Online Personal Counselling System

Table of contents :

Table of Contents

Table of Contents ii

1. Introduction 1

1.1 general description for the project…………………………………………………………………………………………………

1.2 stack holders …………………………………………………………………………………………………………………………………..

2. functional reqirements 2

2.1 User functional reqirements 2

2.2 System functional reqirements 2

3. Non functional Requirements 3

3.1 look and feel 3

3.2 usability&humantiy 3

3.3 performance 3

3.4 maintainbility 3

3.5 cultural…………………………………………………………………………………………………………………………..3

3.6 legal …………………………………………………………………………………………………………………………3

3.7 security……………………………………………………………………………………………………………………….3

4. Diagrams 4

4.1 usecase diagram 4

4.2 usecase description 4

4.1 ER diagram 4

4.2 database tables diagrams 4

4.1 class diagram 4

4.2 activity diagram 4

Sequence diagram ………………………………………………………………………………………………………………………………..

Package diagram ……………………………………………………………………………………………………………………………………..

collaboration diagram……………………………………………………………………………………………………………………………….

5. Others :

Design Patterns ……………………………………………………………………………………………………………………………………………………………

System Architecture …………………………………………………………………………………………………………………………..

Introduction to the project:

In order to improve the facilities related to counselling process, many other virtual counselling schemes came into actuality, which provisions the students in getting essential help about the educational procedures of universities and colleges. Counselling is the process that includes different types of activities such as guiding students towards colleges and universities, supporting them to register with their courses, providing full guidance in getting the information related to their college registration process. This online counselling system even gives great support to universities and colleges through which they can totally reduce the paper usage for counselling and guiding the students towards their university courses. Basically this software includes three different modules in which universities can manage their college records, student’s records as well as counselling sections.

Stack holders :

1- Admin : the person who registers people with the system and activates their personal accounts.

2- User : A person who wants advice from the counselor .

3- IT staff : who are responsible for installing and maintaining of the system .

4- Developer : people who develop the system and follow up on the latest developments for the system .

Functional Requirements:

Firstly: User requirements

1. The Counselling System helps students to show all information about faculty, record courses and show their grades and contact with Officials.

2- The Counselling System helps university can manage their college records, student’s records as well as counselling sections and can totally reduce the paper usage .

System Requirements:

* 1. in September students can registrations in the system.
  2. in October students can records courses and choose favorites class.
  3. students can contact with Officials and show their cv.
  4. students know all information about semester’s.
  5. in January students know time of exomes and their tables.
  6. students know their results and make a complaint.
  7. Faculty members can fully record data on the system in order to avoid mistakes, save time and effort, and also add all college records.
  8. The university can amend the academic courses by adding some courses, removing some of them, or adding new departments.
  9. The system will help the university in counting the number of students, determining the start and end dates of the semester, and easily determining the dates of the exams.
  10. Communicating information about the semester and the course plan in an easy and guaranteed way through the system.
  11. Facilitate the search for students and courses.

Non Functional requirements :

(1Look-and-Feel Requirements :

1- The system shall use a large range of exciting sounds.

2-The system should use a lot of animation.

3- The system shall use a lot of color.

(2Usability & Humanity requirements :

1- The system shall be easy to use by member and should be organized in such away users error should be minimized .

2-The search for college or university should be easy should take from 10 to 30 second.

3-Easy to managing whole counseling procedure it should take 30 minute after 2 hour training .

4-Users must change the initially assigned login password immediately after the first successful login. Moreover, the initial should never be reused.

(3Performance requirements :

1-Easy to managing whole counseling procedure it should take 30 minute after 2 hour training .

2-A website should be capable enough to handle 1000 users with affecting its performance .

3-User can easy take test on general knowledge questions and select correct answer for the asked question within 20 second .

4-users and Councilor can write a feedback message to notify the admin about the working of the system take 400 M byte .

5-the system helps the university to dynamically add records to the databaseWithin 30 second .

6-Admin can add new questions by specifying their options and the correct answer within 20 second .

7-Time to restart after failure for system should be 30 second .

8- The system should keep every id and password safe and hidden .

4 ( Maintainability:

* The system shall be able to be modified to cope with minor changes to supreme council of universities.
* The system shall be able to handle more than 5000 users at the same time within the first year after its publishing.
* The system should be able to support another language with its mother language within one year of publishing .
* The system may be require to be temporarily shutdown and new modules as (ex: php classes ) to be added to the system or it may require a minor update in its graphical design (ex: html – css modules) .

5) Cultural requirements:

* The language that used in the interfaces of the system shall be polite and formal.
* The system shall not use any terms that may harm any one on the planet.
* The system shall use English language in its interfaces.
* System shall not display any violence symbols or words in its interfaces.

6) Legal Requirements:

* The system shall follow restrictions and constraints of the supreme council lf universities.
* The system shall obey to the privacy policy and terms for all users.

7) Security Requirements:

* The system should distinguish between authorized and non-authorized users.

Ex: student only cannot change his grades but only can view it .

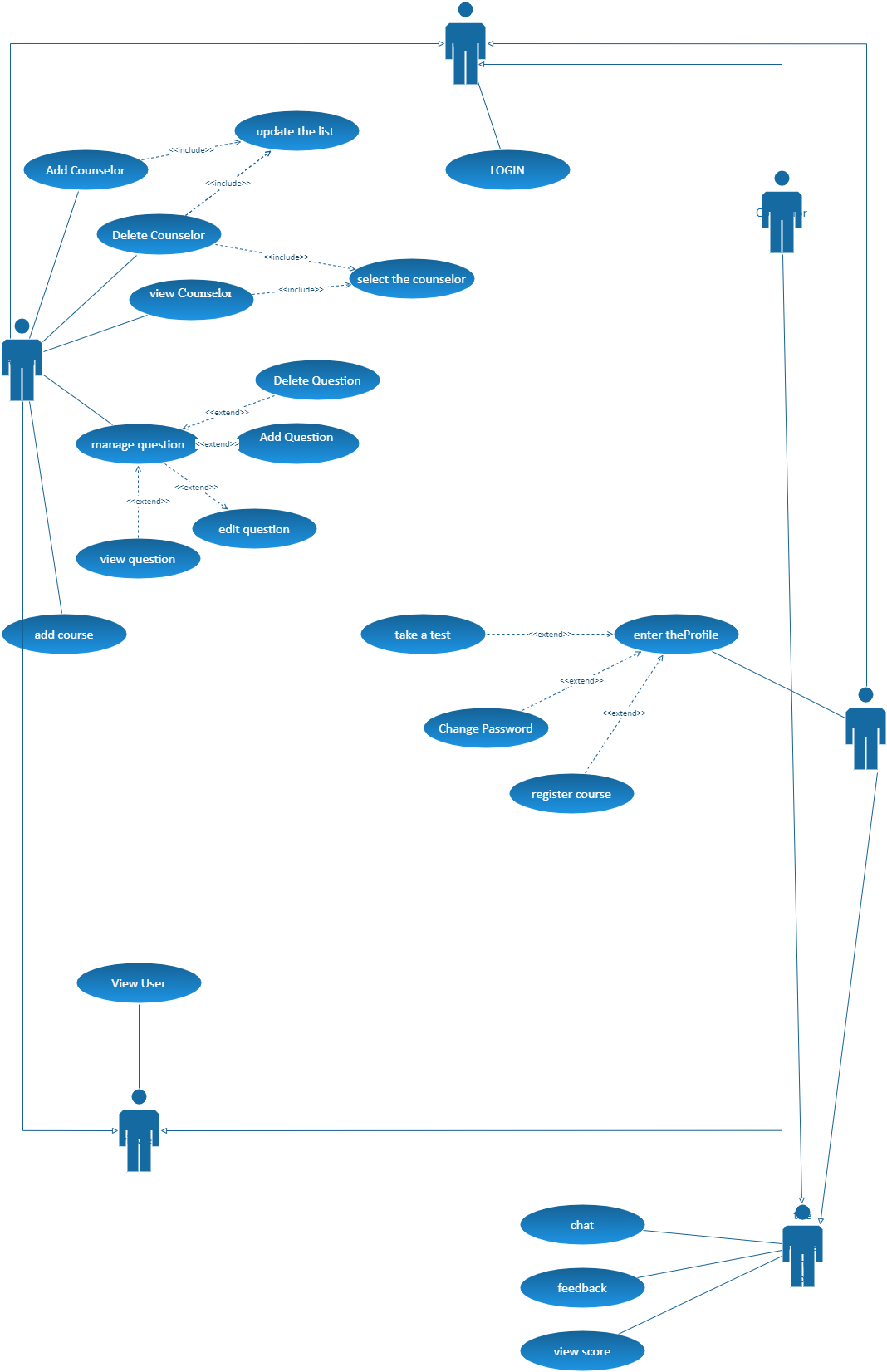
* The system shall maintain information securely and ensure that it is only accessed by authorized users.

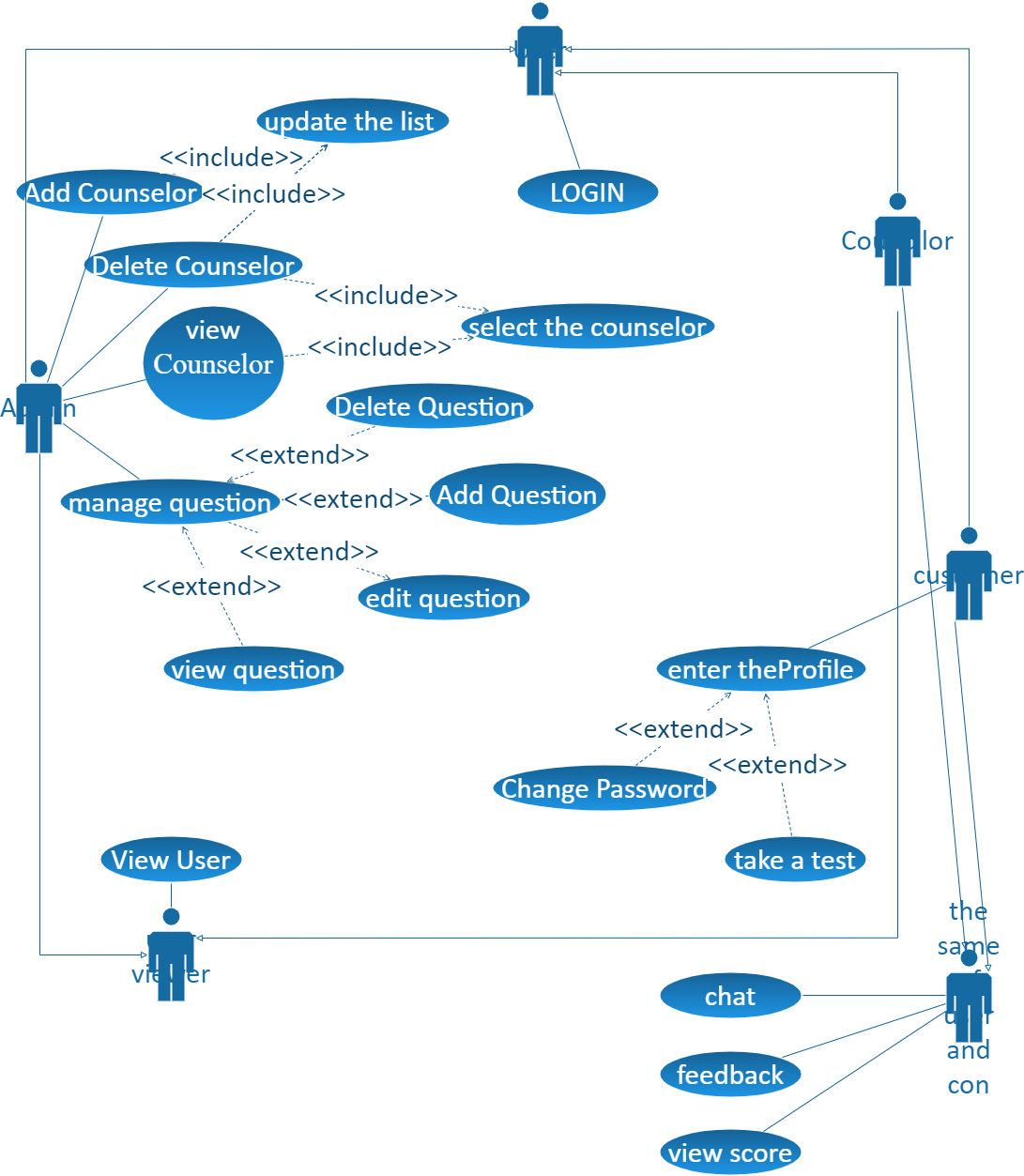
Ex: Counselor cannot view all information about the student but Admin only can do this.

* The system shall provide a facility for users to request personal information and request changes to that information as change password.
* The system shall only allow the transmission of personal users information only to admin and to the user itself.

The diagrams :

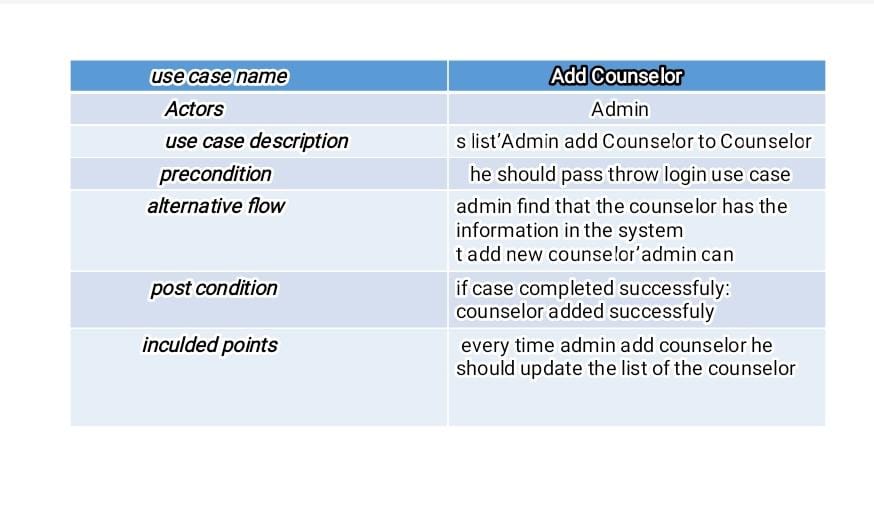
1- Use case diagram :



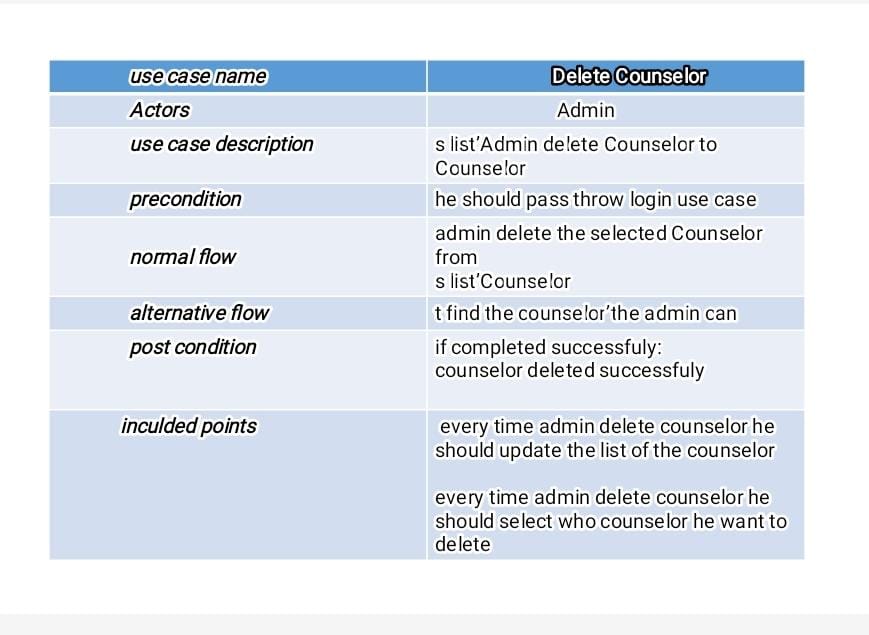


Use case description:

