OBJECT ORIENTED ANALYSIS AND DESIGN COURSE (INT3047)

E-Learning system

**COURSE PROJECT**

**Instructor:** Dr. Truong Ninh Thuan

**Project team:**

* Nguyen Van Vu Student.No: K58CA

|  |  |  |
| --- | --- | --- |
| * Nguyen Huu Thuc | Student.No: | K58CA |
| * Nguyen Van Quan | Student.No: 13020347 | K58CA |

**Requirement:** Give the analysis and design documentation of any software project.

Contents

[I. E-learning system requirements: 4](#_Toc419541228)

[1.1 problem statement: 4](#_Toc419541229)

[1.2 Glosary: 5](#_Toc419541230)

[1.3 Supplementary specification: 5](#_Toc419541231)

[1.4 Use-case model: 7](#_Toc419541232)

[1.4.1 Log in: 12](#_Toc419541233)

[1.4.2 Profiel changing: 12](#_Toc419541234)

[1.4.3 Course searching: 13](#_Toc419541235)

[1.4.4 Course enrolling: 14](#_Toc419541236)

[1.4.5 Homework uploading: 15](#_Toc419541237)

[1.4.6 Tutorial dowloading: 16](#_Toc419541238)

[1.4.7 Student certificate confirming: 17](#_Toc419541239)

[1.4.8 Course feedback: 18](#_Toc419541240)

[1.4.9 Course creating: 19](#_Toc419541241)

[1.4.10 Homework feedback: 20](#_Toc419541242)

[1.4.11 Register: 21](#_Toc419541243)

[1.4.12 Register confirm: 22](#_Toc419541244)

[1.4.13 User management: 22](#_Toc419541245)

[1.4.14 Course management 24](#_Toc419541246)

[II. E-learning system analysis 25](#_Toc419541247)

[2.1 Architectural analysis: 25](#_Toc419541248)

[2.1.1 Key abtraction: 25](#_Toc419541249)

[2.1.2 Upper-Level Components và dependency: 26](#_Toc419541250)

[2.2 Use-case analysis: 27](#_Toc419541251)

[2.2.1 Use-case relization interactions diagram: 27](#_Toc419541252)

[2.2.2 Use-case relization view of participating classes (VOPCs): 34](#_Toc419541253)

[2.2.3 Analysis-class-to-Analysis-machanism map 41](#_Toc419541254)

[III. E-learning system design: 43](#_Toc419541255)

[3.1 Identify design elements: 43](#_Toc419541256)

[3.1.1 Subsystem context diagram: 43](#_Toc419541257)

[3.1.2 Analysis-class-to-design-elemnt map: 44](#_Toc419541258)

[3.1.3 Design-element-to-owning-Packages map 46](#_Toc419541259)

[3.1.4 Architectural components and their dependencies: 48](#_Toc419541260)

[3.1.5 Packages và their dependencies 49](#_Toc419541261)

[3.2 Class design: 50](#_Toc419541262)

[3.2.1 Describe each class or interface: 50](#_Toc419541263)

[3.2.2 Class diagram in total: 62](#_Toc419541264)

# I. E-learning system requirements:

## 1.1 Problem statement:

Nowadays, with the development of the technology, the application of information technology in human activities from an economic, financial, trade to culture, education... has become more and more popular. Information technology has been profoundly transformed traditional fields, industries.

Online learning model is no longer strange all over the world. Many developed countries in North America, Europe ... have strongly implemented E-learning in common educational system nationwide. In recent years, e-learning has been developed for high school students, especially the United States, Korea, Japan ...

Currently, the system of education in our schools are being improved, converted from traditional training model to the model of a modern credit-oriented training and applied with information technology extensively. In this trend, building an online learning management system (E-Learning Management System) performance is really necessary, because it offers many practical benefits for learners, trainers and education manager

Online learning Management System will be deployed as Web-based applications (Web Application), that has primary functions are to contributing and managing courses. Students, teachers and system administrators are logged into the system with different objectives to ensure that the system works at stable mode and carry out the most effective learning and teaching activities. Trainees can search the list of courses (time, respective trainers ...), register courses, download documents, lectures, and assignments, edit personal information. Trainers can create courses, lectures and upload related documents, evaluate learners. Administrators can manage how people enrolling, registering for an account and setting system parameters..

## 1.2 Glossary:

**Glossary definition:**

The glossary contains the working definitions for the key concepts in the E-Learning System.

***User:***

A person who attempts one or more courses

***Instructor:***

A person, who manages directly the teaching in a course

***Administrator:***

A person, who manages the whole system.

***Guest:***

A person, who accesses to the system but not log in yet

***Course:***

A class about one particular subject in the system.

***Course certificate:***

Certificate to confirm a user have finished a particular course

## 1.3 Supplementary specification:

**Objectives:**

The purpose of this document is to define requirements of E-learning system. This supplementary specification lists the requirements that are not readily captured in the use case of use-case model. The supplementary specification and the use-case model together capture a complete set of requirements on the system.

**Scope:**

This supplementary specification applies to the E-learning system, which will be developed by OOAD students.

**Functionality:**

Multiple users must be able to perform their works concurrently.

**Usability:**

The software must be easy to use that a new user can learn how to use the system within 30 minutes. This is very important requirement.

The user interface must be nice and clear.

**Reliability:**

The system shall be available 24 hours a day, 7 days a week, with no more than 10% down.

**Performance:**

The latency of get static data must be less than 10 seconds and that one of other operations are less than 2 seconds.

The GUI transitions must be smooth.

No error of billing

**Supportability:**

None.

**Security:**

Almost changes of the system databases can only done by the administrator. Require password before submitting the changes

**Design constraints:**

The system should provide both Window-based desktop Interface and Mobile Application Interface.

## 1.4 Use-case model :

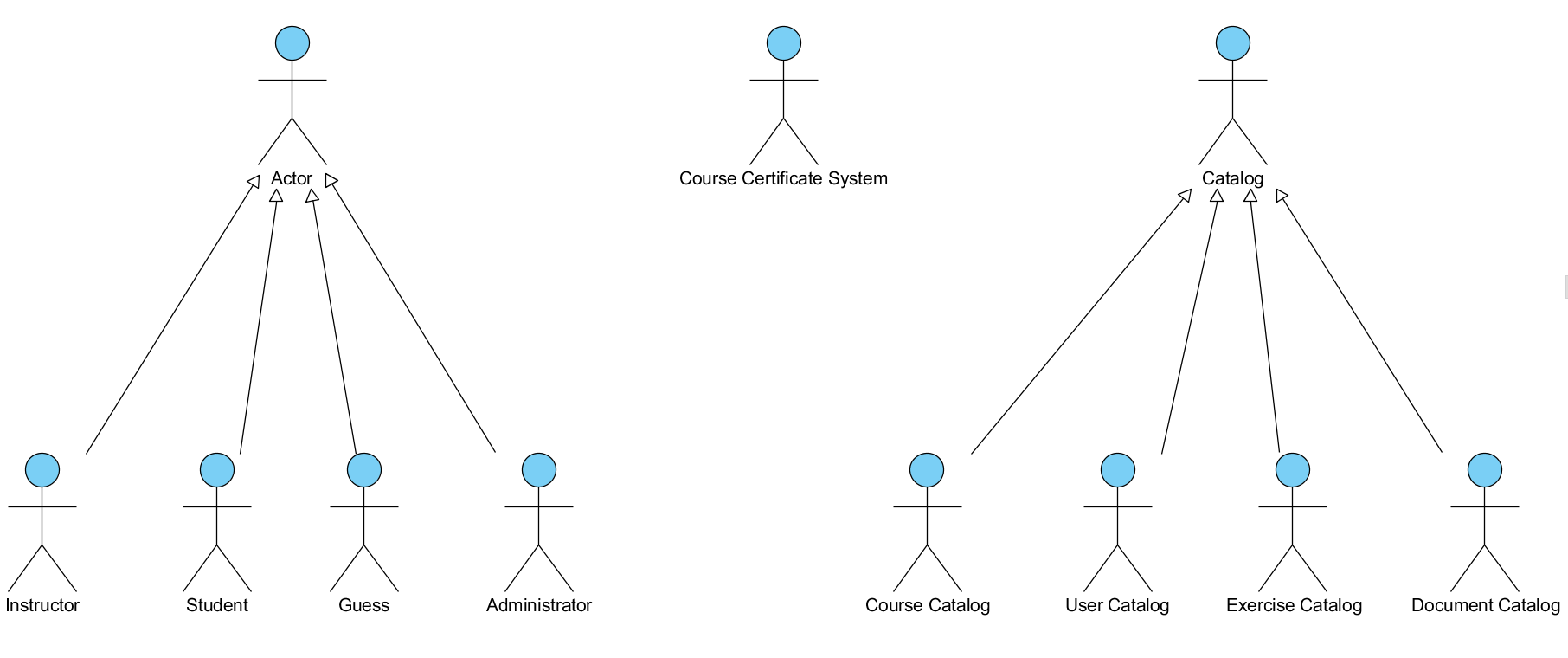
****

Figure 1.4.1: Use-case model – Actors and their dependencies.

