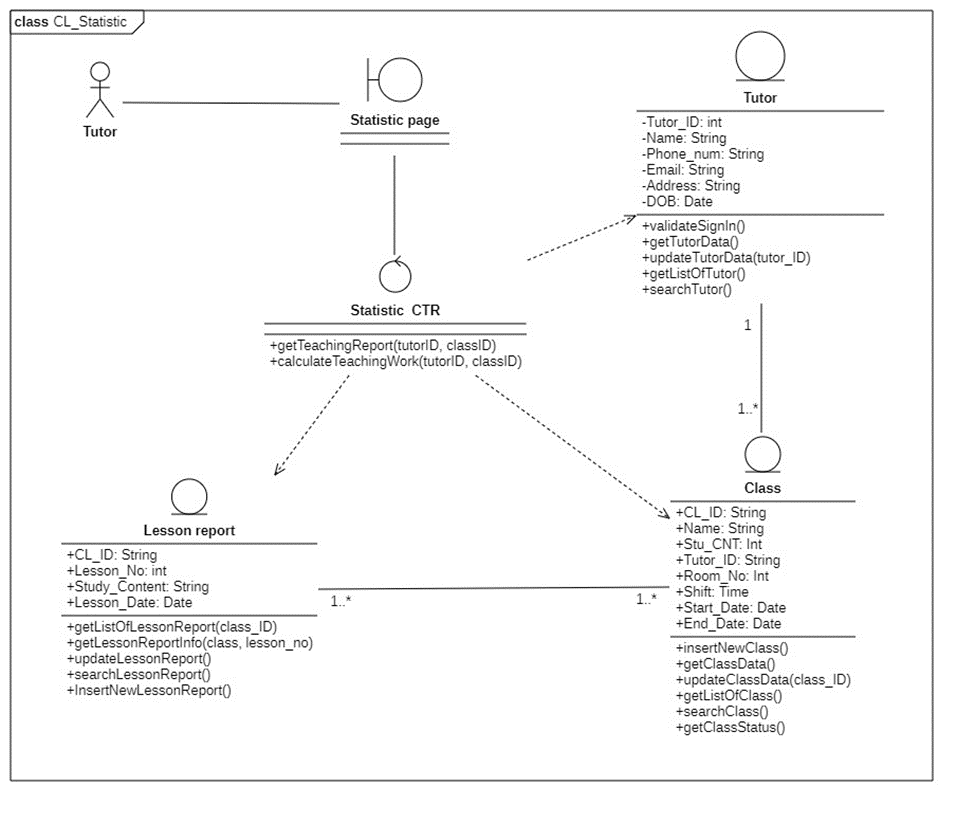
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Grade Report CTR** |  | **getListOfStudent(class\_ID) array** | **get list of students belonging to class\_ID** |
|  | **getListOfStudyReport(class\_ID) array** | **Get list of study report belonging to class\_ID** |
|  | **displayEditGradeForm(studentID, classID)** | **Display edit grade form based on selected student ID and class ID** |
|  | **validateInputDataFormat()** | **Check if input data is valid** |
|  | **updateGrade(studentID, classID) void** | **Update the grade of the selected student in class** |
|  | **refreshPage() void** | **Reload page and display updated data (if any)** |
|  | **searchStudent(ID, name) void** | **Search for student based on name or class** |
| **Grade Report** | **Class\_ID** |  | **Class’s main identification** |
| **Stu\_ID** |  | **Student’s main identification** |
| **PRG\_Mark** |  | **Progress mark** |
| **Midterm** |  | **Midterm mark** |
| **Final** |  | **Final exam mark** |
| **Total** |  | **Total mark** |
|  | **getStudyReport(student)** |  |
|  | **updateStudyReport(studyreport) void** |  |
|  | **insertStudyReport(studyreport) void** |  |
|  | **deleteStudyReport(ID) void** |  |
|  | **getListOfStudyReport(class\_id) void** |  |
| **Student** | **Stu\_ID** |  | **Student’s main identification** |
| **Name** |  | **Student’s name** |
| **Phone\_Num** |  | **Student’s phone number** |
| **Email** |  | **Student’s email** |
| **Address** |  | **Student’s address** |
| **DOB** |  | **Student’s date of birth** |
|  | **insertNewStudent() void** | **Add new student** |
|  | **getStudentData() student** | **Get student data from text field** |
|  | **updateStudentData(stu\_ID) void** | **Get student data** |
|  | **getListOfStudent() array** | **Get list of student from a class from text field** |
|  | **searchStudent() array** | **Search for student from name or ID from text field** |
|  | **getStudentStatus() bool** | **Check if Student is active or not** |
| **Class** | **Class\_ID** |  | **Class’s main identification** |
| **Name** |  | **Class’s name** |
| **Stu\_CNT** |  | **Number of student in class** |
| **Tutor\_ID** |  | **ID of tutor in charge of class** |
| **Room\_No** |  | **Room number** |
| **Time** |  | **Time of study** |
| **Start\_Date** |  | **Start date for class** |
| **End\_Date** |  | **End date for class** |
|  | **insertNewClass() void** | **Create new class** |
|  | **getClassData() class** | **Get class data** |
|  | **updateClassData(class\_ID) void** | **Update class data w for selected class with class\_ID** |
|  | **getListOfClass() array** | **Get list of current classes** |
|  | **searchClass() array** | **Search for a class based on text field** |
|  | **getClassStatus() bool** | **Check if class is still active or not** |
| **Tutor** | **Tutor\_ID** |  | **Tutor’ s main identification** |
| **Name** |  | **Tutor’s name** |
| **Phone\_num** |  | **Tutor’s phone number** |
| **Email** |  | **Tutor’s email** |
| **Address** |  | **Tutor’s home address** |
| **DOB** |  | **Tutor’s date of birth** |
|  | **validateSignIn() bool** | **Check if sign-in is valid** |
|  | **getTutorData() tutor** | **Get tutor’s data** |
|  | **updateTutorData(tutor\_ID) void** | **Update the inserted tutor’s information** |
|  | **getListOfTutor() array** | **Return a list of all tutors** |
|  | **searchTutor() array** | **Search for information about the tutor from the text field** |