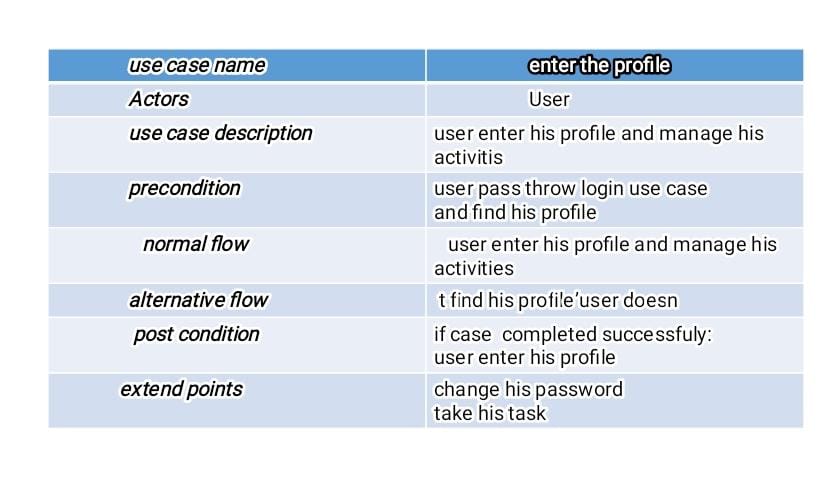
Add counselor :



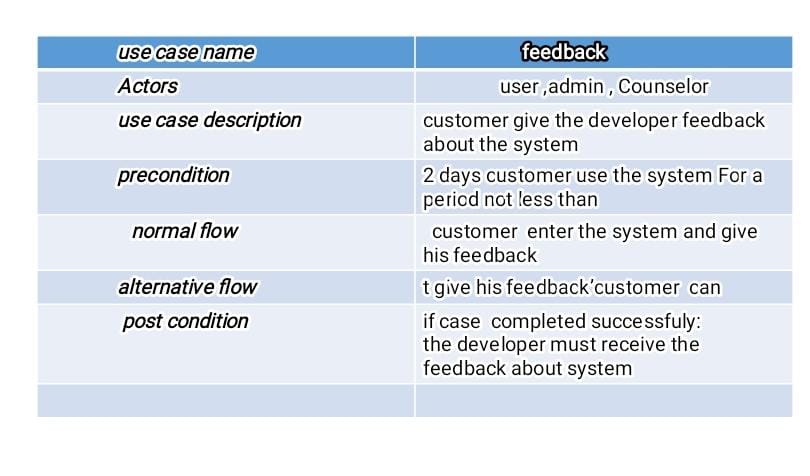
Delete counselor :



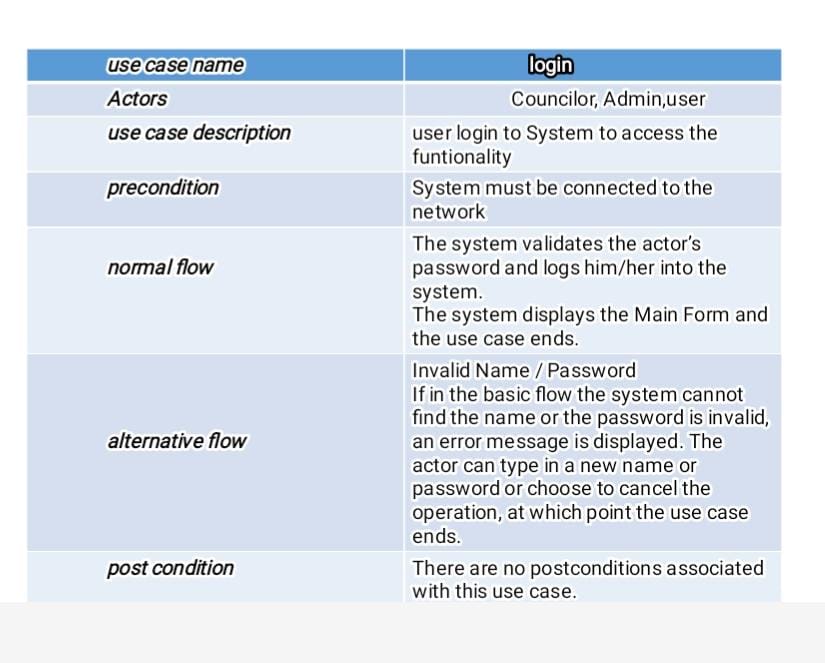
Enter profile :



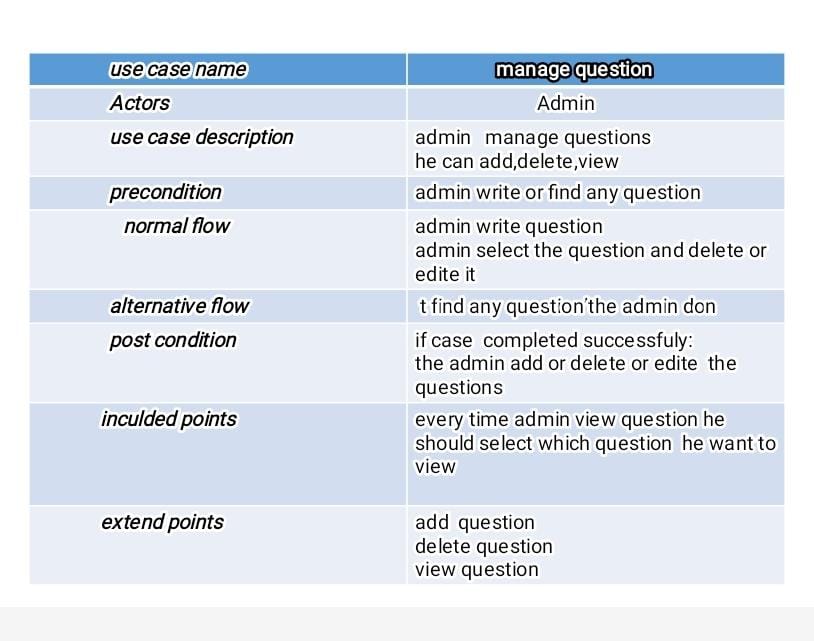
Feedback :



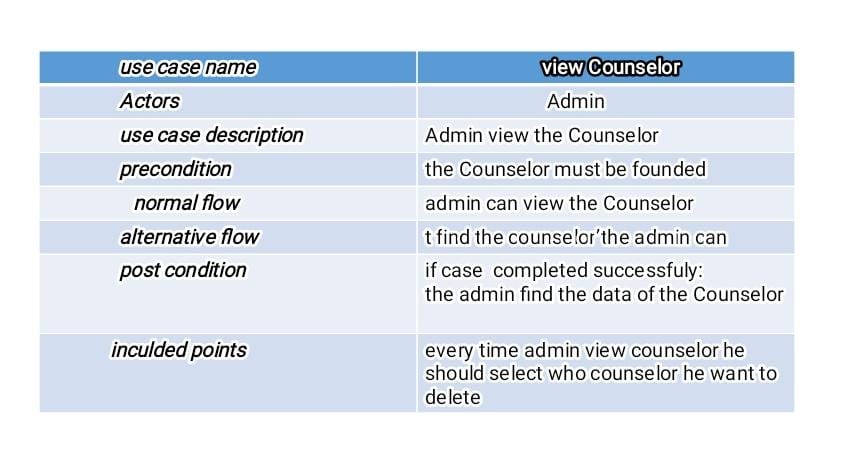
Login :



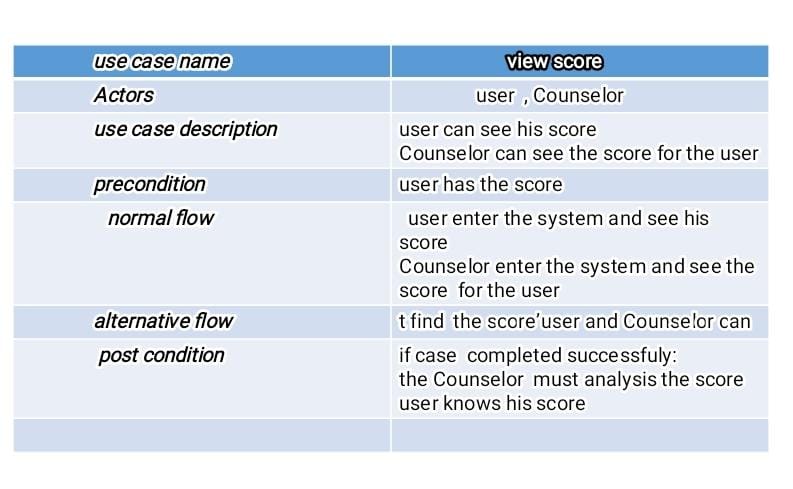
Manage question :



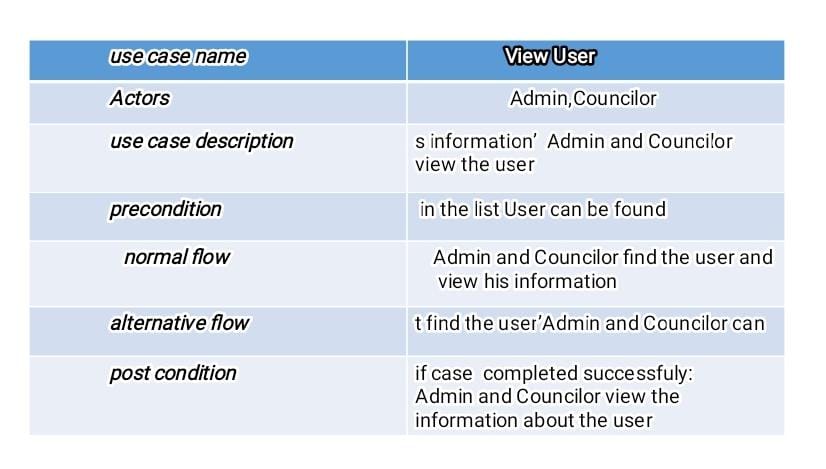
View counselor :



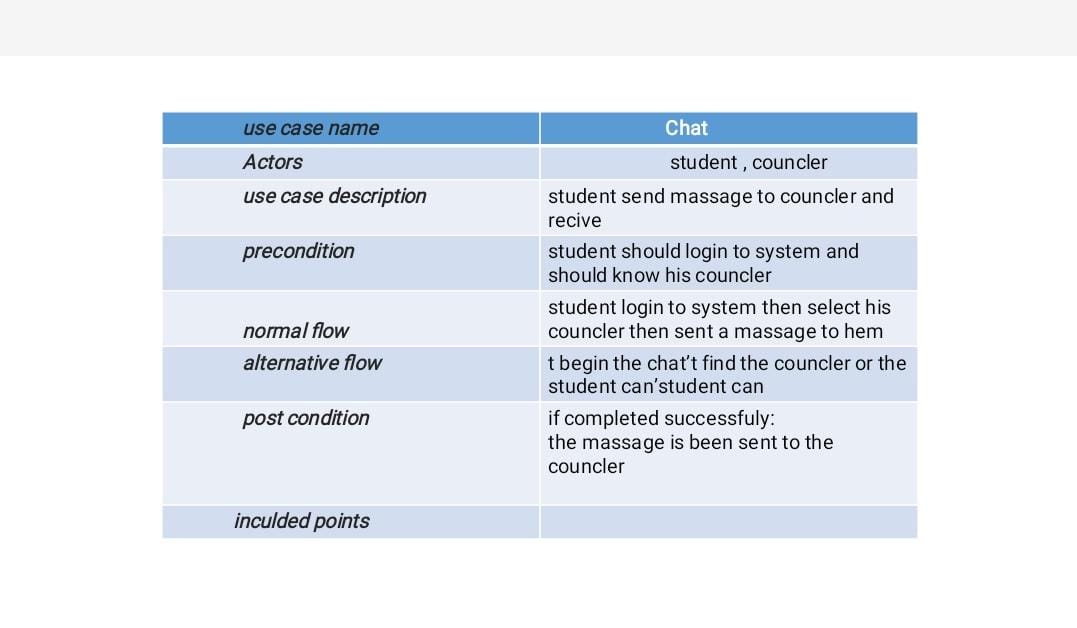
View score :



View user :

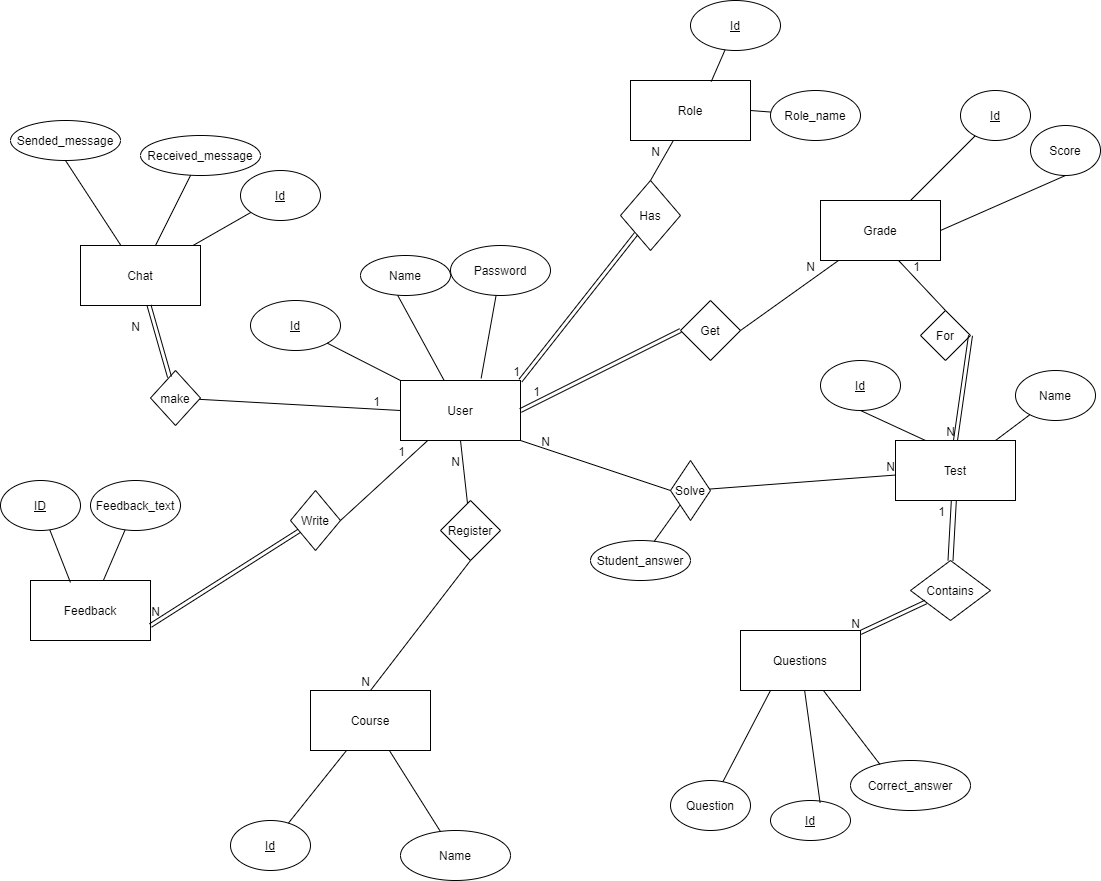


Chat :

