Software Requirements Specification

for

Attendance Management System

Version 1.0 approved

Prepared by

Abu Jafor Mohammad Saleh(170110)

Abdullah Al Jawadul Omar Rifat(170135)

Jashore University of Science and Technology

01-10-2019

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Abu Jafor Mohammad Saleh	30-09-2019		1.0

1. Introduction

1.1 Purpose

This document is to describe all the software requirement specification (SRS) for the Attendance Management system. It explains the purpose and features of the application, the interface through which a user can access a personal account and the constraints that must be satisfied for security purposes. This document is intended for both the customer and the project development.

1.2 Document Conventions

The document uses the following conventions.

Short Form	**Abbreviation**
DB	Database
DFD	Data Flow Diagram
CRC	Class Relationship Collaboration
ER	Entity Relationship

1.3 Intended Audience and Reading Suggestions

The purpose of the attendance management system is restricted within university premises. This project has been implemented under the guidance of Md. Shafiuzzaman (lecturer of Jashrore University of Science of Technology). This is useful for the students and teachers.

1.4 Product Scope

The system has been facing problems due to its paper-based attendance system. Roll calling in manual process kills the valuable time of teachers. Besides, calculating the present of a day, or calculating total present of a student in a semester is a tredious work.

So, the purpose of attendance management system is to ease attendance system and to create a convenient and easy to use application for teachers and studens of universities. It is very easy to give attendance. Students can view their total attendance of a class. Teaches and students can communicate with one another about the class.

1.5 References

https://nevonprojects.com/android-based-self-attendance-system-using-otp/
https://www.udemy.com/ios-13-app-development-bootcamp/
https://www.udemy.com/course/devslopes-firestore-ios/

2. Overall Description

2.1 Domain Analysis

Technical Literature:

Attendance management system is very common and regular process for all the schools, colleges and universities. It is a very familiar to all of us that, teacher enters into the classroom, after that he/she opens the attendance book and starts roll calling. It kills valuable time of the class. At the last of the semester when the teacher needs to know the total attendance of a student he/she calculates all the attendance of the student. Students also can not be able to know about the percentage of total attendance of current time. For this reason, they become careless about their attendance in the class.

Existing applications:

Such kind of application are not available in Play store (for android devices) and App store (for Apple users). But such application is available for web application. All the common features are included in the features such as:

- Attendance system via bluetooth.
- Student, teacher , admin log in, register page.
- Attendance view for students, teachers and admin.

We know that, if we want to want to open a web application we need a browser and after searching the proper address we enter the application. But if use mobile application it will be very easy to run. Now a days, mobile is so available to all and people loves to use mobile application than web application as it is easier than web application.

2.2 Product Features and Prioritization

Normal Requirement: Normal requirement consists of objectives, goals, minimal function and performance that are stated during the meeting with customer. The normal requirement of our project are:

- Student, teacher and admin register.
- Student, teacher and admin login.
- Students, teachers and admins can see the percentage of attendance of a particular student.
- Teacher will be able to create a class for any particular course.
- Students can join the class created by teacher.
- Students and teachers can see the course list.
- Teachers will accept the request of students who want to attend in the class.

Expected Requirement: Expected requirement consists of implicit requirements and may be so fundamental that the customer does not explicitly state them. Their absence will be a cause of dissatisfaction. The expected requirement of our project are:

- Accessible via internet.

- Manage database for students.
- Manage database for attendance.
- Dashboard for students, teachers and admins.

Exciting Requirement: These requirements are for features that go beyond the customer's expectation and prove to be very satisfying when present. The exciting requirement of our project are:

- Teachers will get notification before 30 minutes of class starting.
- If teacher cancels class students will get notification.
- Student will be able to review about the class.

2.3 Stakeholders

We have identified the possible stakeholders for our attendance system.

1. **The University VC (Project Sponsor) :** The University VC is a person who has the final authority over our budget, our personal resources, and ultimately the finished product. He can take any decision and can give any veto for the project.

2. **The controller section (System Head):** The controller section has the direct authority on our project budget and also on our team.

3. **The Chairman(Administrator) :** All the chairman of all the departments have the administrative power to handle the software.

4. **System Operator :** He will be able to interact with the software directly.

5. **Student :** Mainly the software will be made for their purpose. They will present their attendance and can see the percentage of their attendance.

6. **Teachers :** They will take a big part in the system.

7. **Developers :** I can take the developers as stakeholder because the system will be updated day by day and if occurs any system interruption , they will find the problem and try to solve it.

8. **University :** University will finance the project and it has some rules and regulation to maintain our system. We have to follow them strictly.

2.4 User Classes and Characteristics

User of the system should be able to retrieve attendance information of the student's of university. Throughout the system, teacher will be able to create class for the students. Students can be able to join that class and their attendance will be counted automatically in the system. Students, teachers will be able to to see the attendance of any class.

The admin should be able to the following function:

- Add course

- Confirm student and teacher

The teacher should be able to the following function:

- Create class
- Confirm student for the class
- View attendance

The student should be able to the following function:

- Request for join class
- View attendance

2.5 User Story

In the Attendance System Application, as a admin he/she want to register and login so that he/she can confirm the confirmation request of all the students and teachers of the university, monitor them and add courses in the course list. At the same time, as a student he/she want to register and login so that he/she can give my attendance, see the percentage of my attendance in any class and get notified about the class also can give review about the class. Similarly, as a teacher he/she want to login so that he/she can able to take the attendance of my students from the class and see their percentage in the class and also get feedback about my class.

2.6 Operating Environment

Operating environment for attendance management system

- Operating system: IOS
- Database: Firebase
- Platform: Xcode

2.7 Design and Implementation Constraints

The Attendence system shall be an ios based application system. The system shall be developed using Android and Firebase server.

A person who has no knowledge of computers will find it difficult to understand the system. But with a little knowledge it will be very easy to handle the project. This will be a ios based application. So, it is not easy to implement a web based application and android based application.

2.8 Assumptions and Dependencies

- Each user must have a valid user id and password.
- Only admin can be able to block any student.
- Users must log in the system to access any record.

2.9 User Documentation

- Help pages will be providing document with screen shots.
- If the user has more queries regarding this website then he/she can contact with the administrator through contact us page.

3. External Interface Requirements

3.1 User Interfaces

The user interface is designed in ios. The developer will have to study the designing of the product. The use of the controls and the component from the Add items feature of the ios. The user of the product will get very user friendly environment which will be very easy to work with.

3.2 Hardware Interfaces

- Iphone

3.3 Software Interfaces

Software Used	**Description**
Operating system	We have chosen IOS operating system for its best support and user-friendliness.
Database	To save student's attendance records I have chosen firebase database.
Xcode	To implement the project I have chosen swift language for its more interactive support.

3.4 Communications Interfaces

The attendance management system will support all iphones. I have used Bluetooth system to take attendance in this project.

4. System Features

4.1 Teacher

- Class management
- Communicate with students

4.2 Student

- Joining in a class

- View attendance
- Get notified about class

4.3 Admin

- Adding new courses
- Confirming students and teachers

Use Case

Use Cases Description: In this section use case scenarios are described elaborately.

Use case name : Sign up

Primary Actor : Students, Teachers, Admins.

Goal in Context : To register in the system.

Precondition :
- Attendance system has been designed to add admins, teachers and students.
- It has an interfaces for registration.

Triggers : The students , teachers and admins have to register .

Scenario:
- Visit the register page.
- Input required information
- Check availability for username and check validity of password.
- Email send to user e mail address.
- User confirm from his/her e-mail address.
- Confirmation message shown.

Exception:
- Invalid Input.
- Not valid for Registration
- Authentication failed.

Priority : Must be implemented.

When Available : At the first time of registration.

Use case name : Sign In

Primary Actor : Students, teachers, admin.

Goal in context : When user want to enter the system.
Precondition : Must be registered.

Triggers: Need to log In the system for students, teachers and admins.

Scenario:
- Visit the login page.
- Input Username and Password.
- Proceed the next activity.

Exception:
- Username/Email address invalid
- Wrong Password

Priority: Must be implemented.

When Available : First implemented and after log out for the system.

Use case name: Delete Account

Primary Actor: Admin, student, teacher.

Goal in Context: To delete an account from the system.

Precondition:
- Admin, teacher, student can delete any account.
- Must be logged in.
- Password must be given.

Triggers: Need to delete any account from the system.

Scenarios:
- Visit the delete account page.
- Select the Username.
- Confirm the user email address.
- Confirm password.
- Delete the account .
- Proceed to the next activity.

Exceptions:
- Username Invalid.
- Email address invalid.
- Password invalid.

Priority: Must be implemented.

When available : When admin will want to delete any account from the system.
Use case name : Update information
Primary Actor: Student, teacher, admin

Goal in Context: Update personal information.

Precondition:

- Must be logged in.
- Must need password to Update the information.

Triggers: To update the personal information of students, teachers and admins.

Scenarios:
- Visit the Update Information page.
- Update information what is needed.
- Give the current password.
- Save the give information.
- Back to previous page.

Exceptions:
- If given password in unequal.
- If information type invalid.

Priority : Essential, Must be implemented.

When Available: Always available when user want to update information from his/her own page.

Use case name : Apply Course.

Primary actor : Students.

Goal in Context : To join any class , student will apply for the course.

Precondition:
- Must be logged in.
- Must be verified through admin.

Triggers: Student will apply for any course to teacher.

Scenarios:
- Visit Apply course page.
- Select the course.
- Give the password.
- Apply the course.
- Back to the previous page.

Exceptions:
- If password invalid.
- If student not logged in.

Priority: Essential , must be implemented.

When Available: Always available , when student want to apply for the course.

Use Case : Leave Course

Primary Actor : Students.

Goal in Context: If students want to leave any course .

Precondition:
- Must be logged in.
- Must be added in the course.

Triggers: Student will leave the course , if he/she wants.

Scenarios:
- Visit the Current Course page.
- Select course.
- Write the course Id.
- captcha automation.
- Confirm to leave the course.
- Give password.
- Back to previous page.

Exceptions:
- Id course Id is invalid.
- If given password is invalid.
- If captcha is wrong.

Priority: Essential, must be implemented.

When Available : If student is added in the course.

Use Case : View attendance.

Primary Actor : Students, teachers and admins.

Goal in Context : To see the total and particular attendance of a student.

Precondition:
- Must be logged in.
- Course id is needed to check.

Triggers : If students, teachers and admins want to see the attendance of students, they will be able to see it.

Scenarios:
- Visit the Attendance view page.
- Select the Course / select the course and student id.
- See the attendance percentage.
- Back to the previous page or select another course.

Exceptions:
- Course Id and student id must be valid.

Priority: Essential, must be implemented.

When Available: first increment.

Use Case : Update attendance.

Primary Actor : Teacher.

Goal in Context : Update attendance of students.

Precondition :
- Must be the teacher's course.

Triggers : if teacher want to update attendance , he will be able to do it.

Scenarios:
- Log in the system.
- Click the update page .
- Select the course.
- Update the attendance .
- Back to previous page.

Exceptions:
- Must be teacher's own class.

Priority : Essential , must be implemented.

When Available : Only available for the teachers.

Use Case : Forgotten password.

Primary Actor : Students, teachers and admins.

Goal in Context : Get new password for forgotten password.

Precondition:
- Must be registered.

Triggers : If users forget the password he can change it.

Scenarios:
- Click the Forgotten password page.
- Give email address.

- Confirmation message sent to email.
- Give new password.
- Save the password.

Exceptions :

- Must be registered.
  If users forget email address.

Priority : Essential , must be implemented.

When Available : Always available when needed.

Use Case : Confirm apply

Primary Actor : Teacher.

Goal in Context : Confirm the request of students.

Precondition:

- Must be teacher.
-

Triggers : If teacher want to confirm request of the students.

Scenarios:

- Teacher will log in the system.
- Teacher will visit the confirm request page.
- See students details.
- Approve student for the class.

Exceptions:

- If teacher do not create the class.

Priority : Essential, must be implemented.

When Available: Available when , teacher wants to do it.

Use Case : Create class.

Primary Actor : Teacher.

Goal in Context : Create a class for the students.

Precondition:

- The teacher must be confirmed by the admin.
- Create a valid class.

Triggers : Teacher will Create a class for the students.

Scenarios:
- Log in the the system.
- Click the create class button.
- Visit the create class page.
- Give information of the class.
- Create the class.

Exceptions:
- If information is invalid.
- If teacher is not confirmed via admin.

Priority : Essential , Must be implemented.

When Available : When teacher want to create a class.

Use Case : Block account.

Primary Actor : Admin.

Goal in Context : To block the account.

Precondition:
- Must be admin of the system.

Triggers : If admin want to block the account the will block it.

Scenarios:
- Log in to the system.
- Visit to block account page.
- Select a person .
- Block the account.

Exceptions:
- If the system is not logged in through admin.

Priority : Essential , must be implemented.

When Available : Always available when admin will want to block any account.

Use Case : Add course

Primary Actor : Admin.

Goal in Context : To add courses in the system.

Precondition:
- Must be admin.

Triggers : If admin want to add course in the system.

Scenarios:
- Log in the application.
- Click add course button.
- Visit the page.
- Give information of the course.
- Save.

Exceptions:
- If admin is not logged in.

Priority : Essential, must be implemented.

When Available : Always available when admin wants.

Use Case : Confirm request

Primary Actor : Admin

Goal in Context : Accept request of students and teachers.

Precondition:
- Must be admin.
- Must to same department of students and teachers.

Triggers : Accept request of students and teachers to confirm the student and teacher of that department.

Scenarios :
- Log in to the system.
- Visit the Request confirmation page.
- Accept the proper request.

Exceptions:
- If students and teachers are fake.

Priority : Essential, must be implemented.

When Available : Always available , when needed.

Use Case : Log out

Primary Actor : Students, teachers, admins.

Goal in Context : To log out from the system.

Precondition:

- Must be logged in.

Triggers : If users want to log out.

Scenarios:

- Students , teachers or admins must be logged in.
- Click to log out page.
- Click to log out button.

Exceptions:

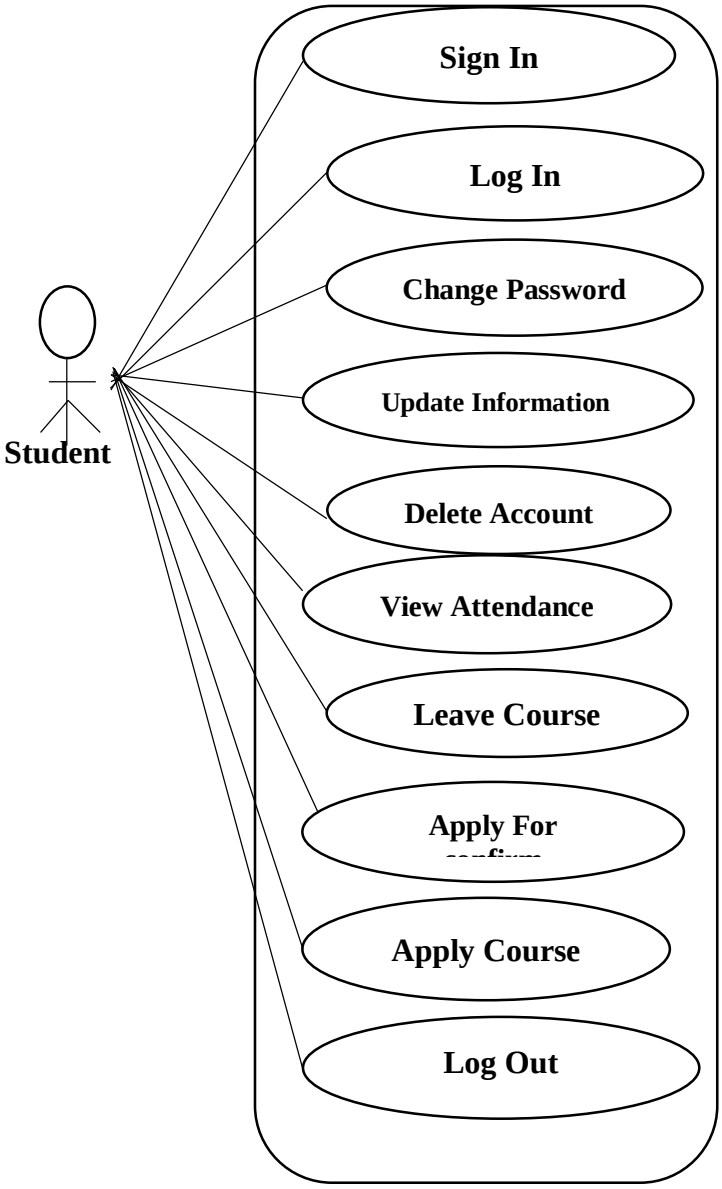
- If users are not logged in in the system.