### 3.5.5 Statistic report

#### 3.5.5.1 Sequence diagram

****

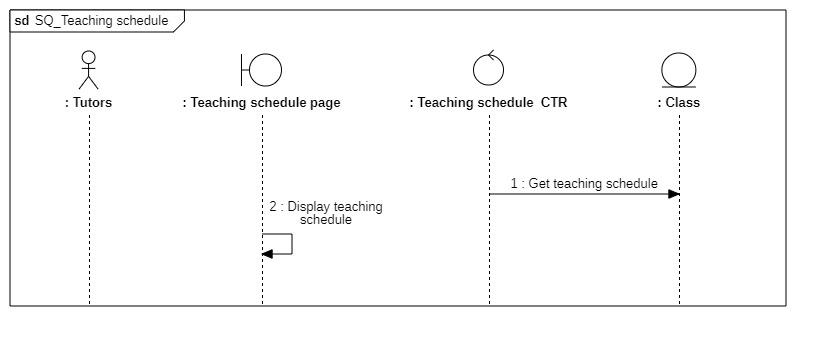
#### 3.5.5.2 Class diagram

****

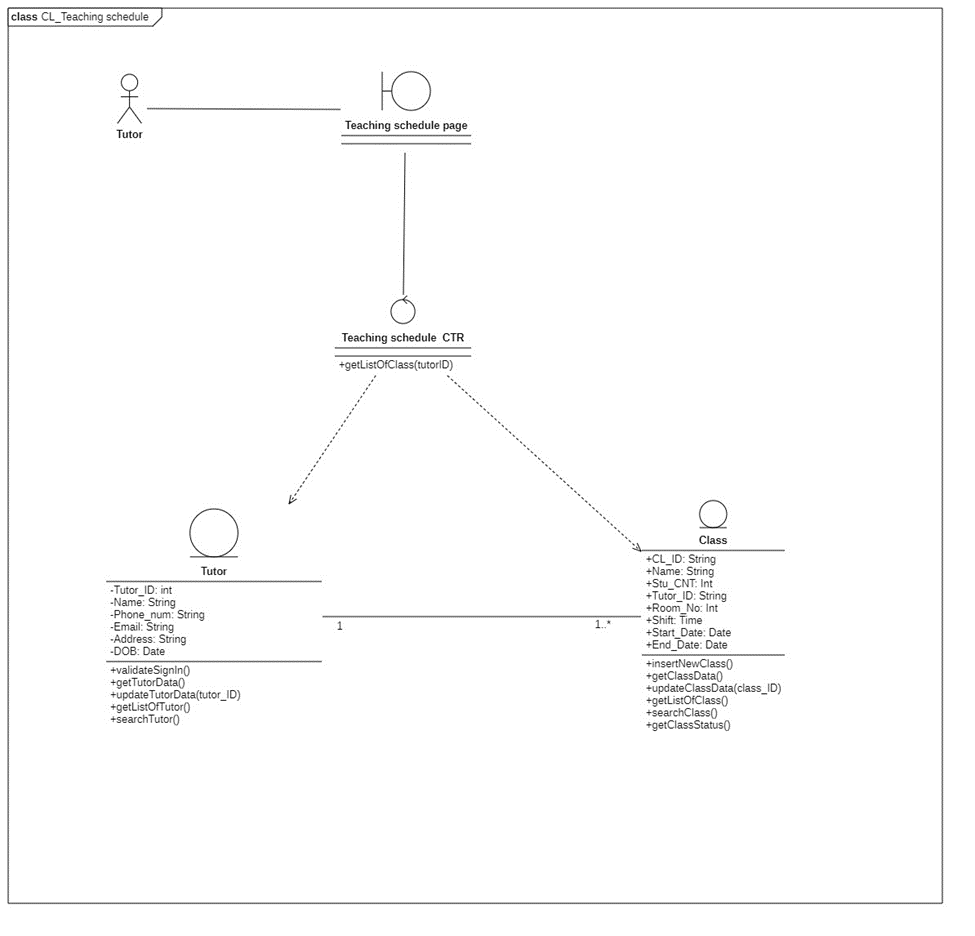
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Statistic** |  | **getTeachingReport(tutorID, classID)** | **get teaching report for tutor based on classes taught** |
|  |  | **calculateTeachingWork(tutorID, classID)** | **Calculate the number of lesson for each class tutor taught** |
| **Lesson Report** | **Class\_ID** |  | **Class’s ID** |
| **Lesson\_No** |  | **Lesson’s cardinal number** |
| **Study\_Content** |  | **Report on the content of the lesson** |
| **Lesson\_Date** |  | **The date of the lesson** |
|  | **getListOfLessonReport(class\_ID) array** | **Get list of lesson report based on the class\_ID** |
|  | **getLessonReportInfo(class\_ID, lesson\_no) lessonreport** | **Return the lesson report info based on the lesson\_no and class\_ID** |
|  | **updateLessonReport() void** | **Update the lesson report of the current day** |
|  | **searchLessonReport() array** | **Search for lesson report based on text field** |
|  | **InsertNewLessonReport() lessonreport** | **Insert a new lesson report for the current day** |
| **Class** | **Class\_ID** |  | **Class’s main identification** |
| **Name** |  | **Class’s name** |
| **Stu\_CNT** |  | **Number of student in class** |
| **Tutor\_ID** |  | **ID of tutor in charge of class** |
| **Room\_No** |  | **Room number** |
| **Time** |  | **Time of study** |
| **Start\_Date** |  | **Start date for class** |
| **End\_Date** |  | **End date for class** |
|  | **insertNewClass() void** | **Create new class** |
|  | **getClassData() class** | **Get class data** |
|  | **updateClassData(class\_ID) void** | **Update class data w for selected class with class\_ID** |
|  | **getListOfClass() array** | **Get list of current classes** |
|  | **searchClass() array** | **Search for a class based on text field** |
|  | **getClassStatus() bool** | **Check if class is still active or not** |
| **Tutor** | **Tutor\_ID** |  | **Tutor’ s main identification** |
| **Name** |  | **Tutor’s name** |
| **Phone\_num** |  | **Tutor’s phone number** |
| **Email** |  | **Tutor’s email** |
| **Address** |  | **Tutor’s home address** |
| **DOB** |  | **Tutor’s date of birth** |
|  | **validateSignIn() bool** | **Check if sign-in is valid** |
|  | **getTutorData() tutor** | **Get tutor’s data** |
|  | **updateTutorData(tutor\_ID) void** | **Update the inserted tutor’s information** |
|  | **getListOfTutor() array** | **Return a list of all tutors** |
|  | **searchTutor() array** | **Search for information about the tutor from the text field** |

### 3.5.6 Tutor’s schedule

#### 3.5.6.1 Sequence diagram

****

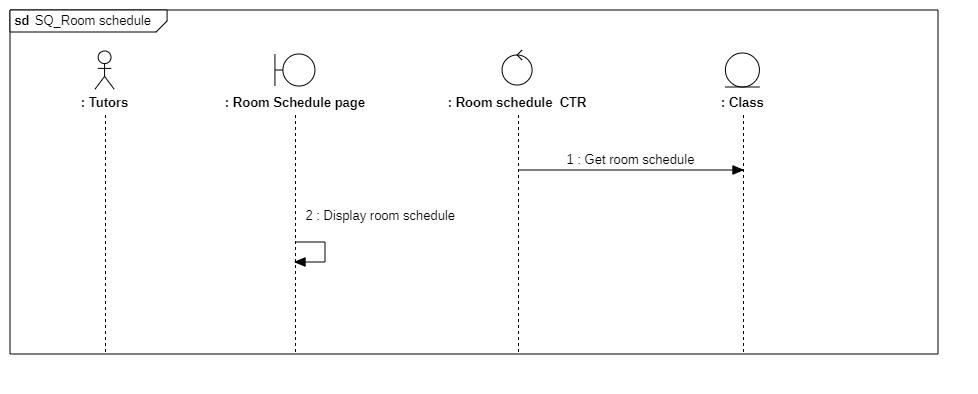
#### 3.5.6.2 Class diagram

****

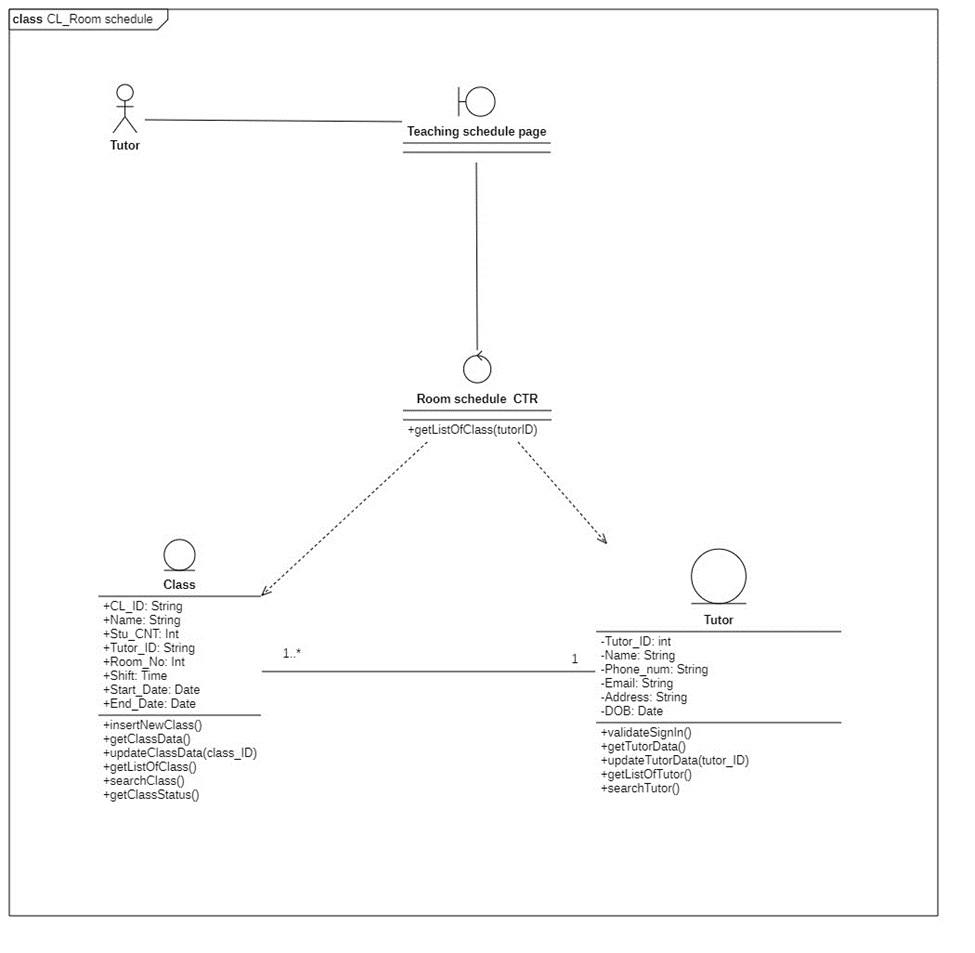
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Teaching schedule CTR** |  | **getListOfClass(tutorID)** | **get list of class tutor have to teach** |
| **Class** | **Class\_ID** |  | **Class’s main identification** |
| **Name** |  | **Class’s name** |
| **Stu\_CNT** |  | **Number of student in class** |
| **Tutor\_ID** |  | **ID of tutor in charge of class** |
| **Room\_No** |  | **Room number** |
| **Time** |  | **Time of study** |
| **Start\_Date** |  | **Start date for class** |
| **End\_Date** |  | **End date for class** |
|  | **insertNewClass() void** | **Create new class** |
|  | **getClassData() class** | **Get class data** |
|  | **updateClassData(class\_ID) void** | **Update class data w for selected class with class\_ID** |
|  | **getListOfClass() array** | **Get list of current classes** |
|  | **searchClass() array** | **Search for a class based on text field** |
|  | **getClassStatus() bool** | **Check if class is still active or not** |
| **Tutor** | **Tutor\_ID** |  | **Tutor’ s main identification** |
| **Name** |  | **Tutor’s name** |
| **Phone\_num** |  | **Tutor’s phone number** |
| **Email** |  | **Tutor’s email** |
| **Address** |  | **Tutor’s home address** |
| **DOB** |  | **Tutor’s date of birth** |
|  | **validateSignIn() bool** | **Check if sign-in is valid** |
|  | **getTutorData() tutor** | **Get tutor’s data** |
|  | **updateTutorData(tutor\_ID) void** | **Update the inserted tutor’s information** |
|  | **getListOfTutor() array** | **Return a list of all tutors** |
|  | **searchTutor() array** | **Search for information about the tutor from the text field** |