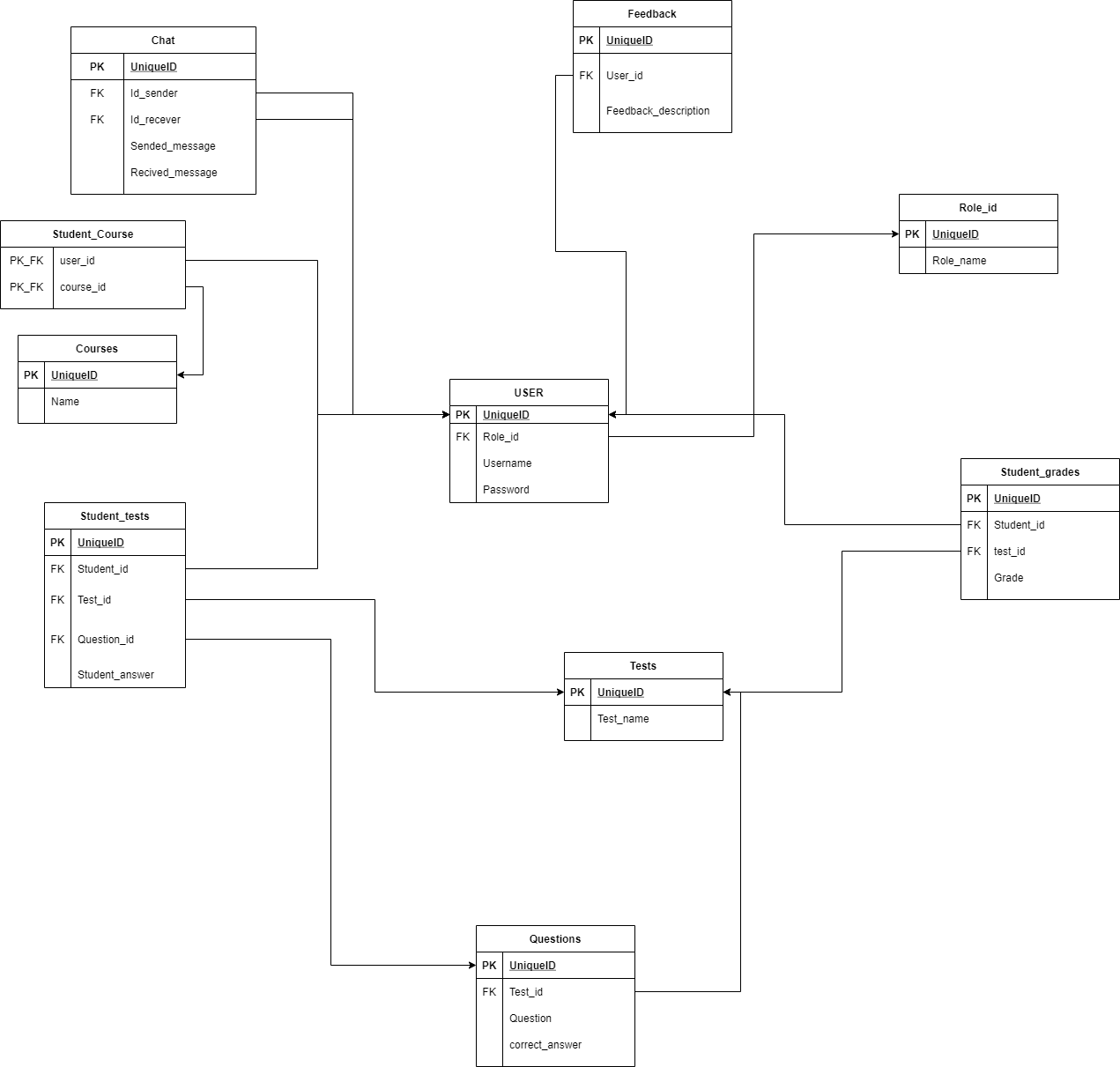


2) Database diagrams :

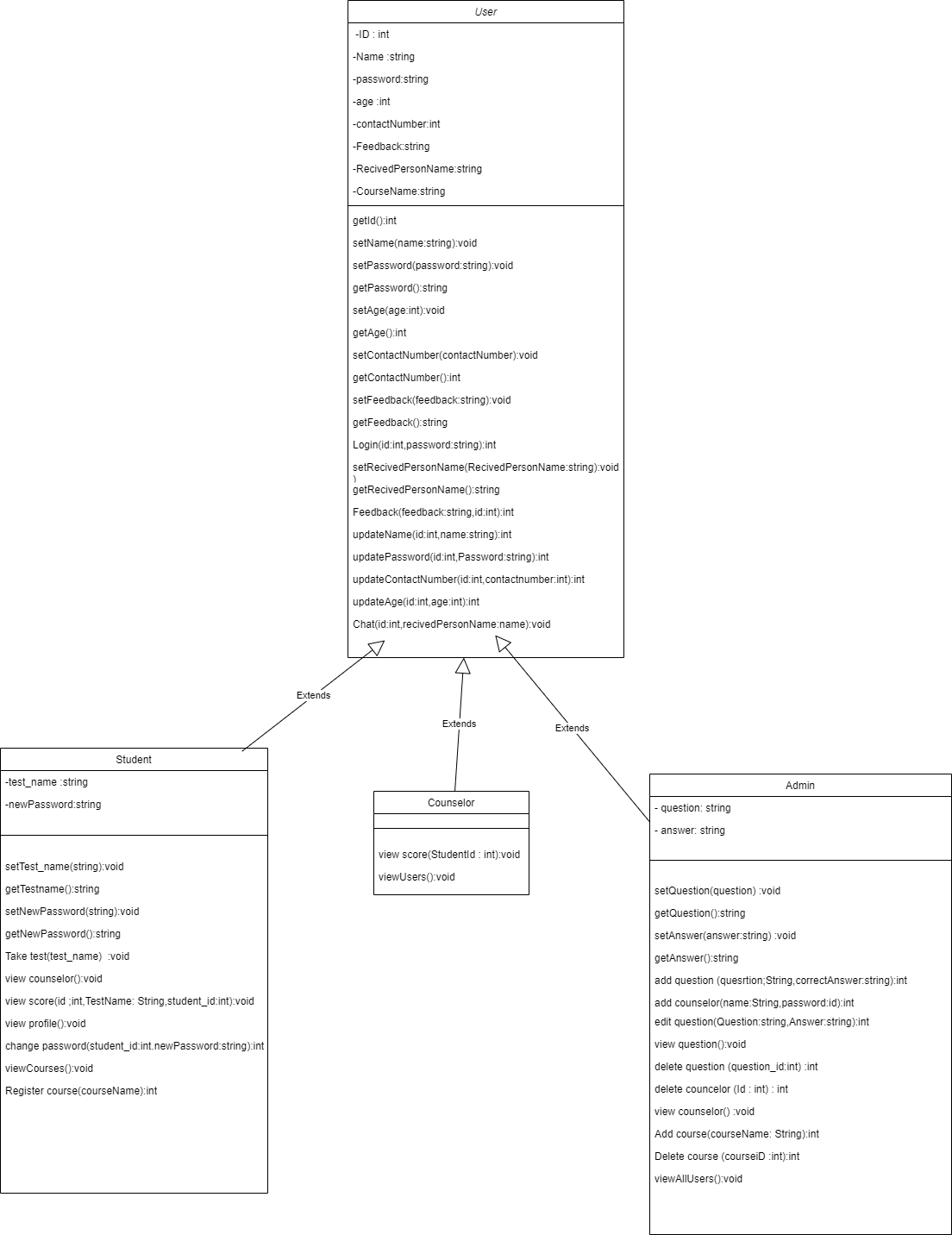
a) ER diagram :



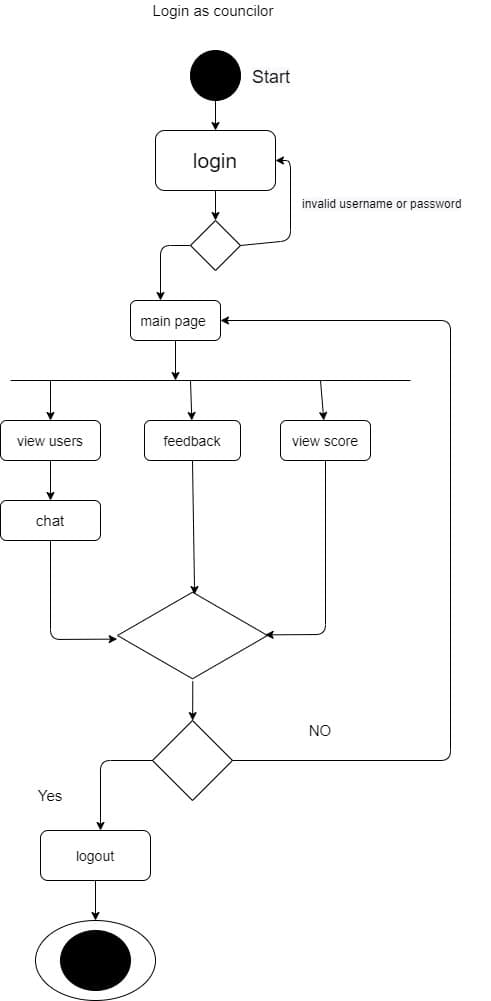
b) database tables diagram :



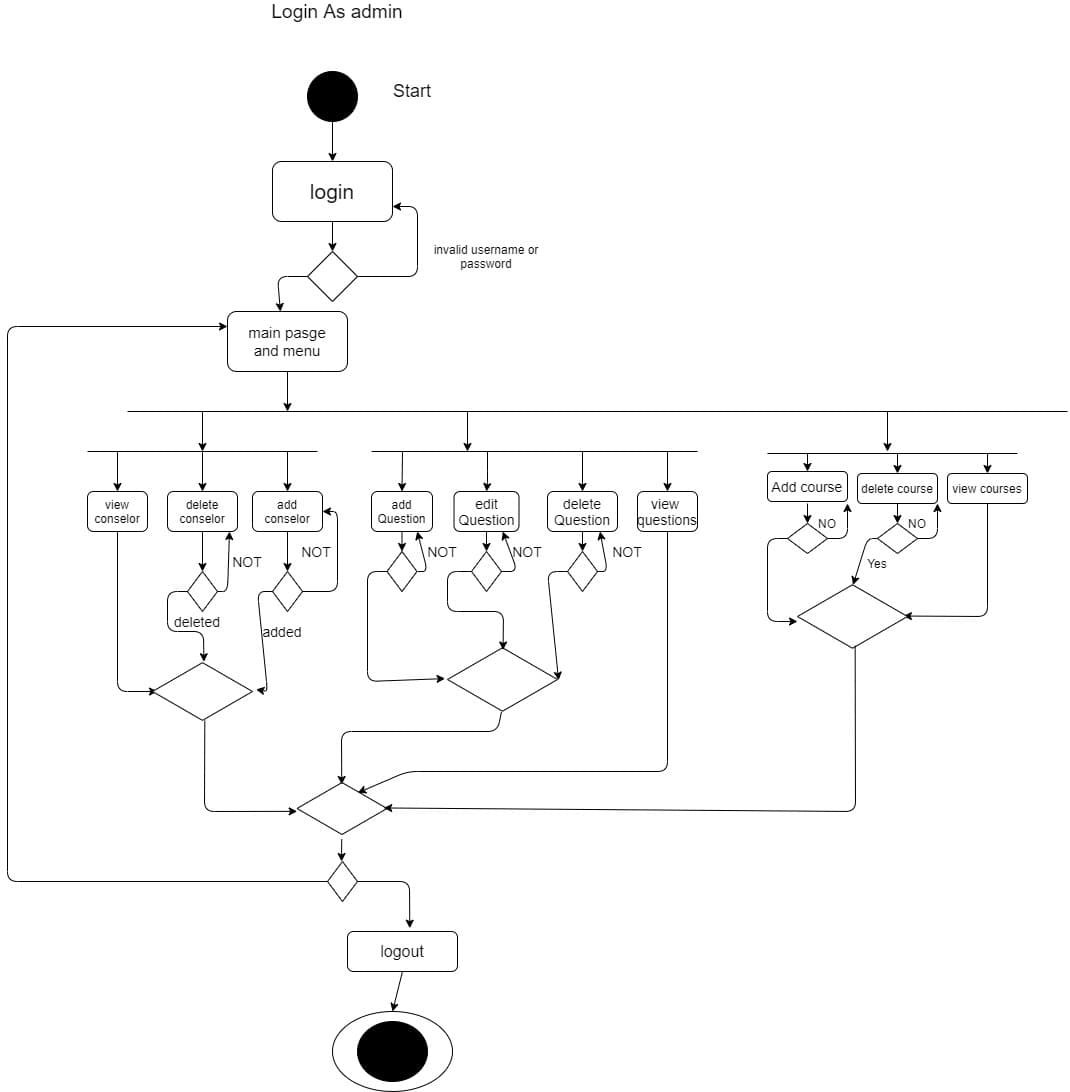
3) Class diagram :



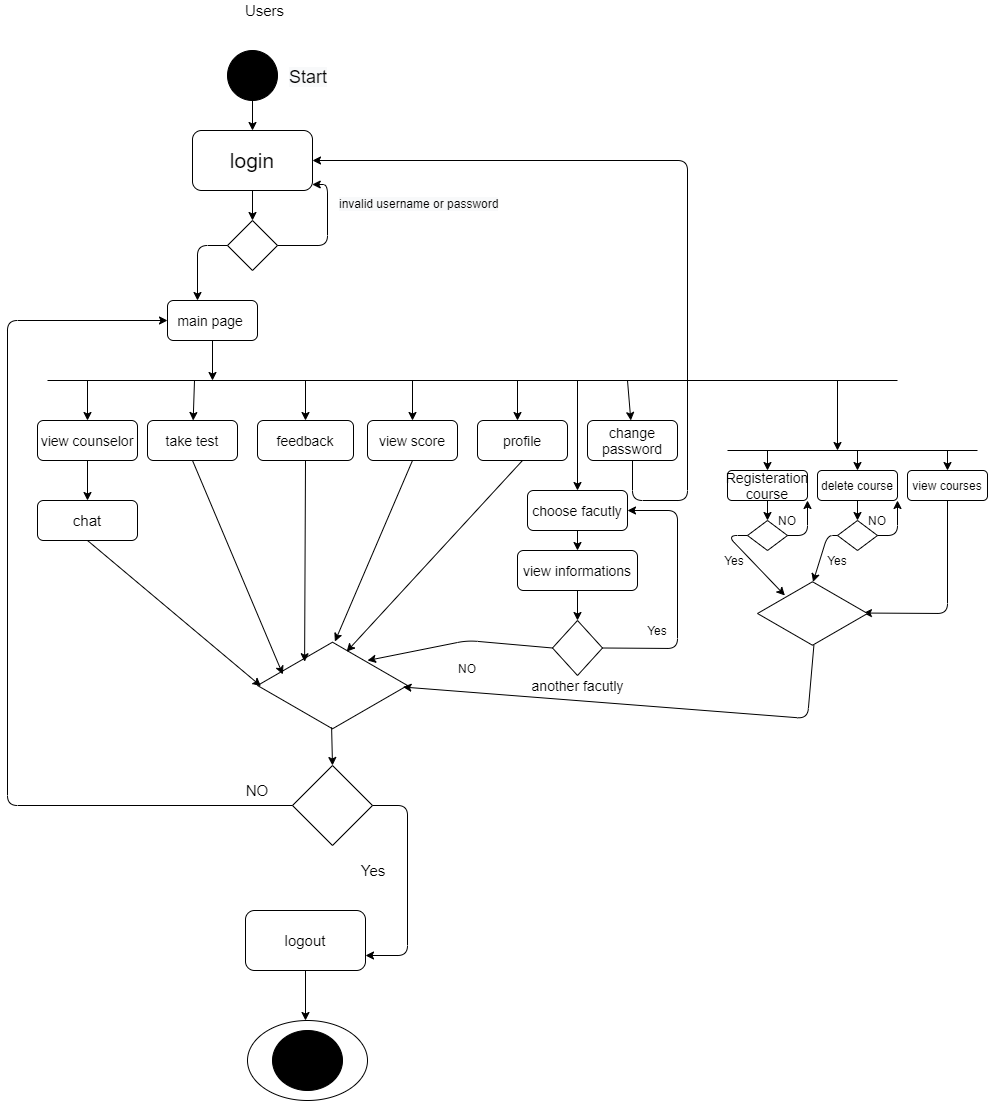
4) Activity diagram : a) login as councelor



Login as Admin :



Login as User :



Design patterns description

1) Singleton pattern (creational pattern) :

Context : It is very common to find classes for which only one instance should exist (singleton) .