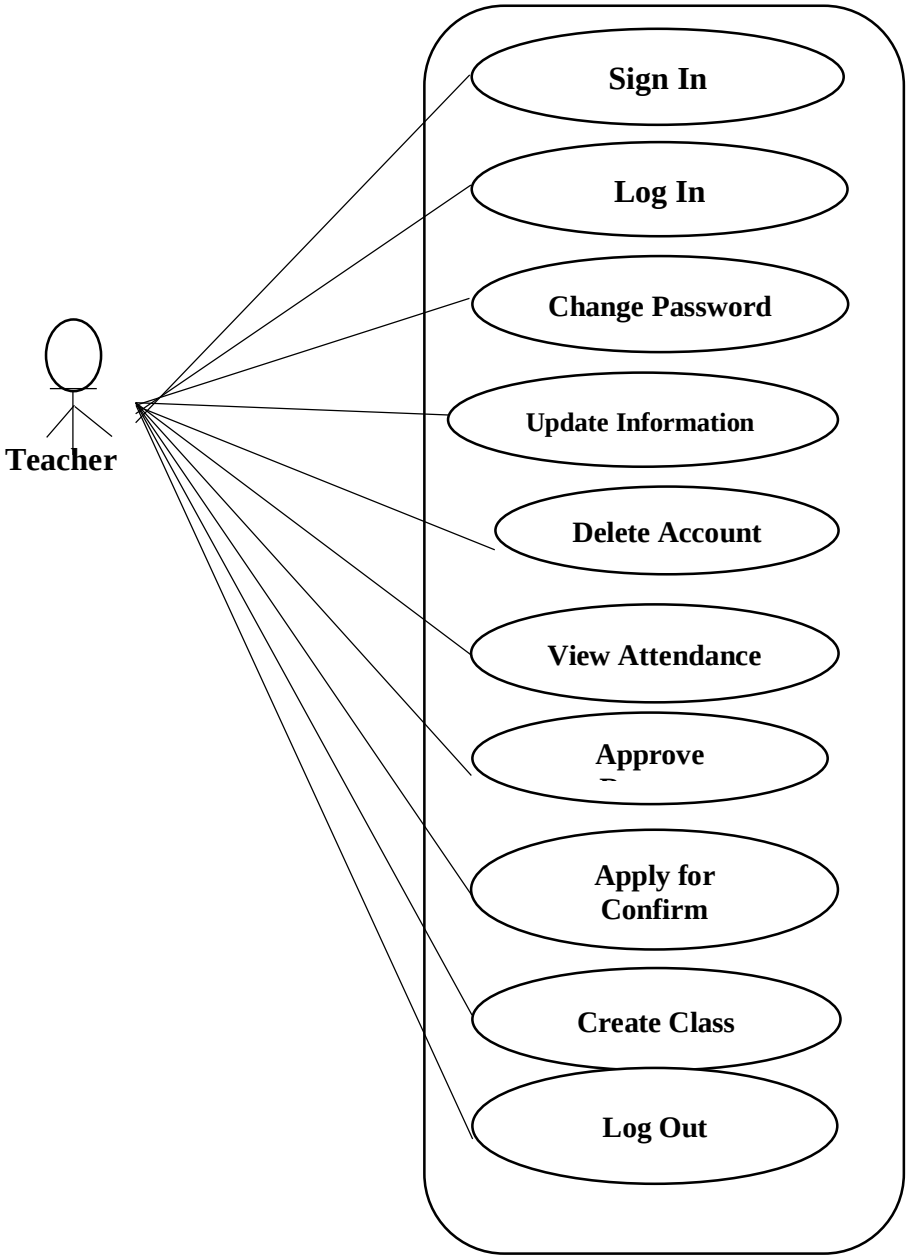
Priority : Essential, Must be implemented.

When Available : Always available.

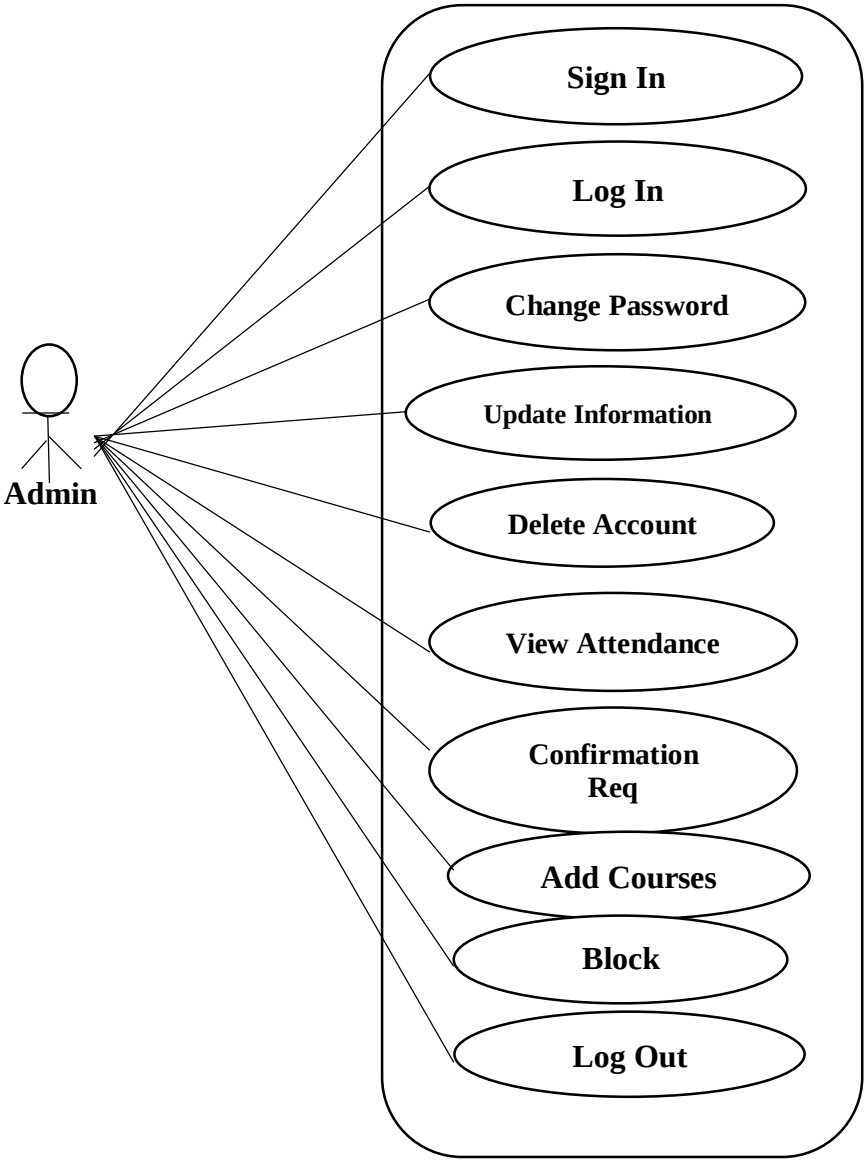
Use Case Diagram

Use Case diagram for Students

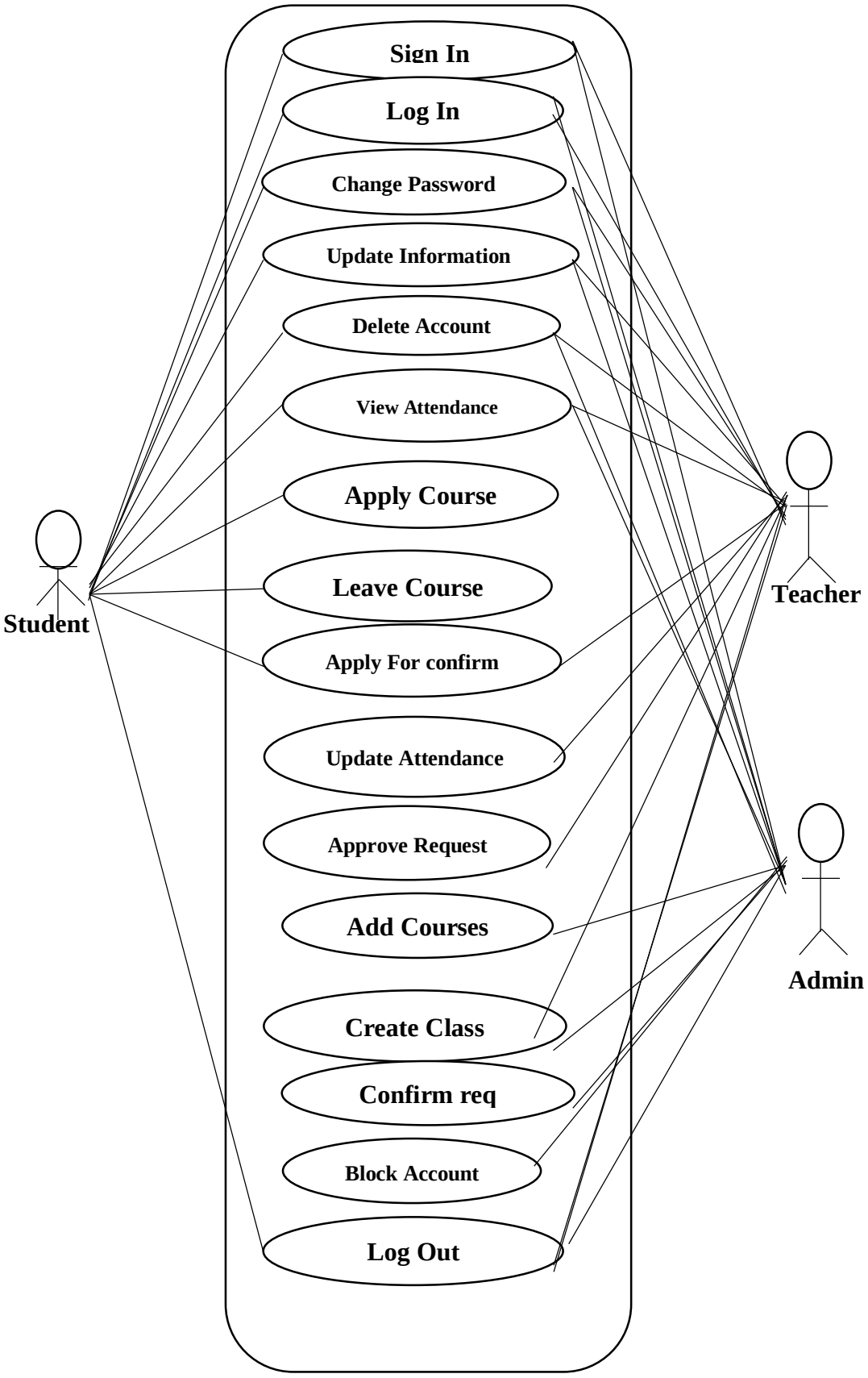




Use Case diagram for Teacher



Use case diagram for Admin

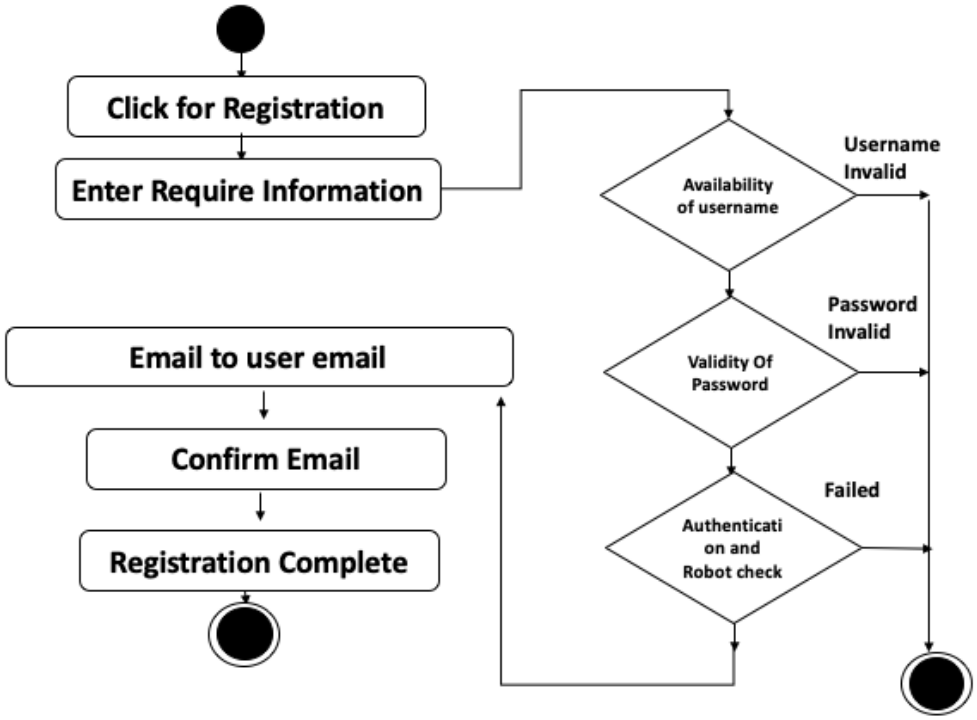


Use case diagram for all the system

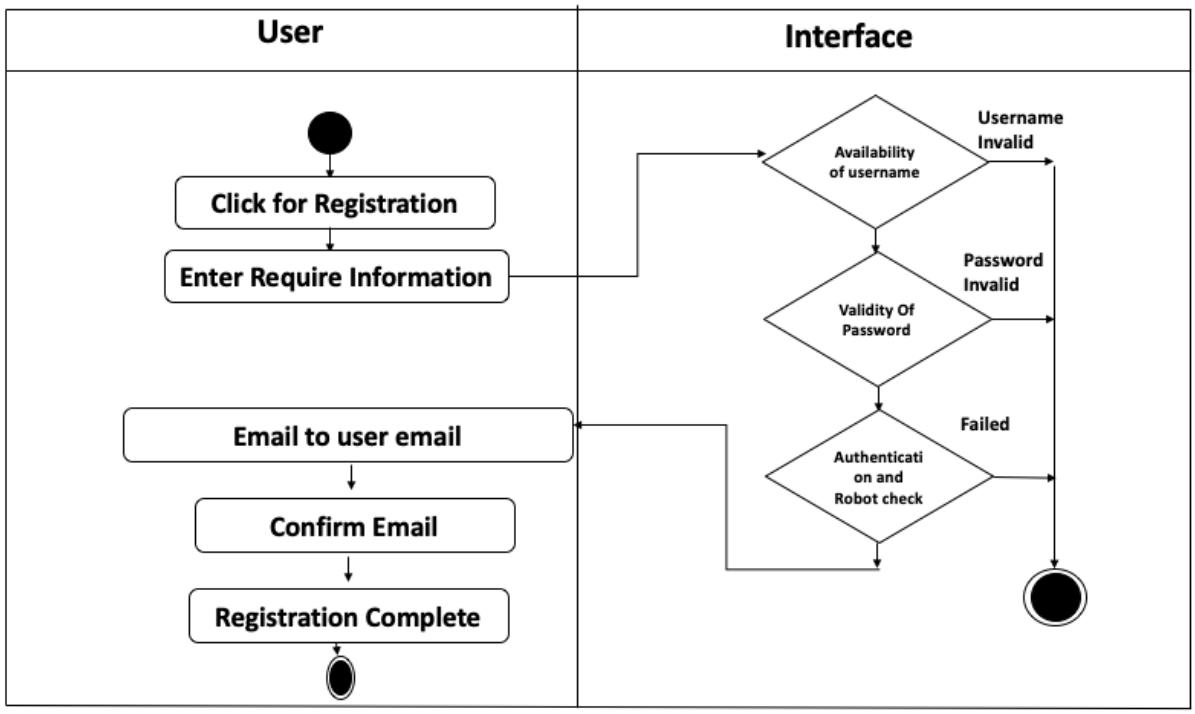
Activity and Swimlane Diagram

Activity Diagram and swimlane diagram :

Sign Up :

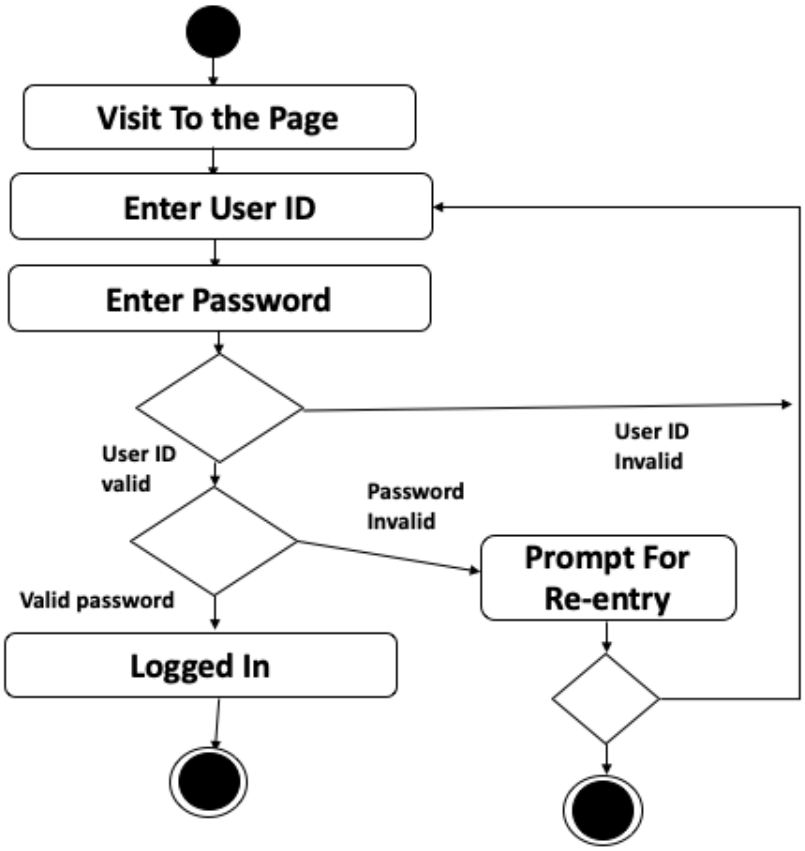


Activity diagram for Sign up

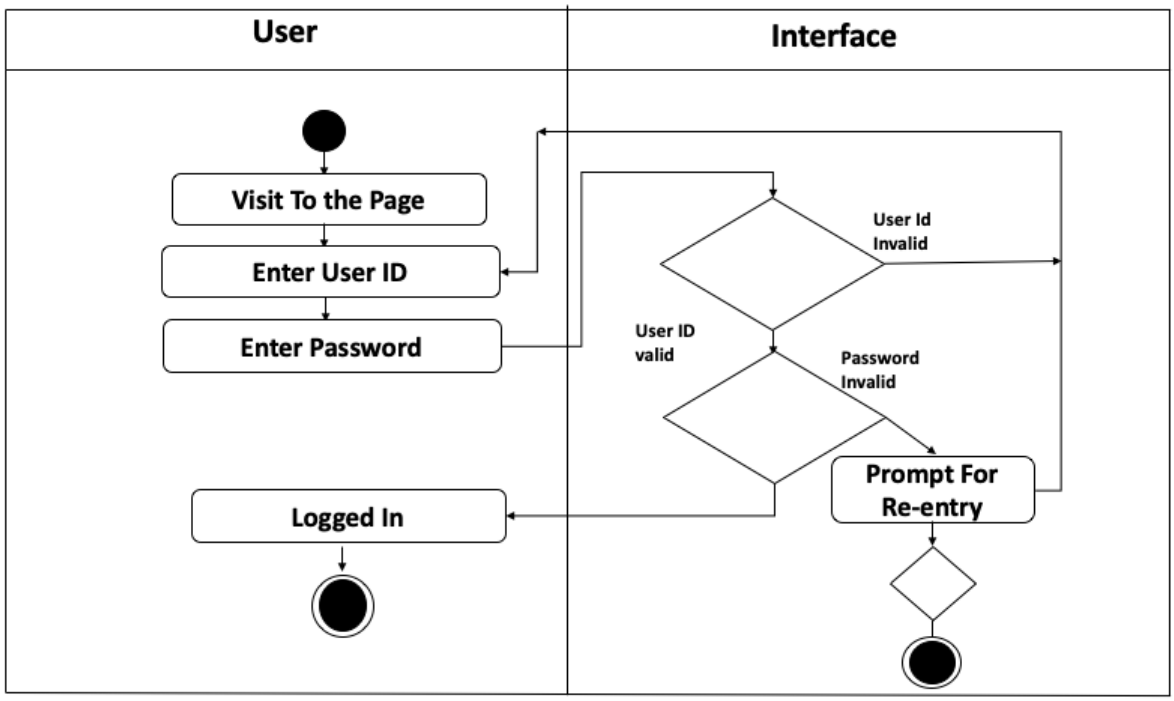


Swimlane diagram for Sign up

Log in:

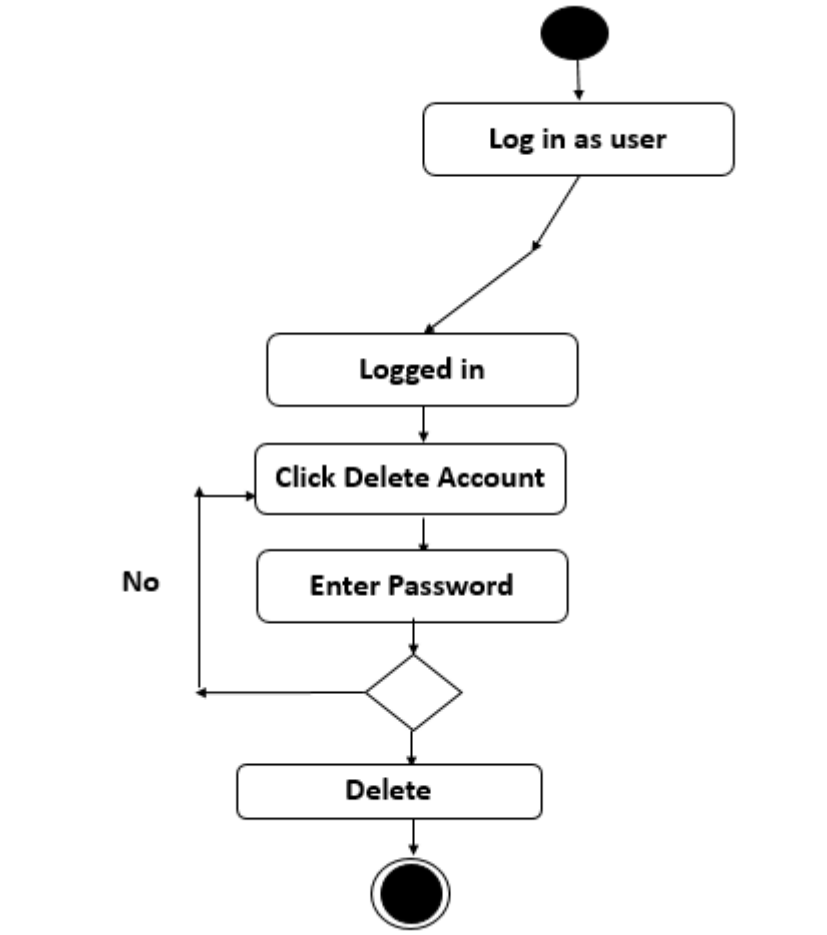


Activity diagram for Log in

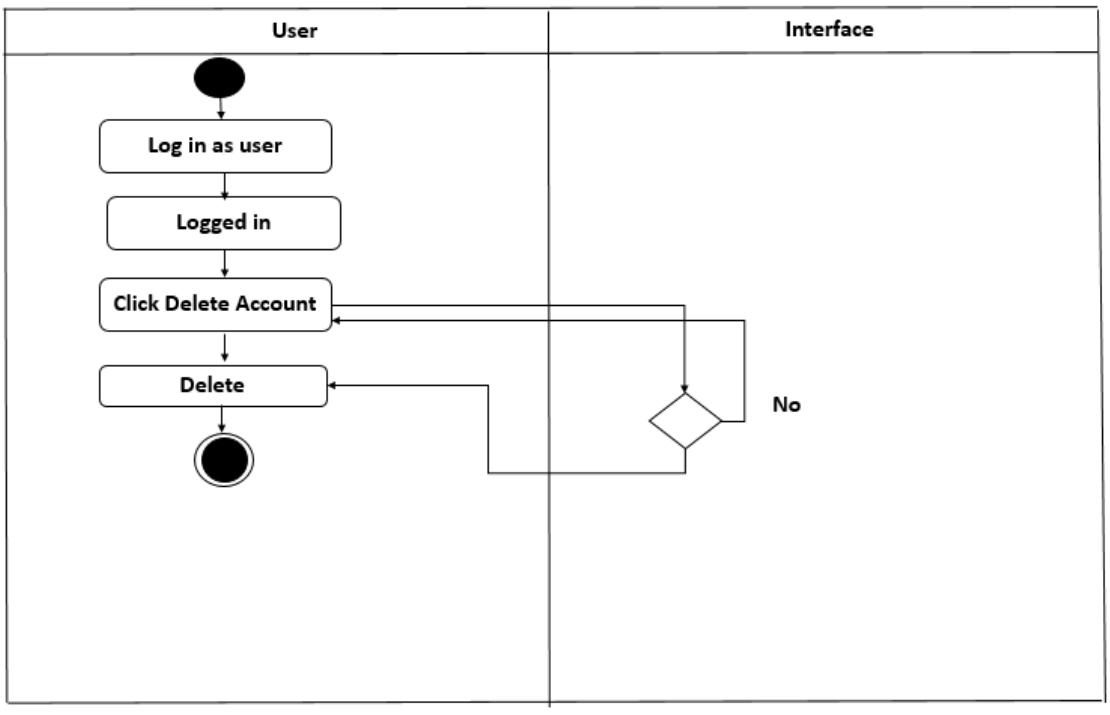


Swimlane diagram for Log in

Delete Account :

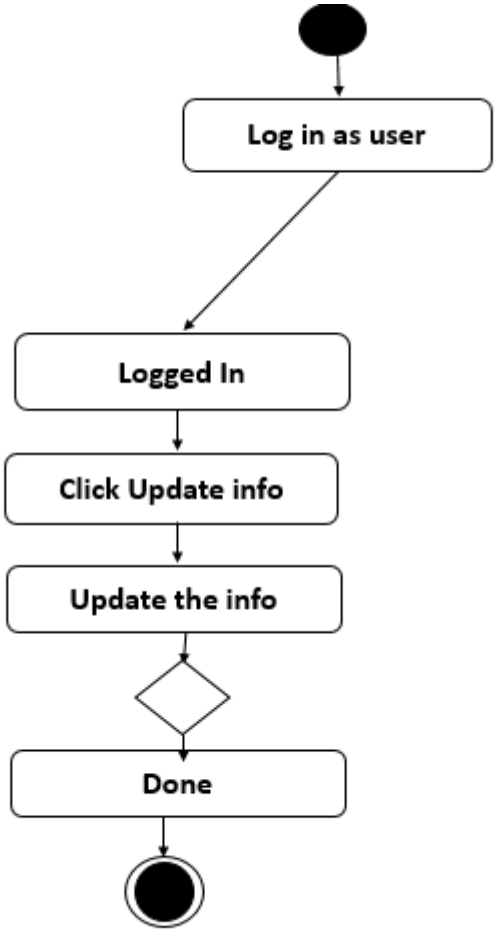


Activity diagram for Delete Account

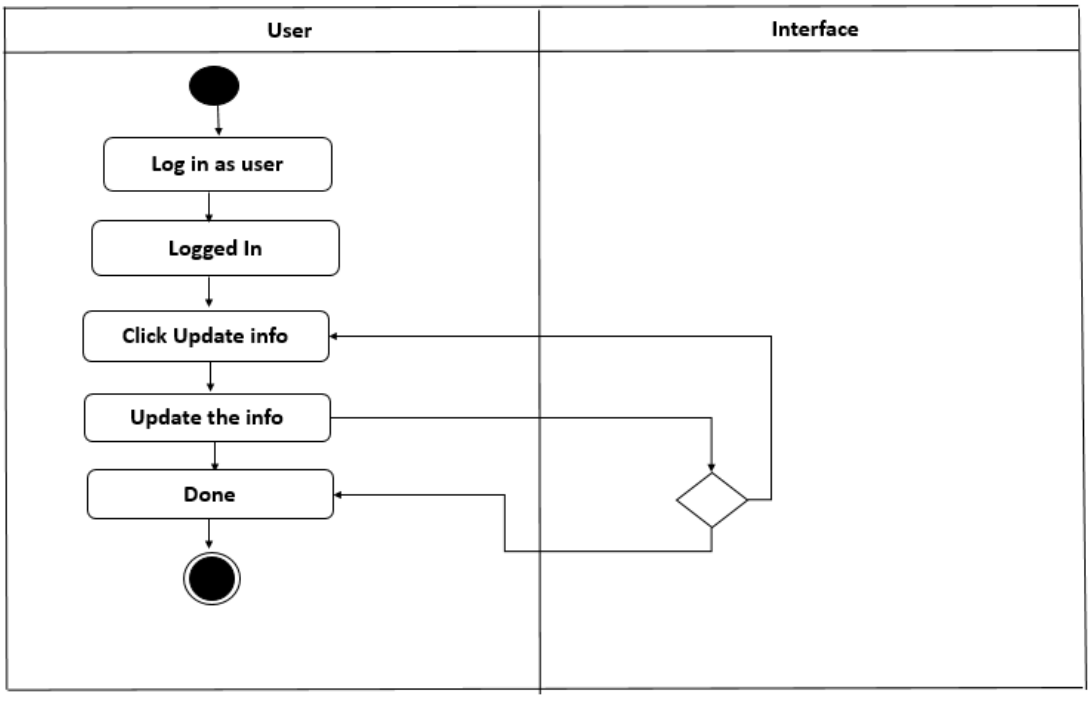


Swimlane diagram for Delete account

Update Information :

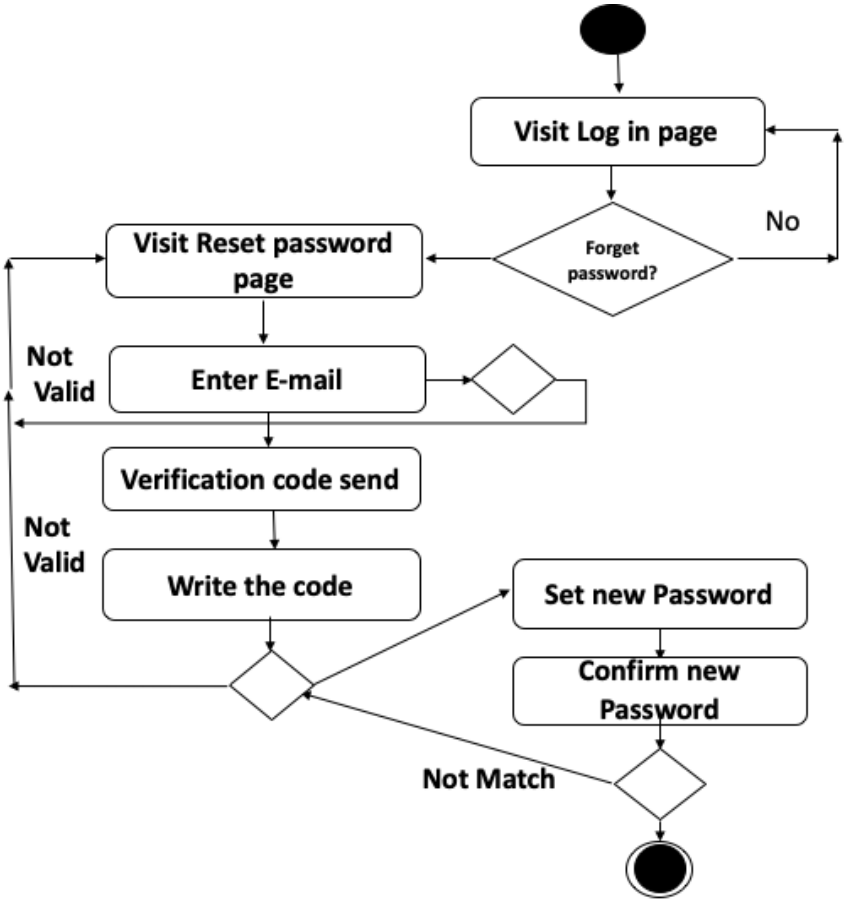


Activity diagram for update information

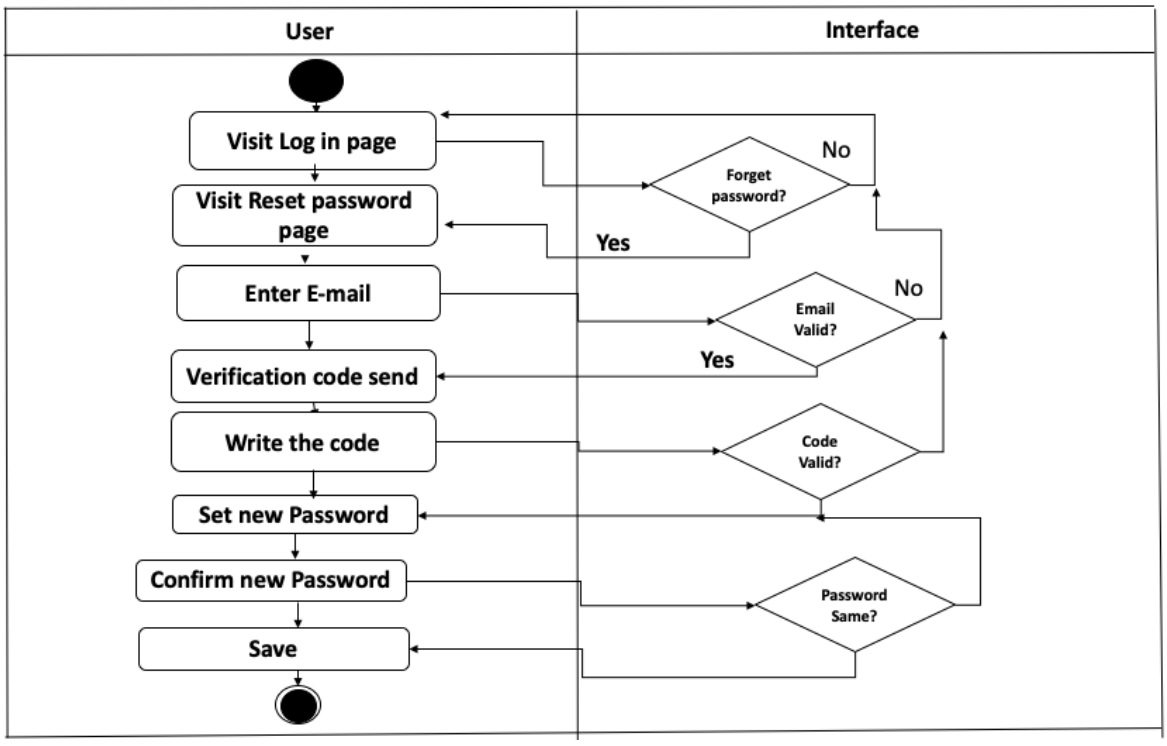


Swimlane diagram for update information

Forget Password :

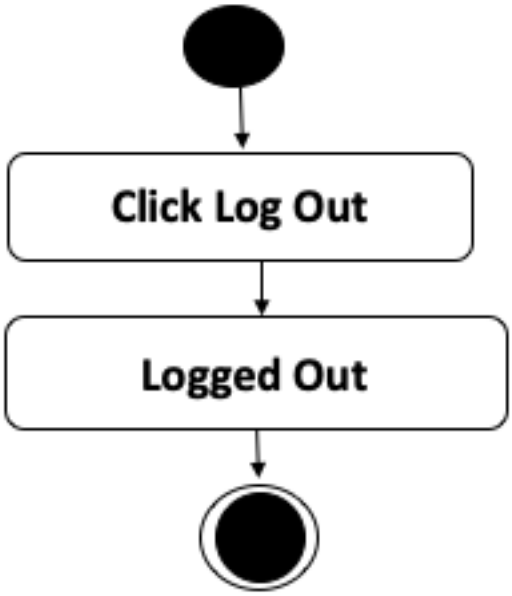


Activity diagram for Forget password

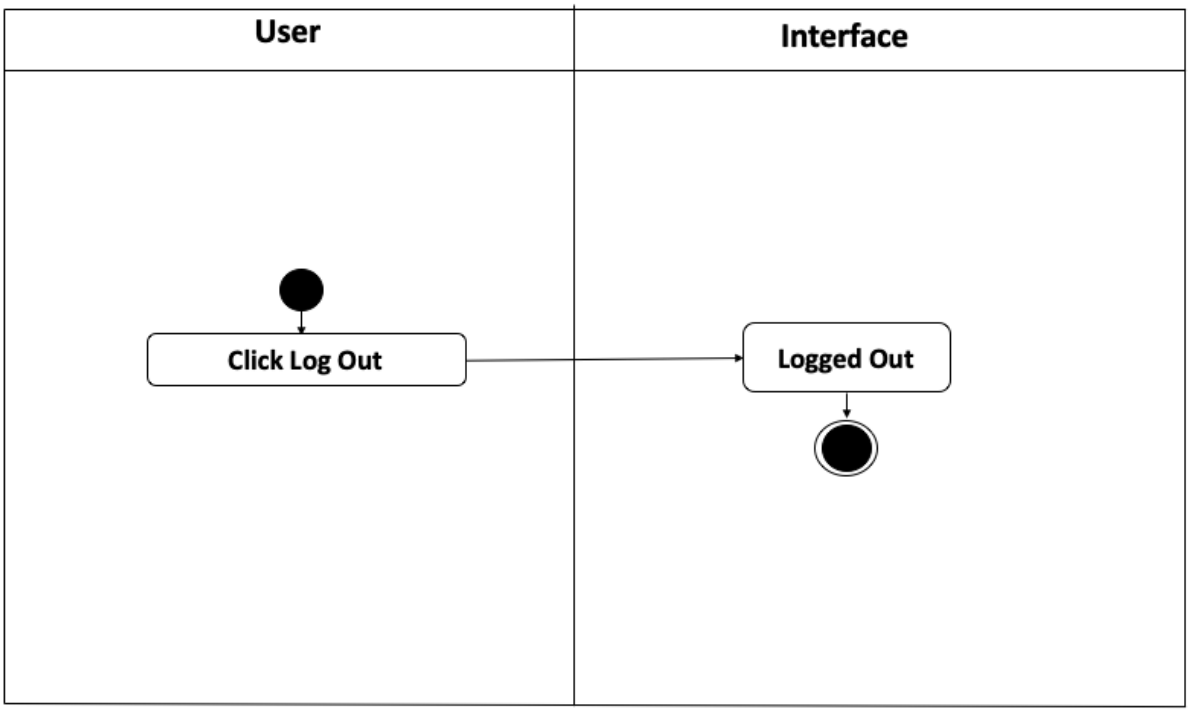


Swimlane diagram for Forget password

Log out:

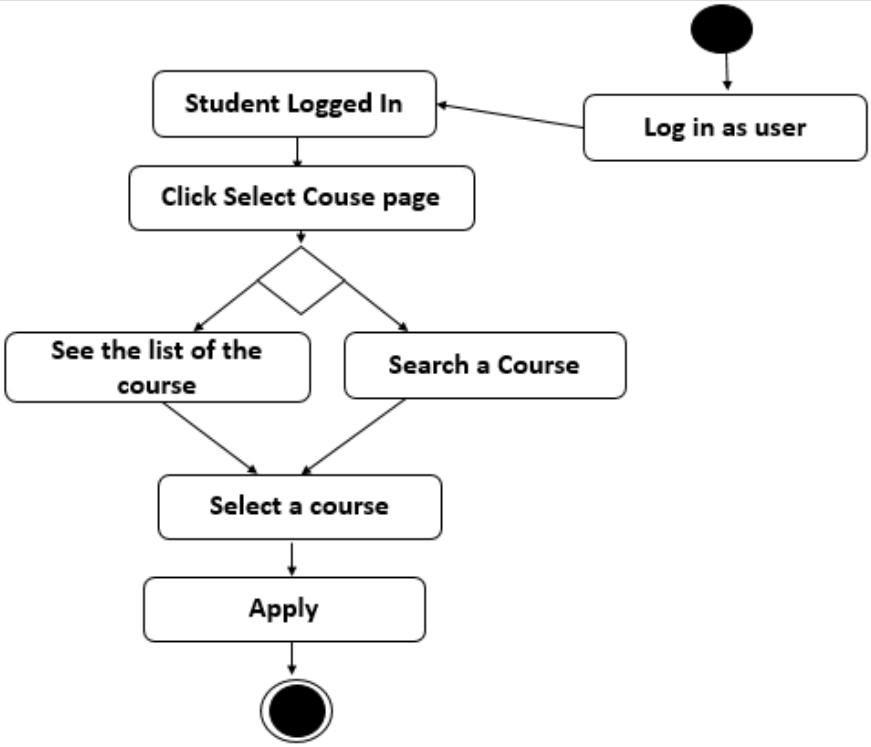


Activity diagram for Log out

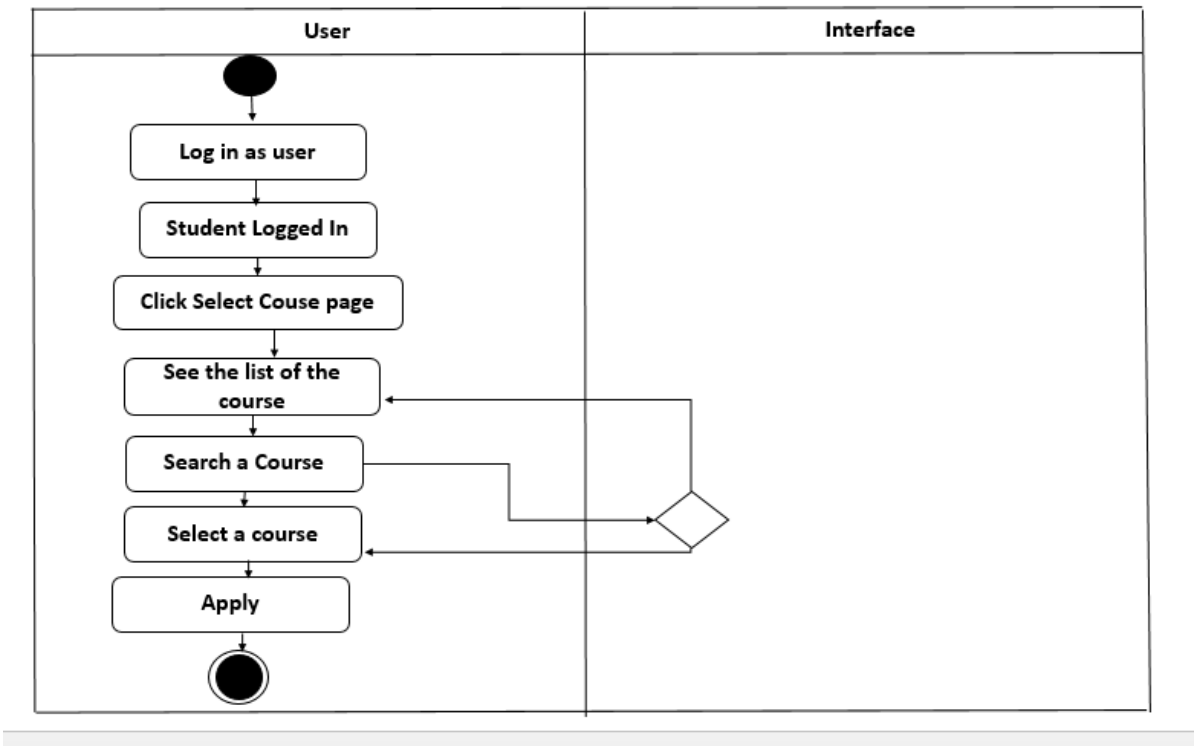


Swim lane diagram for Log out

Apply course :

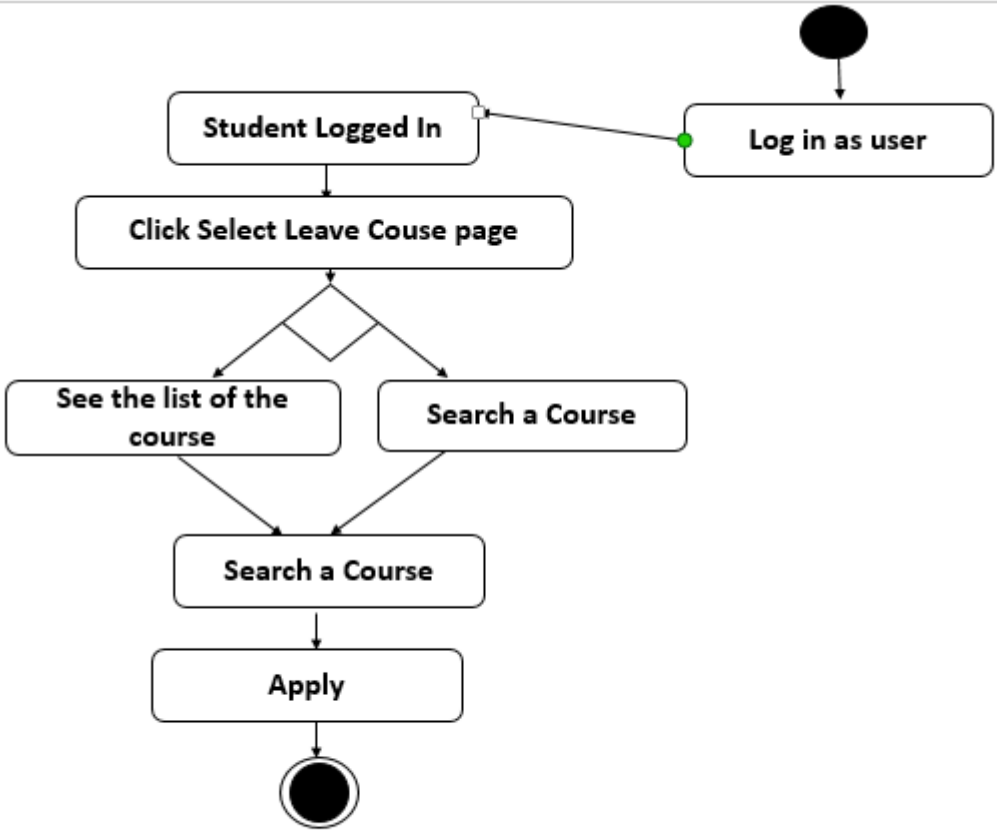


Activity diagram for Apply course

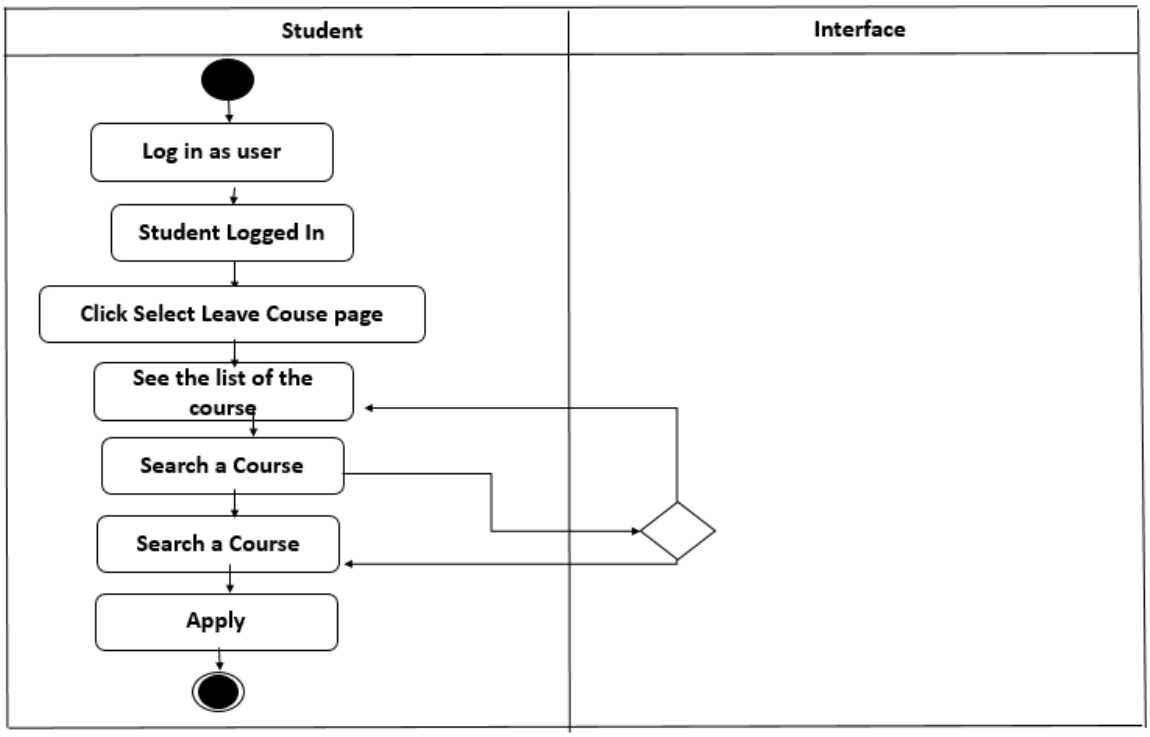


Swimlane diagram for Apply account

Leave course :

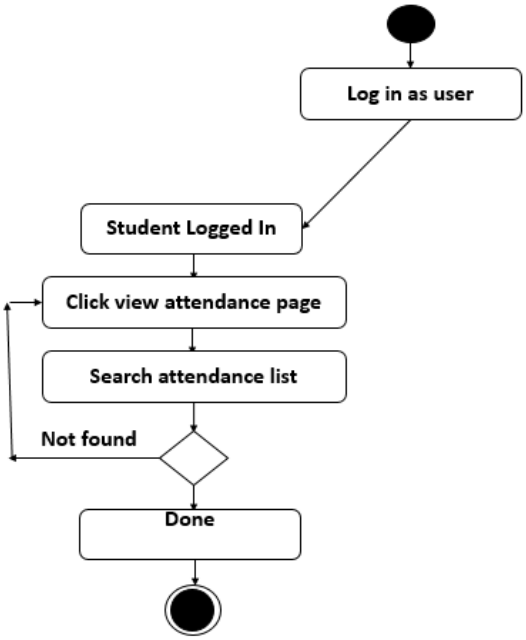


Activity diagram for Leave course

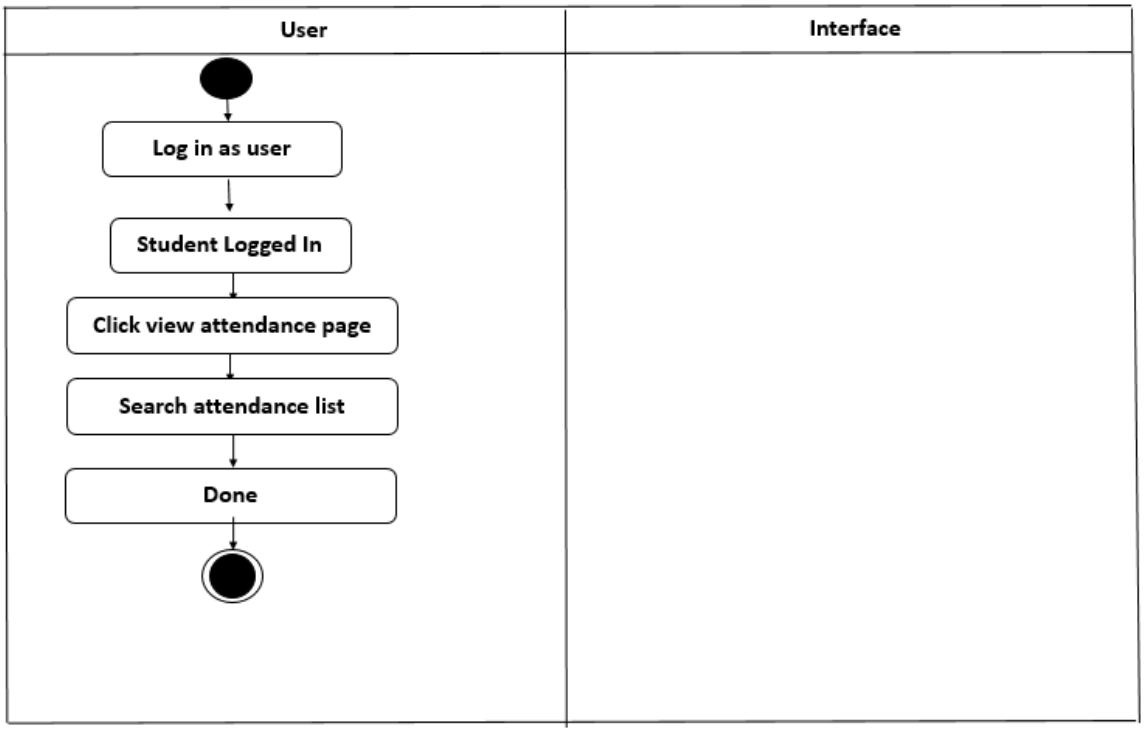


Swimlane diagram for Leave course

View attendance :