### 3.5.7 Room schedule

#### 3.5.7.1 Sequence diagram

****

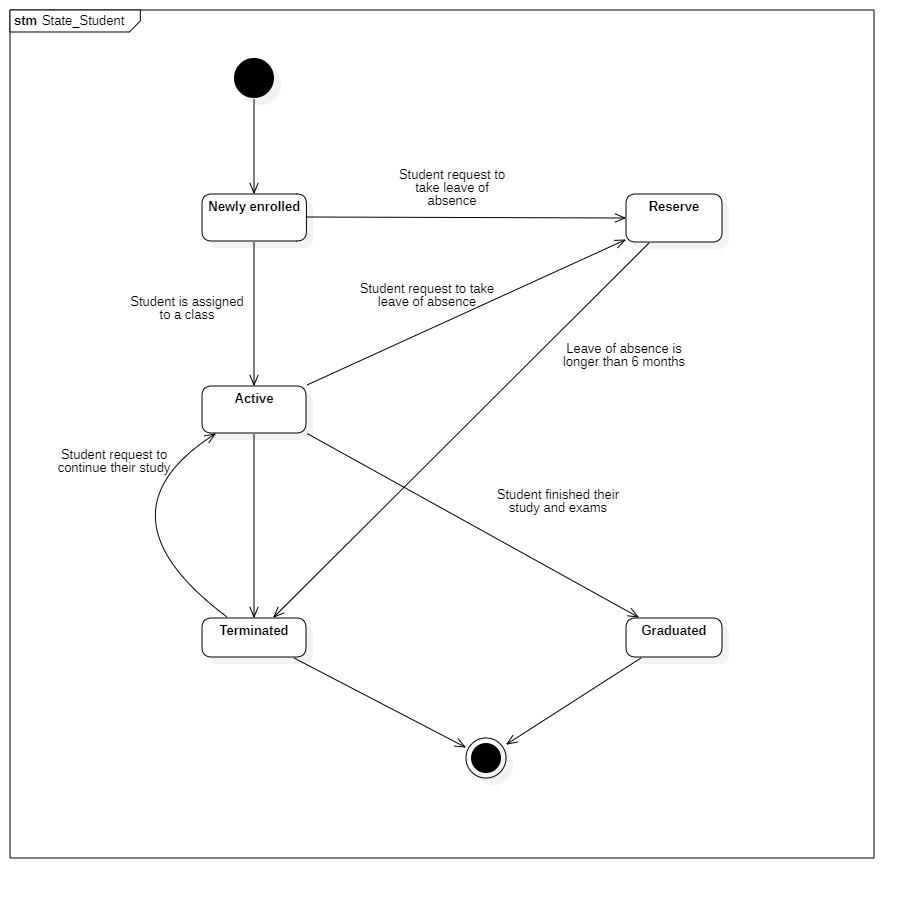
#### 3.5.7.2 Class diagram

****

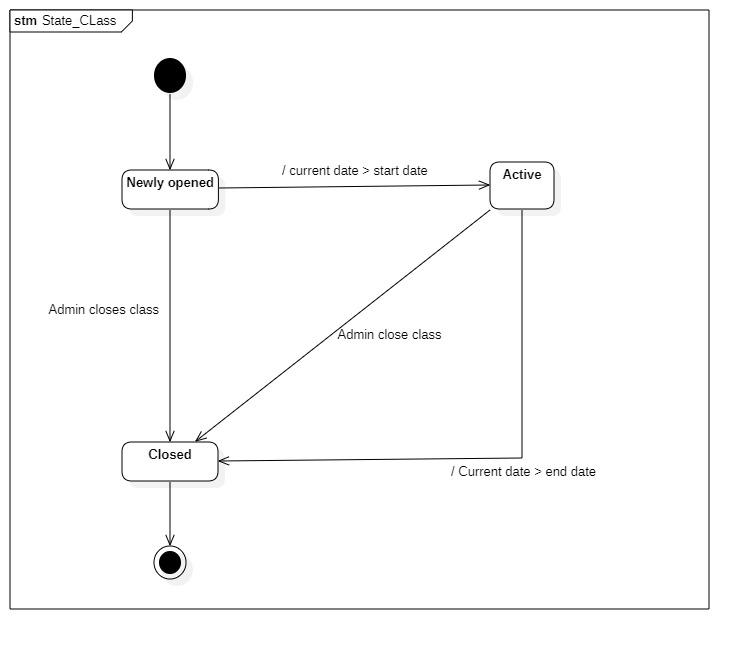
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Attributes** | **Functions** | **Meaning** |
| **Teaching schedule CTR** |  | **getListOfClass(tutorID)** | **get list of class tutor have to teach** |
| **Class** | **Class\_ID** |  | **Class’s main identification** |
| **Name** |  | **Class’s name** |
| **Stu\_CNT** |  | **Number of student in class** |
| **Tutor\_ID** |  | **ID of tutor in charge of class** |
| **Room\_No** |  | **Room number** |
| **Time** |  | **Time of study** |
| **Start\_Date** |  | **Start date for class** |
| **End\_Date** |  | **End date for class** |
|  | **insertNewClass() void** | **Create new class** |
|  | **getClassData() class** | **Get class data** |
|  | **updateClassData(class\_ID) void** | **Update class data w for selected class with class\_ID** |
|  | **getListOfClass() array** | **Get list of current classes** |
|  | **searchClass() array** | **Search for a class based on text field** |
|  | **getClassStatus() bool** | **Check if class is still active or not** |
| **Tutor** | **Tutor\_ID** |  | **Tutor’ s main identification** |
| **Name** |  | **Tutor’s name** |
| **Phone\_num** |  | **Tutor’s phone number** |
| **Email** |  | **Tutor’s email** |
| **Address** |  | **Tutor’s home address** |
| **DOB** |  | **Tutor’s date of birth** |
|  | **validateSignIn() bool** | **Check if sign-in is valid** |
|  | **getTutorData() tutor** | **Get tutor’s data** |
|  | **updateTutorData(tutor\_ID) void** | **Update the inserted tutor’s information** |
|  | **getListOfTutor() array** | **Return a list of all tutors** |
|  | **searchTutor() array** | **Search for information about the tutor from the text field** |