Problem : How do you ensure that it is never possible to create more than one instance of a Admin class. And provide a global point of access to it.

Forces : The use of a public constructor cannot guarantee that no more than one instance will be created. TheAdmin instance must also be accessible to all classes that require it, therefore it must often be public.

Solution : Have the constructor private to ensure that no other class will be able to create an instance of the class Admin. Define a public static method called “GetAdminInstance ()”, The first time this method is called , it creates the single instance of the class “Admin” and stores a reference to that object in a static private variable called “theAdmin”.

Example : Applying on Admin Class

|  |
| --- |
| Admin |
| theAdmin:Admin |
| -Admin():void  +GetAdminInstance():Amin |

If (theAdmin==null)

theAdmin=newAdmin

return theAdmin;

2) Immutable pattern (structural pattern):

Context: An immutable object is an object that has a state that never changes after creation.

Problem: How do you create a class whose instances are immutable? In our project the class is the Database\_connection class.

Forces: There must be no loopholes that would allow ‘illegal’ modification of an immutable object.

Solution: Ensure that the constructor of the “Database\_connection” class is the only place where the values of instance variables are set or modified.

Instance methods which access properties must not change instance variables.

Example : applying on Database\_connection class

|  |
| --- |
| Database\_connection |
| -ConnectionAddress: string |
| Database\_connection(connectionAddress):Boolean  GetConnectionAddress():string |

3) Delegation pattern (structrual): “applying on many classes in our the project”

Context : You are designing a method in a class . You realize that another class has a method which provides the required service. Inheritance is not appropriate (e.g. because the is-a rule does not apply).

Problem : How can you most effectively make use of a method that already exists in the other class?

Forces : You want to minimize development cost by reusing methods.

Solution: The delegating method in the delegator class in our example (Student and counselor classes ) calls a method in the delegate class in our example (Feedback class)to perform the required task. An association must exist between the delegator and delegate classes.

Example :

Delegate class Delegator classes

|  |
| --- |
| Feedback |
|  |
| SetFeedback(string):boolean |

|  |
| --- |
| Counselor |
|  |
| MakeFeedback(string):boolean |

|  |
| --- |
| Student |
|  |
| MakeFeedback(string):boolean |

4) Abstraction Occurrence (Structural pattern) applying on (Test and Questions classes) :

Context: In a domain model we find a set of related objects “occurrences”; the members of such a set share common information but also differ from each other in important ways as in our example “Test class(abstraction)”and “Question class” (Occurrence).

Problem: What is the best way to represent such sets of occurrences?

Forces: You want to represent the members of each set of occurrences without duplicating the common information .

Solution: Create an “abstraction” class as “Test class” as that contains the common data. Then create an “occurrence” class as “Question class” representing the occurrences of this abstraction. Connect these classes with a one-to-many association.

Example :

|  |
| --- |
| Question |
|  |
|  |

Abstraction class Occurrence class

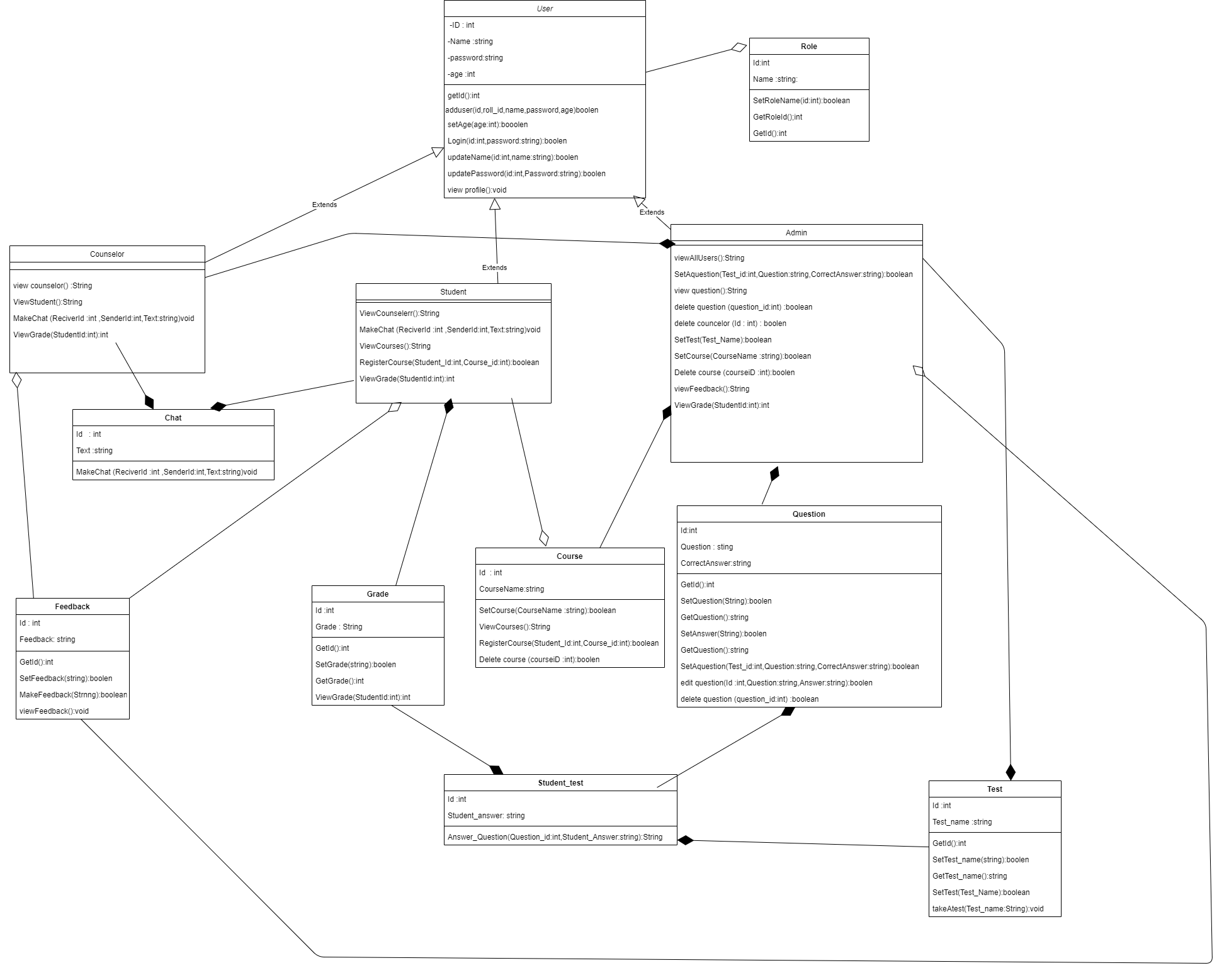
1

\*\*\*\*

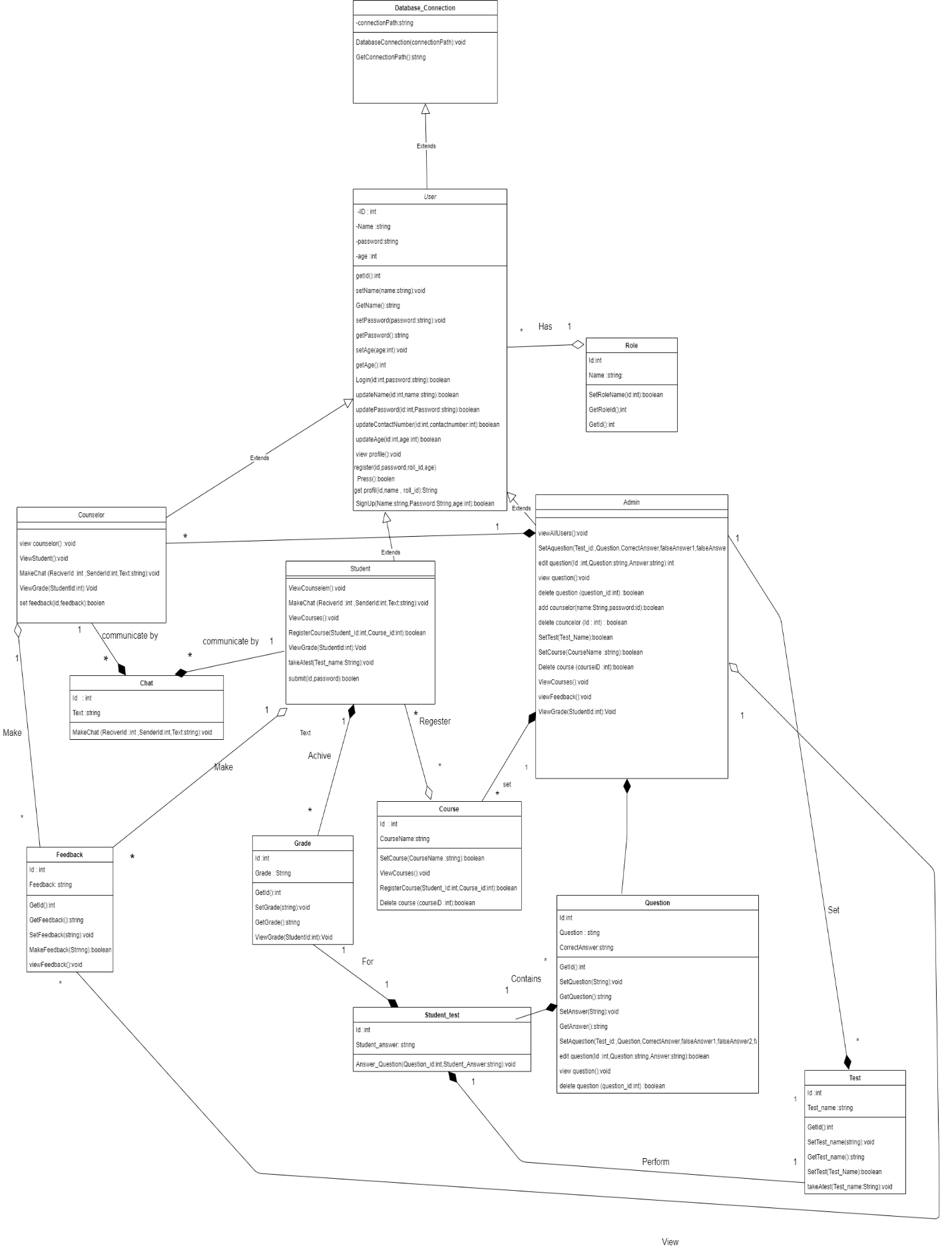
|  |
| --- |
| Test |
|  |
|  |

Class Diagrams (3 Versions)

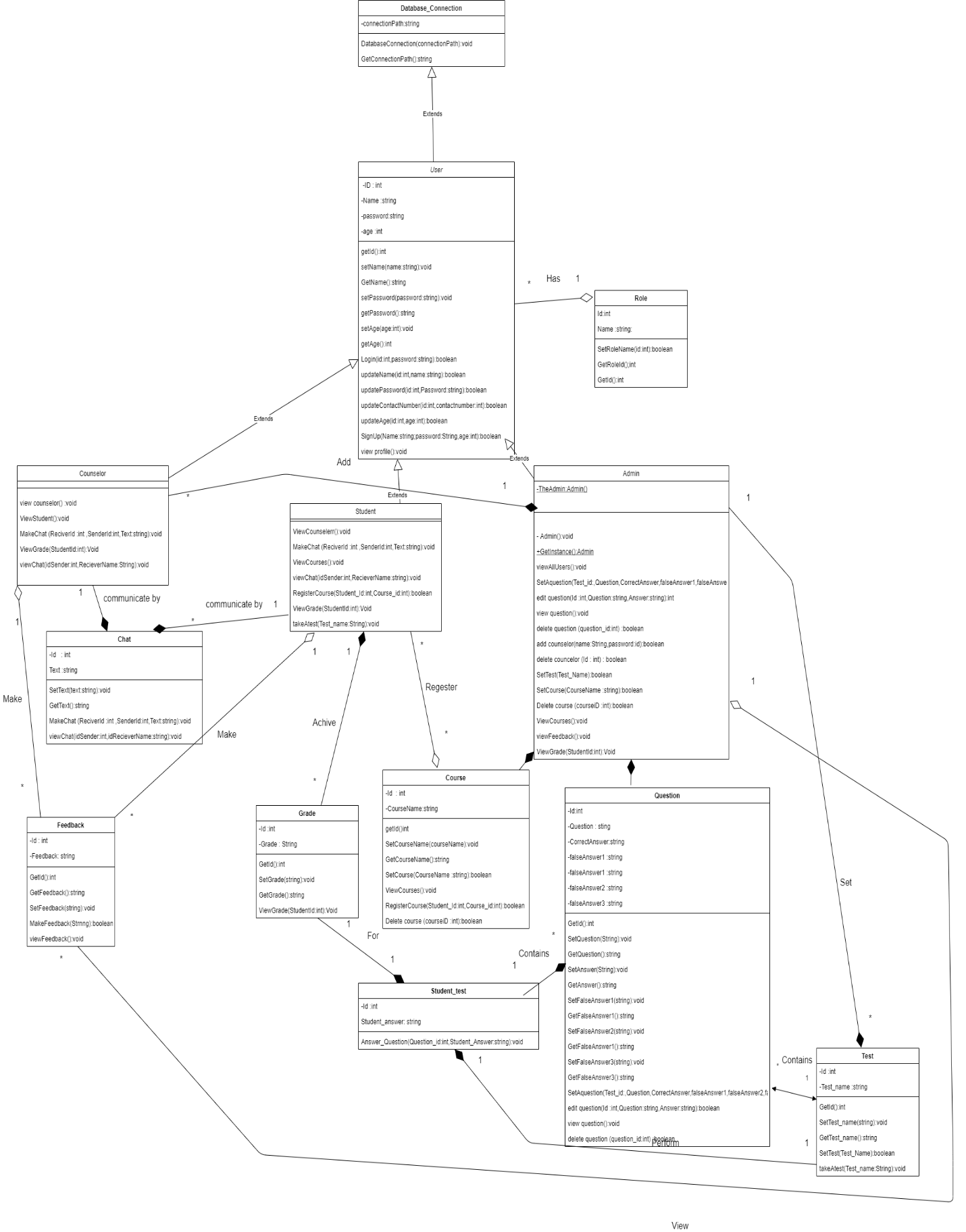
Version One :



Version Two :