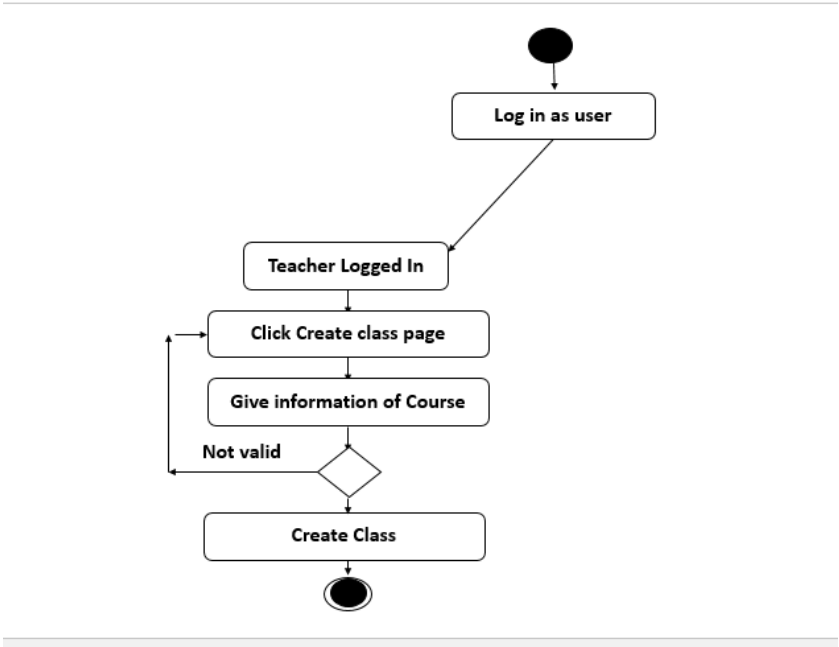


Activity diagram for View attendance

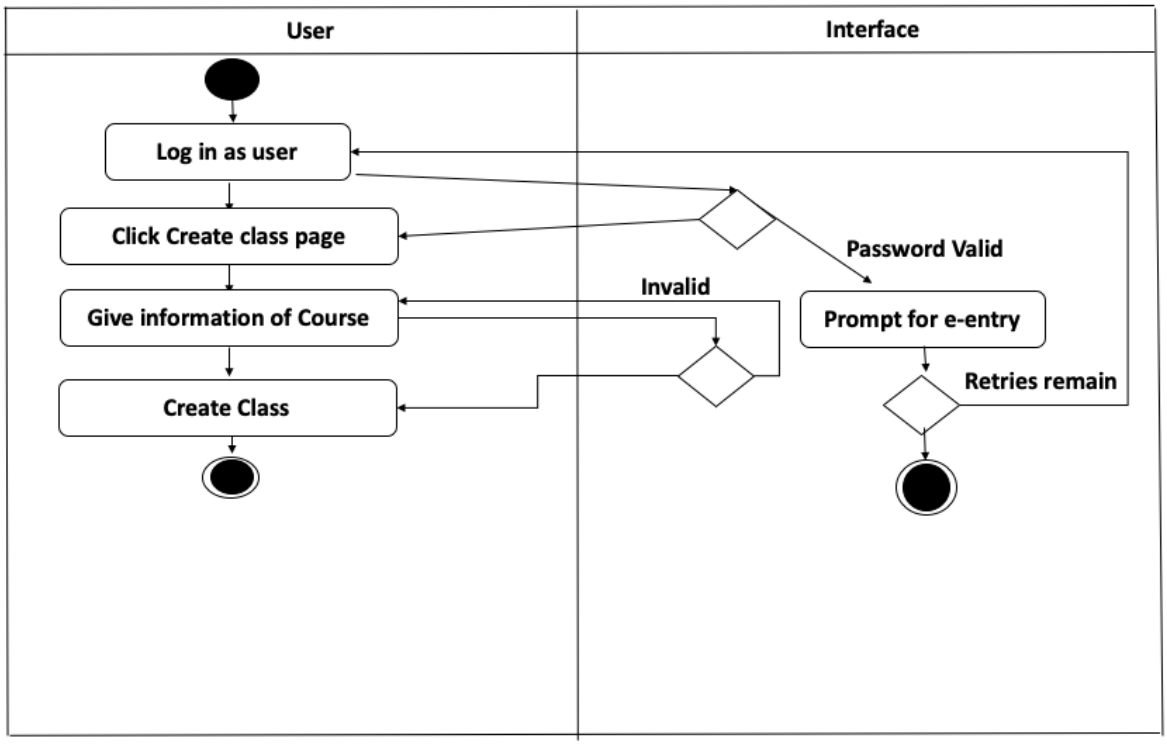


Swimlane diagram for View attendance

Create Class :

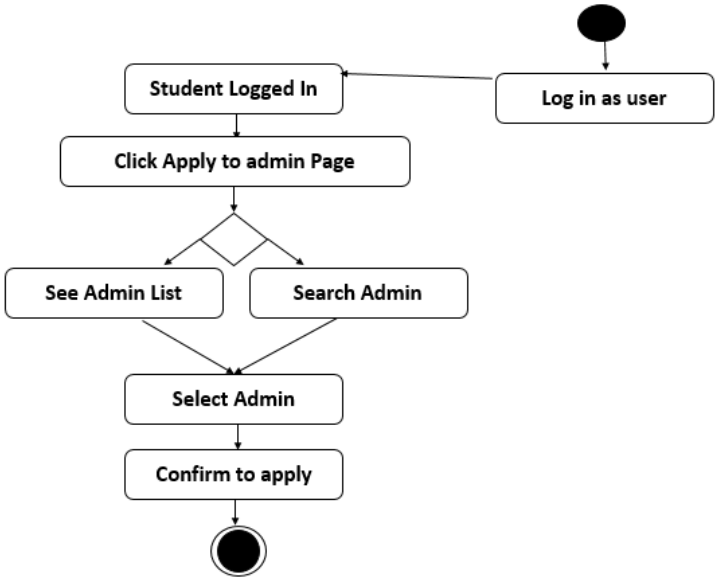


Activity diagram for create class

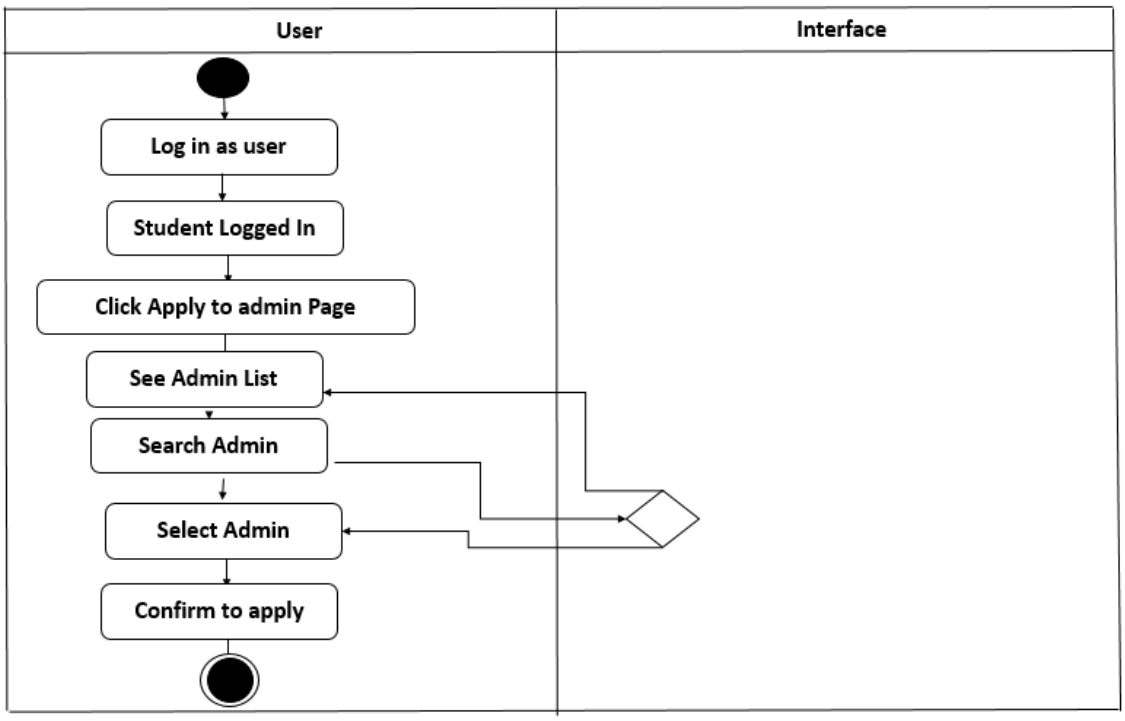


Swimlane diagram for Create class

Apply for Confirm :

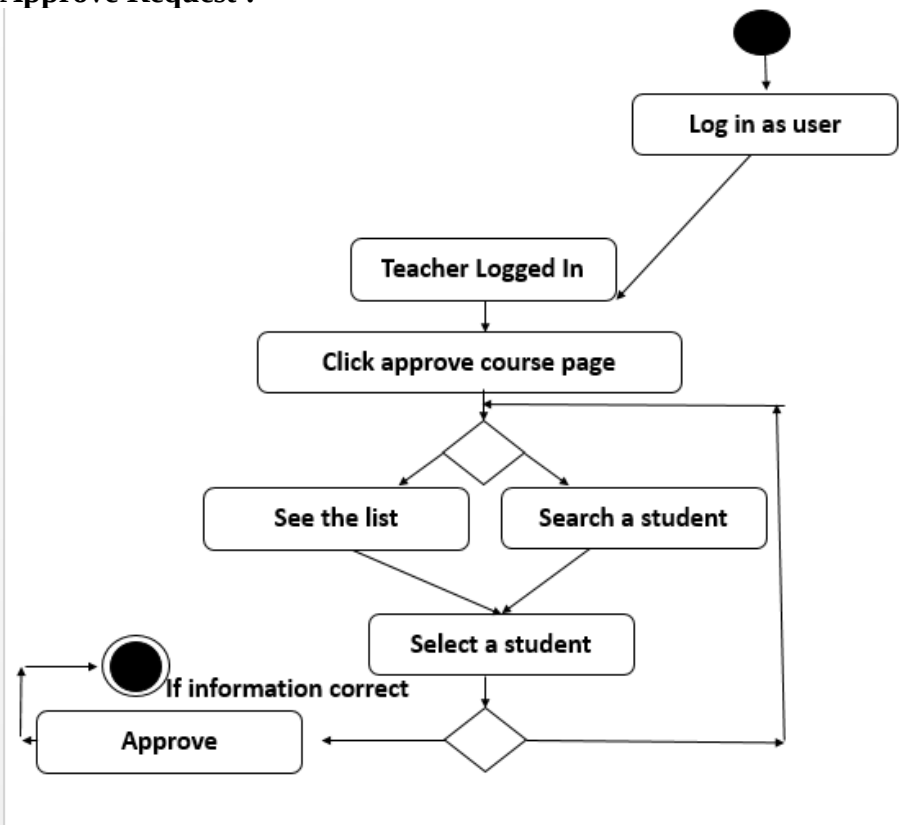


Activity diagram for Apply for Confirm

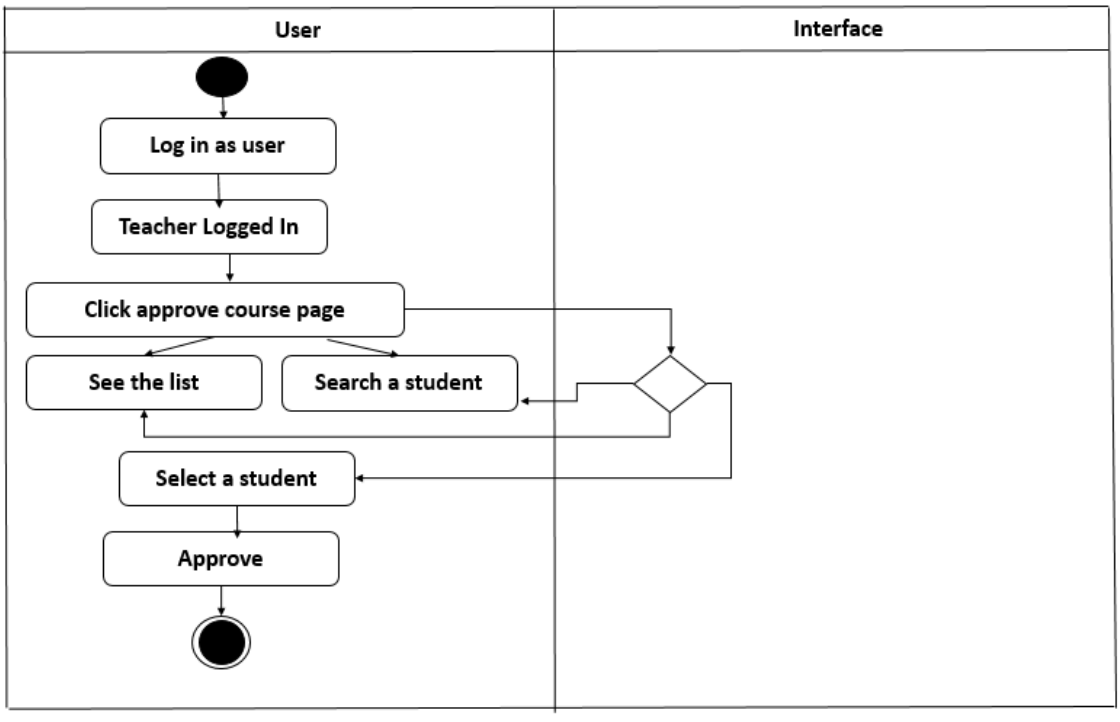


Swimlane diagram for Apply for confirm

Approve Request :

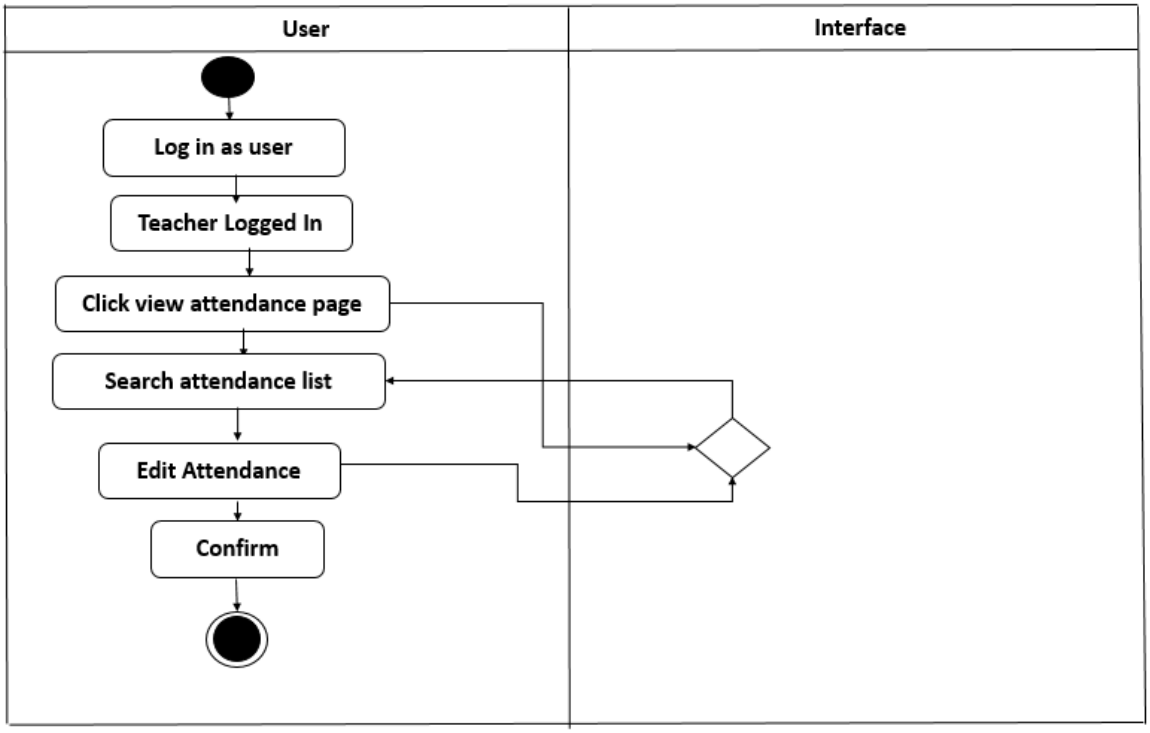


Activity diagram for Approve Request



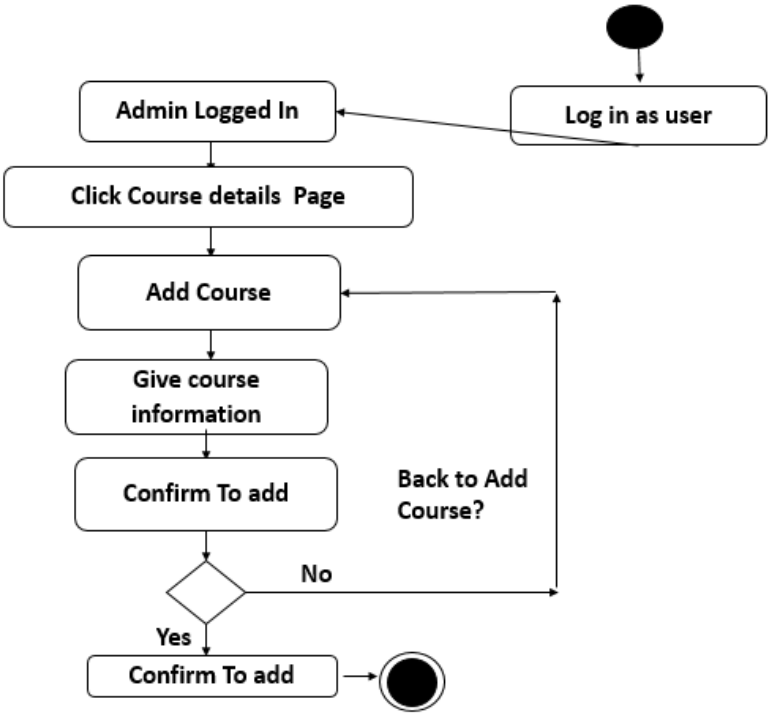
Swimlane diagram for Approve Request

Update attendance :

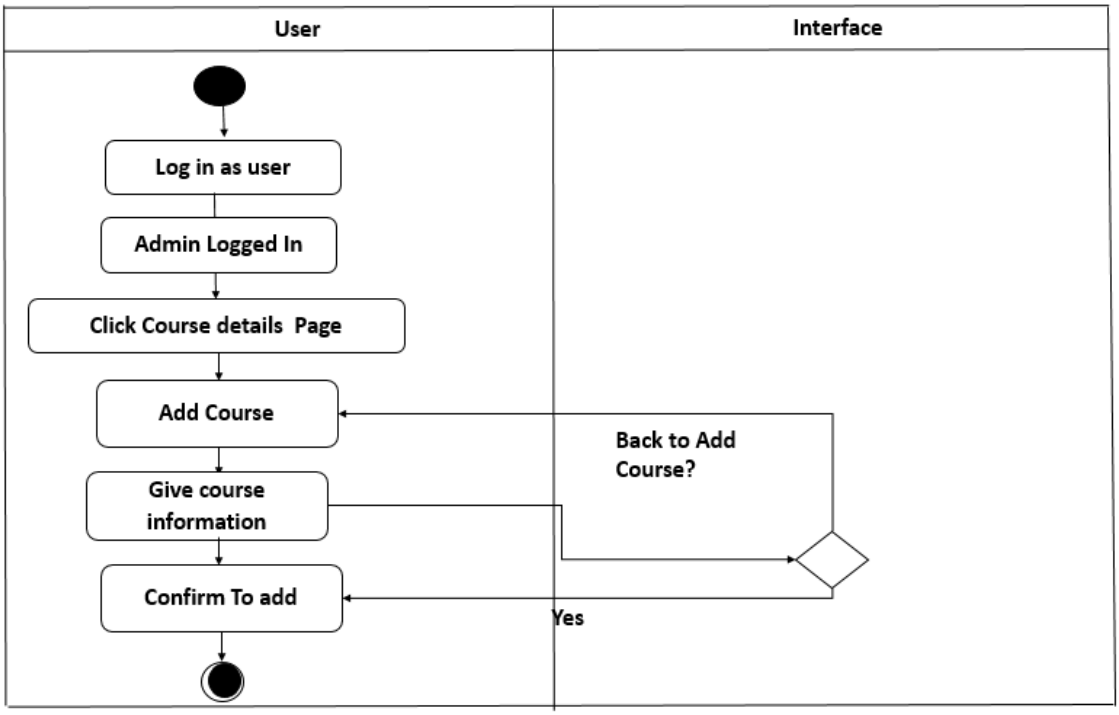


Activity diagram for Update attendance

Add course :