## 3.6 State diagram

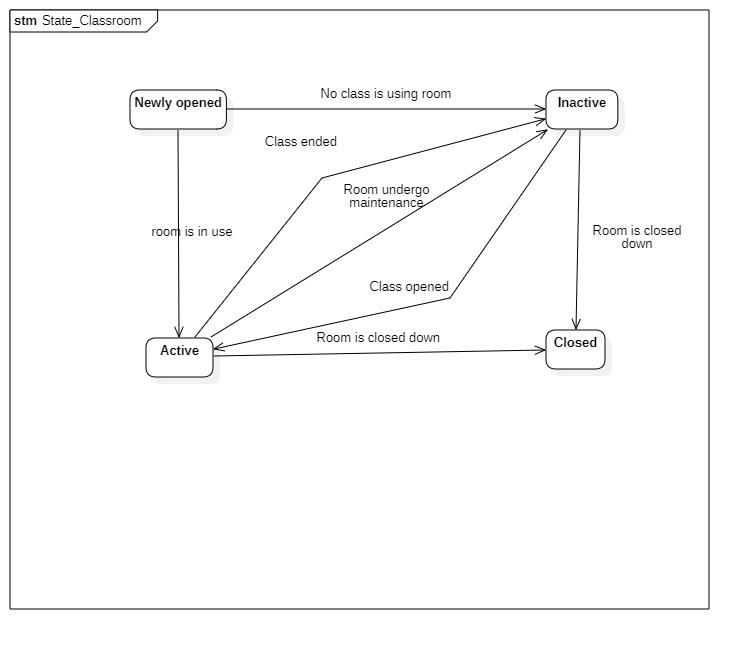
### 3.6.1 Student

****

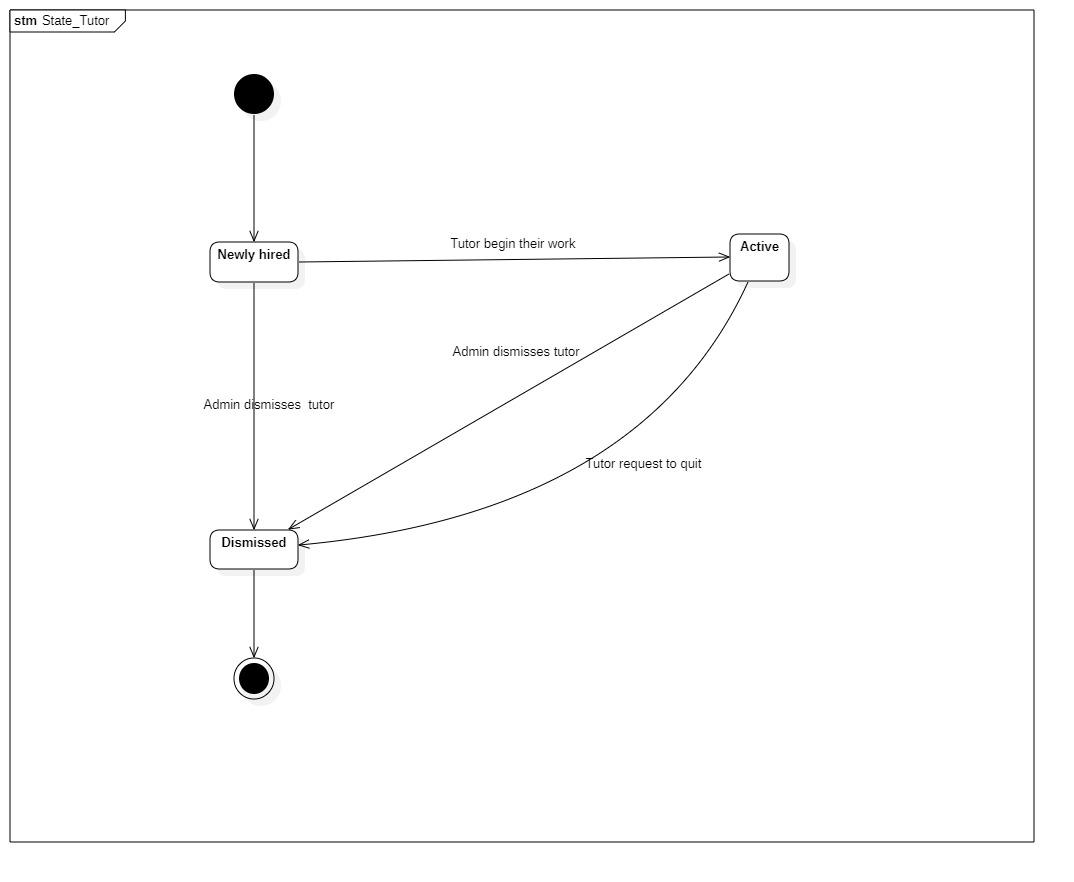
### 3.6.2 Class

****

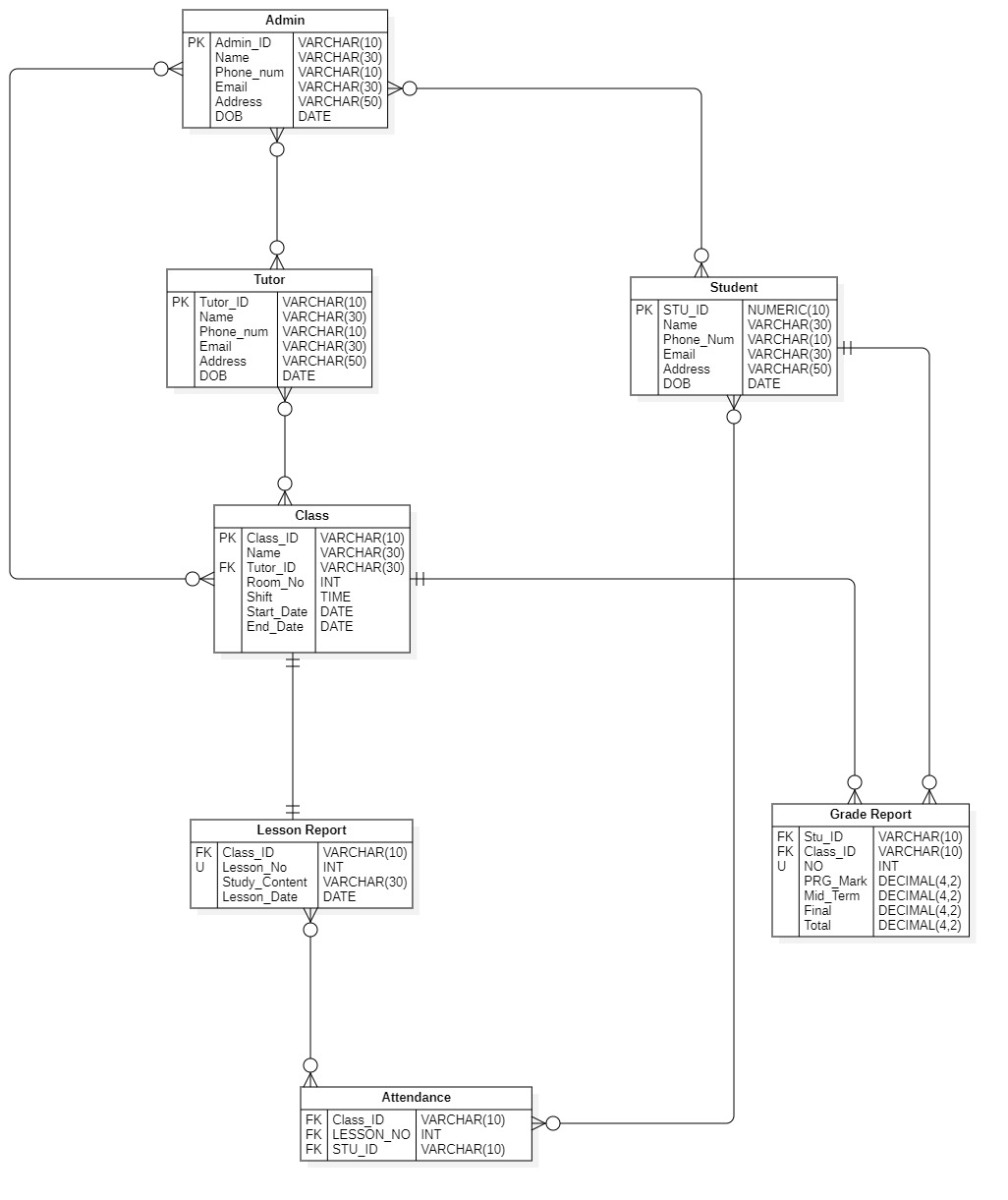
### 3.6.3 Classroom

****

### 3.6.5 Tutor

****

## 3.7 Entity relational diagram



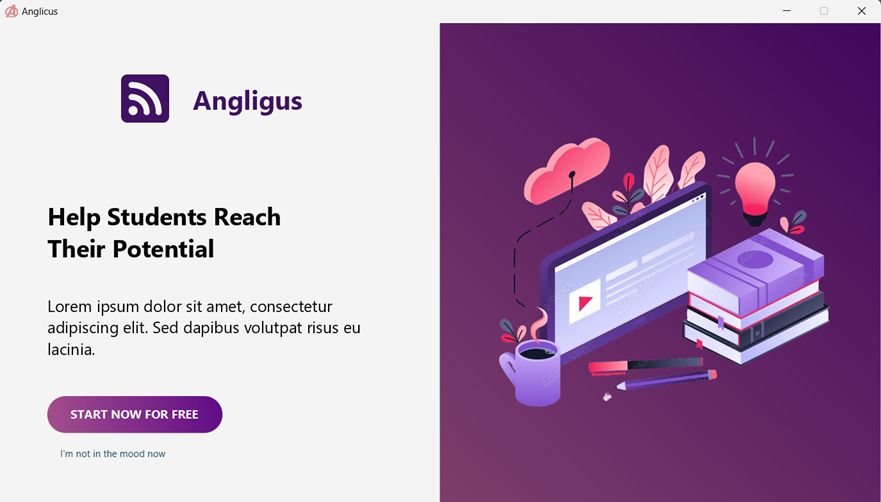
# Chapter 4: Implementation and testing

## 4.1 Introduction to the implementation environment.

* Operating system: Windows 11.
* Database: MySQL Server 8.0.33.
* Code Editor: IntelliJ IDEA.
* Language: Java, Javafx.
* Dependencies package: mysql-connector-j 8.0.33, JavafxLibrary.