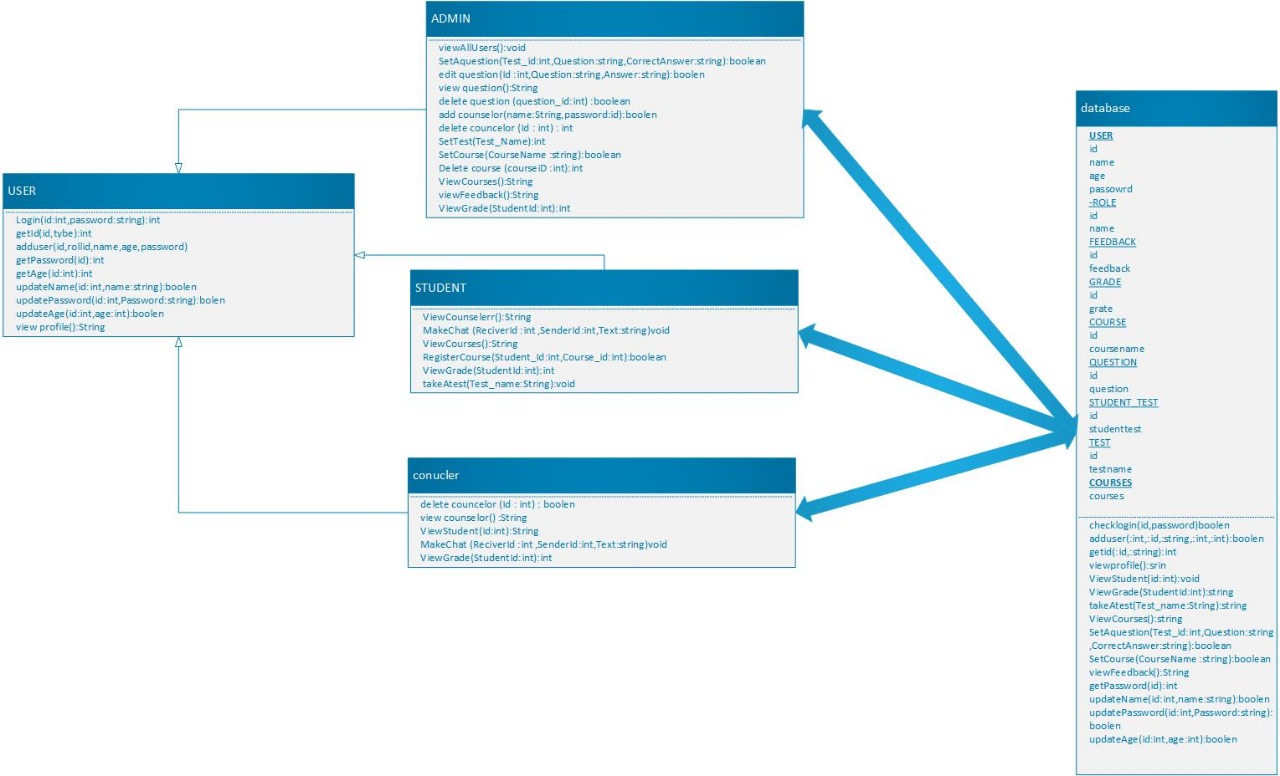


Version Three :

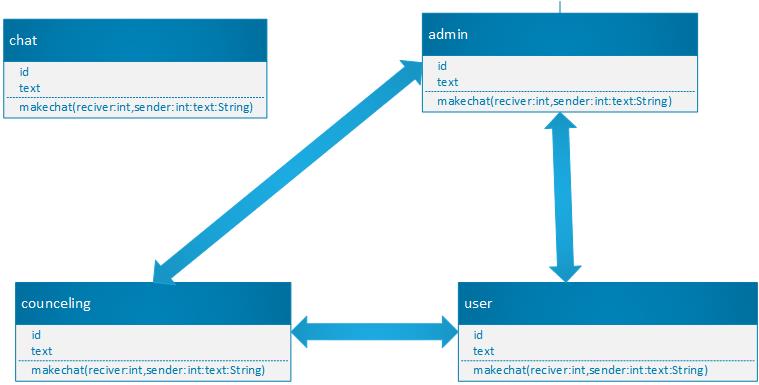


System Architecture :

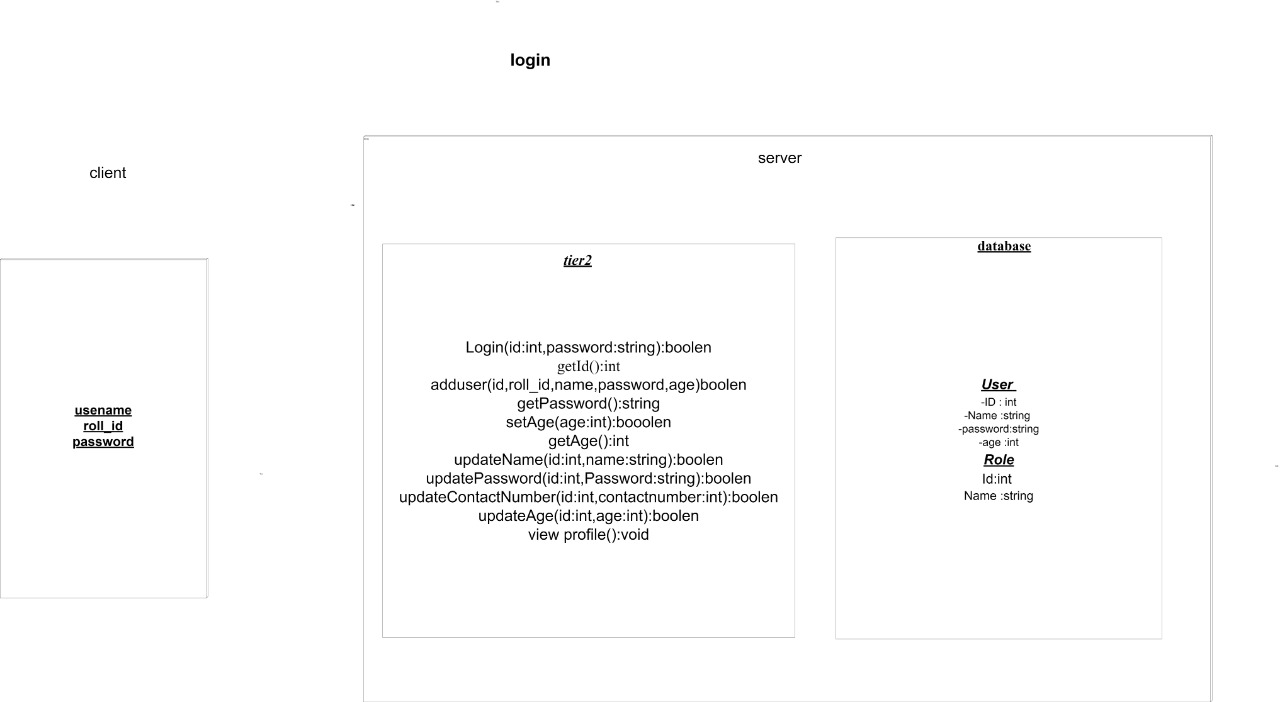
1)Data Centre :

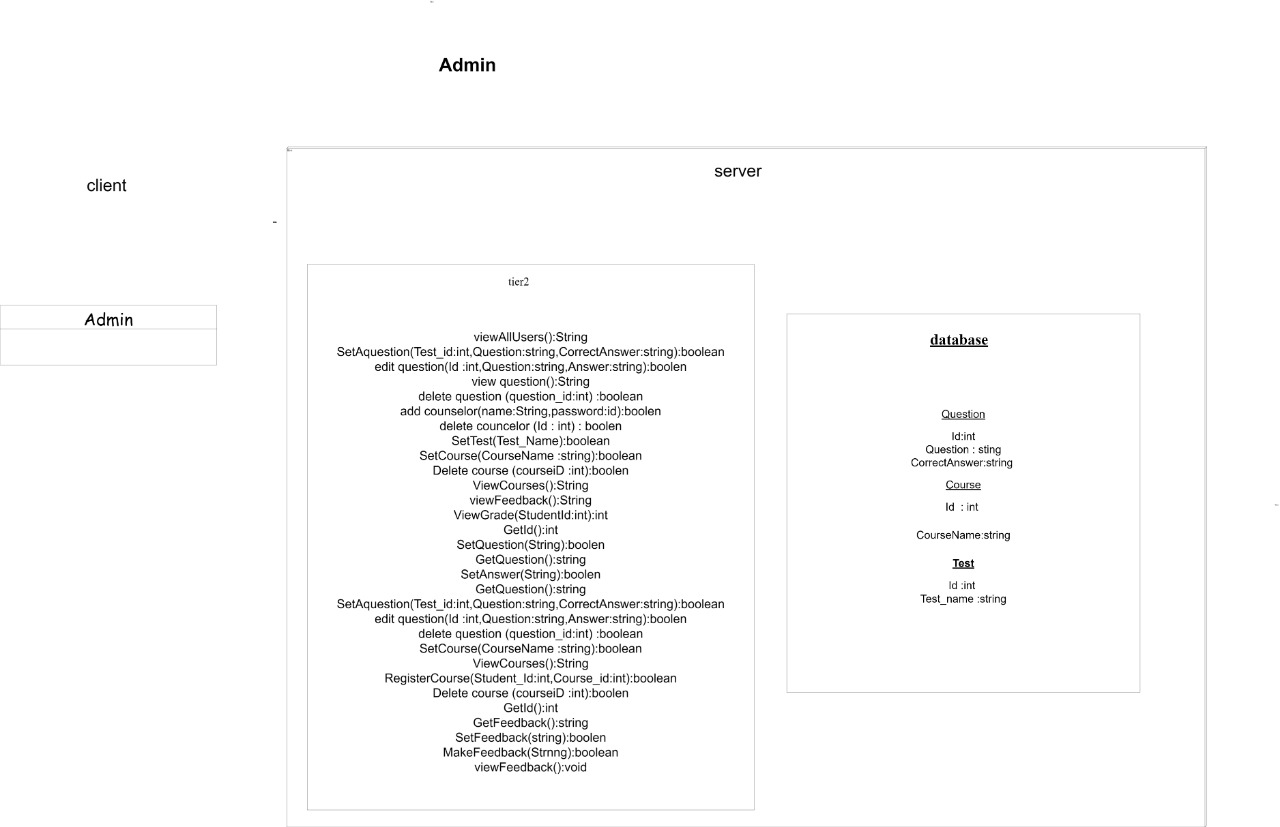


2) Peer to Peer :

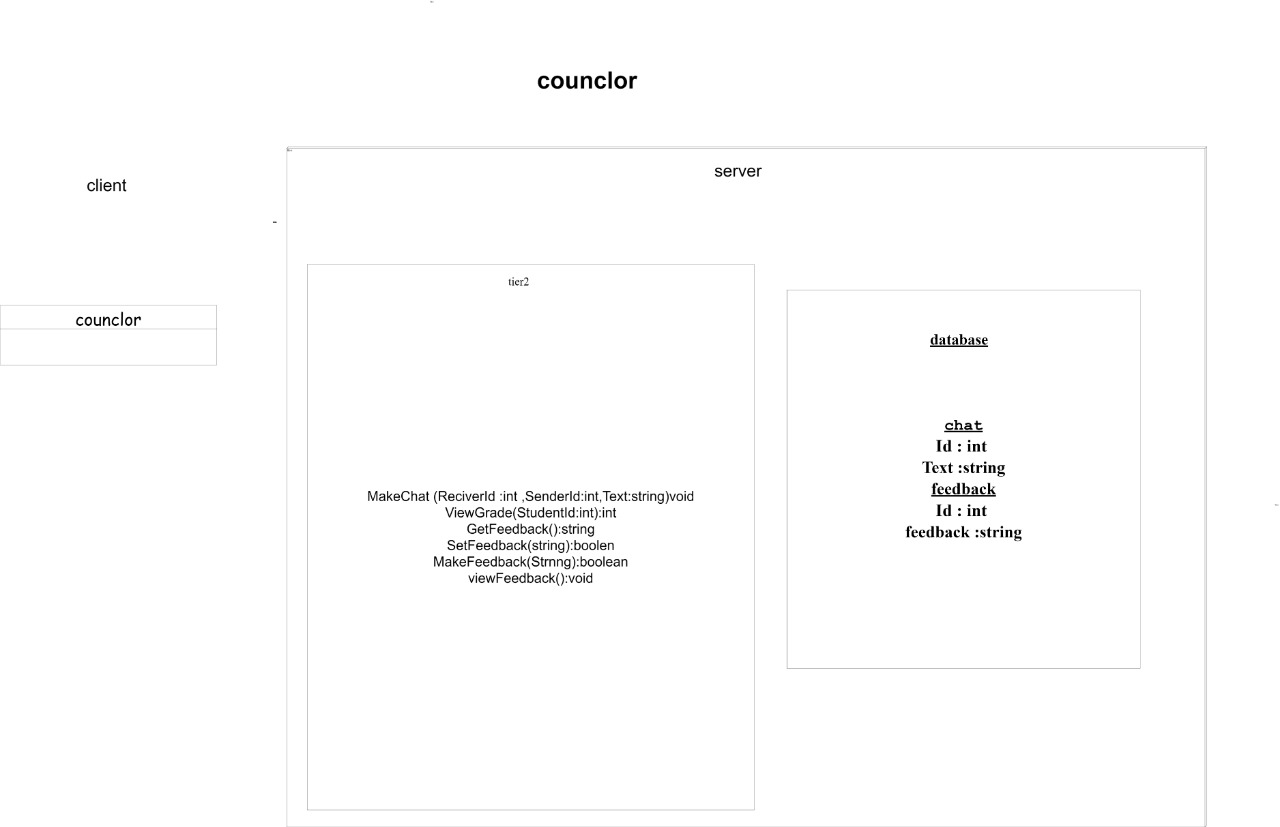


3) Client –Server :



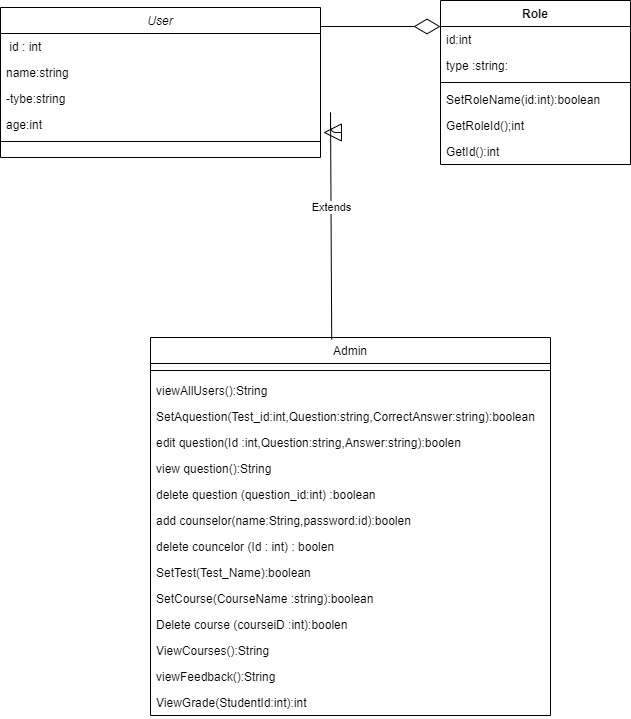




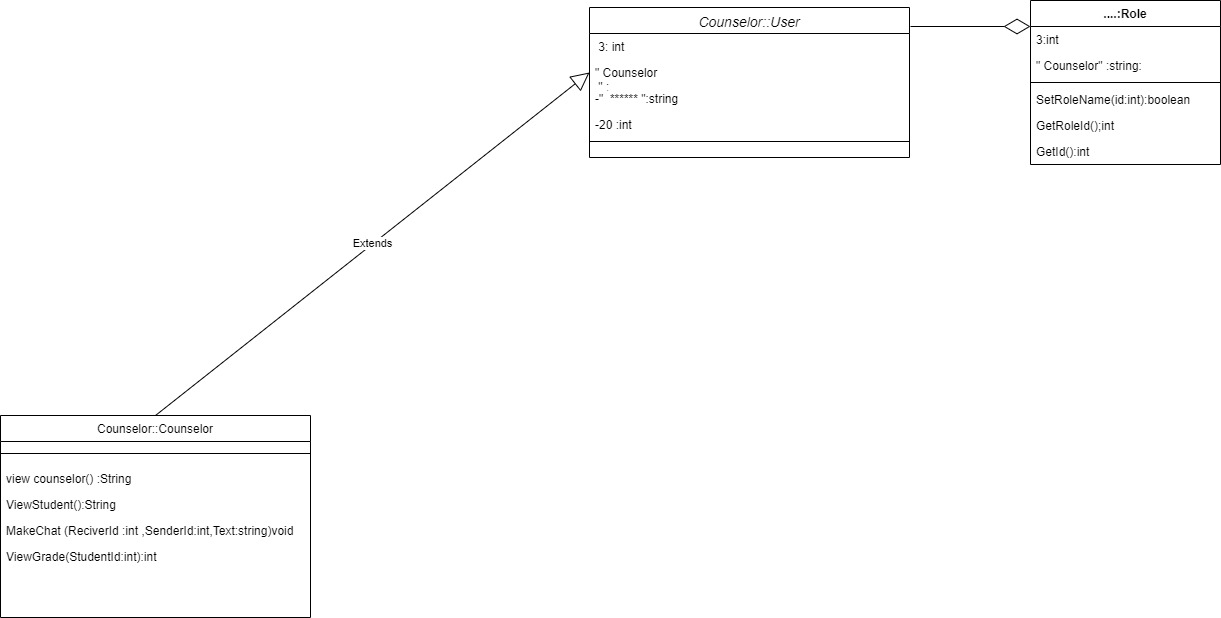


Object diagram :