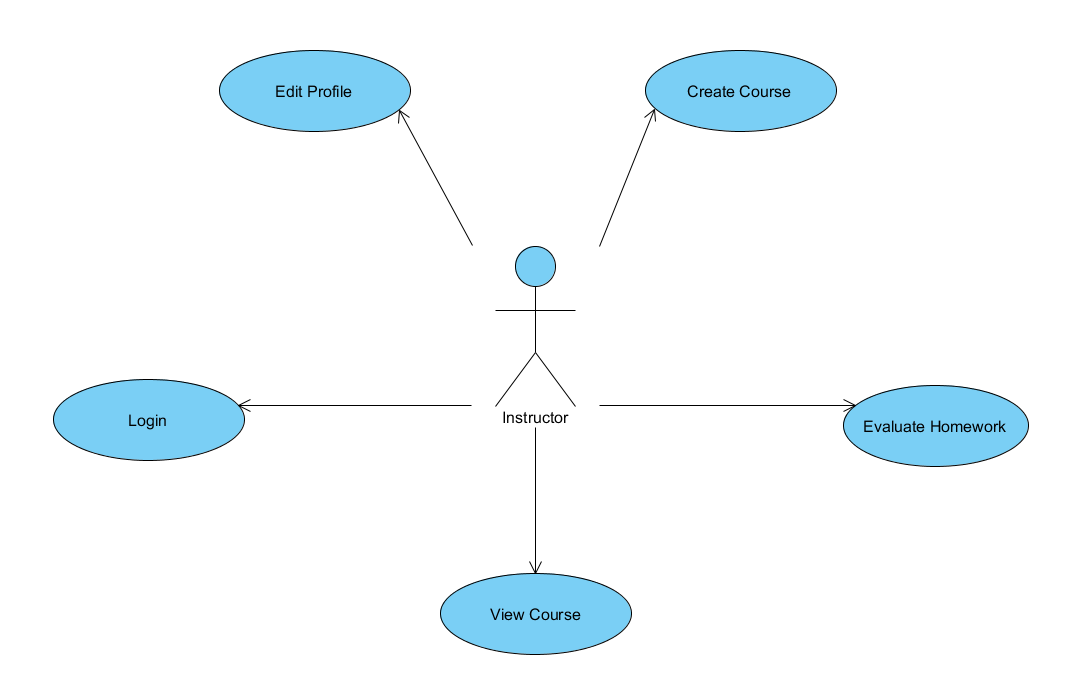


Figure 1.4.2: Use-Case Model - Instructor

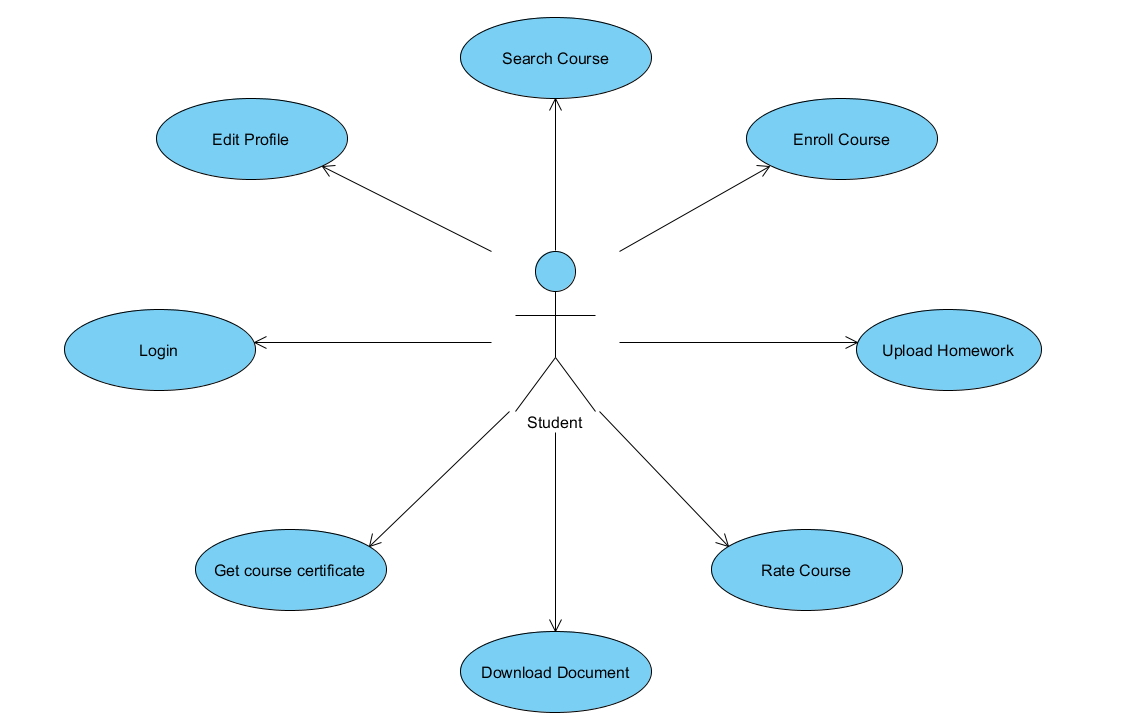
****

Figure 1.4.3: Use-Case Model - Student

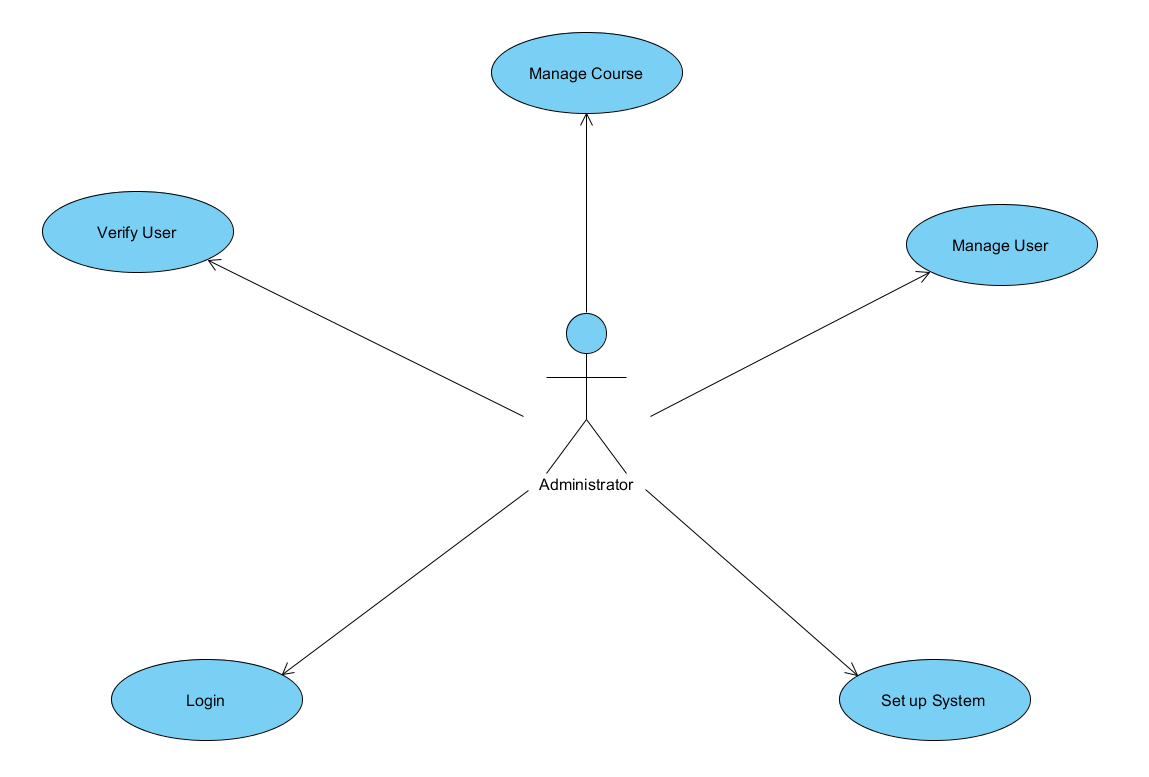
****

Figure 1.4.4: Use-Case Model - Administrator

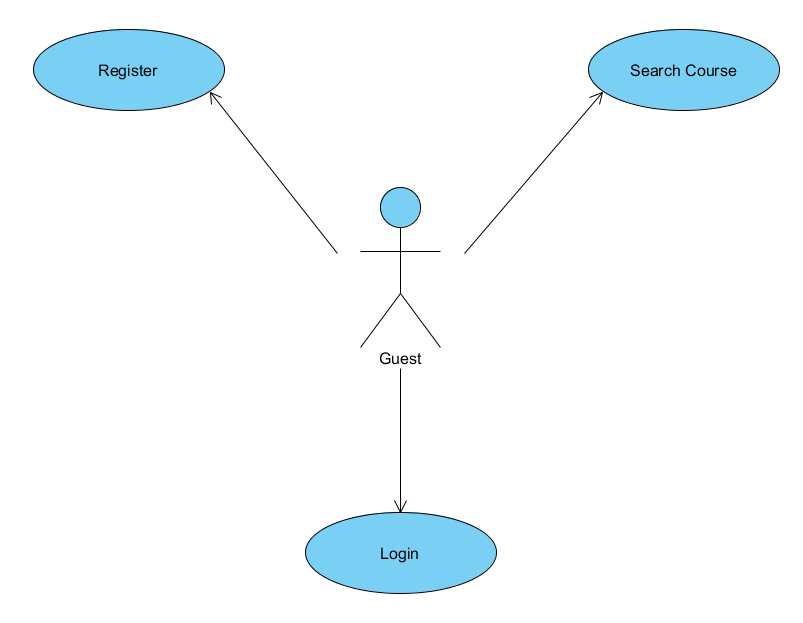
****

Figure 1.4.5: Use-Case Model - Guest

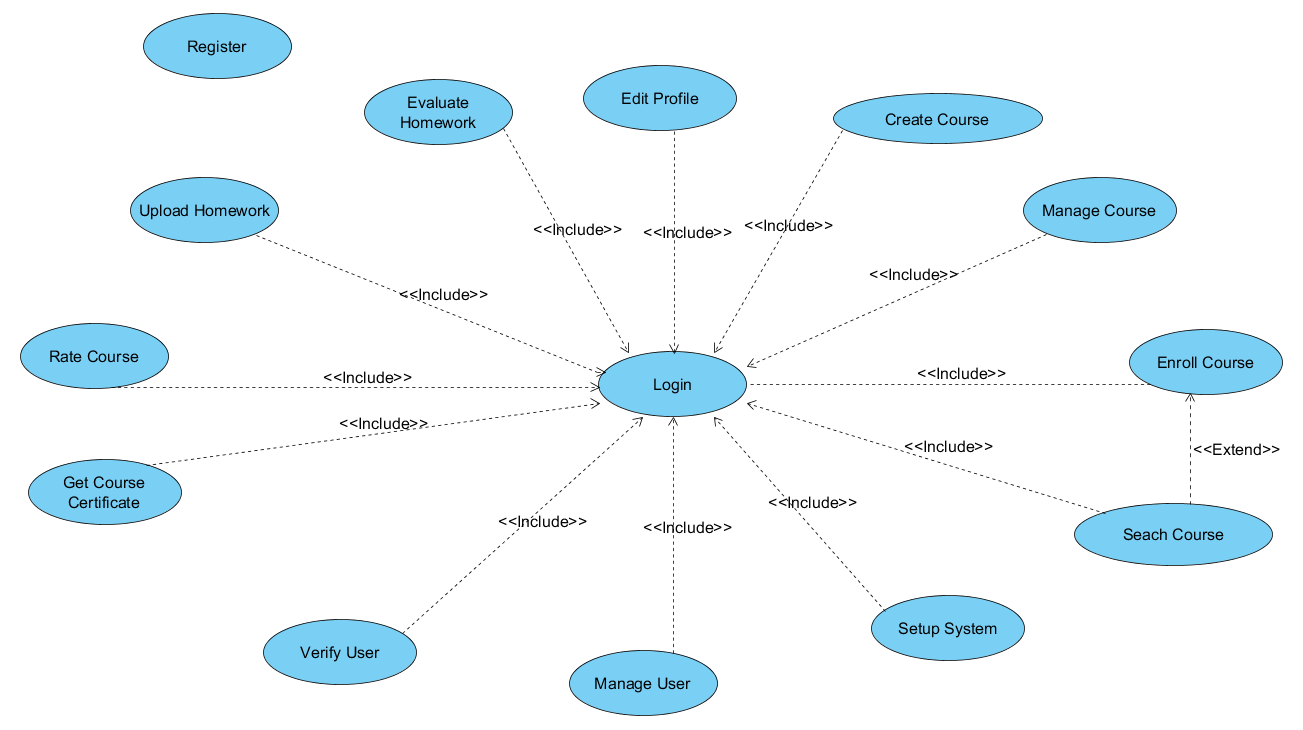
****

Figure 1.4.5: The Use-Case dependencies

### 1.4.1 Log in:

|  |  |
| --- | --- |
| **Use Case** | Log in |
| **Actor** | User, Instructor, Administrator. |
| **Brief Description** | Describes how users log into the system |
| **Main Flow** | 1. System shows log in page.  2. Users enter their user names and passwords.  3. System validates the entered name and password and logs the actor into the system. |
| **Alternative Flows** | 1. Invalid name/password: System displays an error messages.  2. The actor can choose either return to the beginning of Basic Flow or cancel login. |
| **Pre-condition** | None |
| **Post-conditions** | If logging in is successful, the actor is now logged into the system. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.2 Change profile:

|  |  |
| --- | --- |
| **Use Case** | Change profile. |
| **Actor** | User, Instructor. |
| **Brief Description** | Describes how users can change their information. |
| **Main Flow** | 1. The system displays the detail information of the actor.  2. The actors make the desired change.  3. System saves the change and update into database. |
| **Alternative Flows** | Invalid input: If actor entered invalid type of input, the system displays and highlights the errors. |
| **Pre-condition** | Actor have done log in state and clicked change profile bottom. |
| **Post-conditions** | If the use case is successful, the actor’s information is updated. If not, the system is unchanged and display the actor’s information. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.3 Course searching:

|  |  |
| --- | --- |
| **Use Case** | Course searching. |
| **Actor** | User, Guest. |
| **Brief Description** | This use-case is used to search one or more particular courses. |
| **Main Flow** | 1. Actors click on “search course” in homepage.  2. System shows form to search, including limitation.  3. Actors fill the form and click “submit”.  4. System shows the list of course that contain actors’ search keyword. |
| **Alternative Flows** | 1. User click on the course in the list, system turn into view course.  2. If there are no course in the list, send the message.  3. Actors fill the form again and move to stage 4. |
| **Pre-condition** | Actors had finished log in stage. |
| **Post-conditions** | None. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.4 Course enrolling:

|  |  |
| --- | --- |
| **Use Case** | Course enrolling. |
| **Actor** | User. |
| **Brief Description** | Used to enroll a course in the system. |
| **Main Flow** | 1. Users click on the “enroll” bottom.  2. System show the register form.  3. Users compete the form and submit.  4. System show the list of satisfied courses. |
| **Alternative Flows** | Invalid satisfied course: send users the message. |
| **Pre-condition** | Users finish log in stage. |
| **Post-conditions** | If use case was successful, this user will be automatically a member of chosen course. |
| **Extension point** | Course searching. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.5 Homework uploading:

|  |  |
| --- | --- |
| **Use Case** | Homework uploading. |
| **Actor** | Users. |
| **Brief Description** | User shall upload the homework of each lesion during the course for Instructor’s grading and Instructor’s grading at the end of the course. |
| **Main Flow** | 1.System shows the “upload” bottom in each lesion.  2.Users choose file to upload and click “submit”. |
| **Alternative Flows** | Cancel uploading: If users decide to cancel uploading chosen file, the system back into upload page.  If size of chosen file is larger than limited size, system send the error message. |
| **Pre-condition** | User must log in and must be a member of this course. |
| **Post-conditions** | If use case was successful, system sends the message and save the uploaded files.  If system confirms fail, back to upload page. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.6 Tutorial downloading:

|  |  |
| --- | --- |
| **Use Case** | Tutorial downloading. |
| **Actor** | User. |
| **Brief Description** | In each enrolled course, users can download tutorial in each lesson of this course |
| **Main Flow** | 1.System show download page.  2.User chooses a file to download.  3.Click download buttom. |
| **Alternative Flows** | Cancel downloading: If user decides to cancel downloading chosen file, system back into download page.. |
| **Pre-condition** | User must finish log in stage.  User must be a member of this course. |
| **Post-conditions** | If this use case was successful, system sends message.  If not, system back into downloading page. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.7 Student certificate confirming:

|  |  |
| --- | --- |
| **Use Case** | Student certificate confirming. |
| **Actor** | User |
| **Brief Description** | Users get the certificate to confirm that they have finished this course. The rate of certificate based on their grade during the course, graded by Instructor. |
| **Main Flow** | 1. System shows the page to get the certificate online.  2. User click “save” bottom to save this certificate.  3. User can get the certificate offline by fill in the form and contact to Administrator. |
| **Alternative Flows** | If input is invalid, system sends message. |
| **Pre-condition** | User must finish log in stage.  User must be a member of this course.  User must finish all lesson of this course, by other words, they have to finish the course. |
| **Post-conditions** | If use case was successful, system sends message to thank user for enrolling the course and congratulate them finishing this course. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.8 Course feedback:

|  |  |
| --- | --- |
| **Use Case** | Course feedback. |
| **Actor** | User. |
| **Brief Description** | After finishing a course, user shall send feedback to Instructor and Administrator. |
| **Main Flow** | 1.System shows sending feedback page.  2.User fill in the form, including course quality, Instructor behavior, homework grading,etc…  3. User click “submit” buttom. |
| **Alternative Flows** | If user haven’t finished the course yet, system sends message that they can’t send feedback until they finish the course. |
| **Pre-condition** | User have finish the course. |
| **Post-conditions** | If use case was successful, feedbacks are saved into system. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.9 Course creating:

|  |  |
| --- | --- |
| **Use Case** | Course creating. |
| **Actor** | Instructor. |
| **Brief Description** | Instructor fill in the form about course information: starting time, period, fee, limited users,…Course information will be showed in home page for user’s searching and enrolling. |
| **Main Flow** | 1. System shows course creating page.  2. Instructor fill course information in the form and click “create”.  3. System saves the course and sends Instructor a message.  4. Instructor create lessons and homework for each lessons. |
| **Alternative Flows** | Invalid input: if type of input is invalid, system sends a warning message. |
| **Pre-condition** | Instructor must finish log in stage. |
| **Post-conditions** | If use case was successful, system saves the course and shows course information for Instructor for whether anything need to be changed.  If not, system sends a warning message and back to creating course page. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.10 Homework grading:

|  |  |
| --- | --- |
| **Use Case** | Homework grading. |
| **Actor** | Instructor. |
| **Brief Description** | Instructor downloads uploaded homework by users, grades and saves the grade for deciding users’ certificate in the end of the course. |
| **Main Flow** | 1. System shows list of uploaded files in each lesson.  2. Instructor click on “save” bottom to save it in their PC.  3. Instructor grades homework of each users and update to the system. |
| **Alternative Flows** | Instructor can cancel updating grade, system back to upload page. |
| **Pre-condition** | 1. Instructor must finish log in stage.  2. Homework files must be existed. |
| **Post-conditions** | If use case was successful, system updates grade of each users in each lessons. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.11 Register:

|  |  |
| --- | --- |
| **Use Case** | Register. |
| **Actor** | Guest. |
| **Brief Description** | Use to create a new account. |
| **Main Flow** | 1. On homepage, guest click on “register” bottom.  2. System shows the form including name, date of birth, ID,email, etc…  3. Guest fills in the form and click “submit”.  4. System sends a message to confirm. |
| **Alternative Flows** | Invalid input: system sends a warning message.  If this email is already used to create another account, system sends a warning message. |
| **Pre-condition** | None. |
| **Post-condition** | If guest register to be a student, system automatically turn to interface to users.  If guest register to be an Instructor, this account must be confirmed by Administrator before being activated. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.12 Register confirming:

|  |  |
| --- | --- |
| **Use Case** | Register confirming. |
| **Actor** | Administrator. |
| **Brief Description** | Use to confirm the register of being an Instructor. |
| **Main Flow** | 1. In the home page, administrator click on “pending registration” ( for instructor).  2. System shows the list of unconfirmed accounts.  3. Administrator clicks on valid accounts to activate this account.  4. System sends a message to this account to confirm activating successfully. |
| **Alternative Flows** | Account is out of confirmed date/ account had been deleted: system ignore activating this account and delete the message. |
| **Pre-condition** | Guest registered to be an instructor. |
| **Post-conditions** | None. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

### 1.4.13 User management:

|  |  |
| --- | --- |
| **Use Case** | User management. |
| **Actor** | Administrator. |
| **Brief Description** | Administrator has right to manage all accounts in the system: edit, add, delete. |
| **Main Flow** | 1. In the homepage, administrator clicks on “user management”.  2. System shows the list of users in the system.  3. Administrator chooses one account to interact.  4. System shows activities to interact with this account. |
| **Alternative Flows** | If list is empty, system sends a warning message.  Administrator chooses to delete user, system sends confirm message, delete this user and back to the user management page.  Administrator chooses to modify user profile, System turns to edit profile of user (run as administrator).  If editing is invalid, system sends a warning message and back to user management page. |
| **Pre-condition** | Administrator must finish log in stage. |
| **Post-conditions** | None. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