## 4.2 User interface.

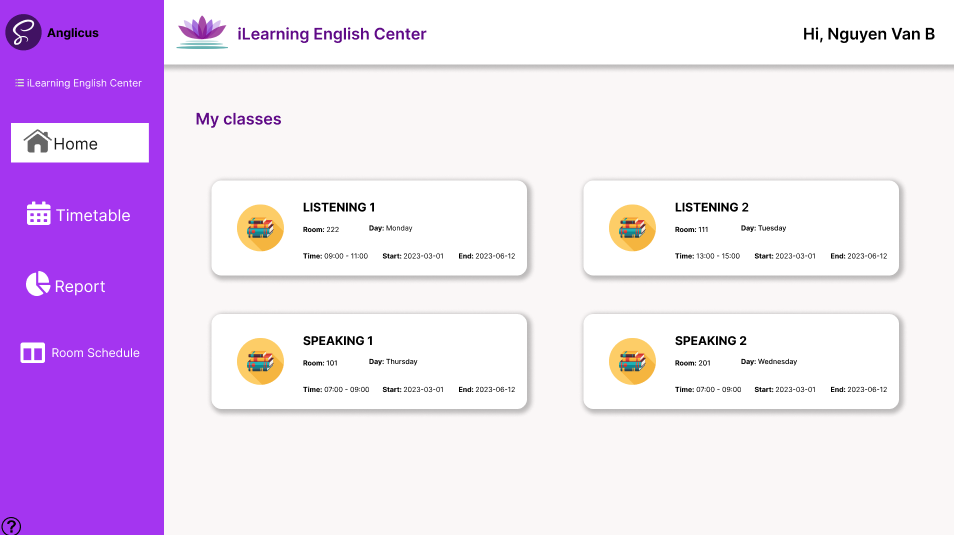
### 4.2.1 Tutor’s interface.

****

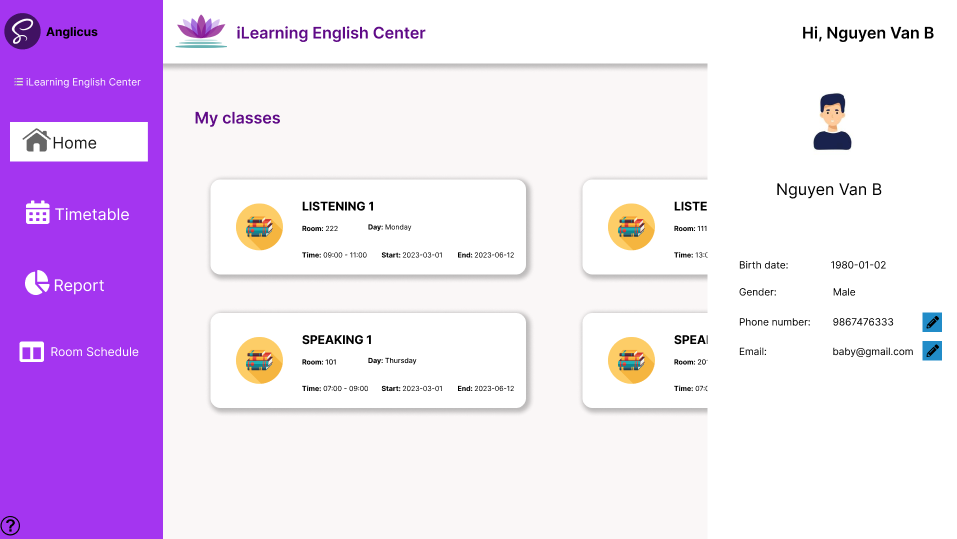
Picture 1. Welcome page.



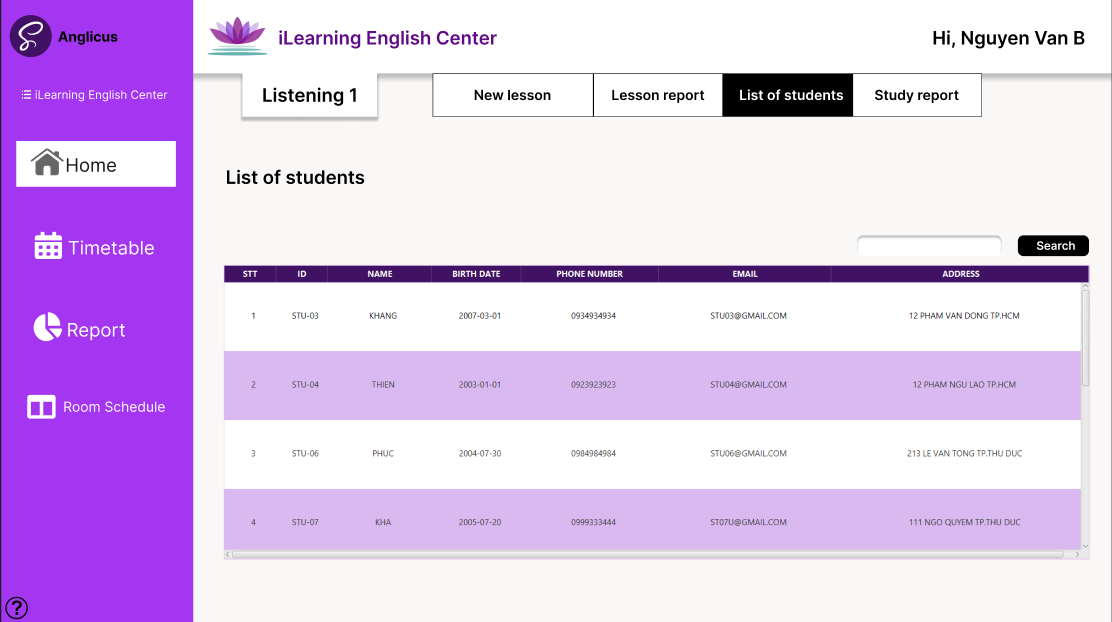
Picture 2. Login page.



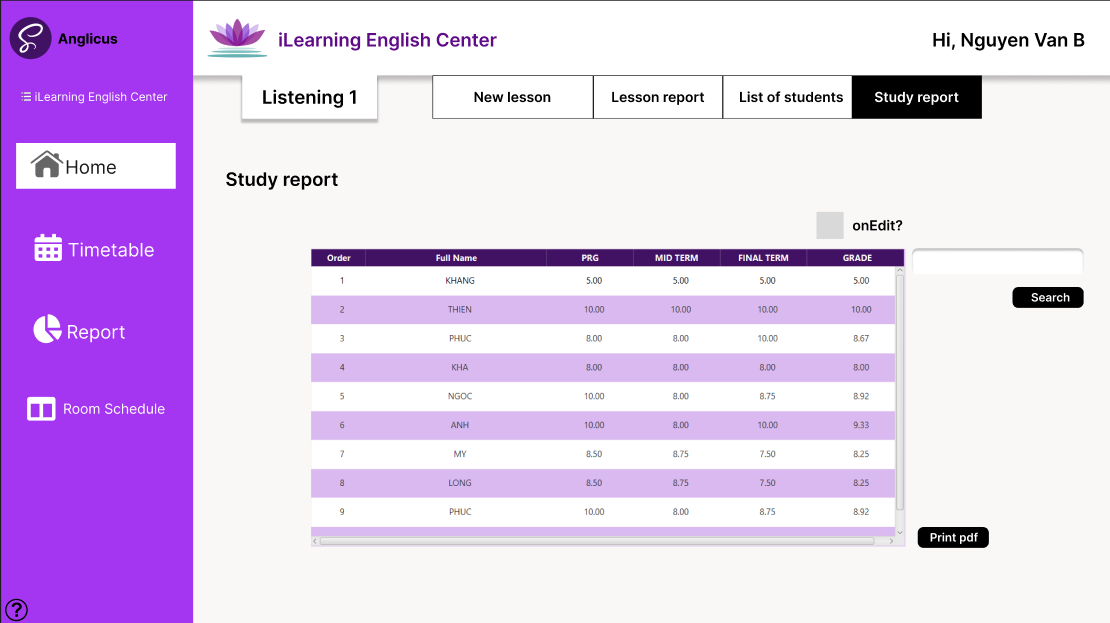
Picture 3.1. Homepage



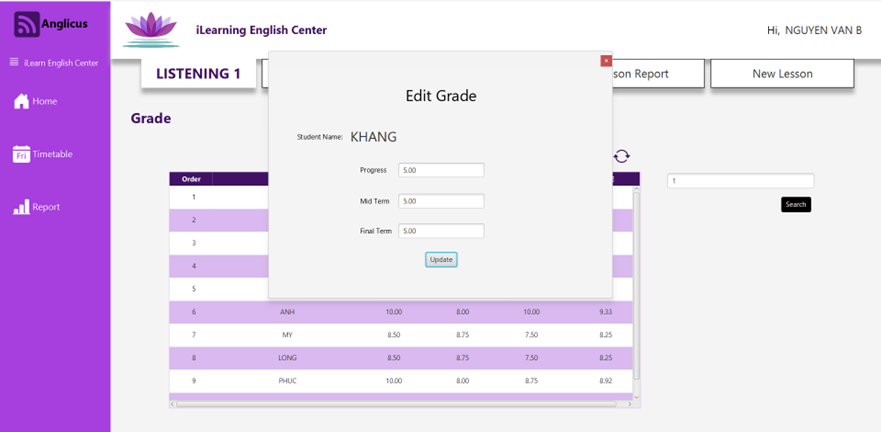
Picture 3.2. Homepage.