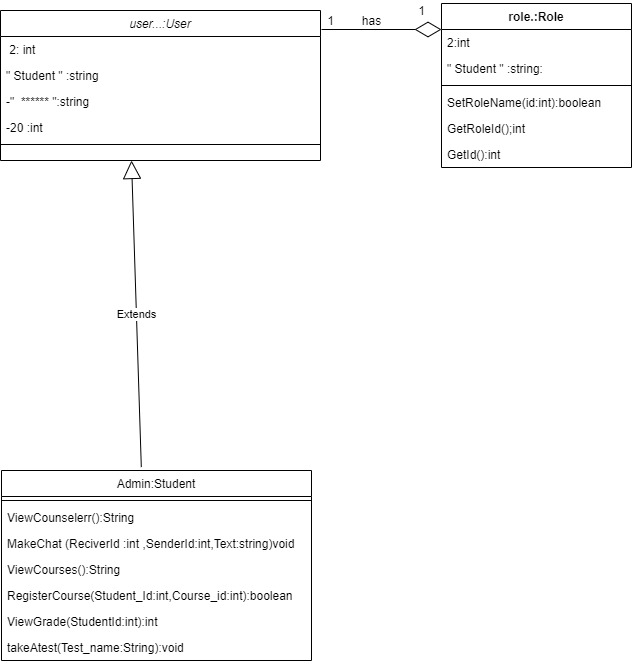
1)



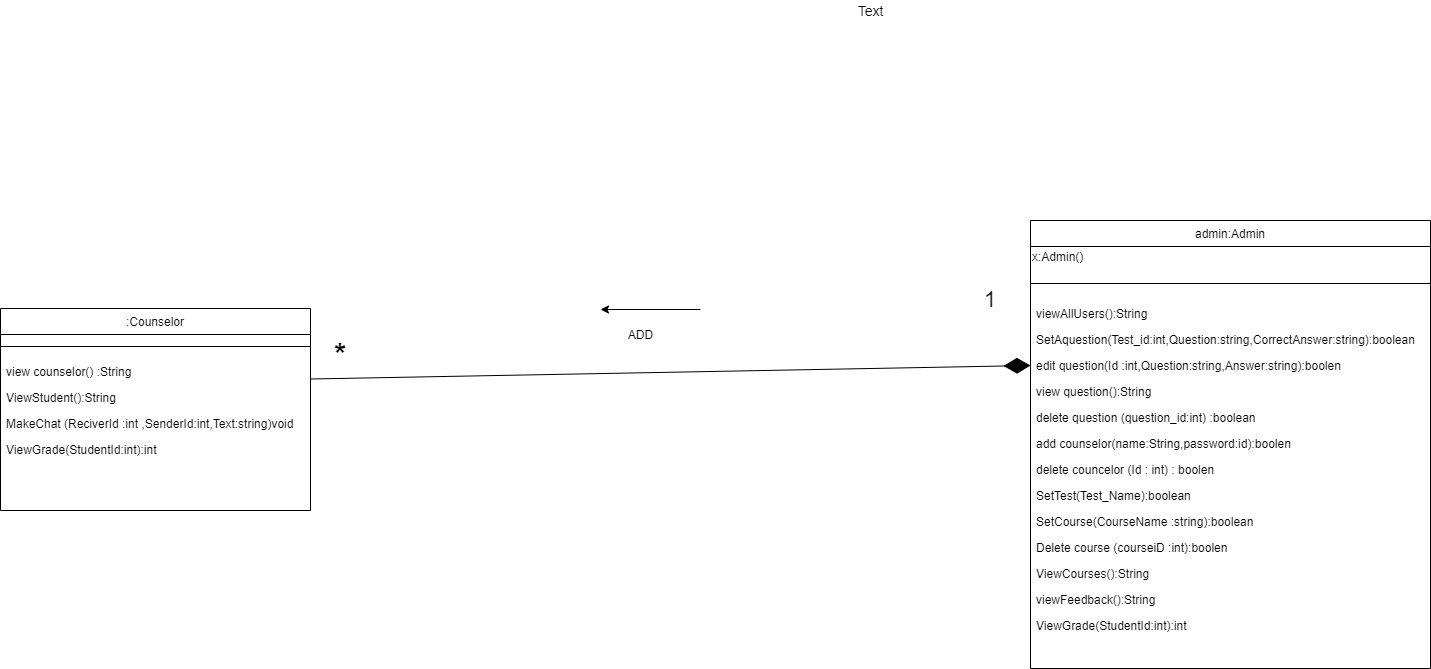
2)



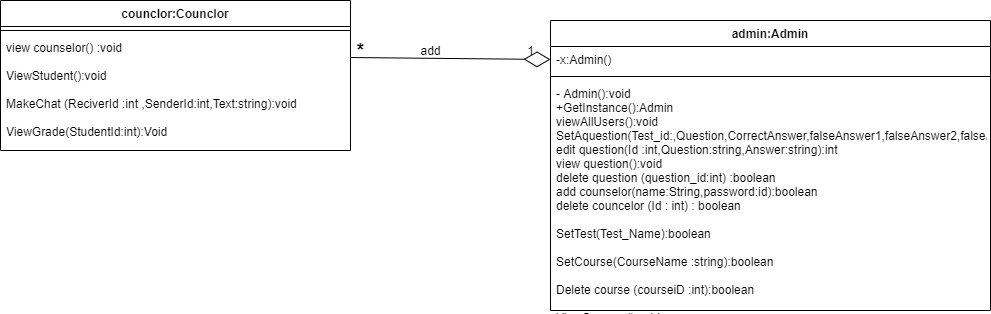
3)



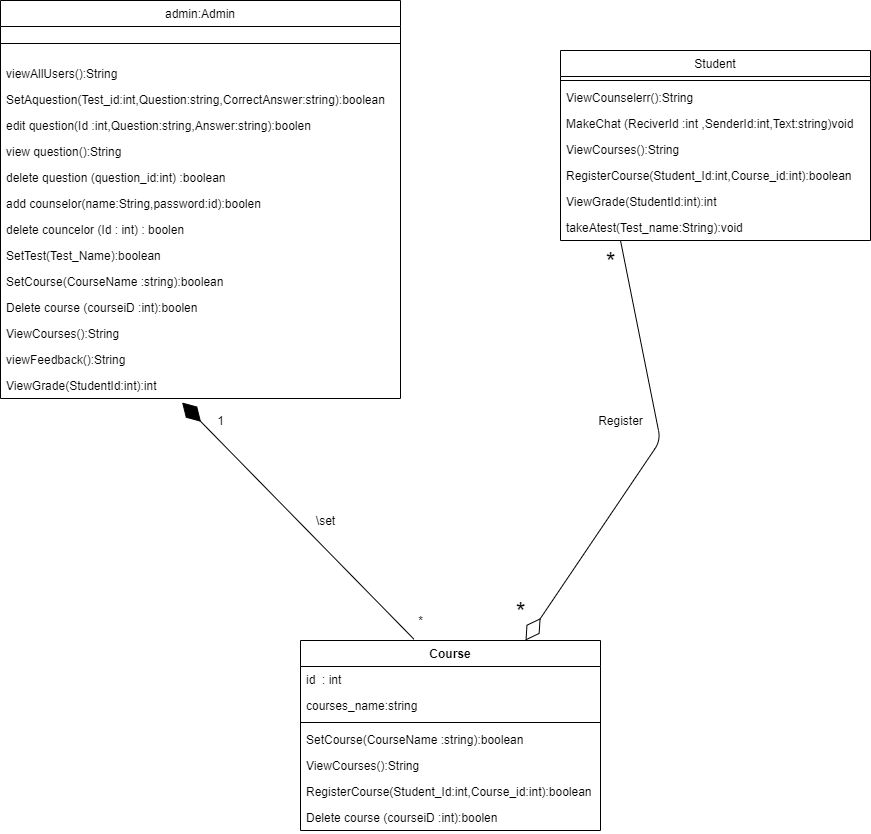
4)



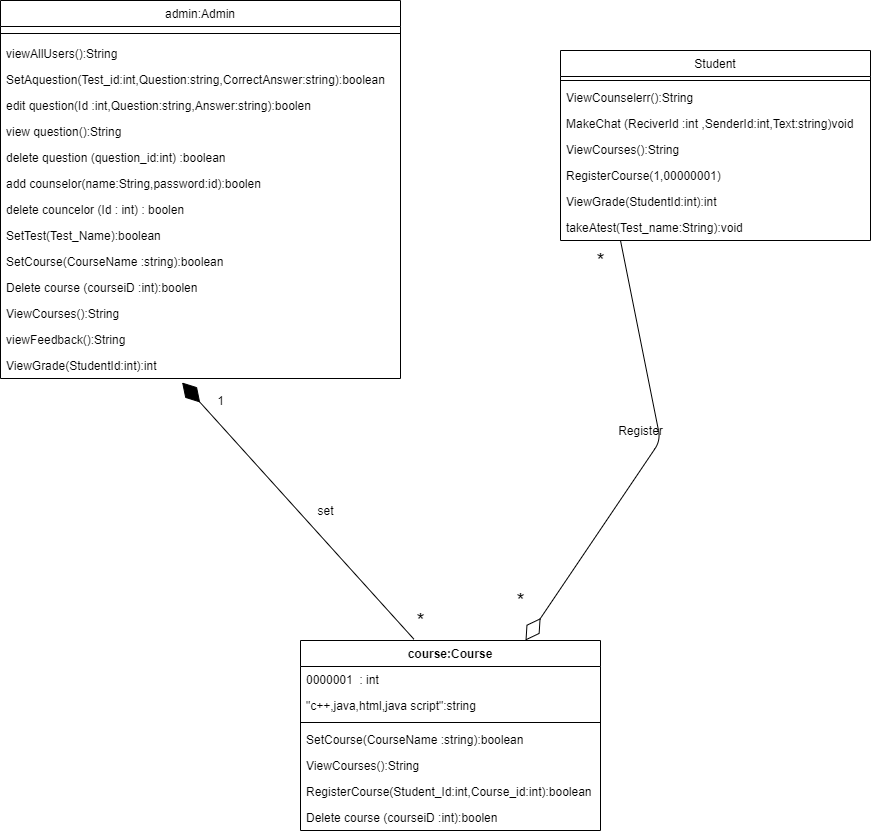
5)



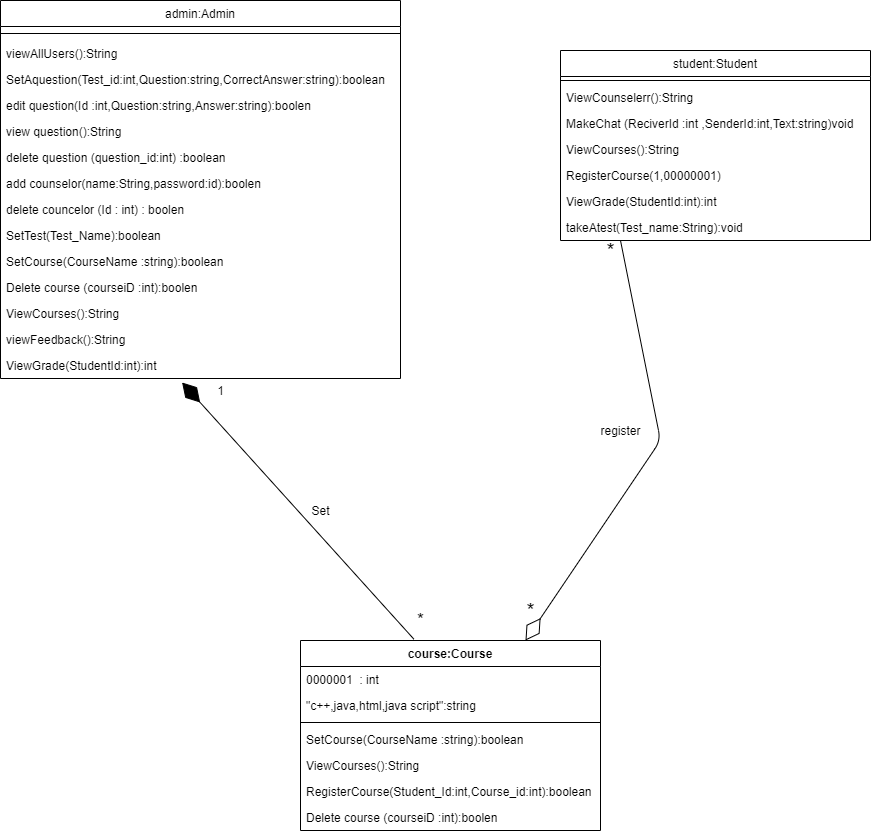
6) course (1):



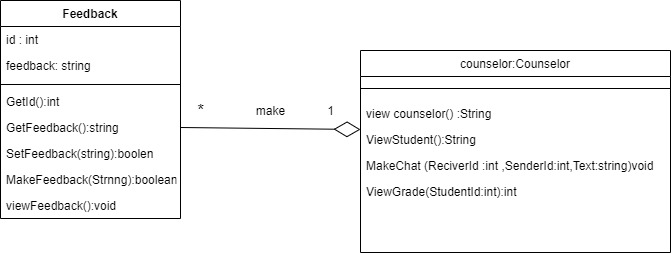
7) course 2



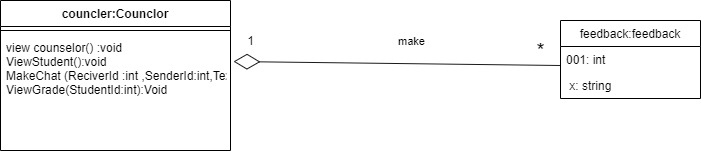
8) course 3 :



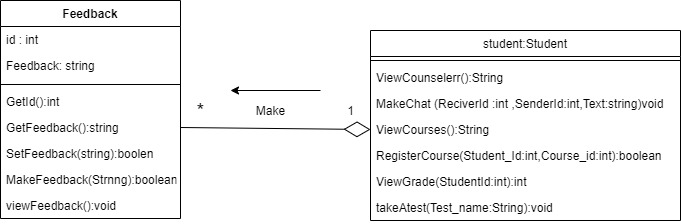
9)



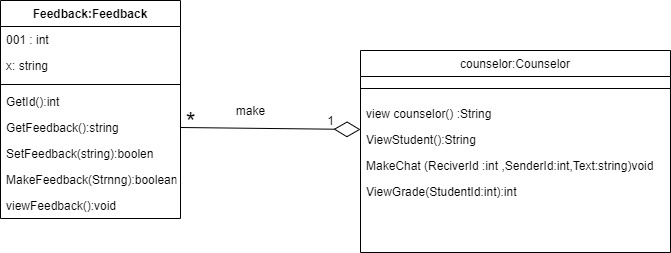
10) :