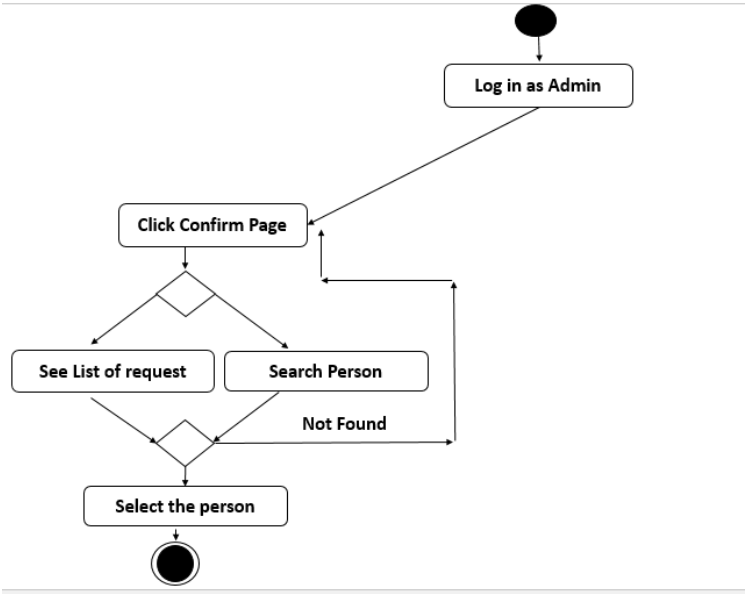


Activity diagram for Add course

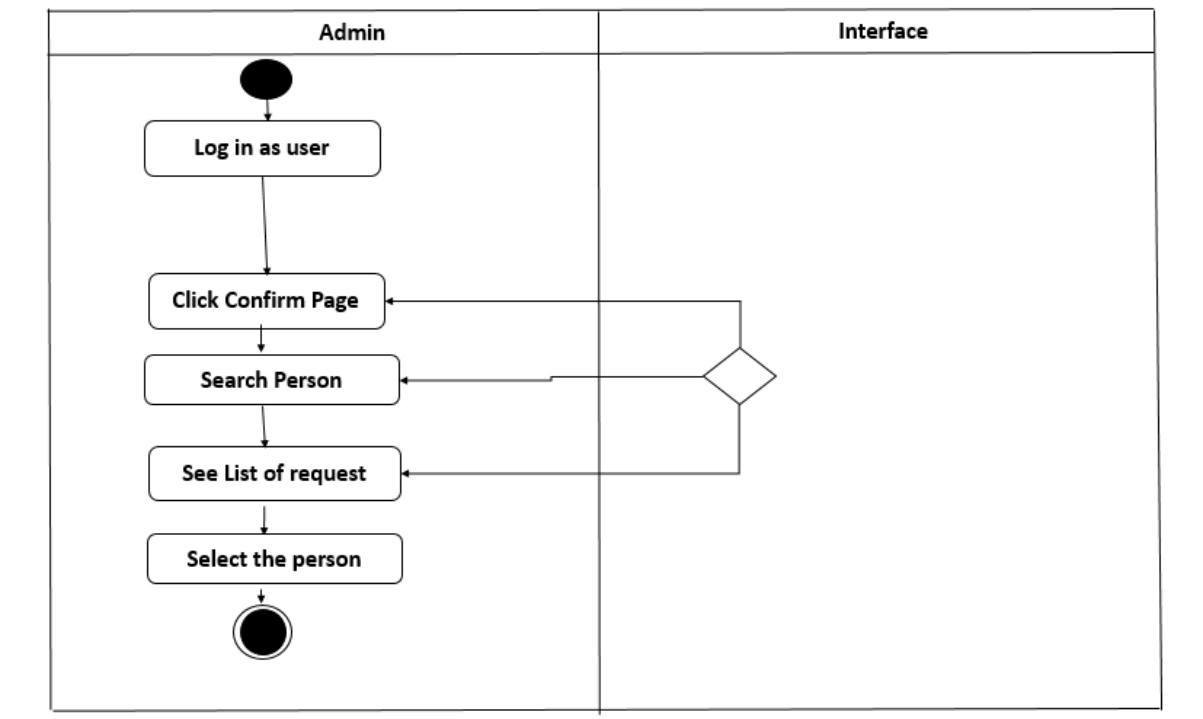


Swimlane diagram for Add course

Confirm request :

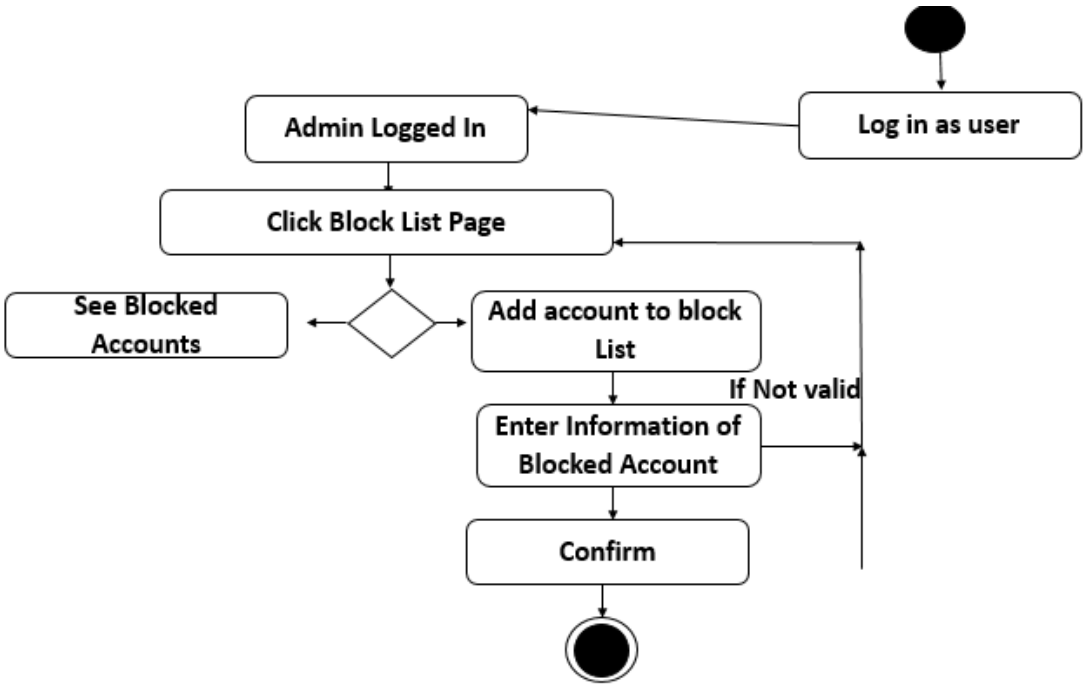


Activity diagram for Confirm request

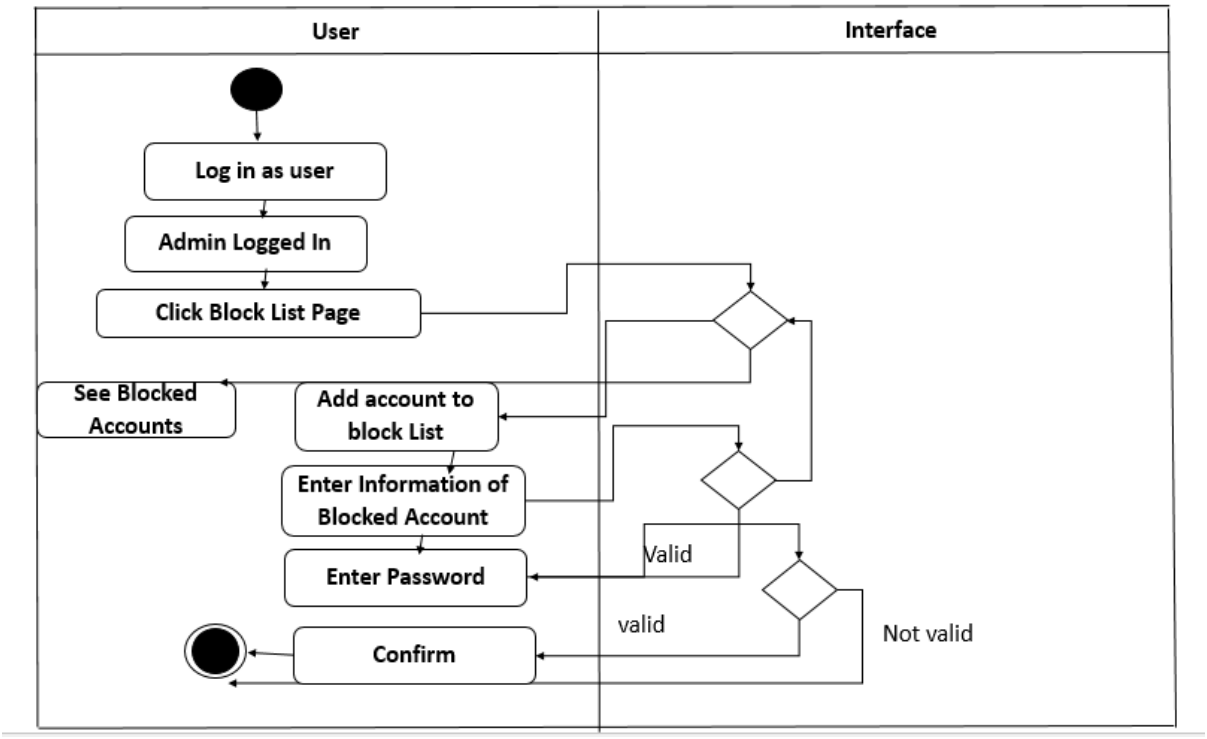


Swimlane diagram for for Confirm request

Block Account :



Activity diagram for Block Account



Swimlane diagram for Block Account

5. Nonfunctional Requirements

5.1 Performance Requirements

The application should have high performance and low failure rates. It manages facilities required by the casual users quickly and easily. Application should have their security not compromised by viruses. All database queries and data receiving/transmitting should be done in higher security transmission.

5.2 Safety Requirements

- In attendance management system, admin must confirm the students and teachers of his department so that any fack students or teachers should not be certified as students or teachers.
- In Doctor Appointment System I have used firebase database. Data is stored as JSON and synchronized in real time to every connected client. Firebase backups all the data instantly.

5.3 Security Requirements

All the administrative and data entry operators have unique logins so that system can understand who is logged in to system right now. No intruders are allowed except system administrative who can change any informations. Because, here many information of students, teachers, admins. So only the legitimate users are allowed to use the application.

5.4 Software Quality Attributes

- **Reliability:** Good validations of user inputs will be done to avoid incorrect storage of records.
- **Availability:** The system will be available all the time.
- **Maintainability:** During the maintenance stage, SRS document can be referred for any validations.
- **Portability:** This system can be installed in any iphone.
- **Flexibility:** The system keeps on updating the data according to the transactions that takes place.
- **Security**: Security of the system is maintained by giving access to only authenticated user id and password.

6. Analysis Models

6.1 Data Flow Modeling

Data Flow Diagram (DFD):
The DFD takes an input process output view of a system. In the figures, data objects are represents by labeled arrows and transformations are represented by circles.