Picture 4. Showing list of students in chosen course.



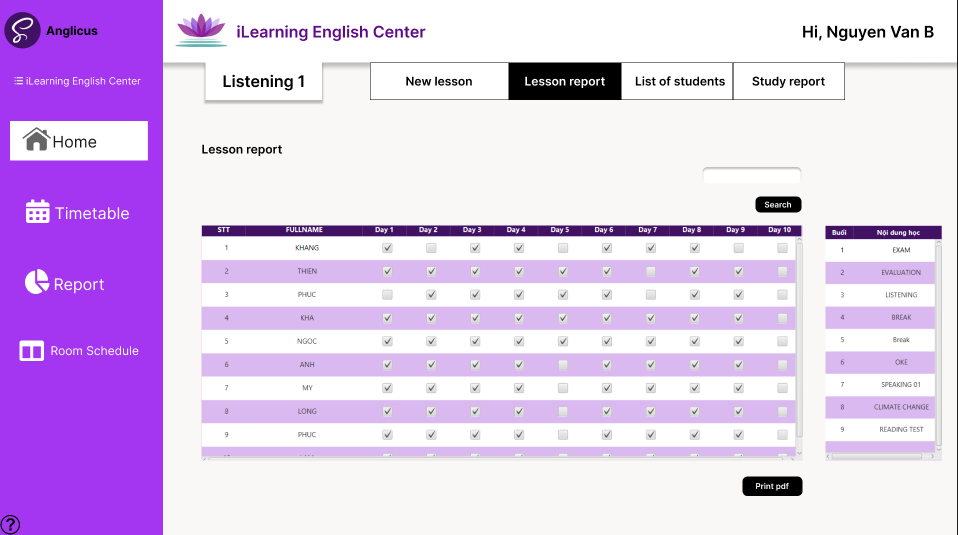
Picture 5. Showing students’ grade.



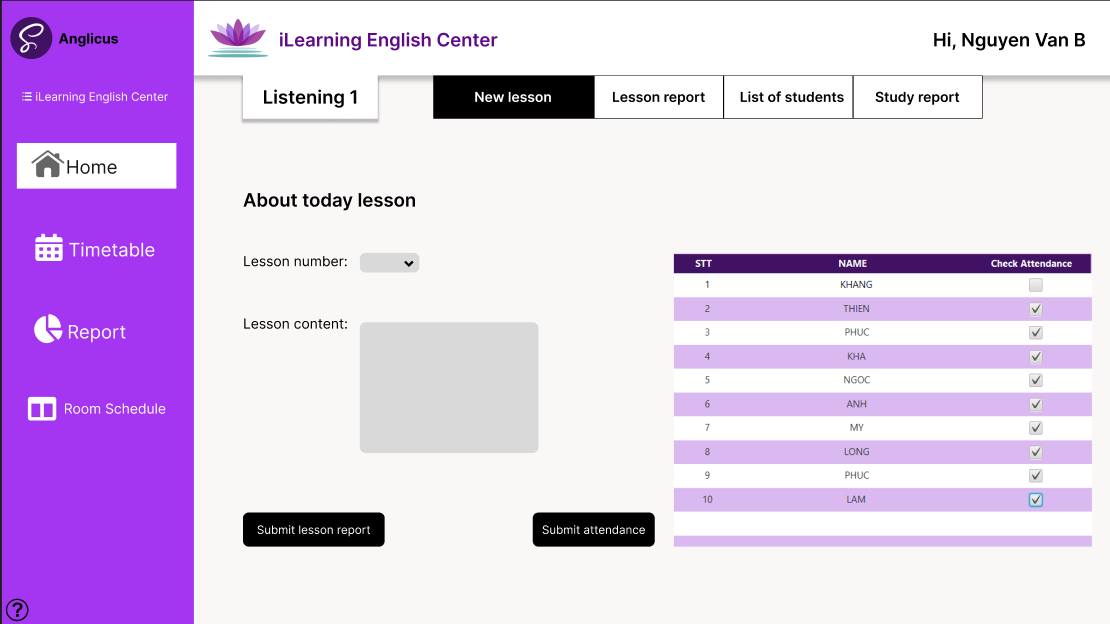
Picture 6. Editing grade for student (Study report page).



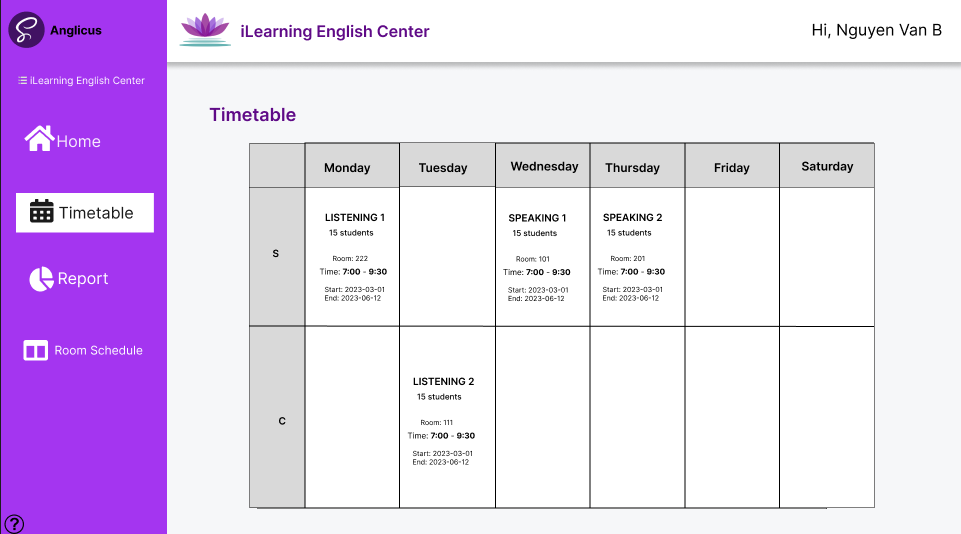
Picture 7. Showing students’ grade (after editing).



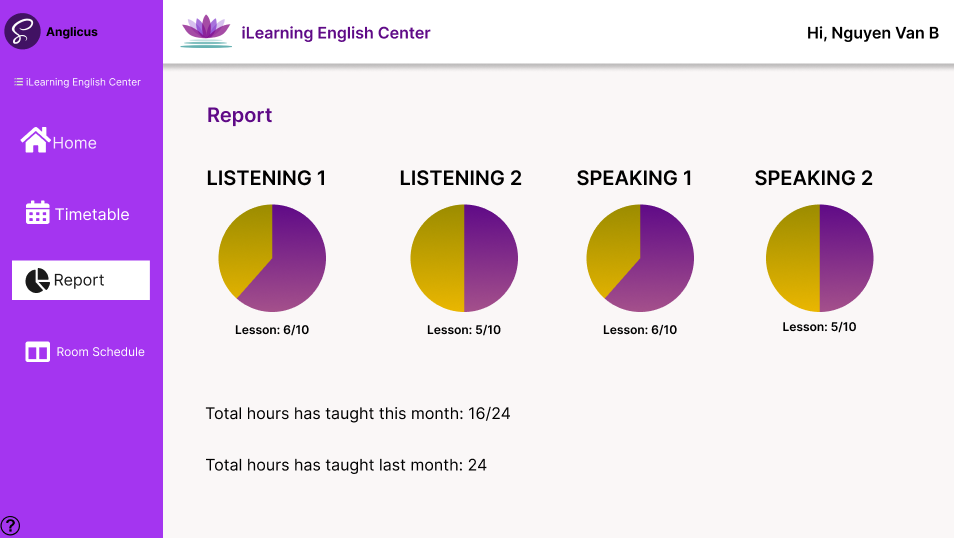
Picture 8. Showing students’ attendance and lesson report.



Picture 9. Adding new lesson content and updating students’ attendance.