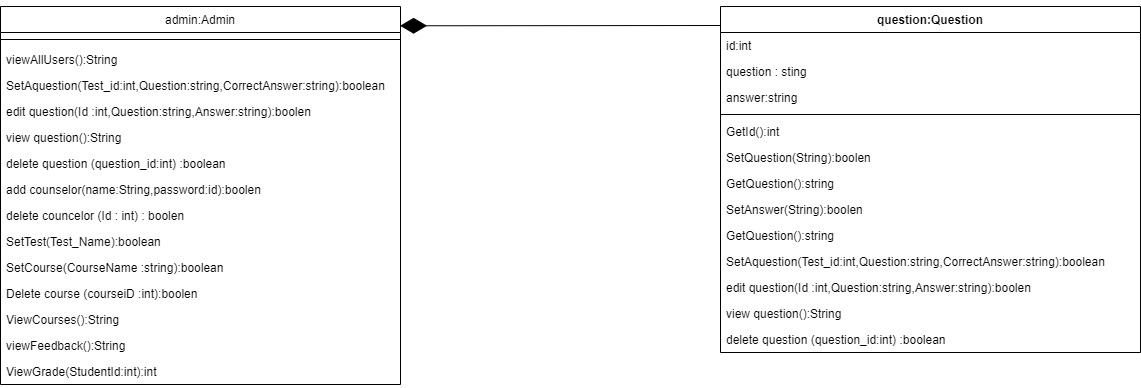
11) :



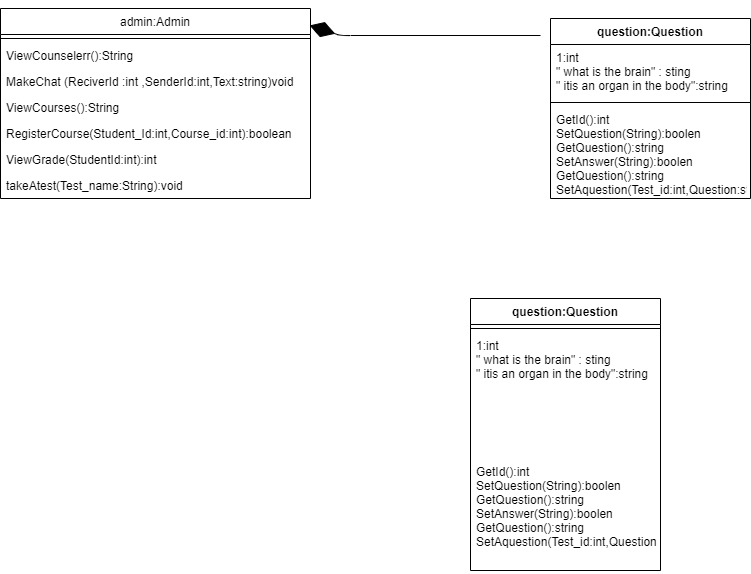
12) :



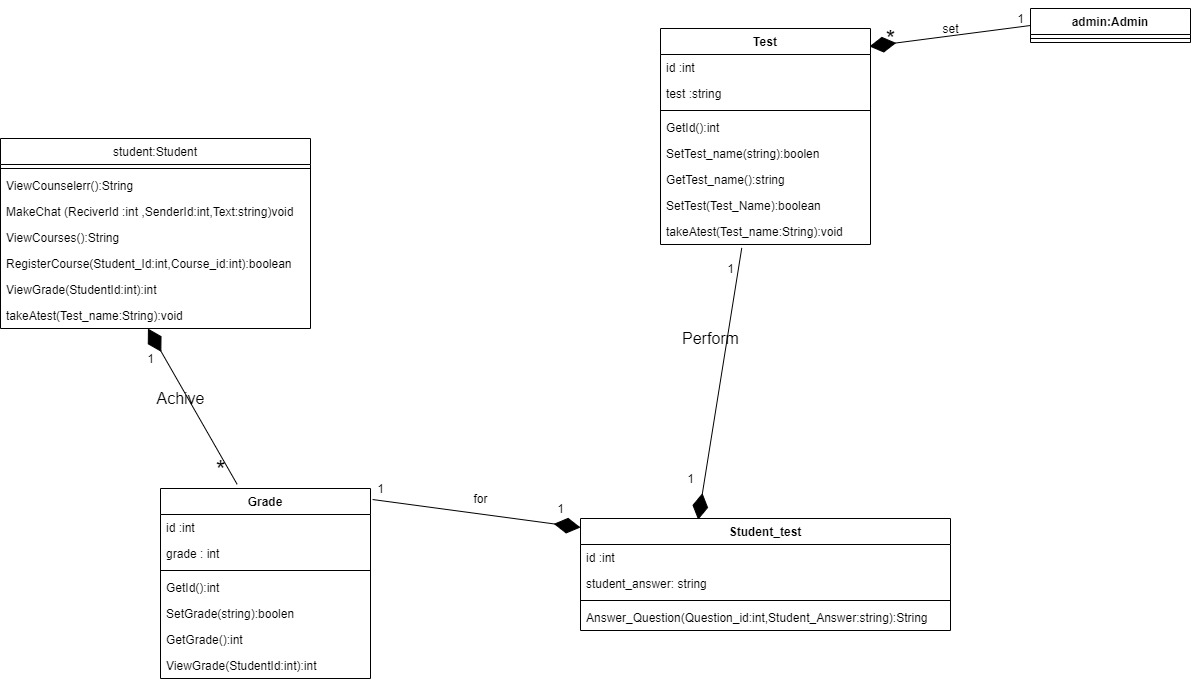
13) :



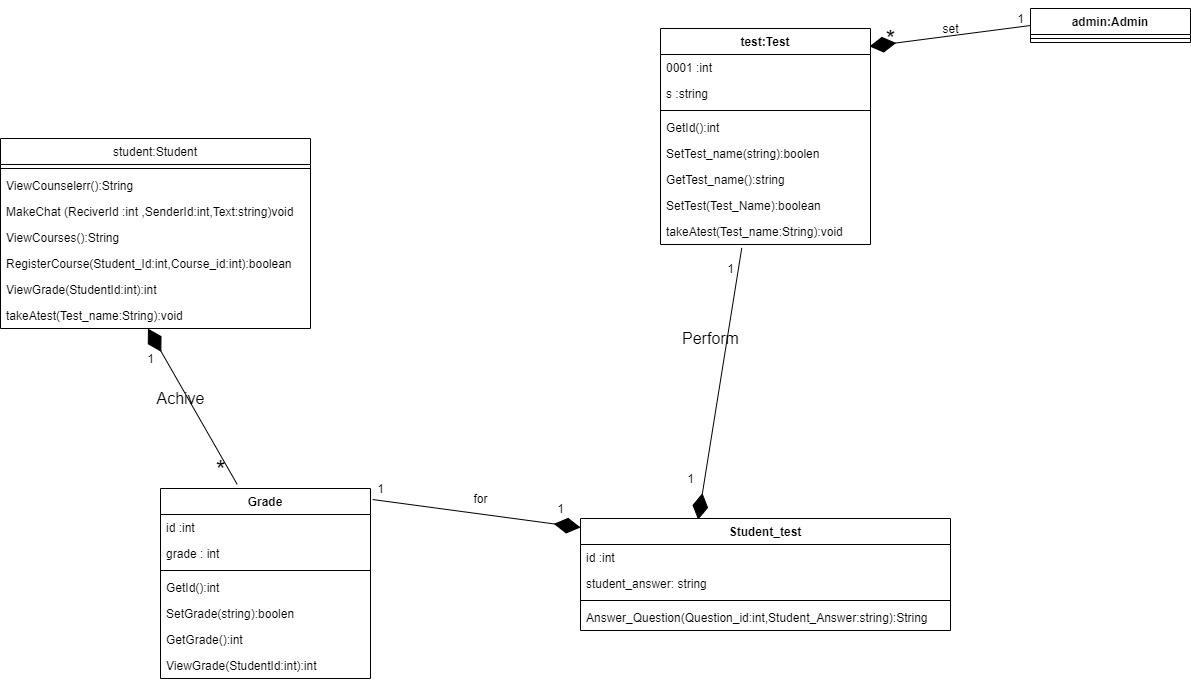
14) :



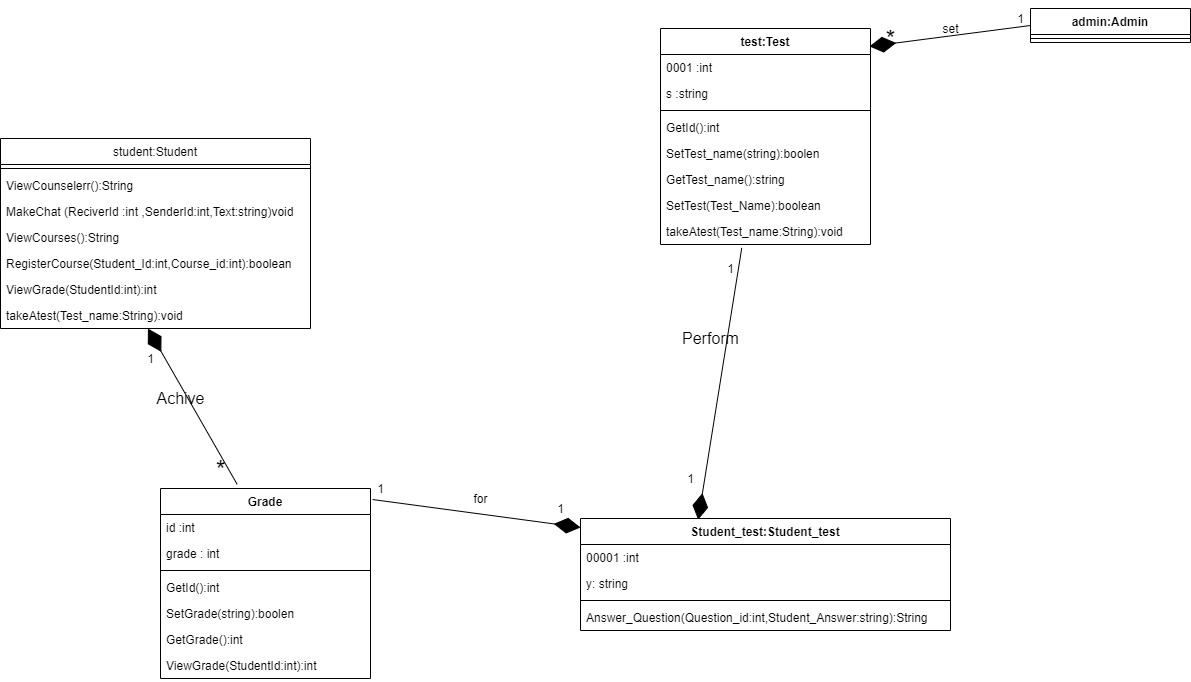
15) :



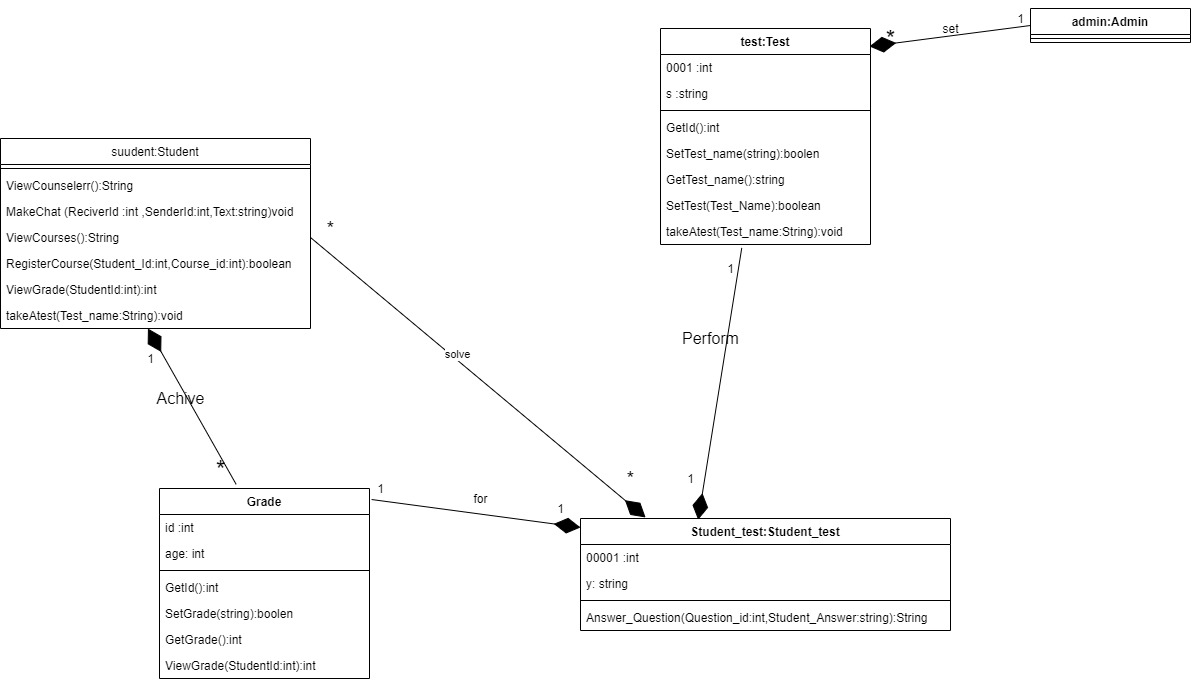
16) :



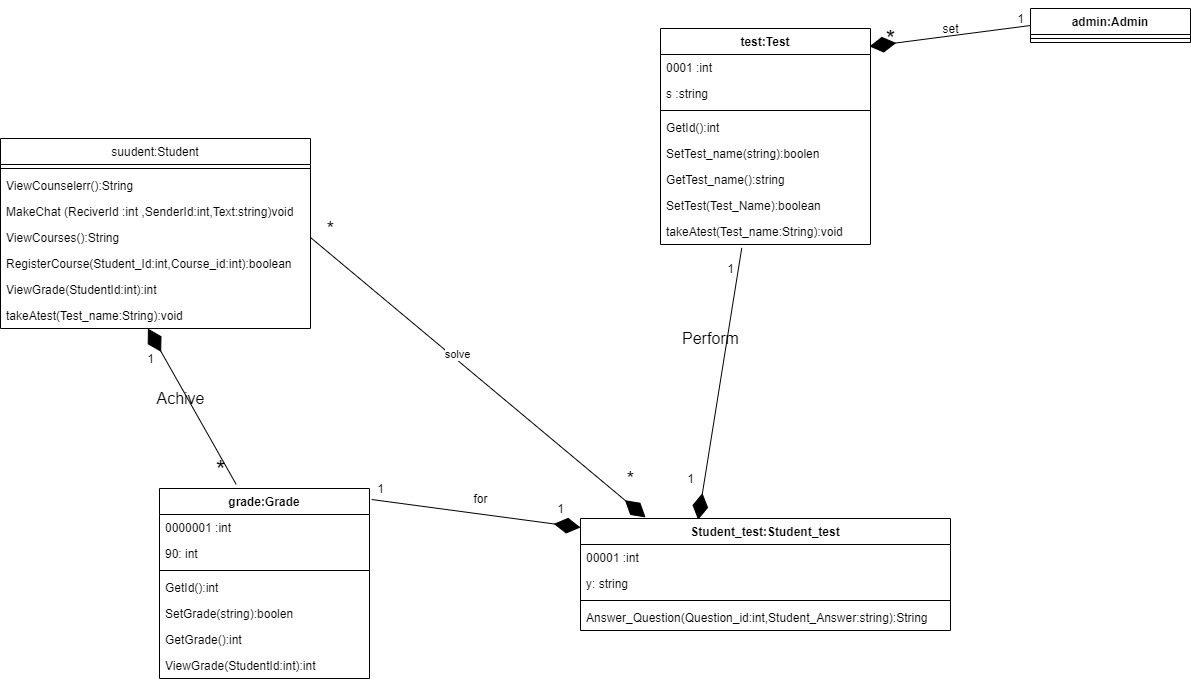
17):