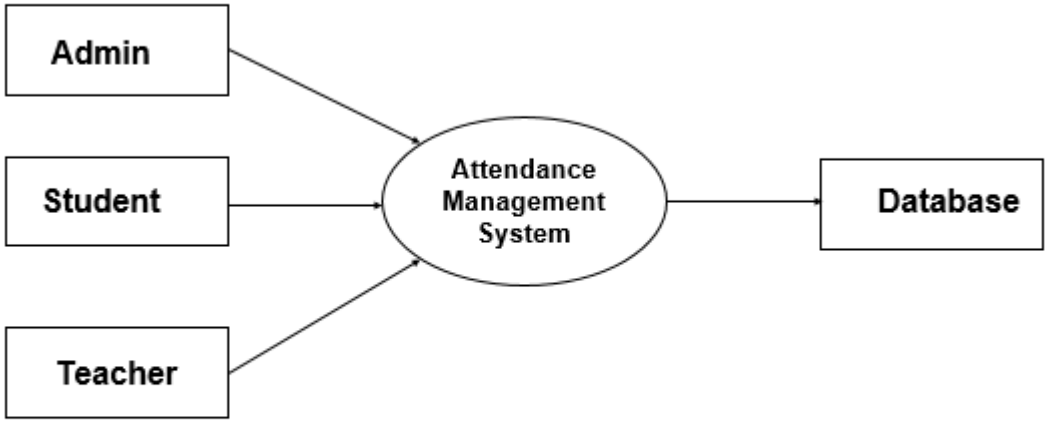


Figure : Level 0 Attendance Management System

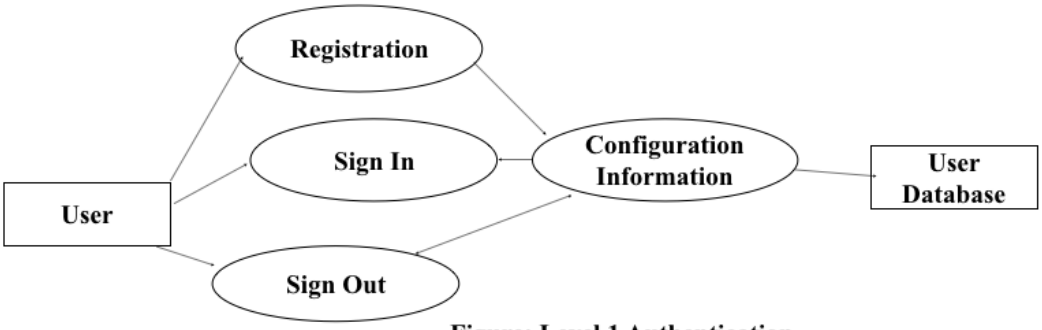


Figure: Level 1 Authentication

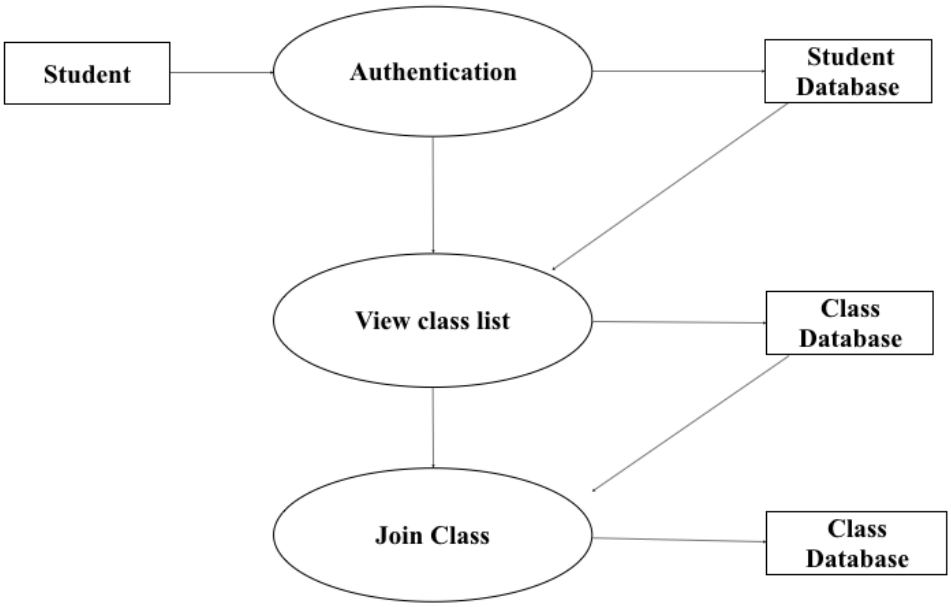


Figure: Level – 2.1 (Student) for Attendance Management System

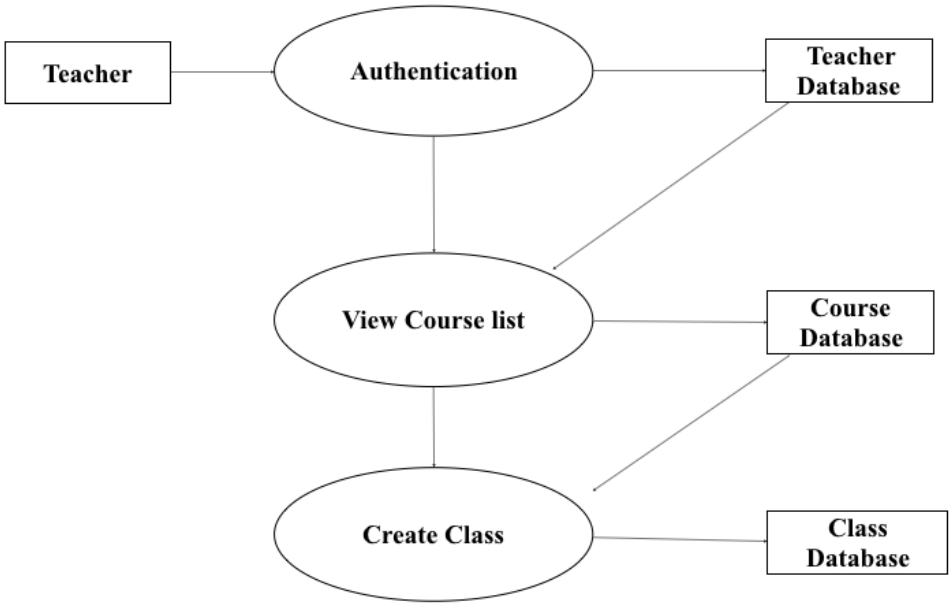


Figure: Level – 2.2 (Teacher) for Attendance Management System

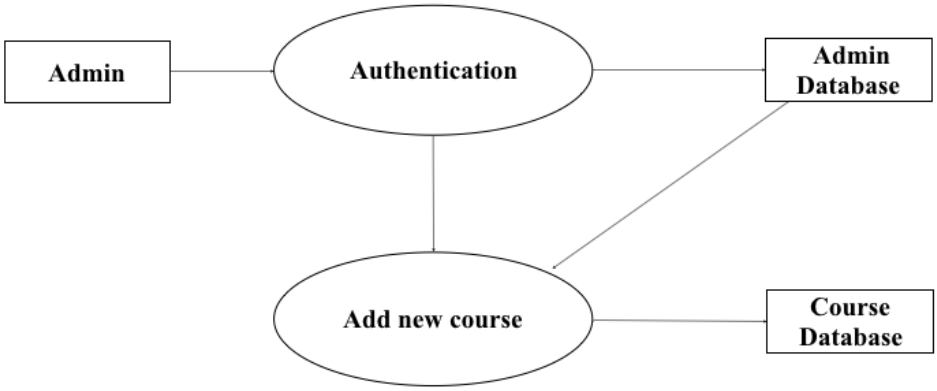


Figure: Level – 2.3 (Admin) for Attendance Management System

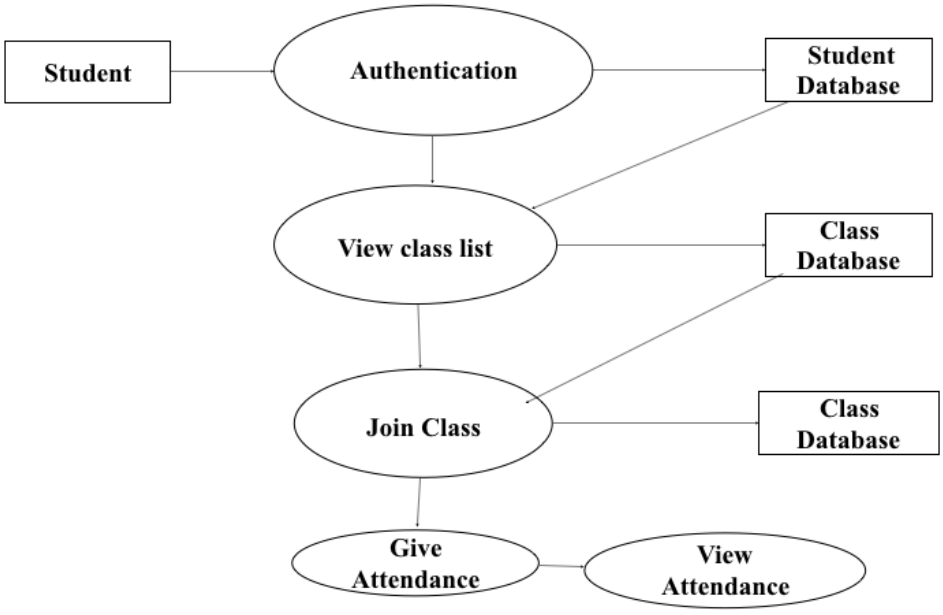


Figure: Level – 3.1 (Student) for Attendance Management System

6.2 Data Modeling

Entity Identification:

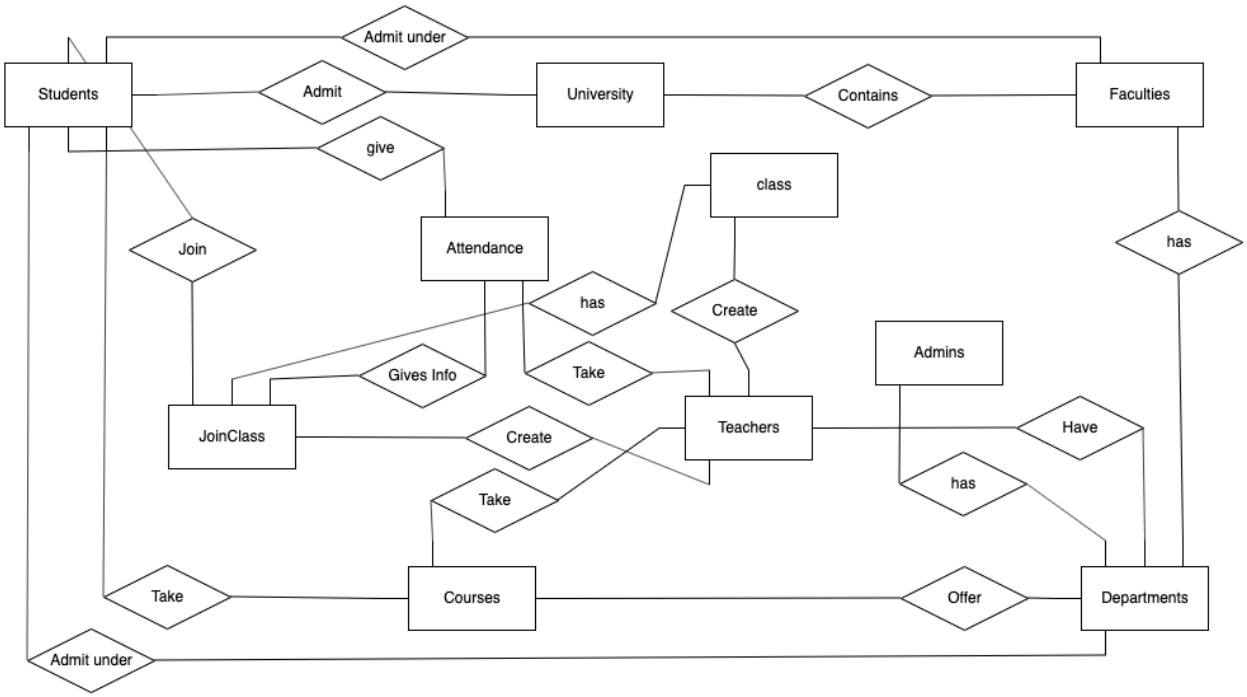
The entities of the scenario is :

1. Universities
2. Faculties
3. Departments
4. Teachers
5. Students
6. Admins
7. Courses
8. Attendance
9. Joinclass
10. Confirmation

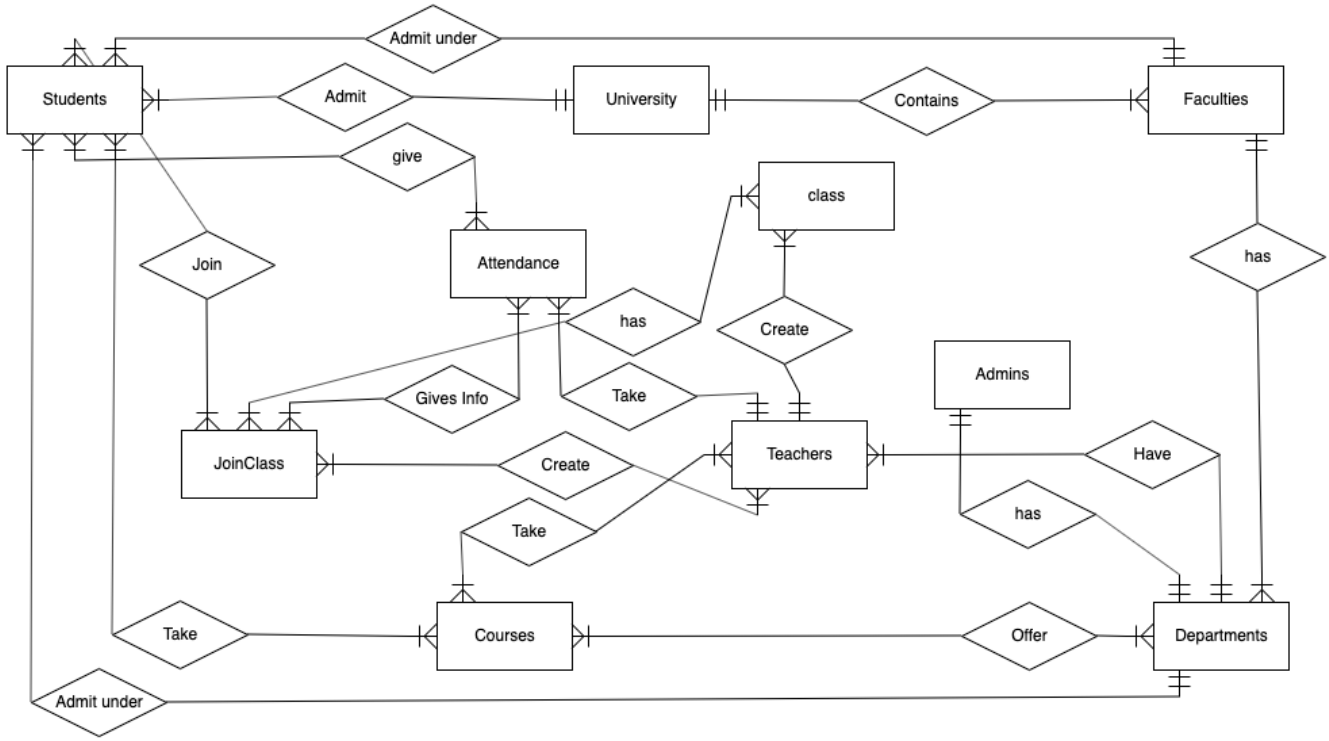
Relationship Identification:

	Universities	Students	Teachers	Faculties	Departments	Admins	Courses	Joinclass	Attendance	confirmation
Universities				**contains**						
Students		**admit**		**Admit under**	**Admit under**		**take**		**give**	
Teachers									**take**	
Admins										**Confirm**
Faculties					**Divided into**	**has**	**offers**			
Departments			**Have**							
Courses										
Joinclass								**Give info**		
Attendance										
Confirmation										

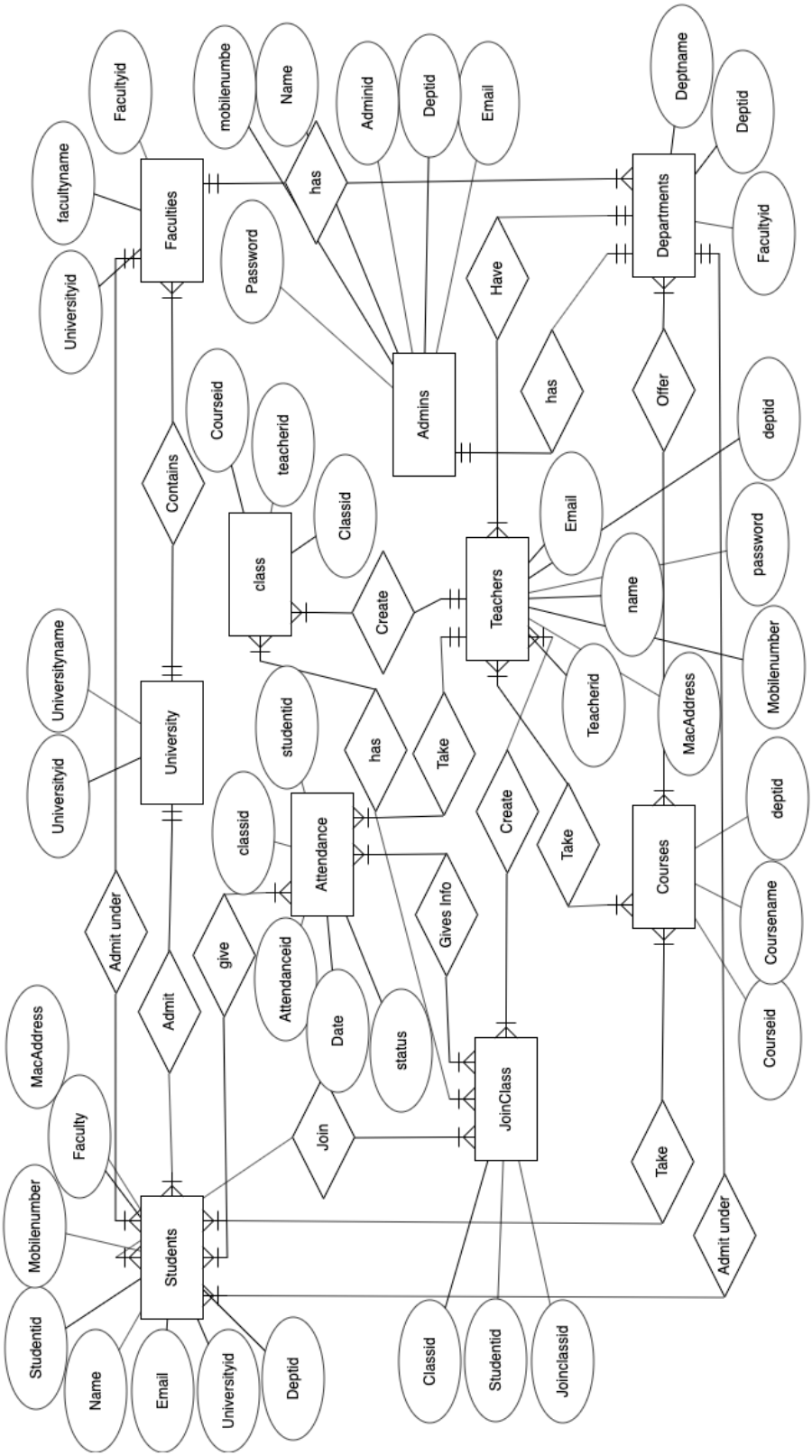
ER Diagram



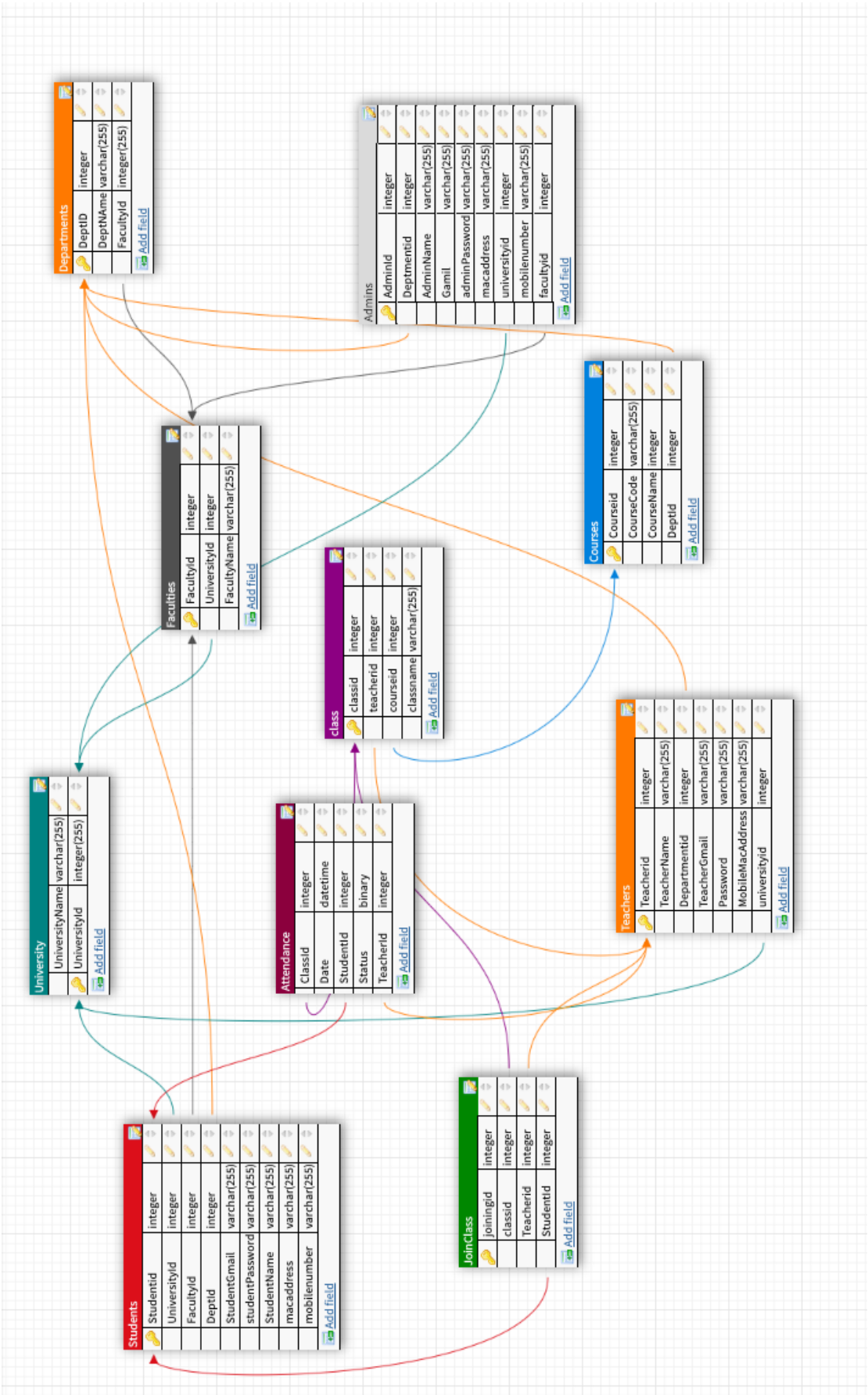
Cardinality Identification :