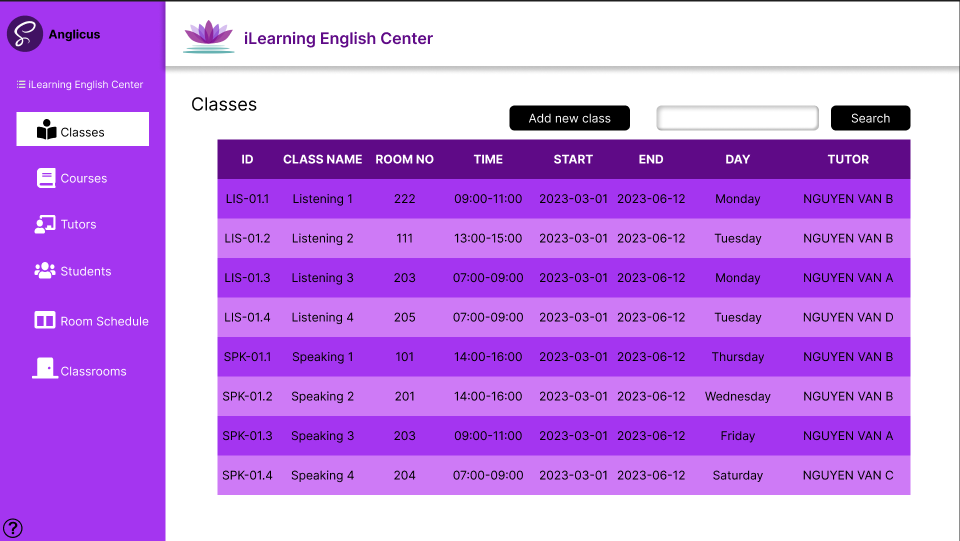


Picture 10. Showing timetable.

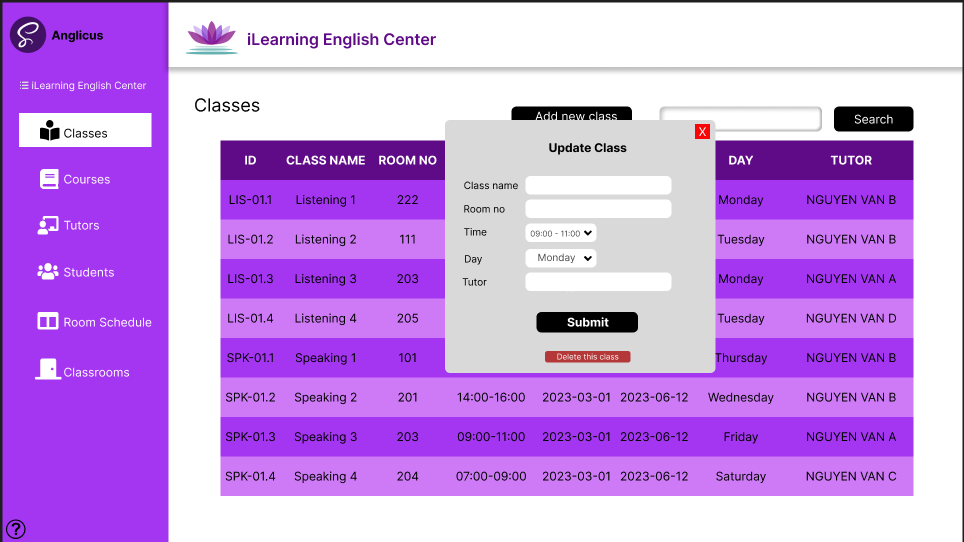


Picture 11. Showing tutor’s report.

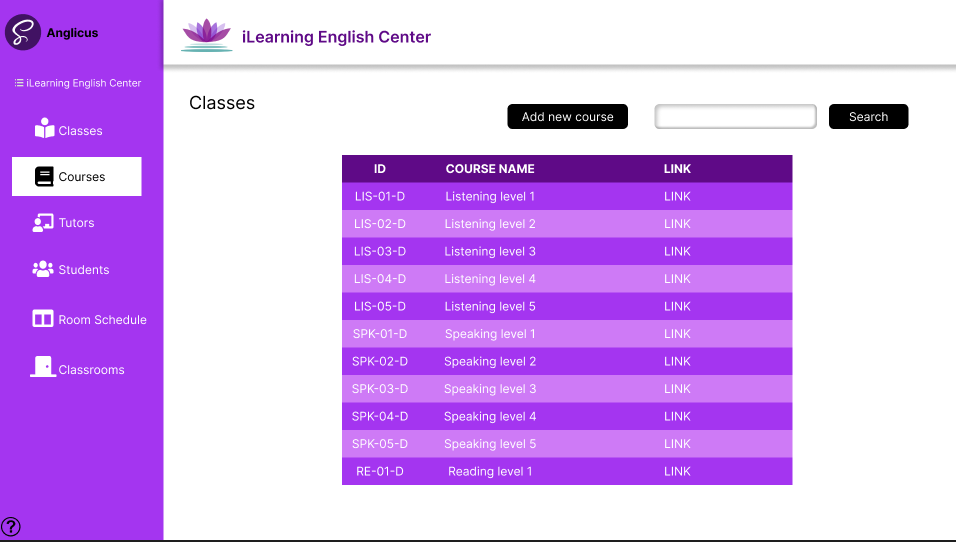
### 4.2.2 admin’s interface.

****

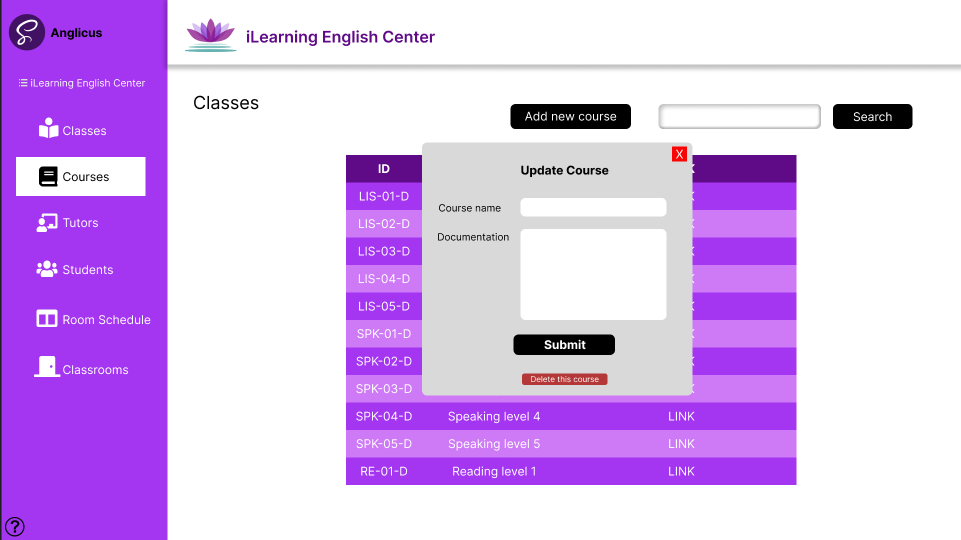
Picture 12. Showing list of classes.



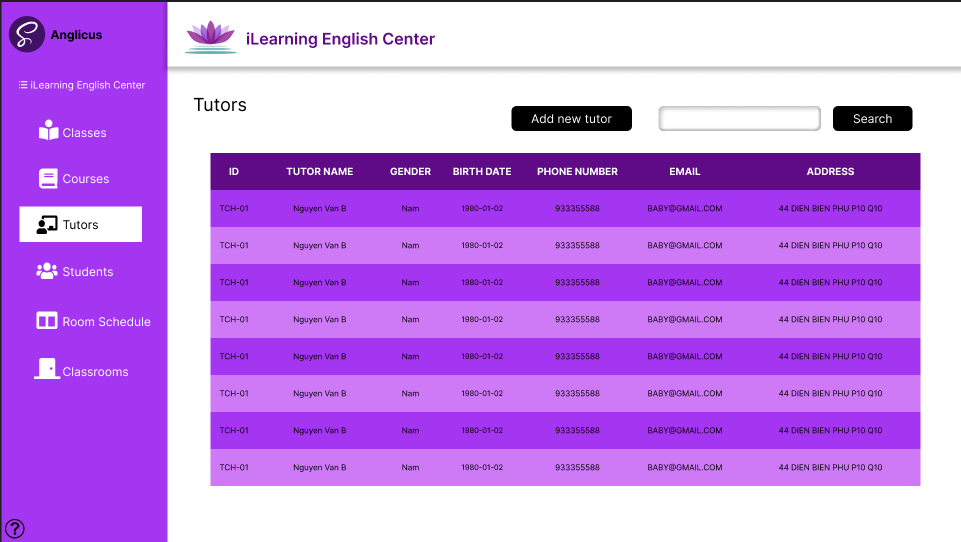
Picture 13. Update or delete a class.