18) :

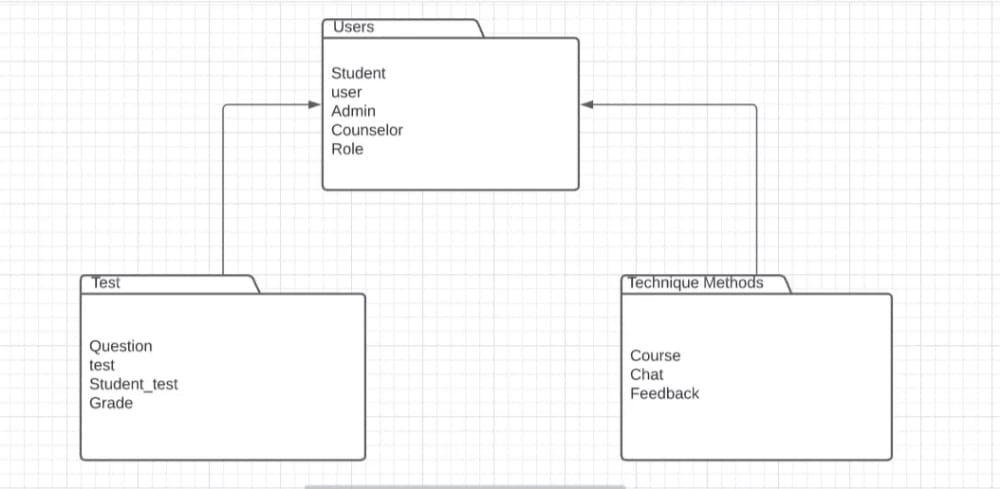


19) :

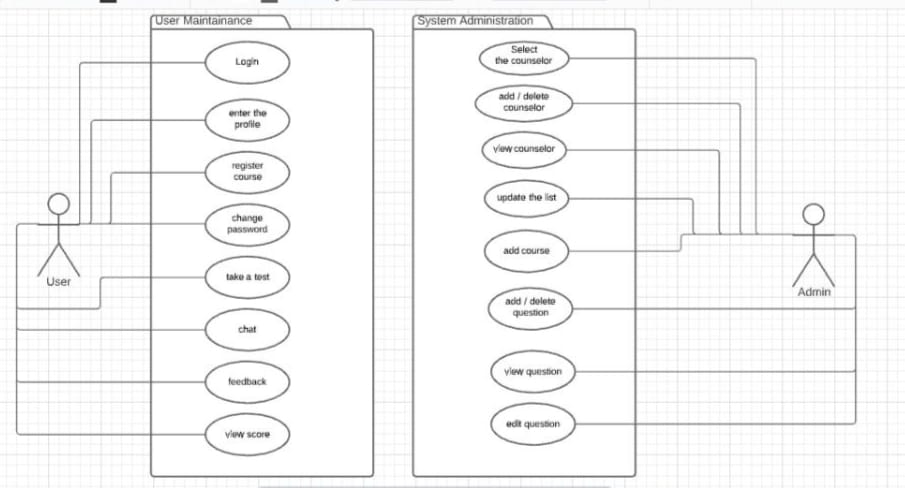


Package diagram :

1)

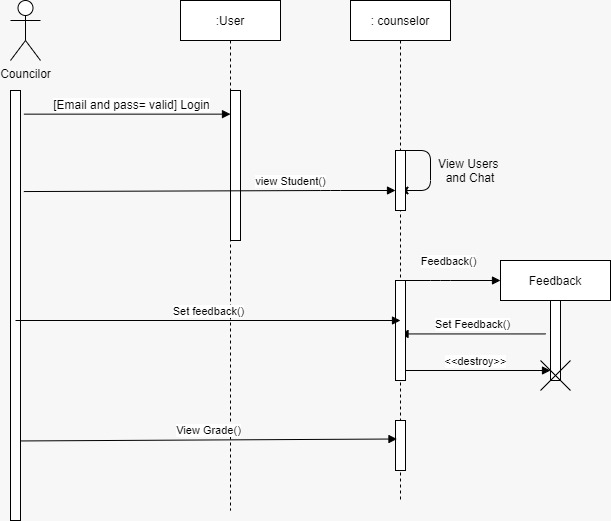


2)

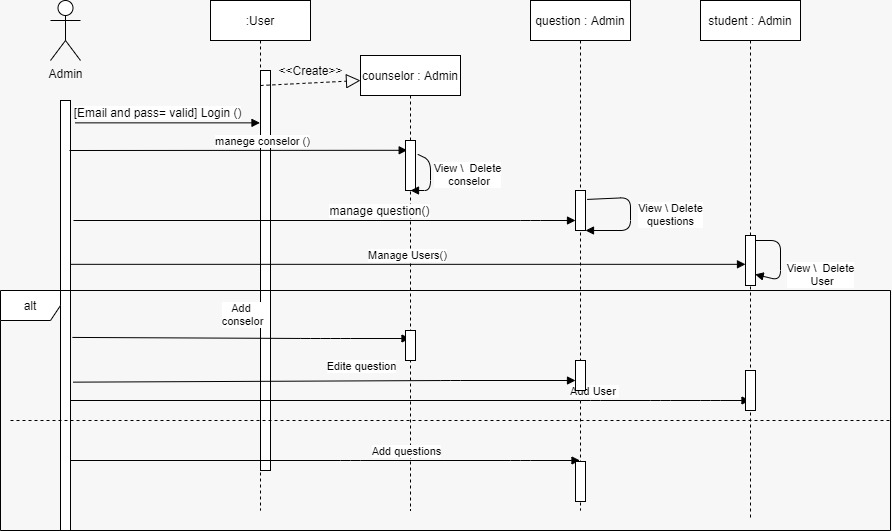


Sequence Diagram :

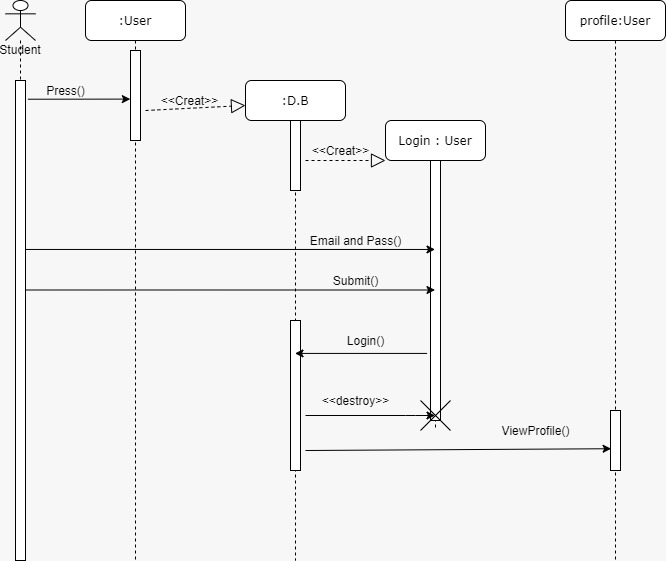
Councleor :



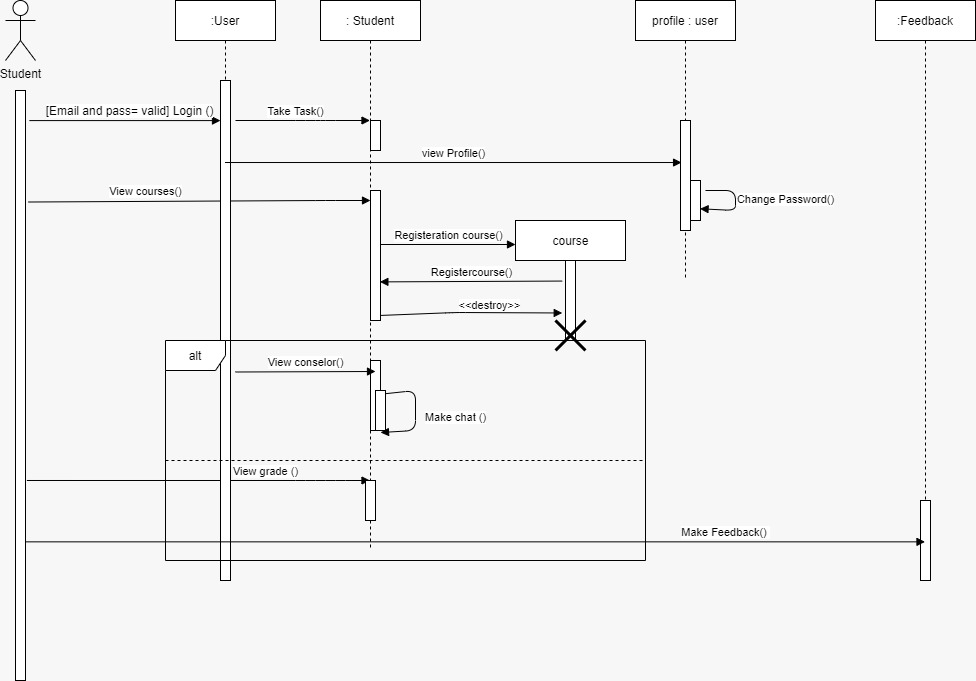
2) Admin :



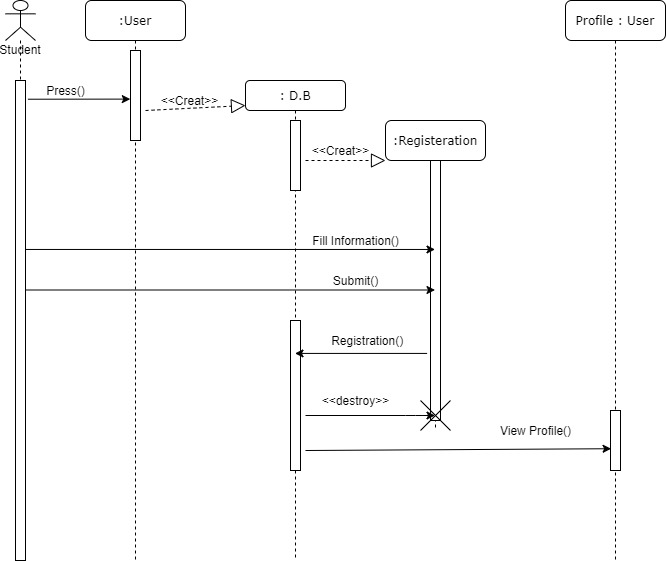
3) student (1):



4) student (2) :

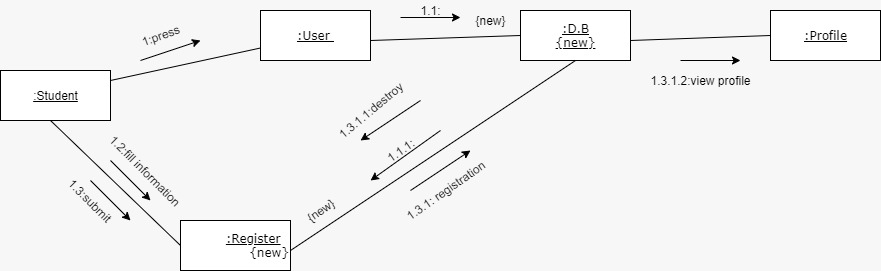


5) student (3) :

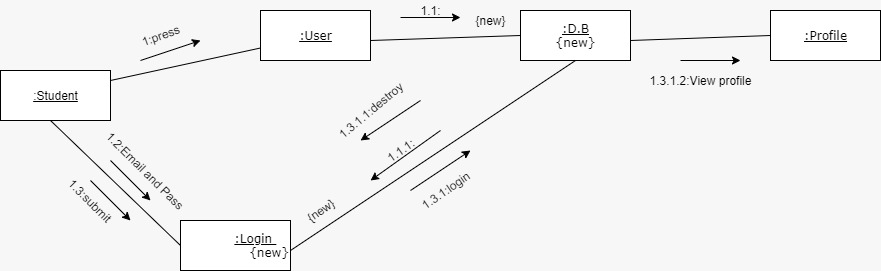


Collaboration diagram :

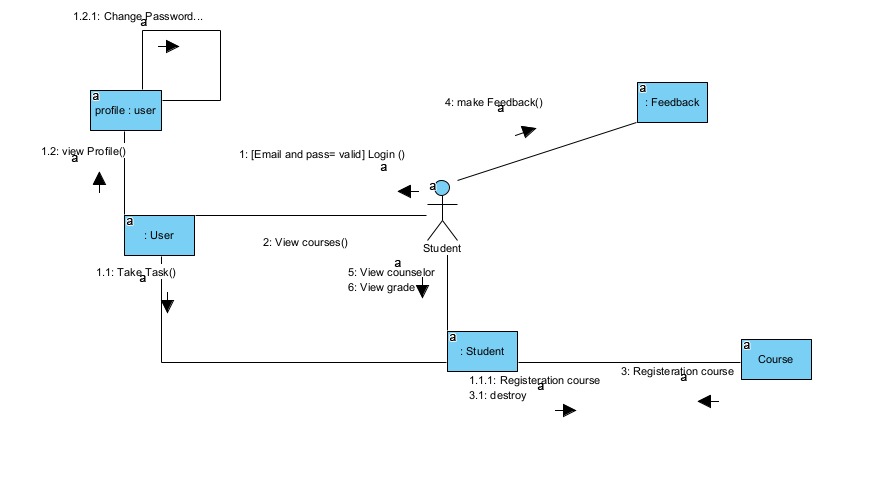
1) :



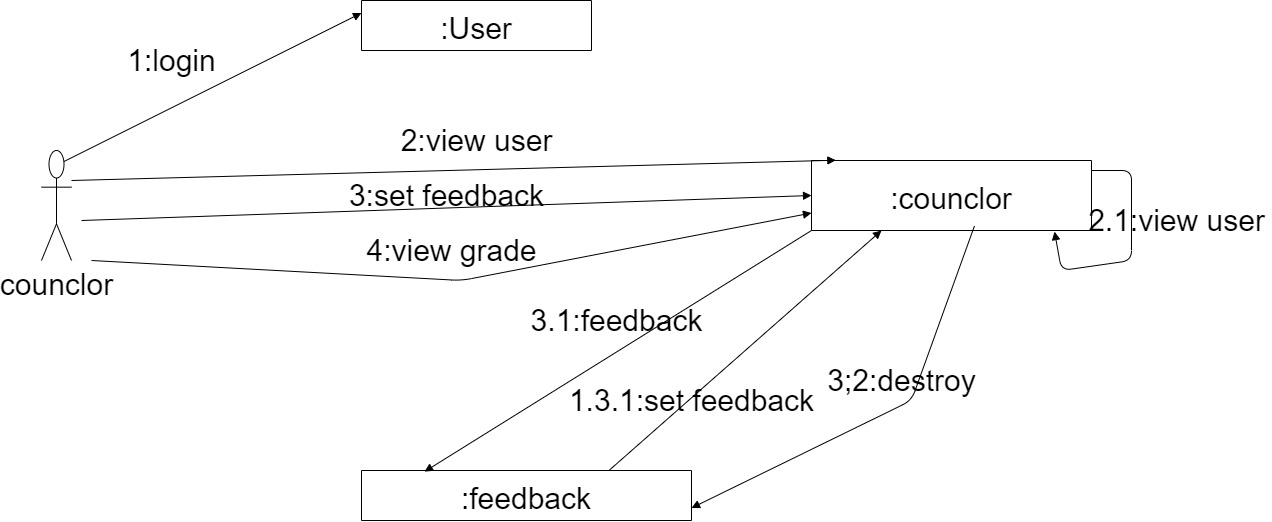
2) :



3) :



4) :



5) :