Create ER Diagram:



Database Schema



6.3 Class Based Modeling

Identifying and categorize all nouns:

External Entities	Students, faculty, Teachers, Admins
Things	Interface, Email, Button, Password, Message, Add course
Occurrence or events	Search, Update information, Apply for class, Apply to Admin, Create Class, Add Courses, Confirm as Student or teacher, Accept Request for class, Block students, Join Class
Roles	Admins
Organizational units	University/College, Department
Places	
Structures	System, Internet, Server, Computer, Mobile

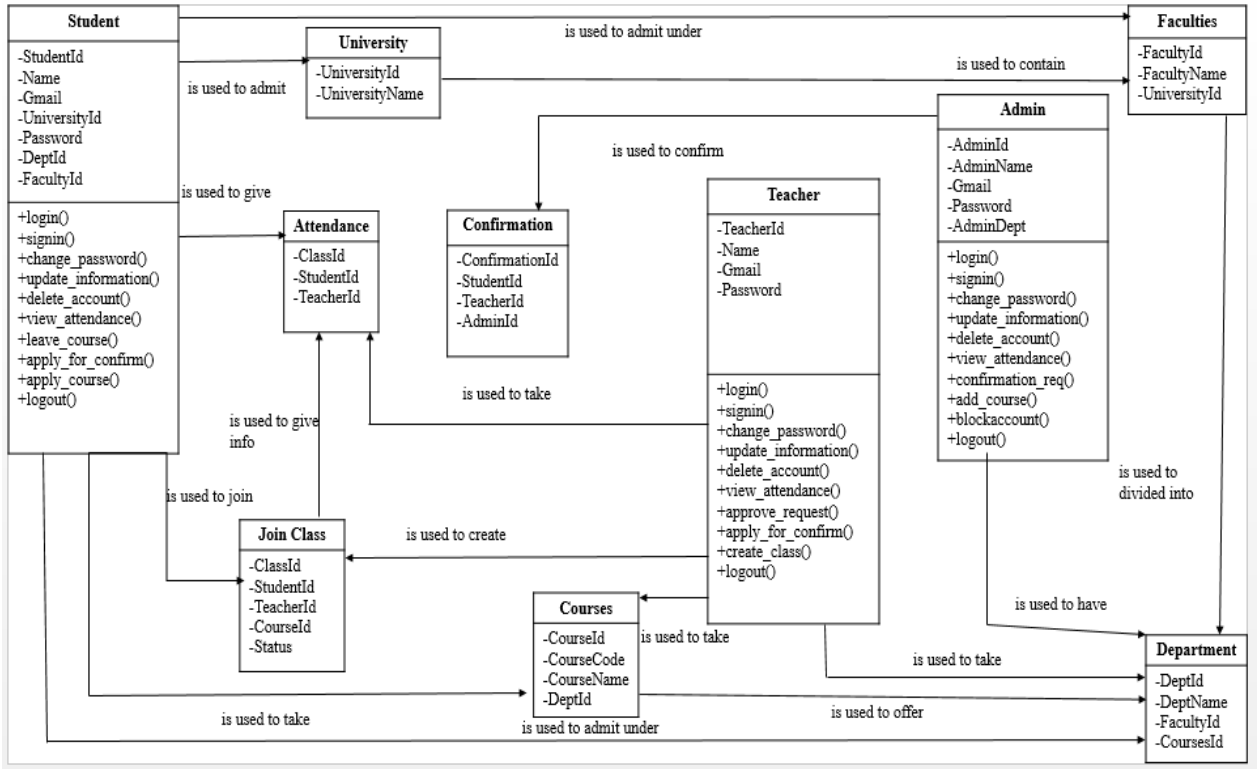
Selection of potential class

Selection characteristics:

1. Retained Information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential Requirements

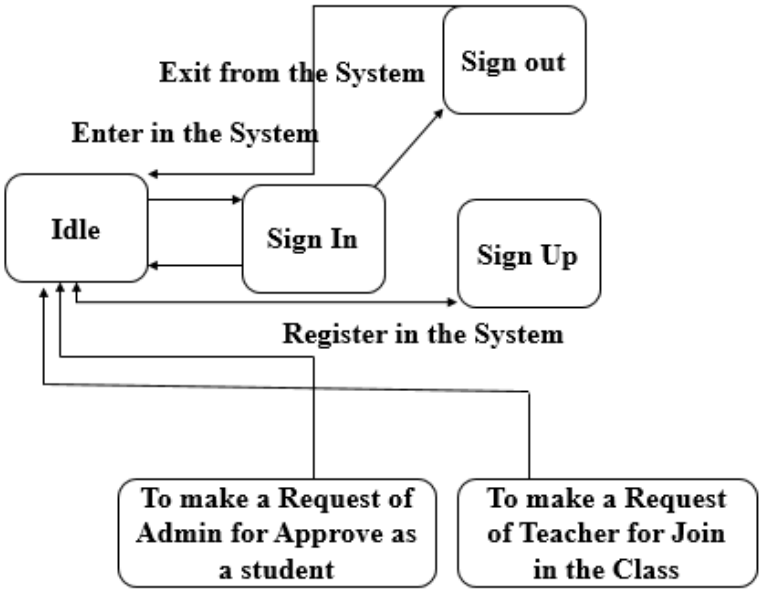
Potential class	**Characteristics Number That Applies**
Students	Apply: 1, 2, 3, 4, 5, 6
Faculty	Apply: 1, 2, 3, 4, 6
E-mail	Apply: 2, 4, 5
Button	Apply: 2, 4
Password	Apply: 1, 2, 4, 5
Teachers	Apply: 1, 2, 3, 4, 5,6
Search	Apply: 1, 2, 4, 5
Join Class	Apply: 1, 2, 6
Interface	Apply: 1, 2, 3, 4, 5, 6
Add course	Apply: 1, 2, 6
Apply for class	Apply: 1, 2, 3, 4, 5, 6
Update	Apply: 1, 2, 4, 5
Apply to Admin	Apply: 1, 2, 6
Create class	Apply: 1, 2, 6
Confirm as Student or teacher	Apply: 1, 2, 6
Accept Request for class	Apply: 1, 2, 6
Admins	Apply: 1, 2, 3, 4, 5, 6
Block students	Apply: 1, 2
Internet	Apply: 1, 2, 6
System	Apply: 1, 2, 3, 4, 5, 6
Computer	Apply: 1, 2, 6
Server	Apply: 1, 2, 6
University/College	Apply: 1, 2, 3
Department	Apply: 1, 2, 3

Class Responsibility Collaborator (CRC)

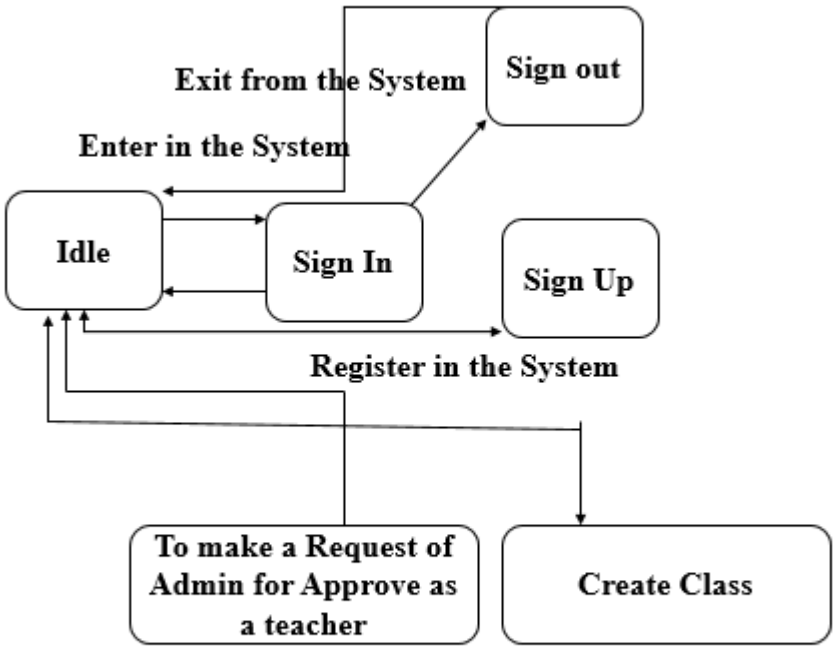


6.4 Behavioral Modeling

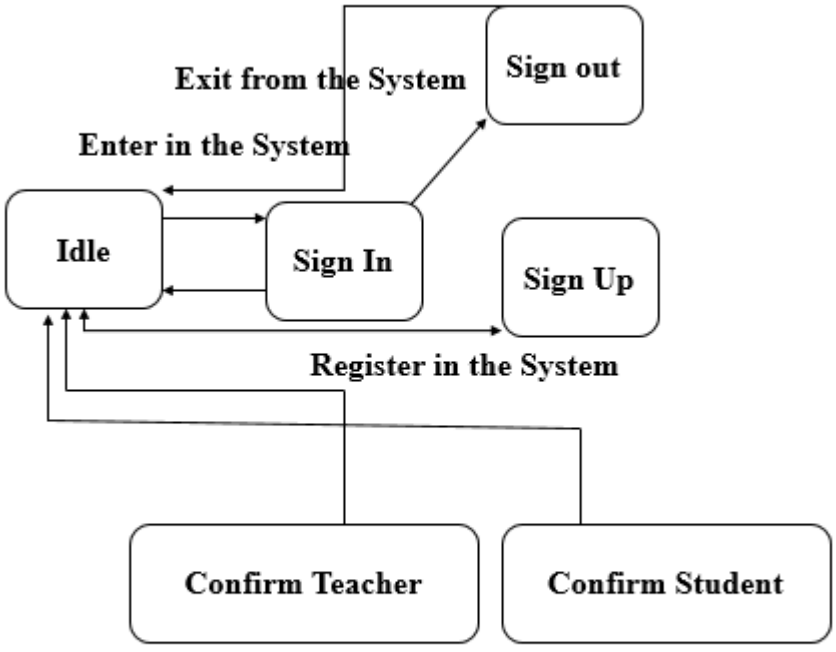
State Diagram



State Diagram: Student Class



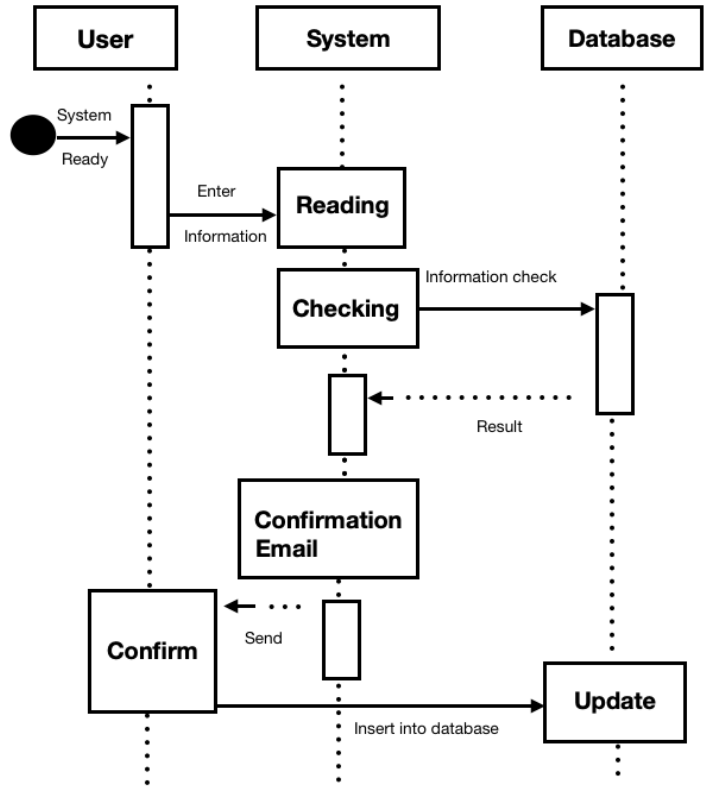
State Diagram: Teacher Class



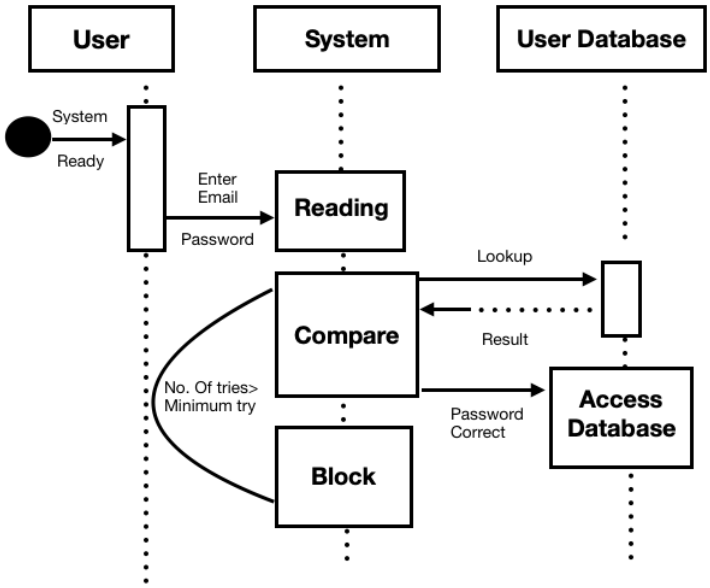
State Diagram: Admin Class

Sequence Diagram

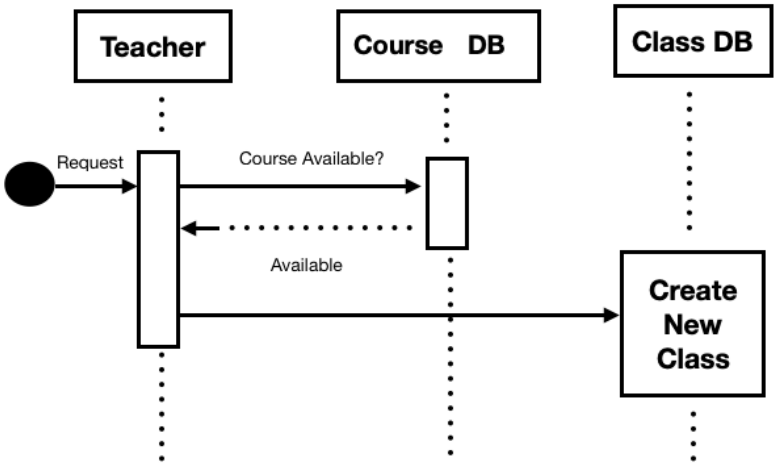
Sequence diagram:



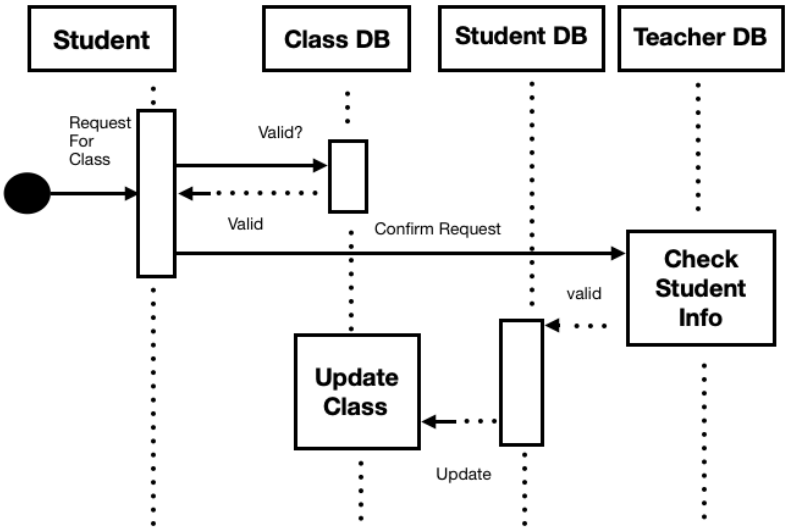
Sequence diagram (Registration)



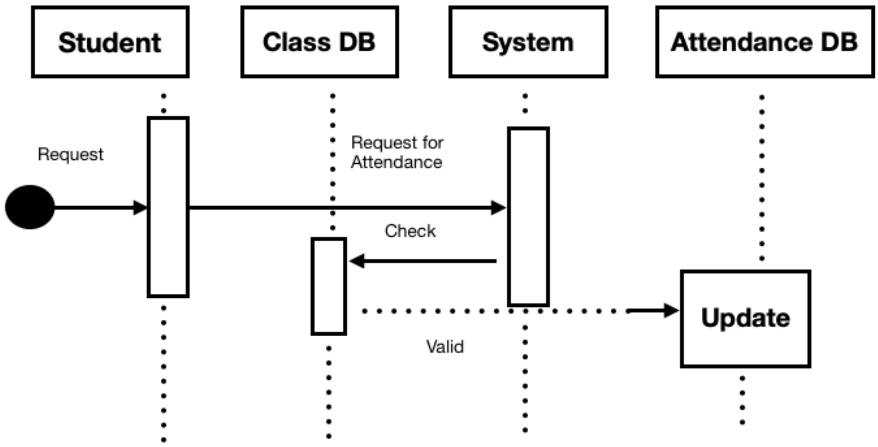
Sequence Diagram (Log in)



Sequence diagram (Creating Class)



Sequence diagram (Joining Class)



Sequence diagram (Attendance)

Appendix A: Glossary

1. **DB:**
    a. **D**ata**B**ase.
    b. Big memory address block which contains large set of data.
2. **CRC:**
    a. **C**lass **R**elationship **C**ollaboration.
3. **SRS:**
    a. **S**oftware **R**equirements **S**tatement.
    b. Statement clarifying the what a software project is supposed to be engineered to doc. Specifies the limits, constraints, and big-picture, abstract plan of the software engineering.
4. **SQL**:
    a. **S**tructured **Q**uery **L**anguage.
5. **DFD:**
    a. **D**ata **F**low **D**iagram.
6. **ER:**
    a. **E**ntity **R**elationship.