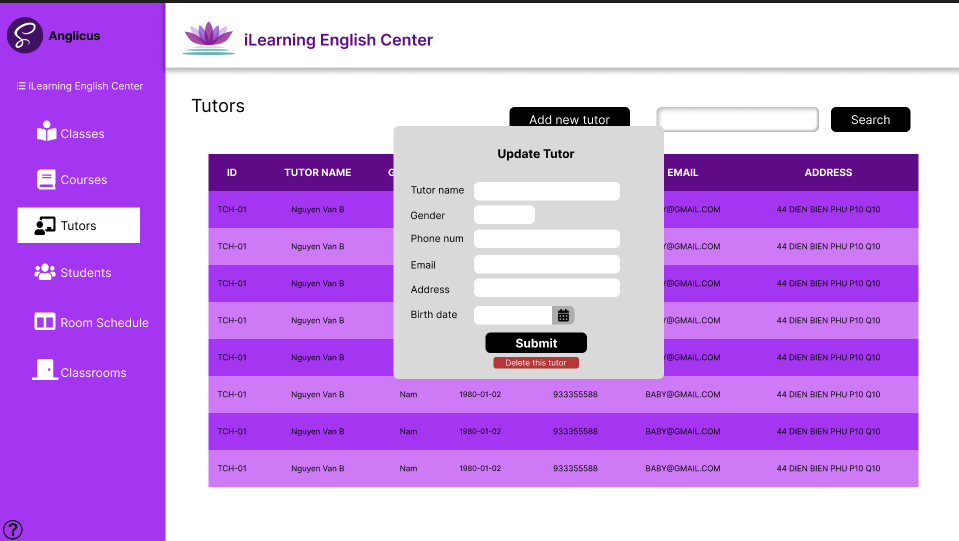
Picture 14. Showing list of courses.



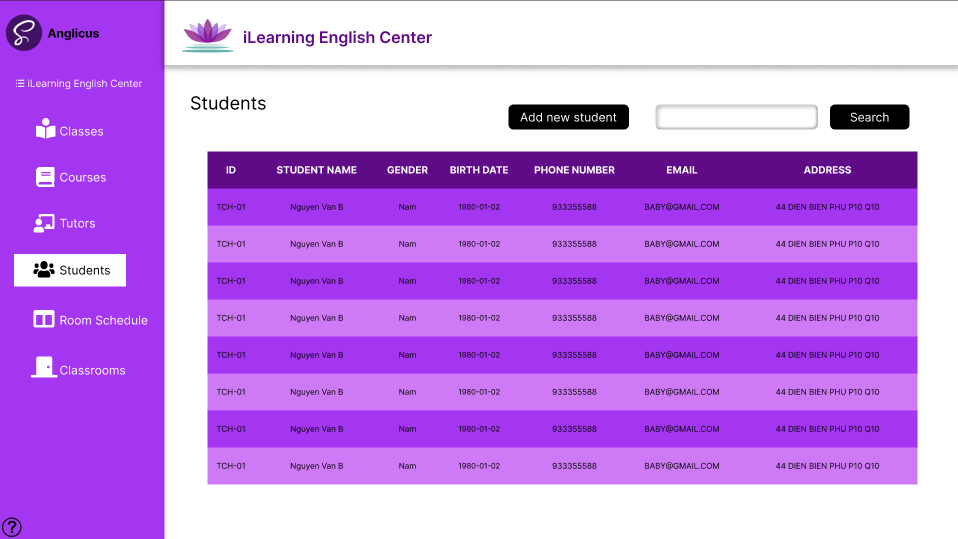
Picture 15. Update or delete a course.



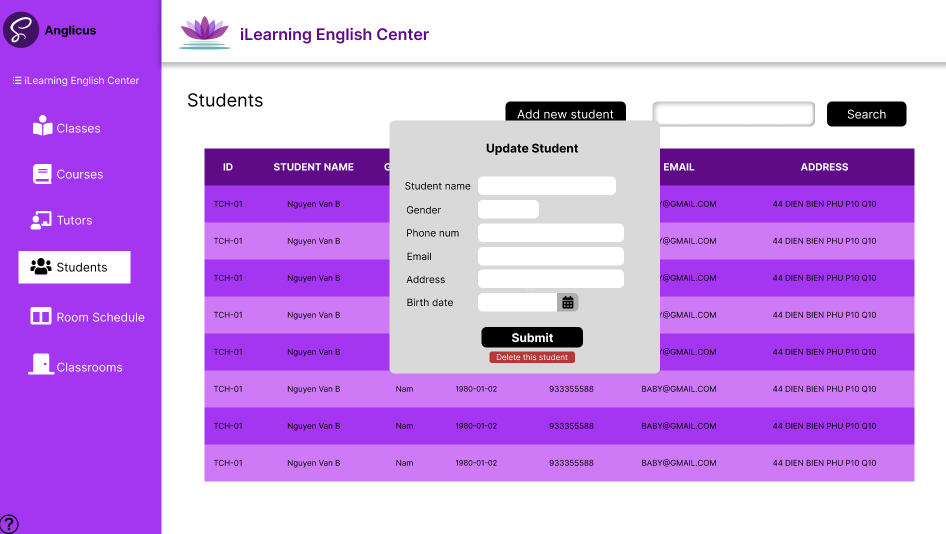
Picture 16. Showing List of tutors.



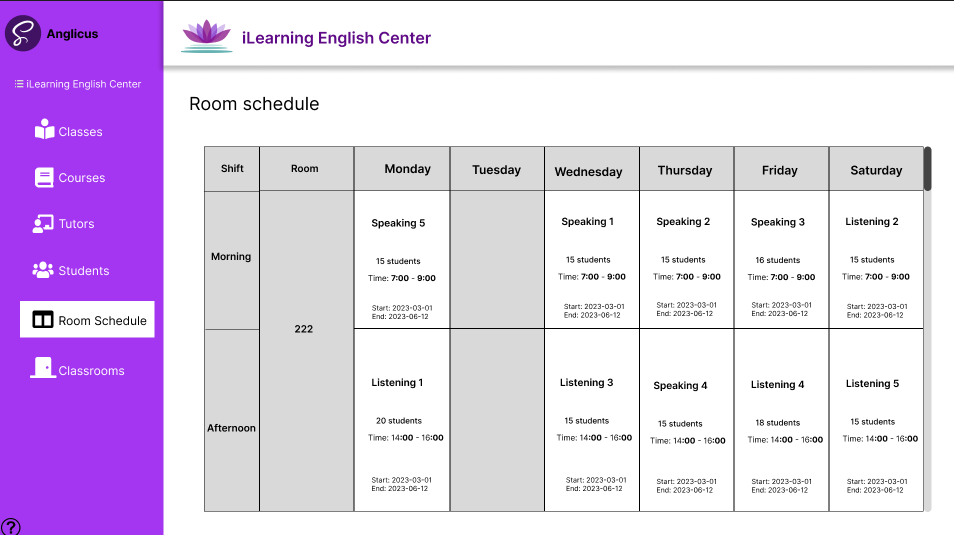
Picture 17. Update or delete a tutor.



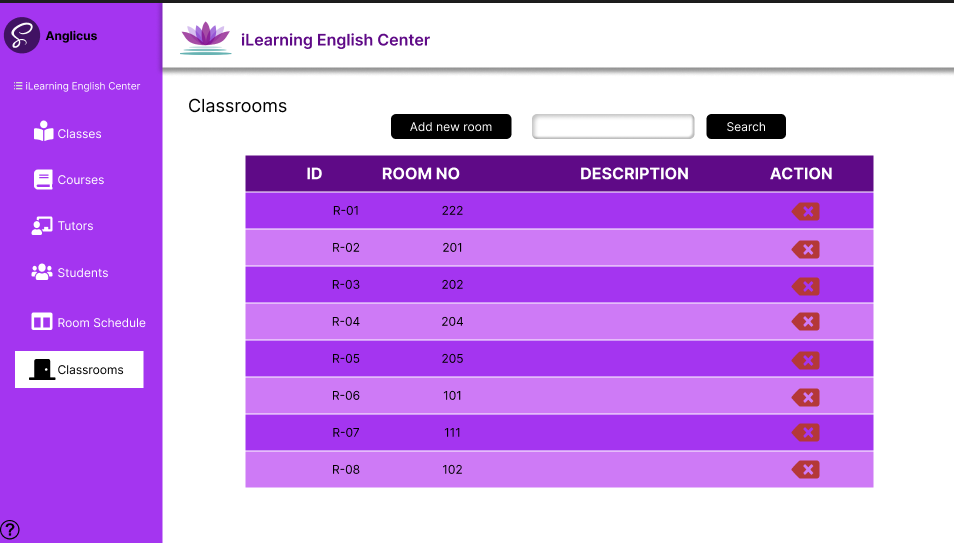
Picture 18. Showing list of students.



Picture 19. Update or delete a student.



Picture 20. Showing room schedule.



Picture 21. Showing classrooms’ information.

# Chapter 5: Conclusion

## 5.1 Achievements

Thanks to the effort of our team members, we were able to create a management app that despite its flaw, is able to satisfy the faculty personnel. The app provides usage for both tutors to manage their schedule and student academic study and for admin to manage the inner working of the center. Our app contains the following functions:

* Manage student information
* Manage tutor information
* Manage lesson report
* Manage grade report
* Manage classes
* Manage classrooms
* Statisitc report
* Teaching schedule
* Room schedule

## 5.2 Limitations

Althought the project is completed, it still far from perfection, the app still has many feature that are yet to be realized. Mainly due to the time constraints and our own lack of experiences, the following features are not present in our app:

* Feature for tutor to upload and store their teaching document.
* Feature for tutor and admin to directly send email to a class of student.
* Feature for admin to manage courses.
* Feature to automatically design teaching schedule for tutor.