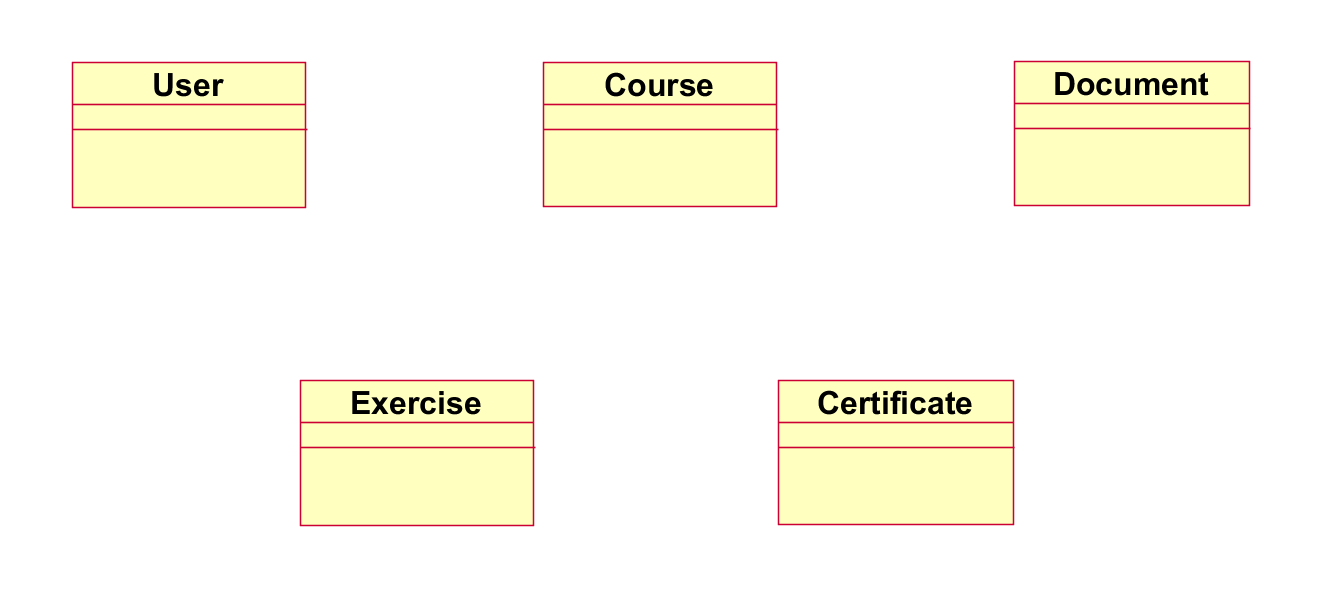
### 1.4.14 Course management

|  |  |
| --- | --- |
| **Use Case** | Course management. |
| **Actor** | Administrator. |
| **Brief Description** | Administrator has right to manage all courses in the system: edit, add, delete. |
| **Main Flow** | 1. On the home page, administrator clicks on “course manager”.  2. System shows the list of courses in the system.  3. Administrator chooses a course to interact.  4. System shows all activities to interact with this course.  4.1 If administrator chooses delete course, system confirms, and wait for the confirming of instructor.  4.2 If administrator chooses edit course, system turns to editing course information (run as administrator). |
| **Alternative Flows** | If administrator cancel changing course, back to course management page. |
| **Pre-condition** | Administrator had logged in the system. |
| **Post-conditions** | Save the edited course. |
| **Extension point** | None. |
| **Special Requirements** | 1. Friendly user interface.  2. Delay time less than 5 seconds. |

# II. E-Learning analysis

## 2.1 Architectural analysis:

### 2.1.1 Key abstraction:

****

#### 2.1.1.1 Key abstraction definition:

**Course:** the basic unit of the system.

Analysis mechanism: persistency, security, process control, synchronization.

**Document:**  all the related information in the system, including users information, courses information, homework, grade, etc…

Analysis mechanism: persistency, security.

**User:** Is an account in the system, including: user, guest, instructor, administrator.

Analysis mechanism: persistency, security.

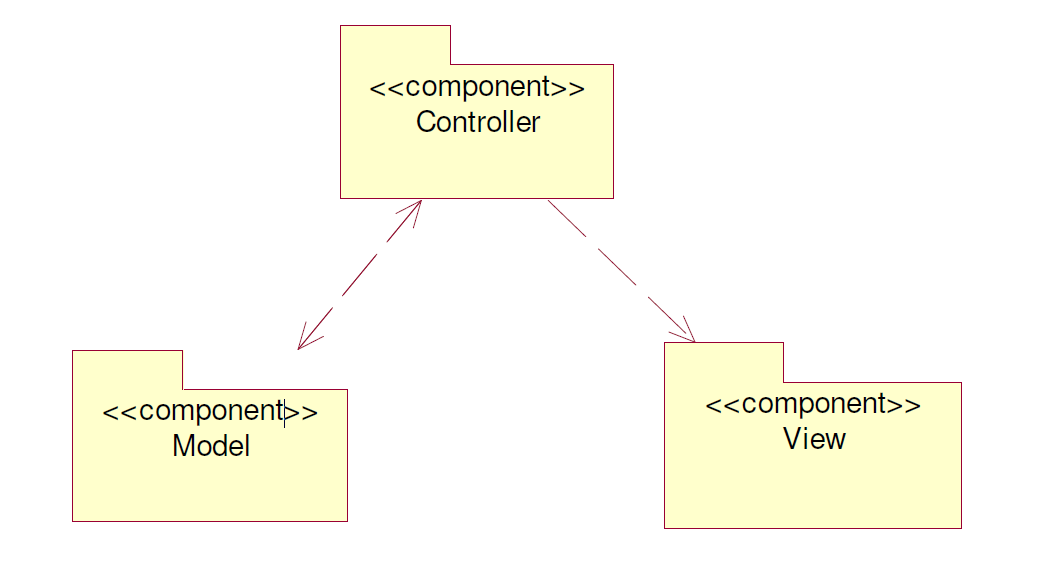
**Exercise :** homework.

Analysis mechanism: persistency, security.

**Certificate:** document to confirm that a user had finished a course in the system.

Analysis mechanism: persistency, security.

### 2.1.2 Upper-Level Components and their dependencies.

****

#### 2.1.2.1 Components definition:

**Controller:** Keep navigation tasks received request from the user and call the accordance methods to handle them. For example, this component will receive the request from URL and form to manipulate directly with database.

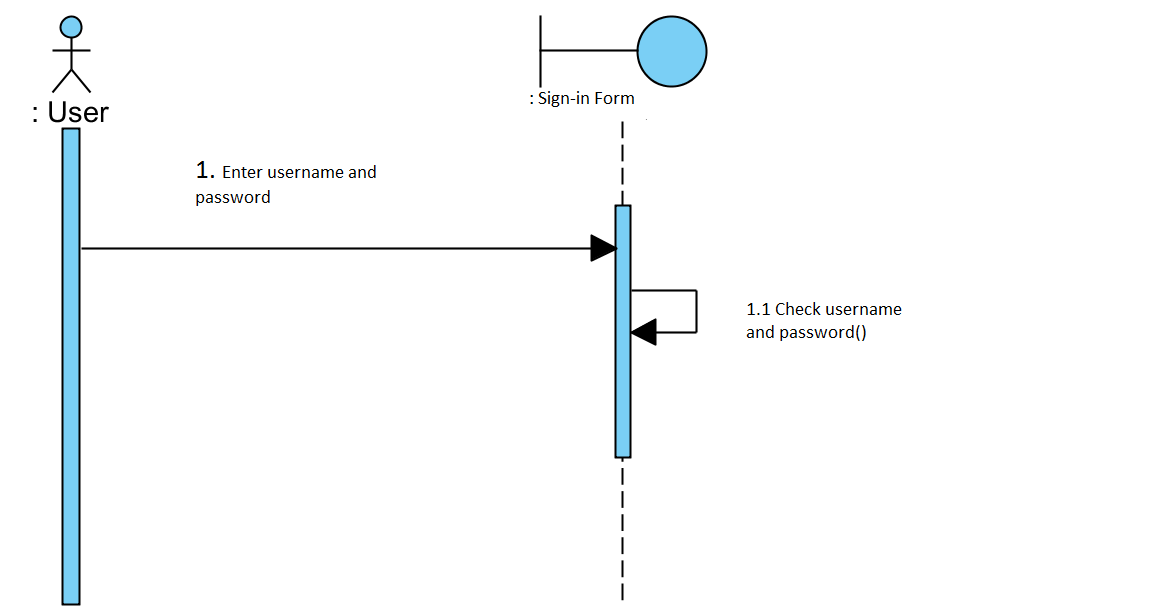
**Model:** This is the component that contains the static data (database), method to access, query and process with it.

**View:** Assume the information display, interaction with users, which contains all the GUI objects such as text boxes, images… By another word, it is a set of form or HTML files.

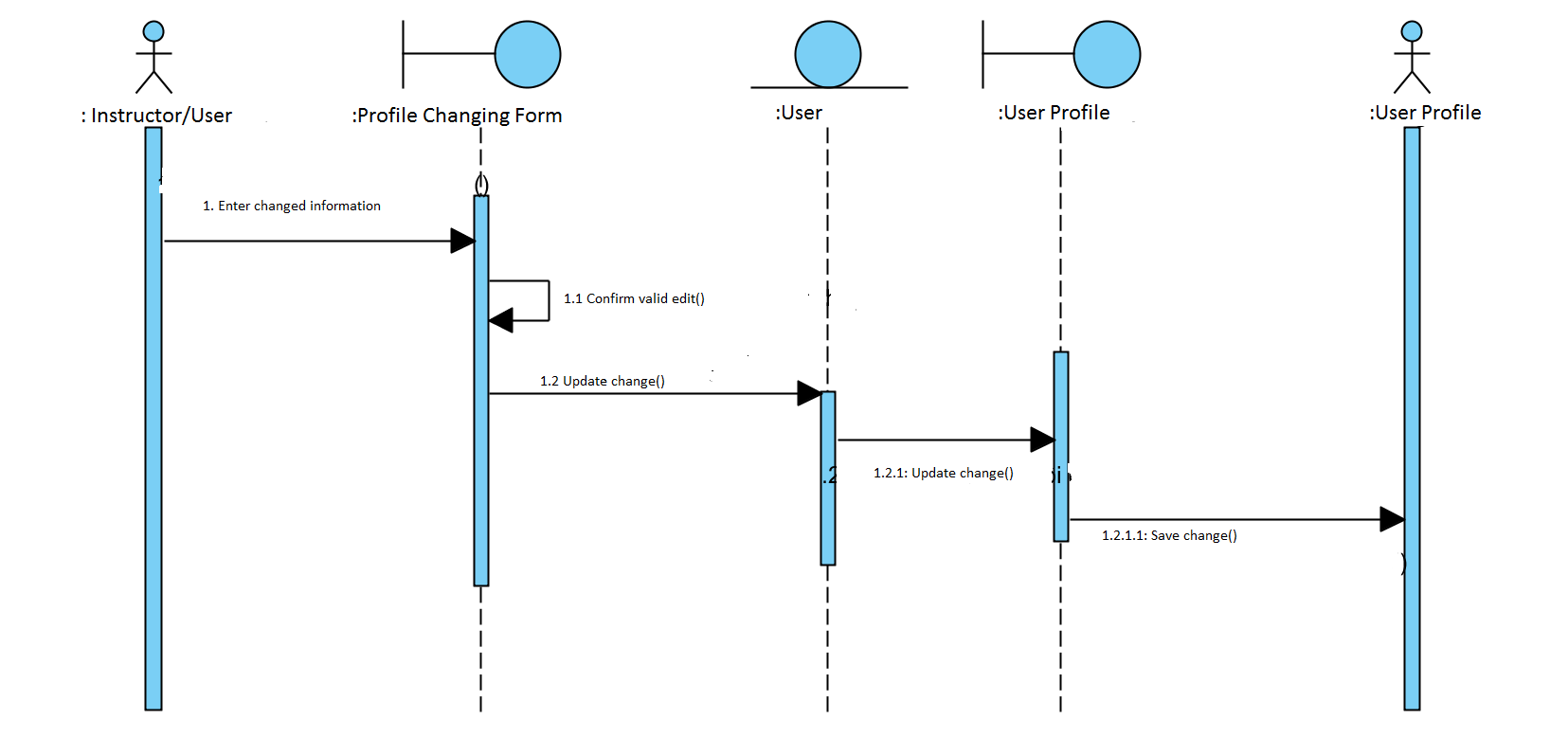
## 2.2 Use case analysis:

### 2.2.1 Use case realization interaction diagram:

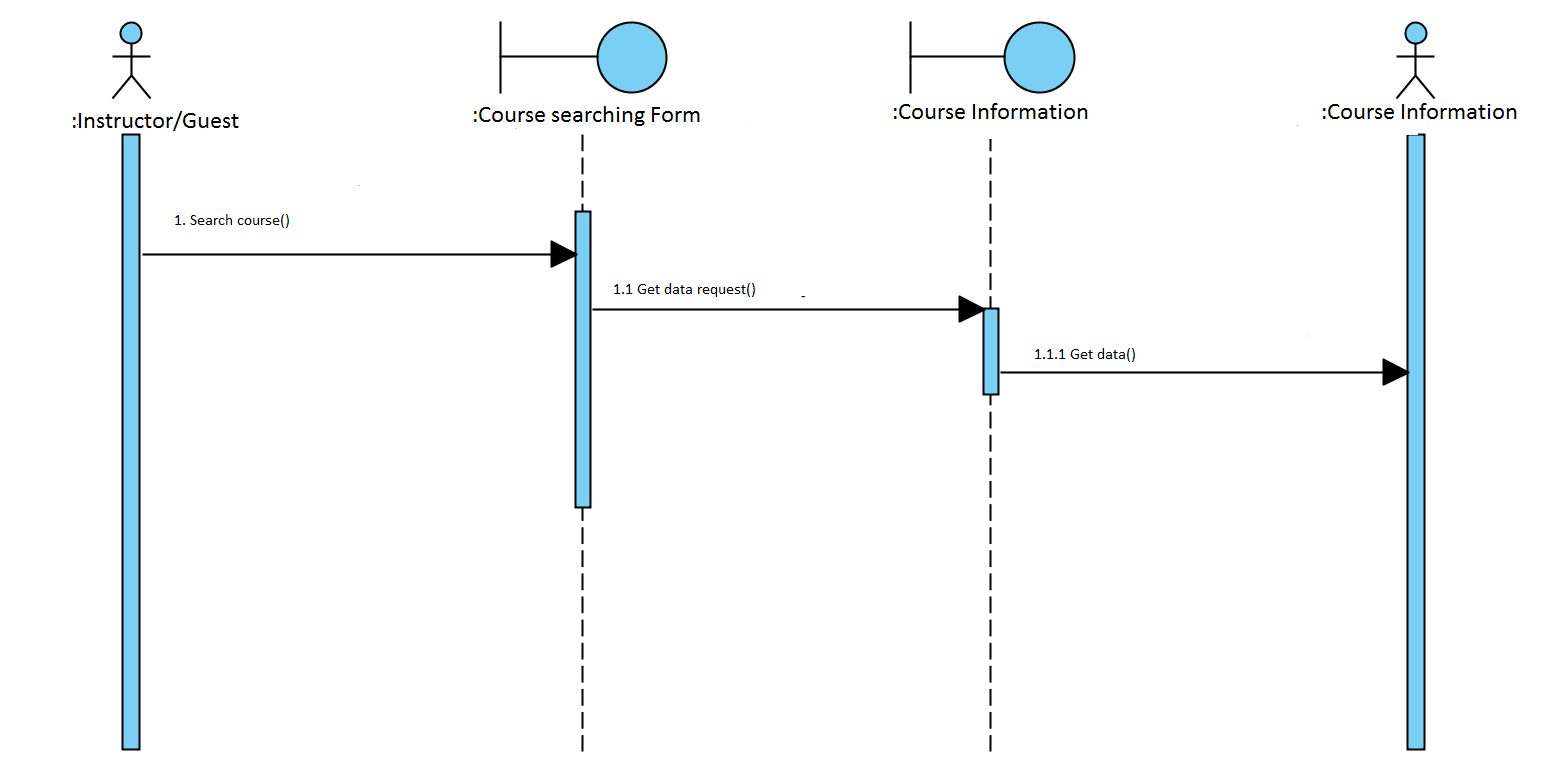
#### 2.2.1.1 Log in:

****

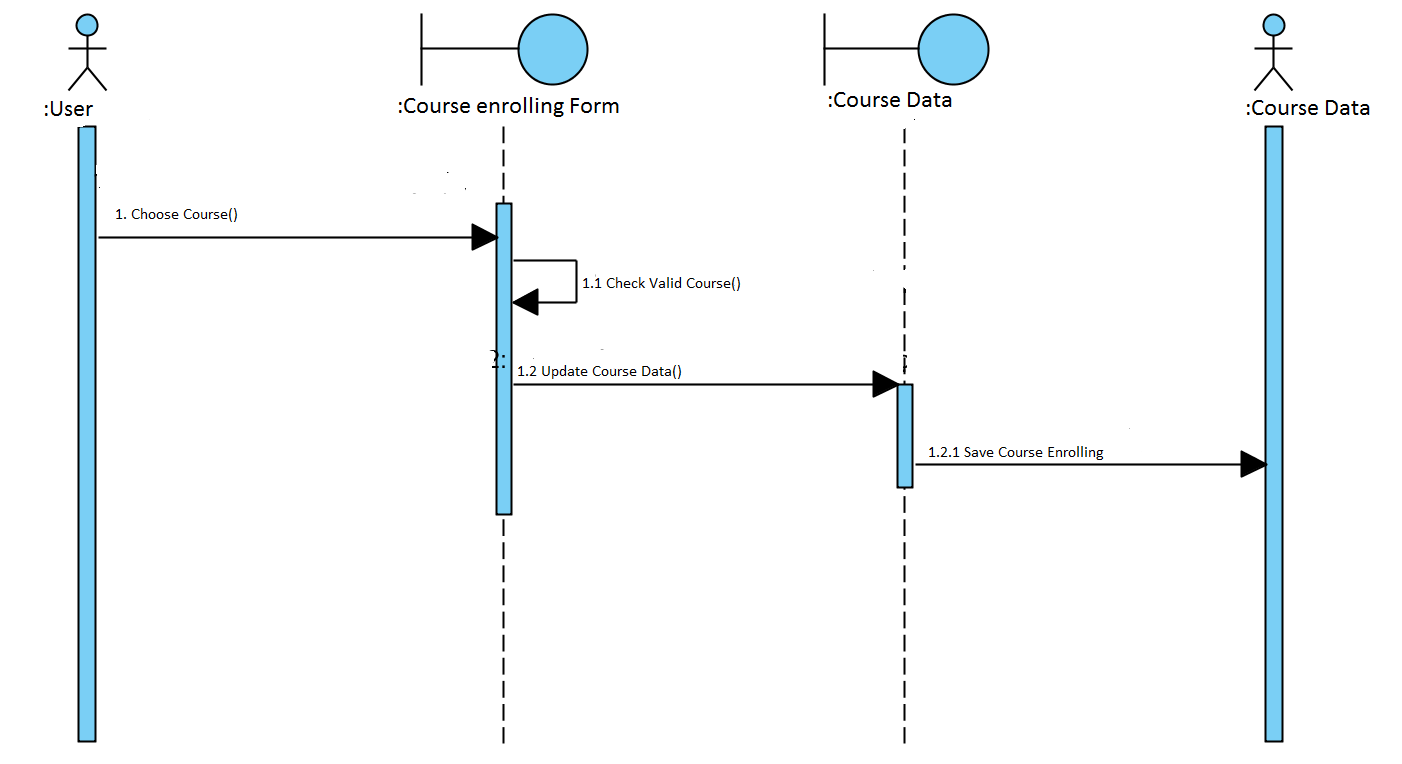
#### 2.2.1.2 Change profile:

****

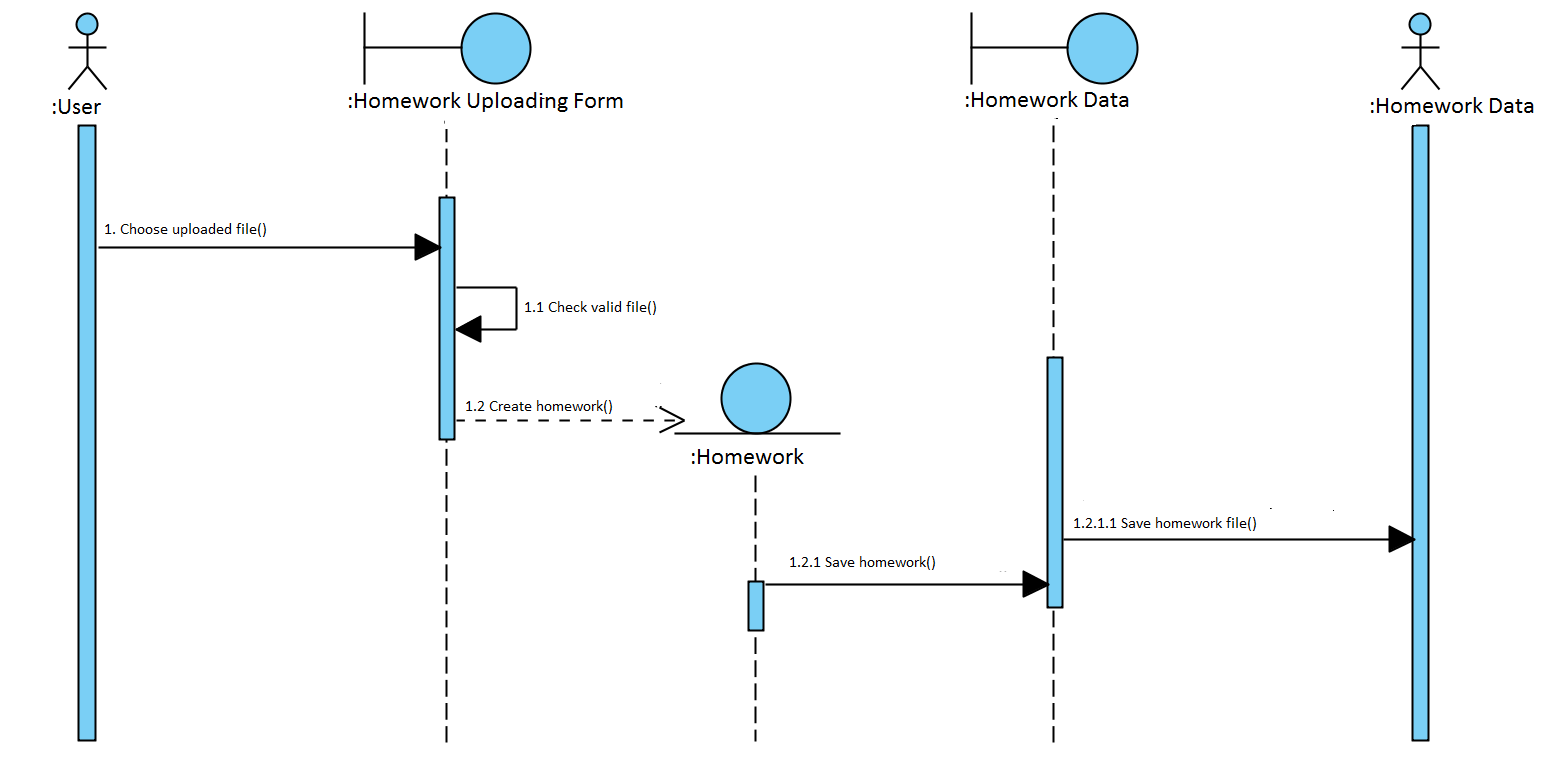
#### 2.2.1.3 Course searching:

****

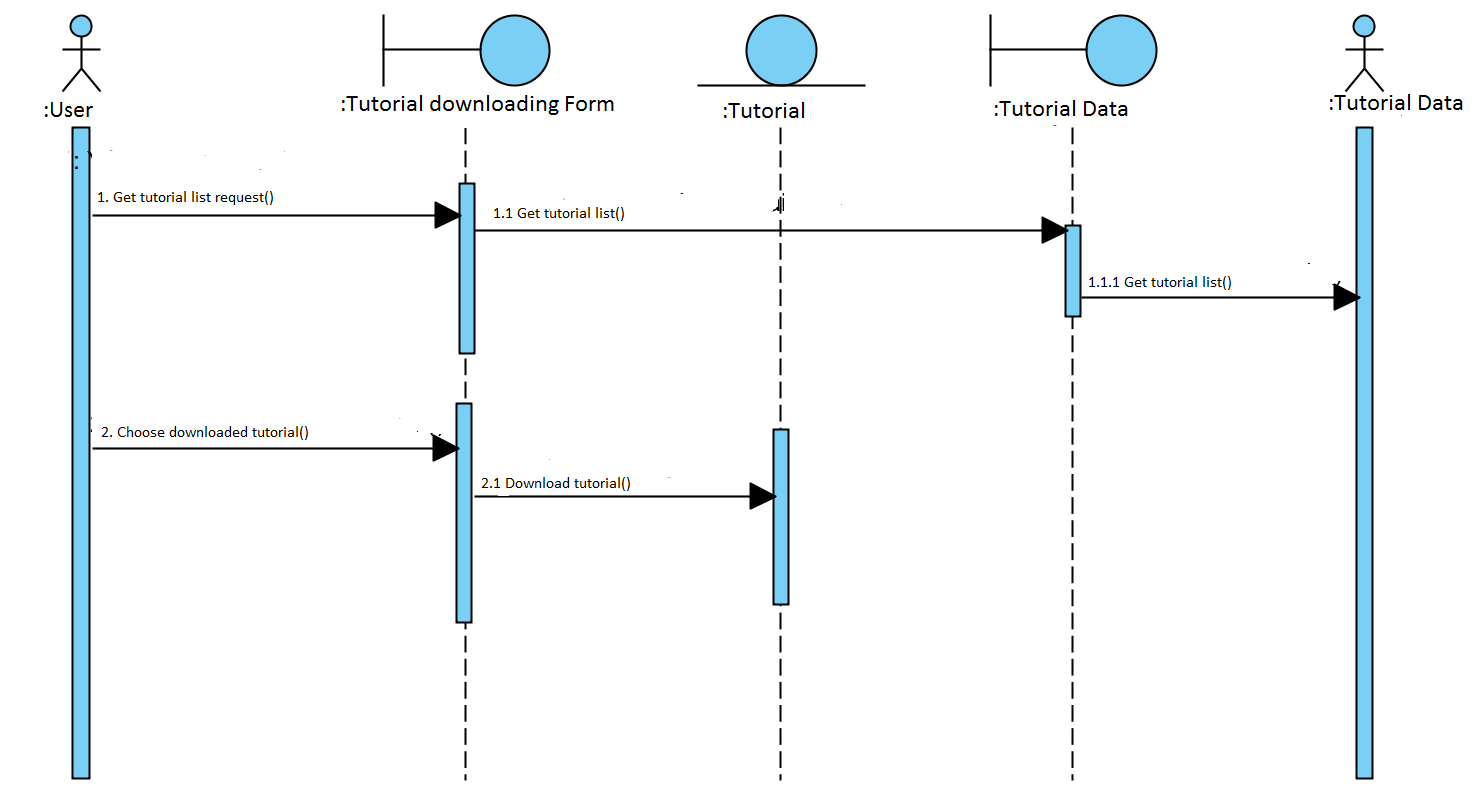
#### 2.2.1.4 Course enrolling:

****

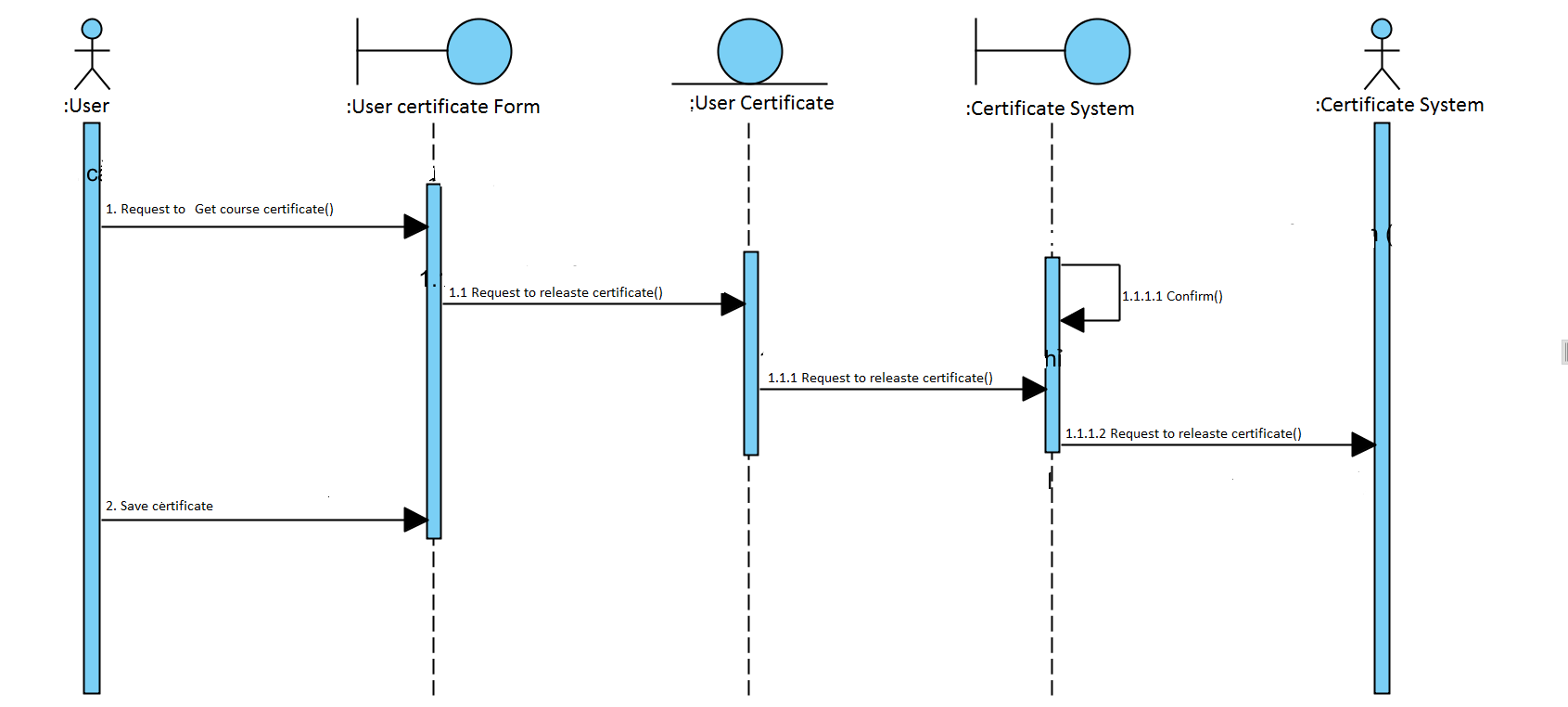
#### 2.2.1.5 Homework uploading:

****

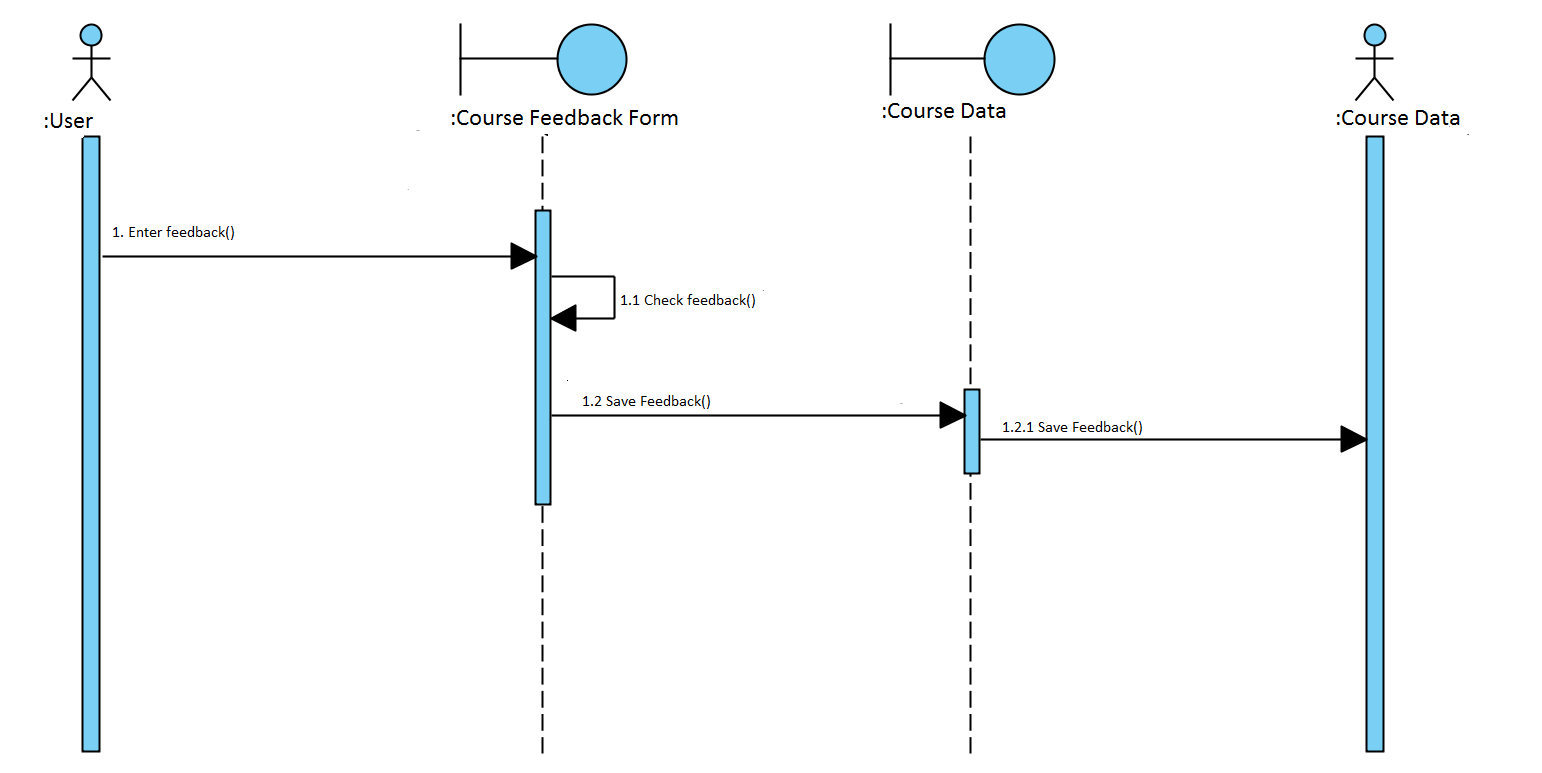
#### 2.2.1.6 Tutorial downloading:



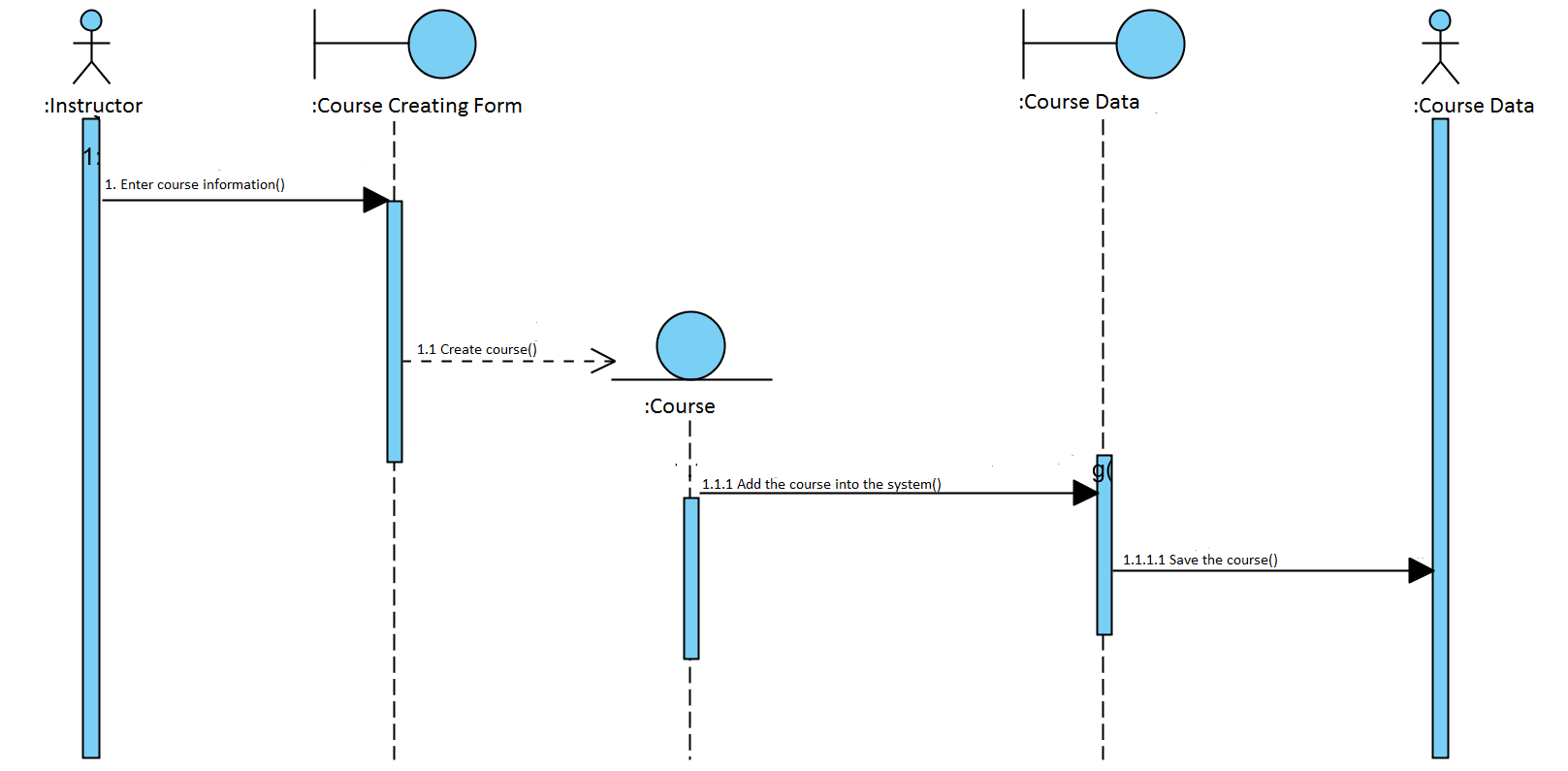
#### 2.2.1.7 Student certificate confirming:

****

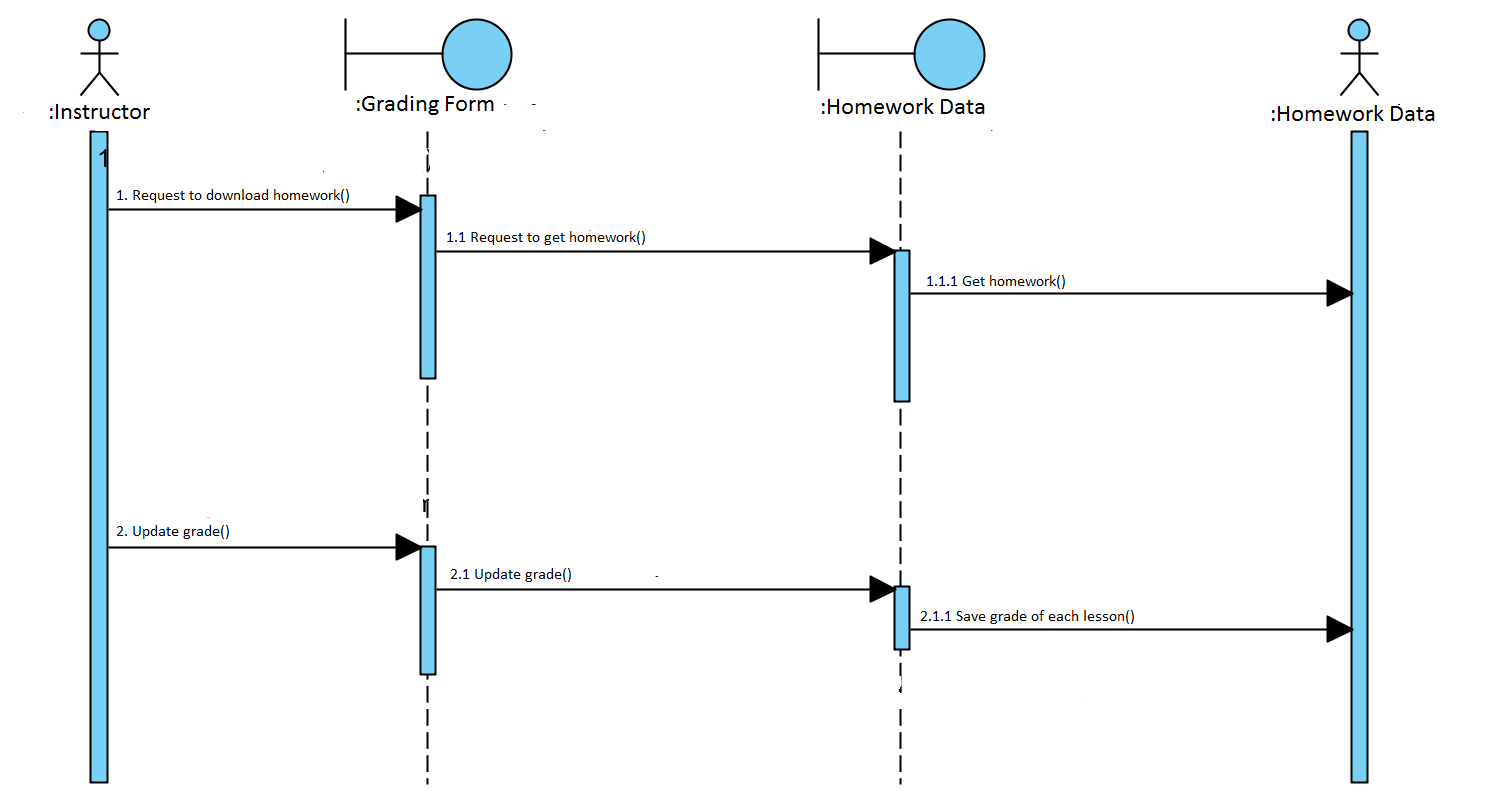
#### 2.2.1.8 Course feedback:

****

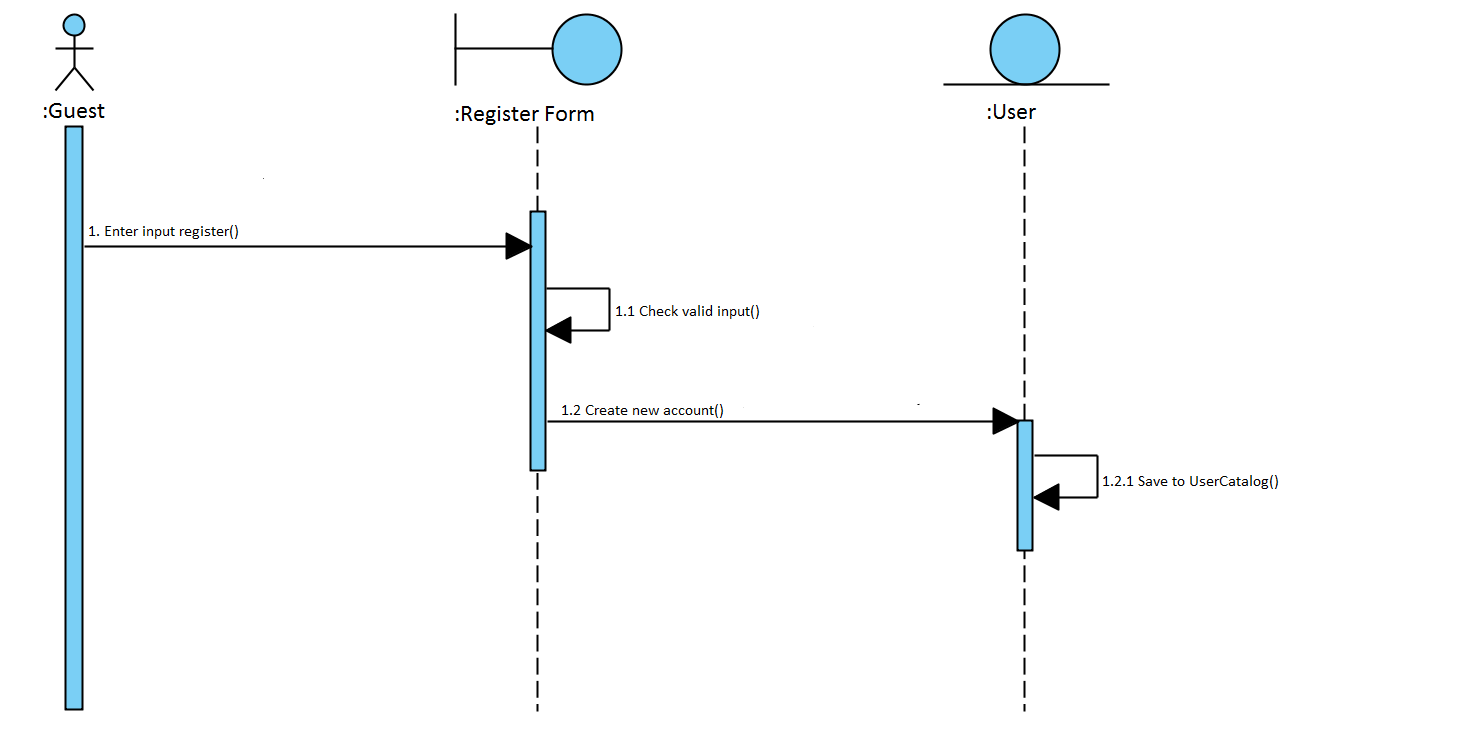
#### 2.2.1.9 Course creating:

****

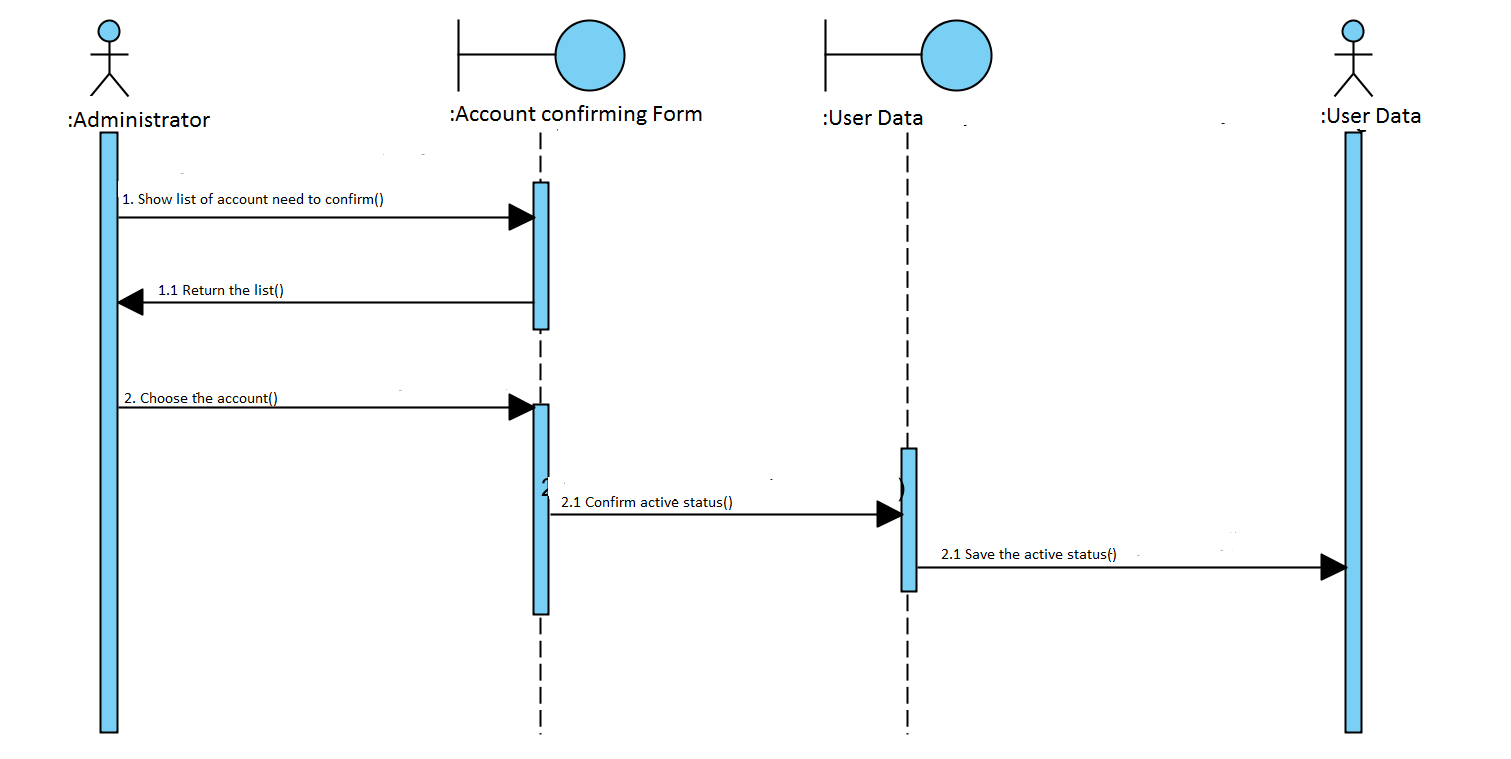
#### 2.2.1.10 Homework grading:

****

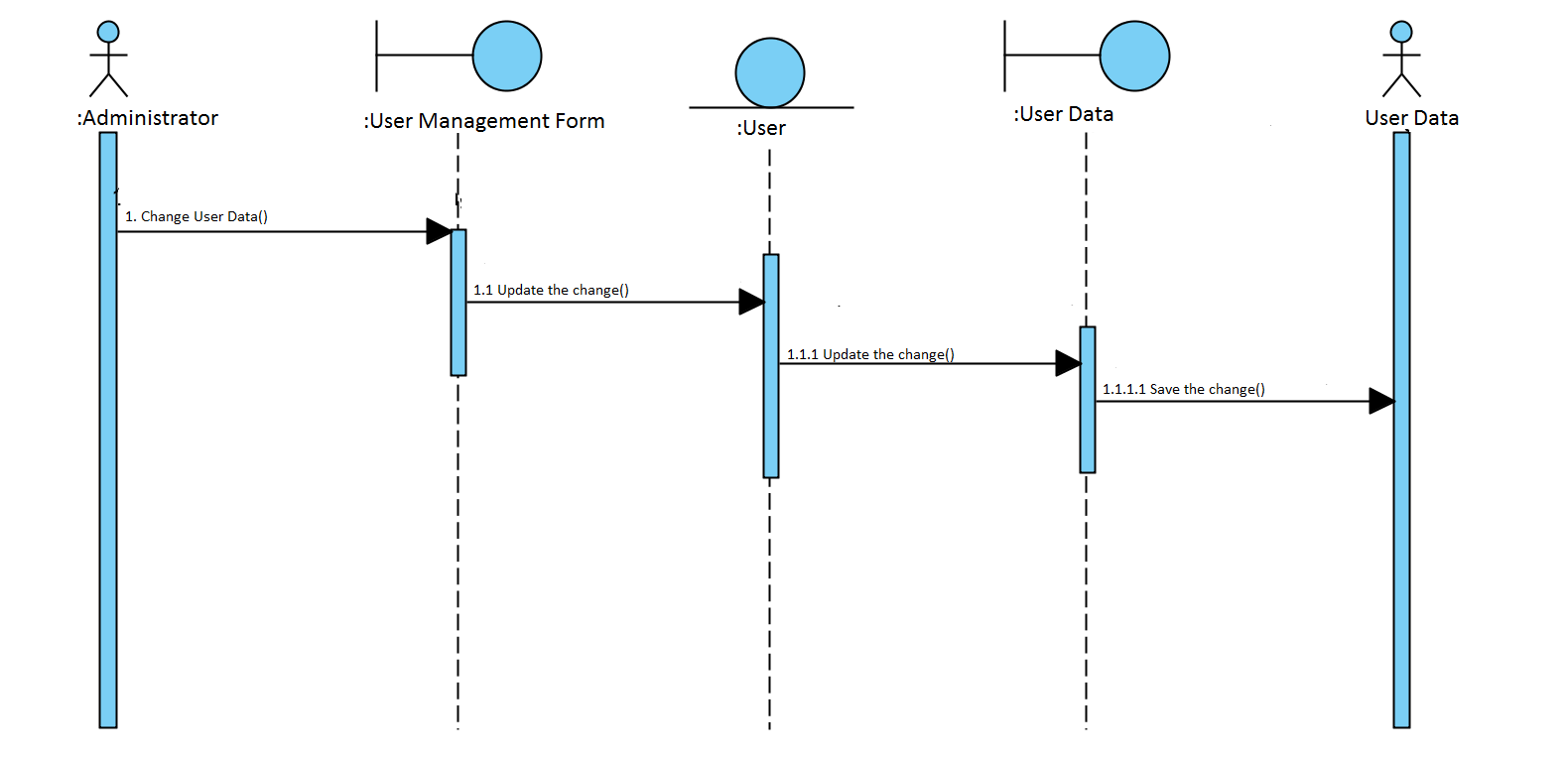
#### 2.2.1.11 Register:

****

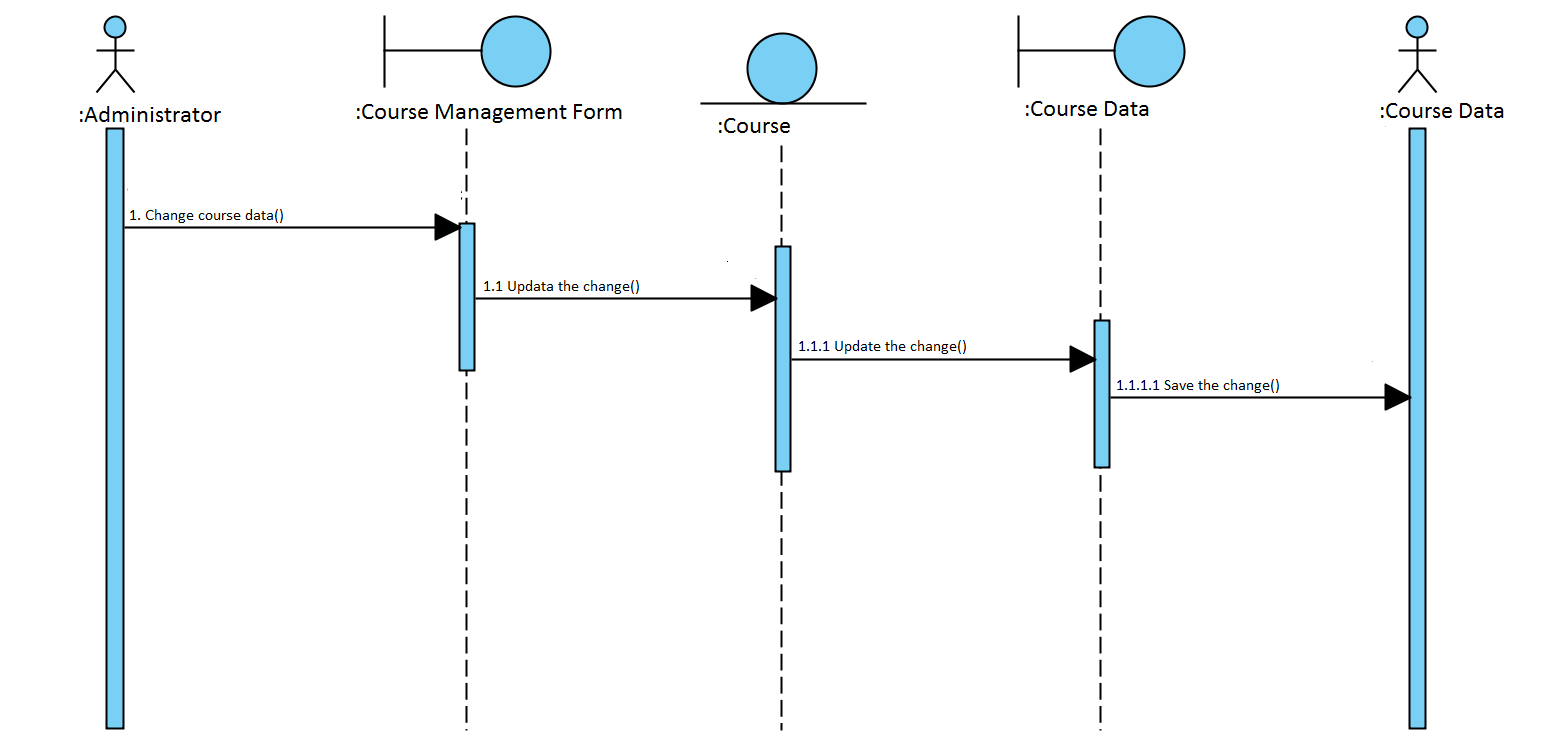
#### 2.2.1.12 Register confirming:

****

#### 2.2.1.13 User management:

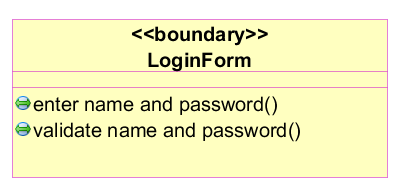
****

#### 2.2.1.14 Course management:

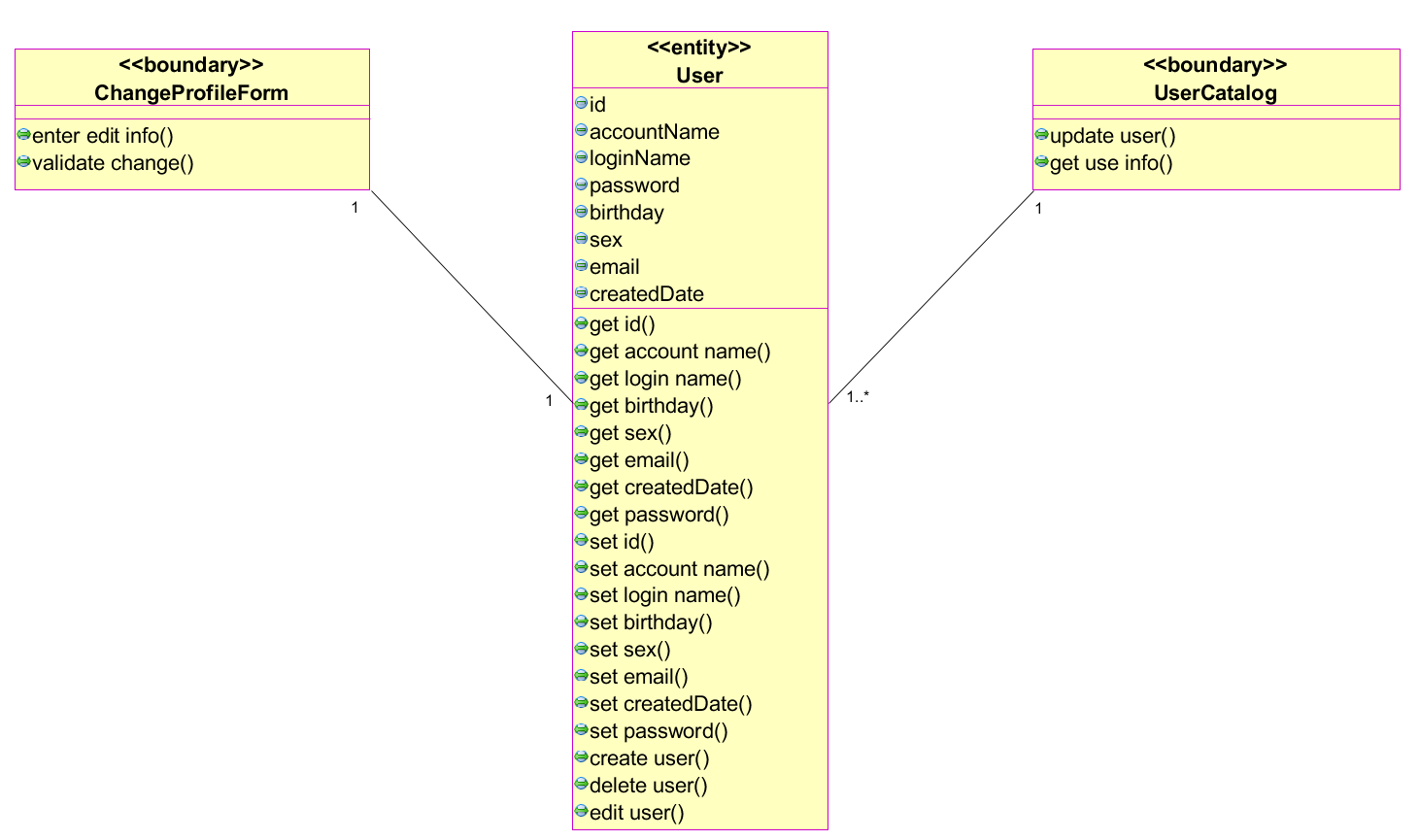
****

### 2.2.2 Use case realization view of participating classes (VOPCs):

#### 2.2.2.1 Log in:



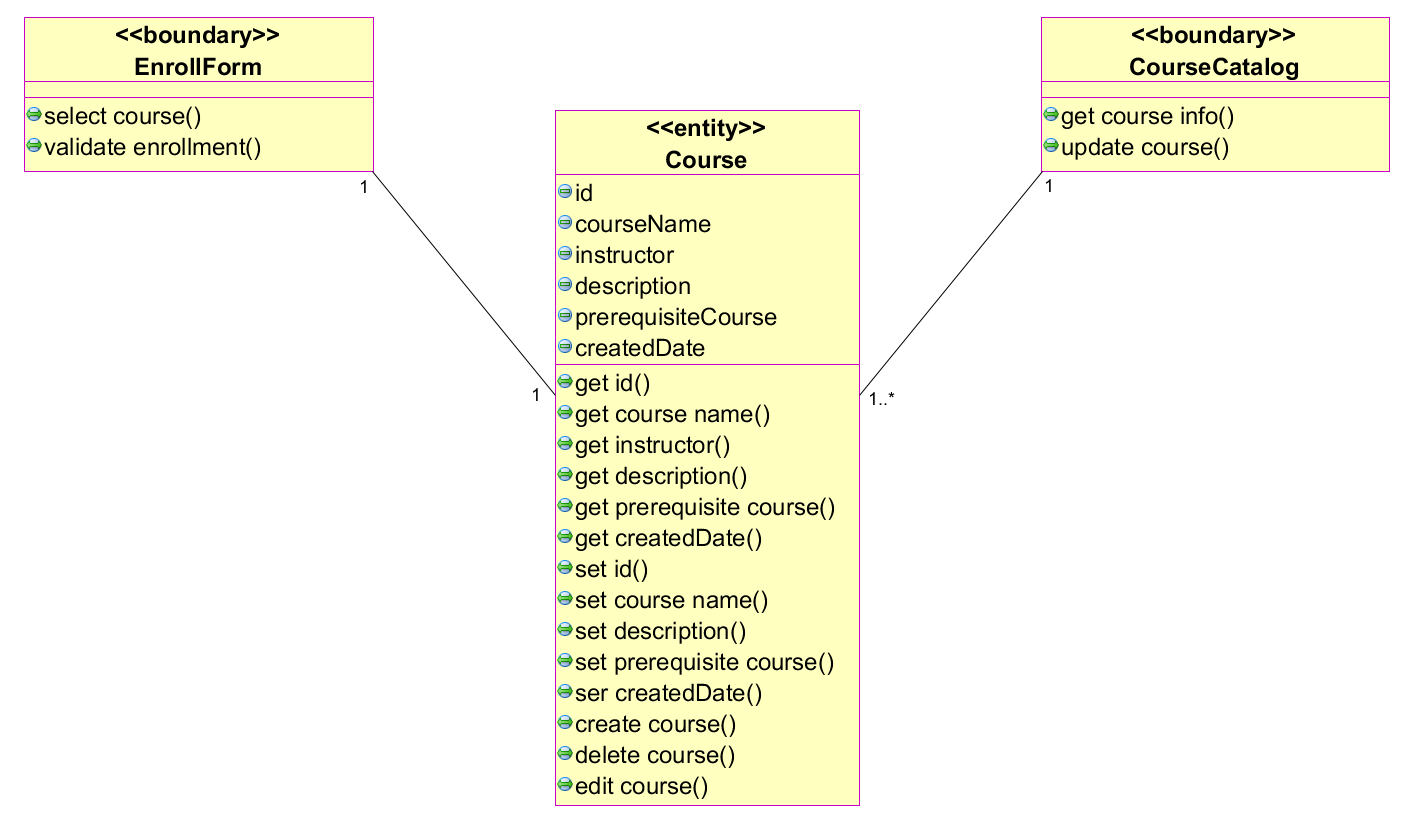
#### 2.2.2.2 change profile:



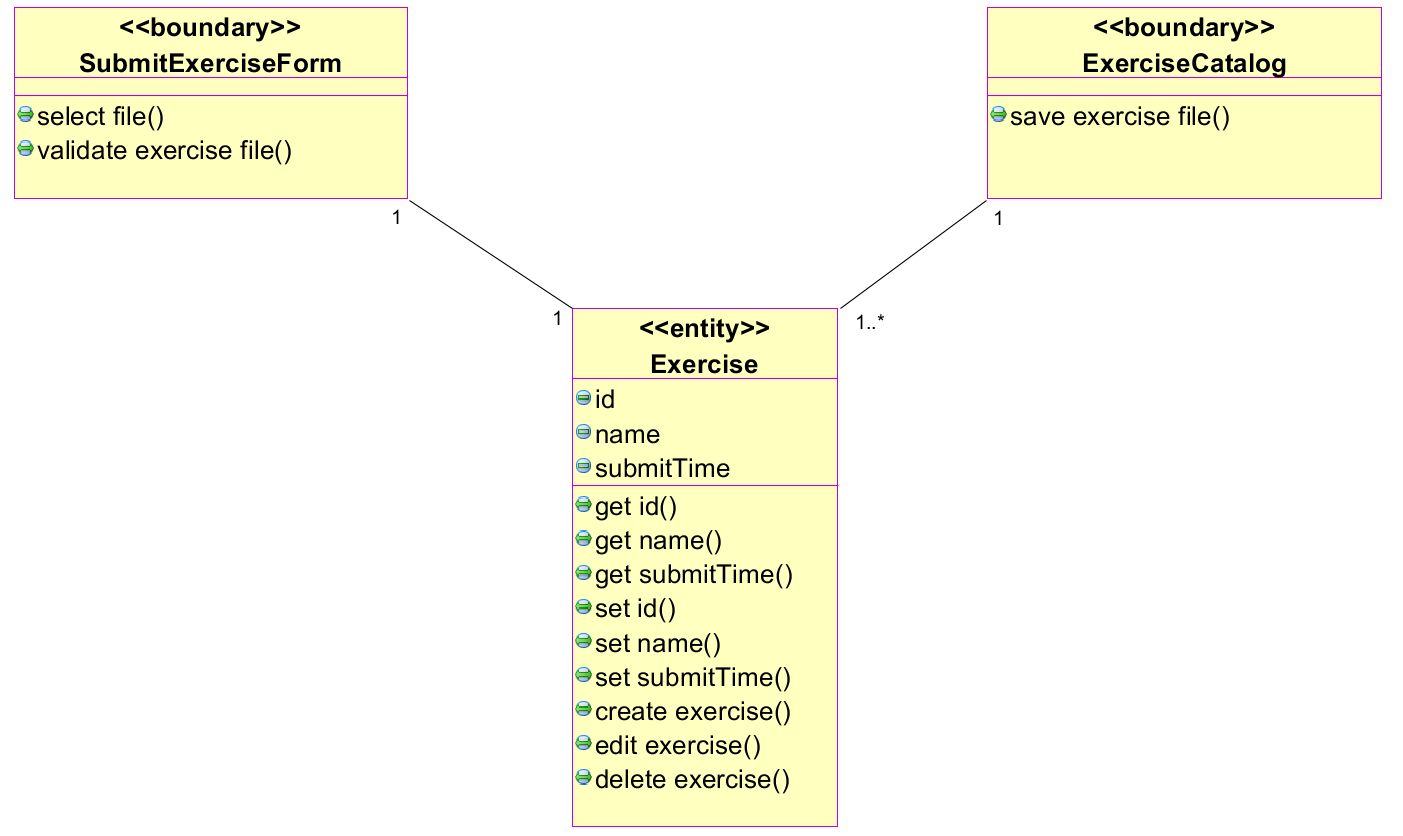
#### 2.2.2.3 Course searching:



#### 2.2.2.4 Course enrolling:

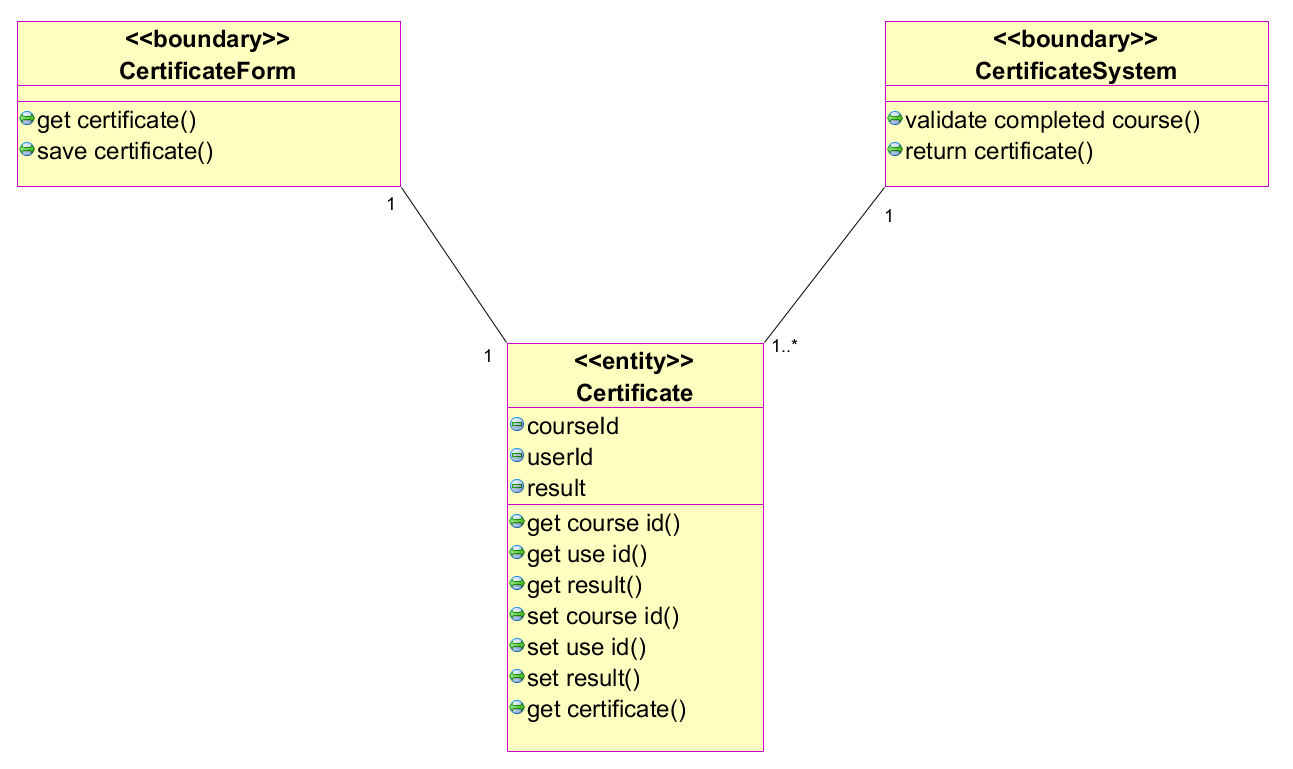
******

#### ***2.2.2.5 Homework uploading:***

******

#### 2.2.2.6 Tutorial downloading:

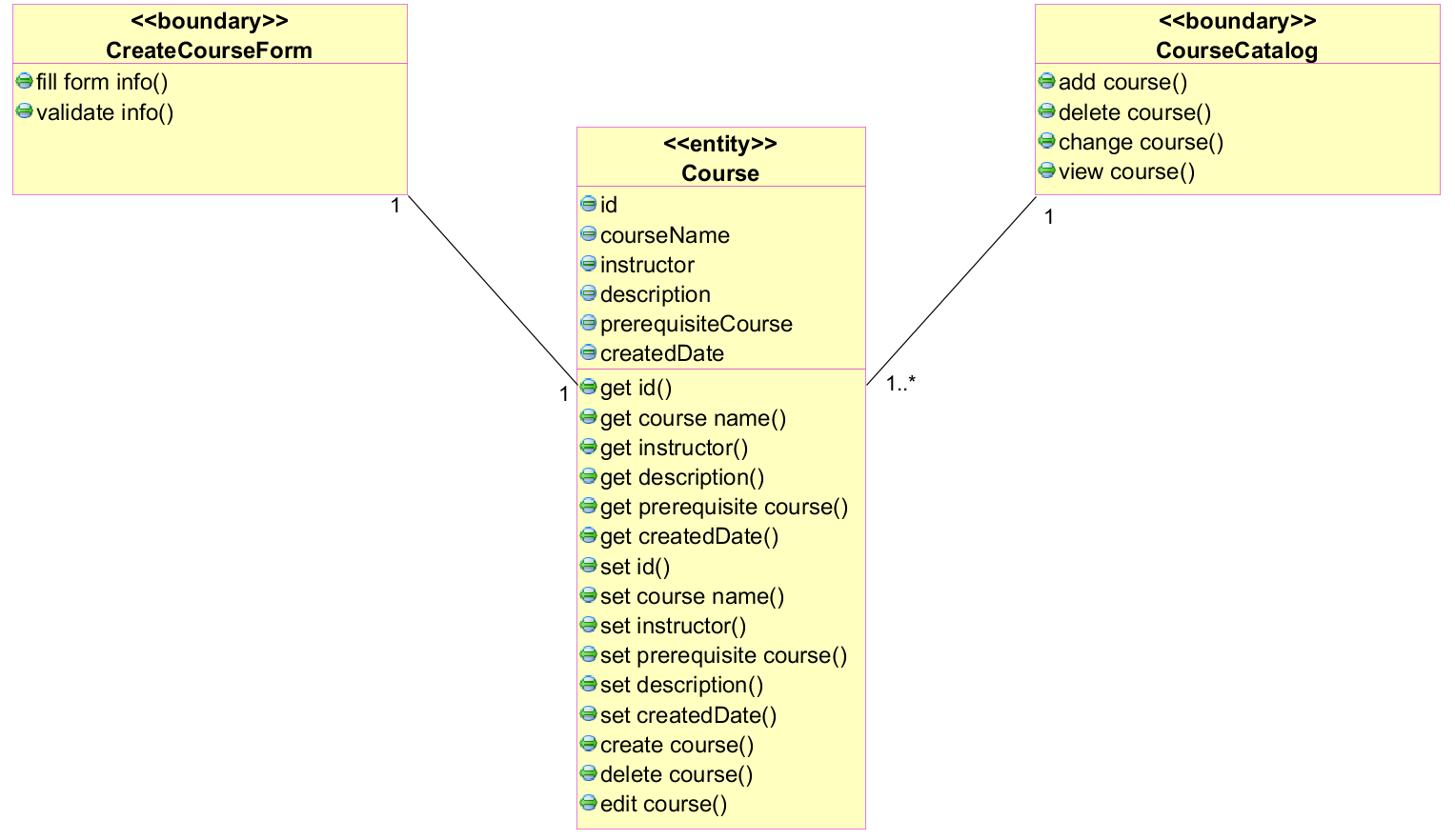
#### 2.2.2.7 Student certificate confirming:



#### 2.2.2.8 Course feedback:



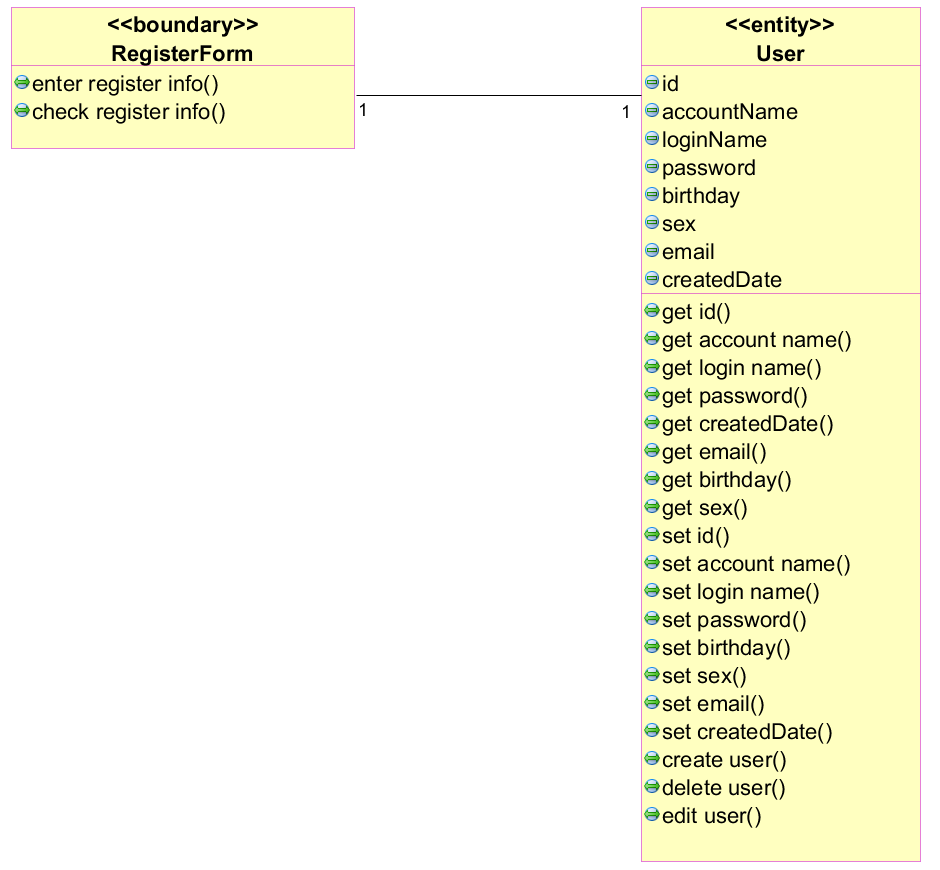
#### 2.2.2.9 Course creating:



#### 2.2.2.10 Homework grading:



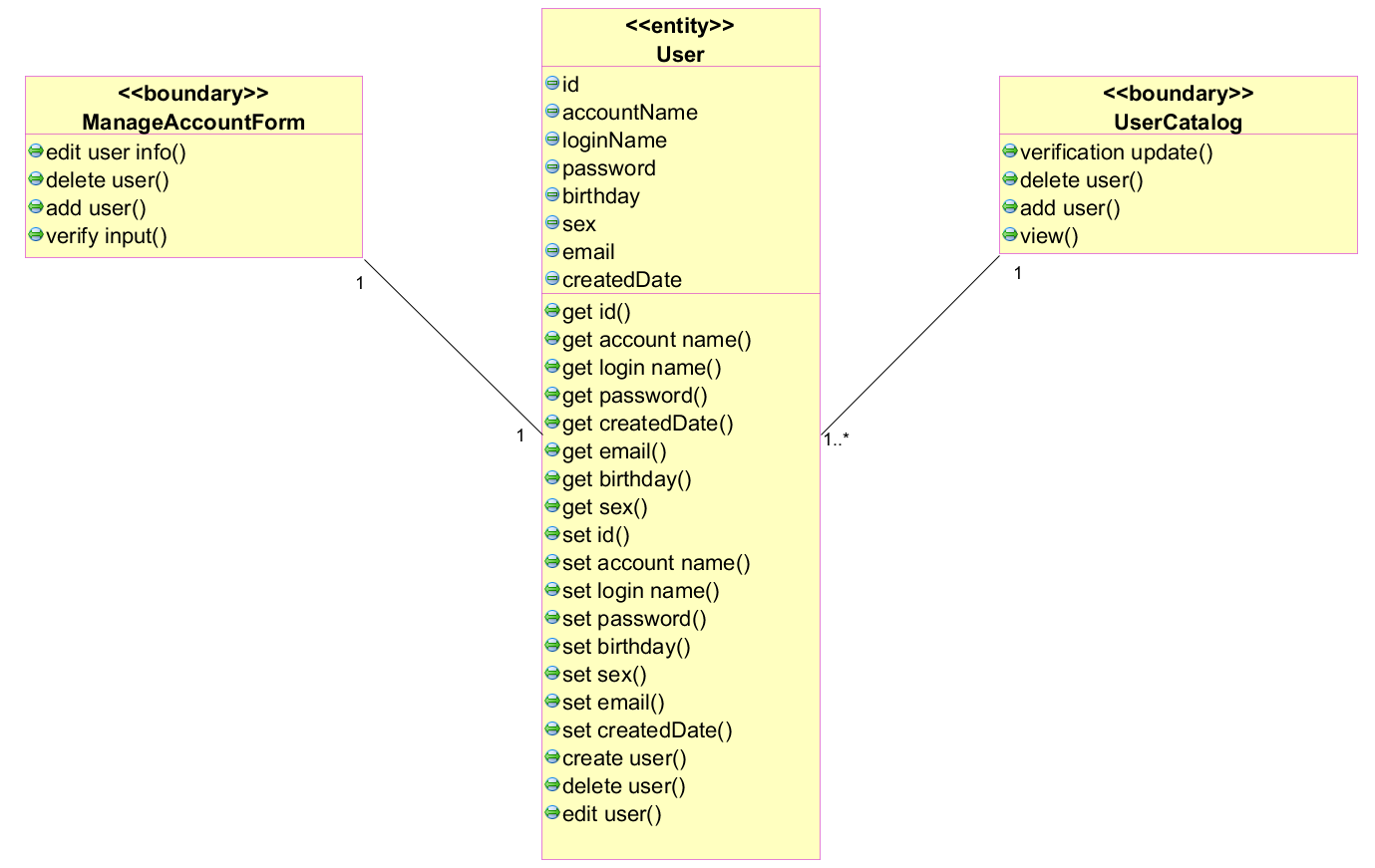
#### 2.2.2.11 Register:



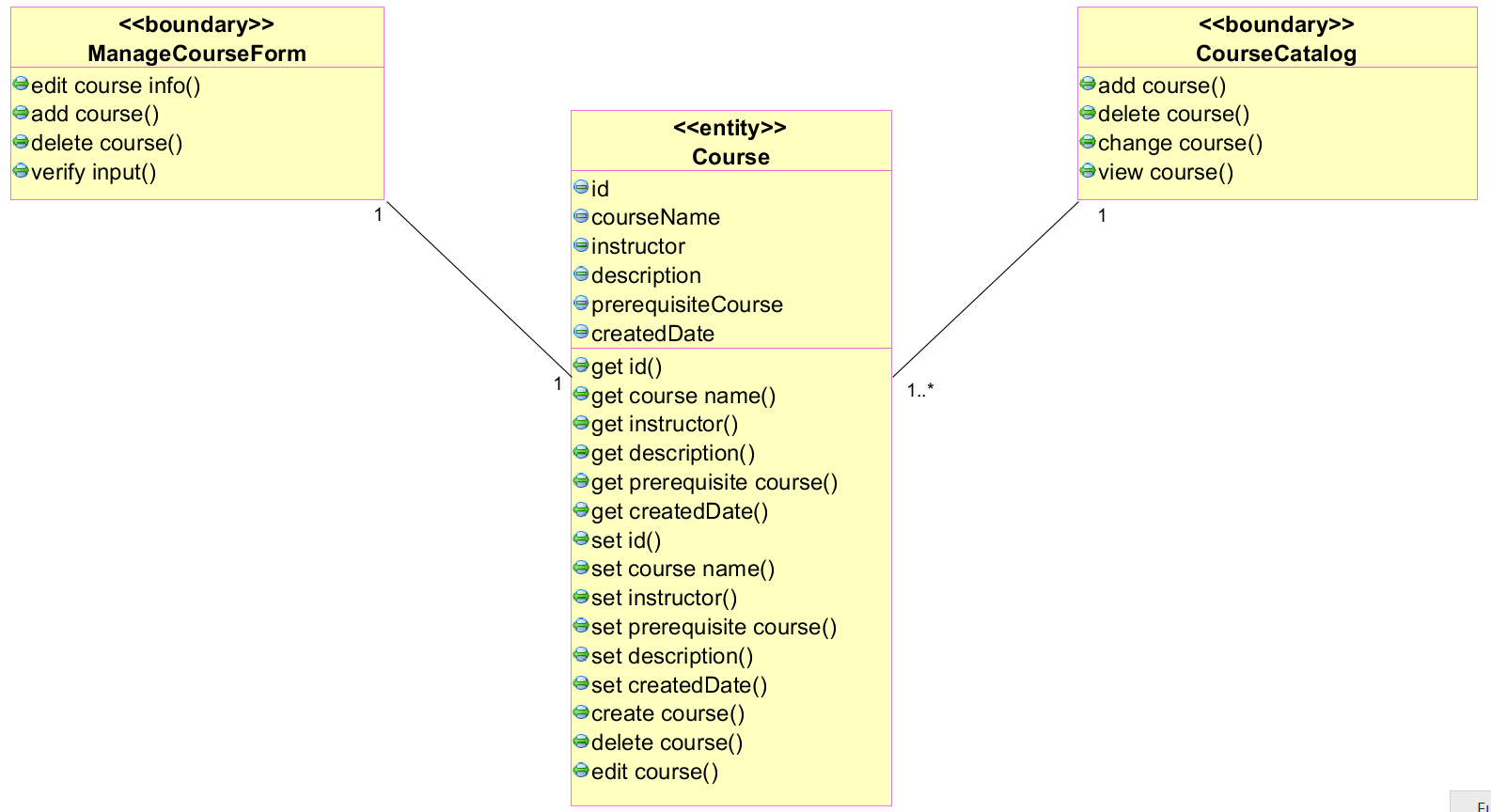
#### 2.2.2.12 Register confirming:



#### 2.2.2.13 User management:



#### 2.2.2.14 Course management:



### 2.2.3 Analysis-Class-To-Analysis-Mechanism Map

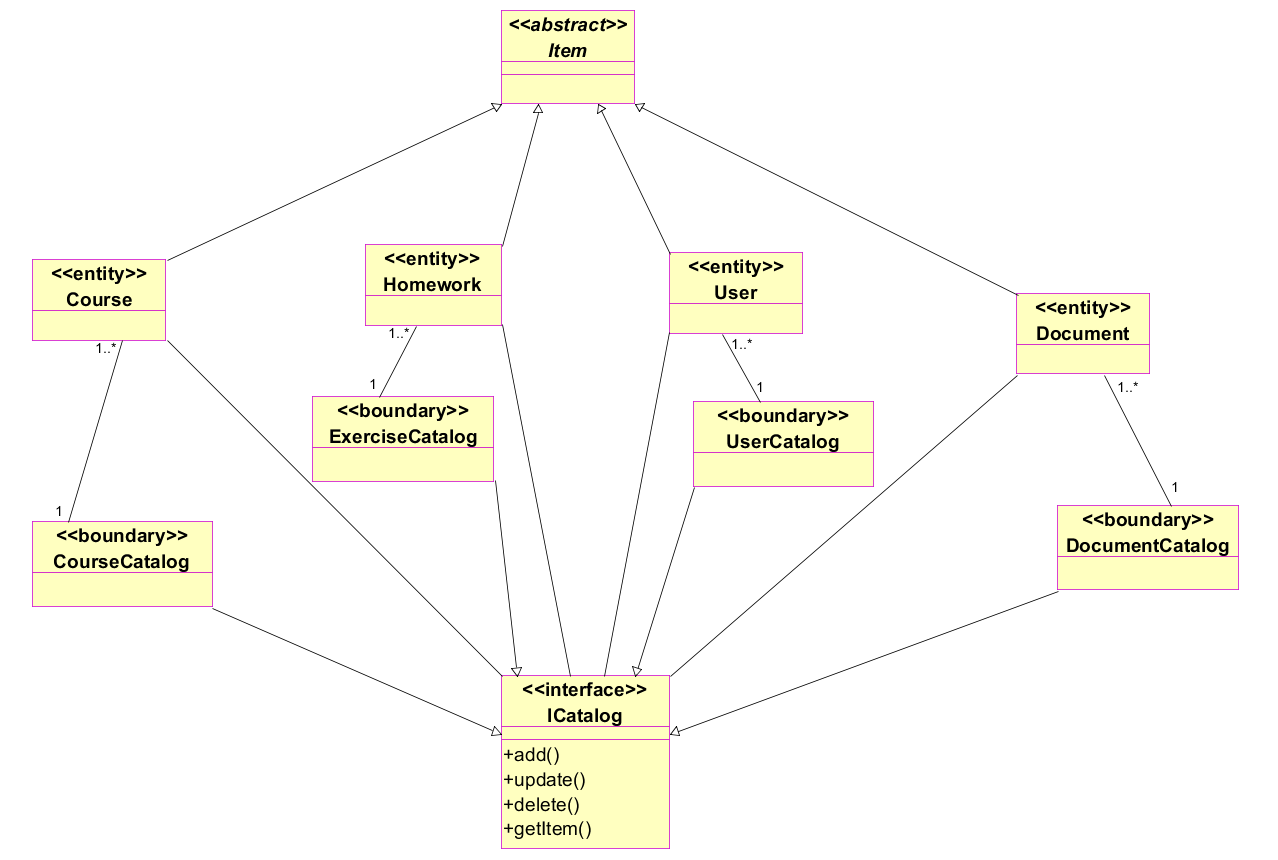
|  |  |
| --- | --- |
| **Analysis class** | **Analysis Mechanism** |
| **LoginForm** | Error detection/handling. |
| **ChangeProfileForm** | Error detection/handling. |
| **User** | Security. |
| **UserCatalog** | Persistency, security. |
| **SearchCourseForm** | None. |
| **CourseCatalog** | Persistency, security. |
| **EnrollForm** | Error detection/handling. |
| **Course** | Security. |
| **SubmitExerciseForm** | Error detection/handling. |
| **Exercise** | Security. |
| **ExerciseCatalog** | Persistency, security. |
| **Document** | Distribution. |
| **DocumentCatalog** | Persistency, security. |
| **DownloadDocumentForm** | Error detection/handling. |
| **CertificateForm** | Error detection/handling. |
| **Certificate** | Security. |
| **CertificateSystem** | Security. |
| **RatingForm** | None. |
| **CreateCourseForm** | Error detection/handling. |
| **AssessExerciseForm** | None. |
| **RegisterForm** | Error detection/handling. |
| **AccountVerificationForm** | Error detection/handling. |
| **ManageAccountForm** | Error detection/handling. |
| **ManageCourseForm** | Error detection/handling. |

# III. E-Learning design:

## 3.1 Identify design element:

### 3.1.1 Subsystem context diagram:

#### 3.1.1.1 Information management system:



System interface description:

ICatalog: encapsulates functions of information objects, including view, edit, add, delete an object

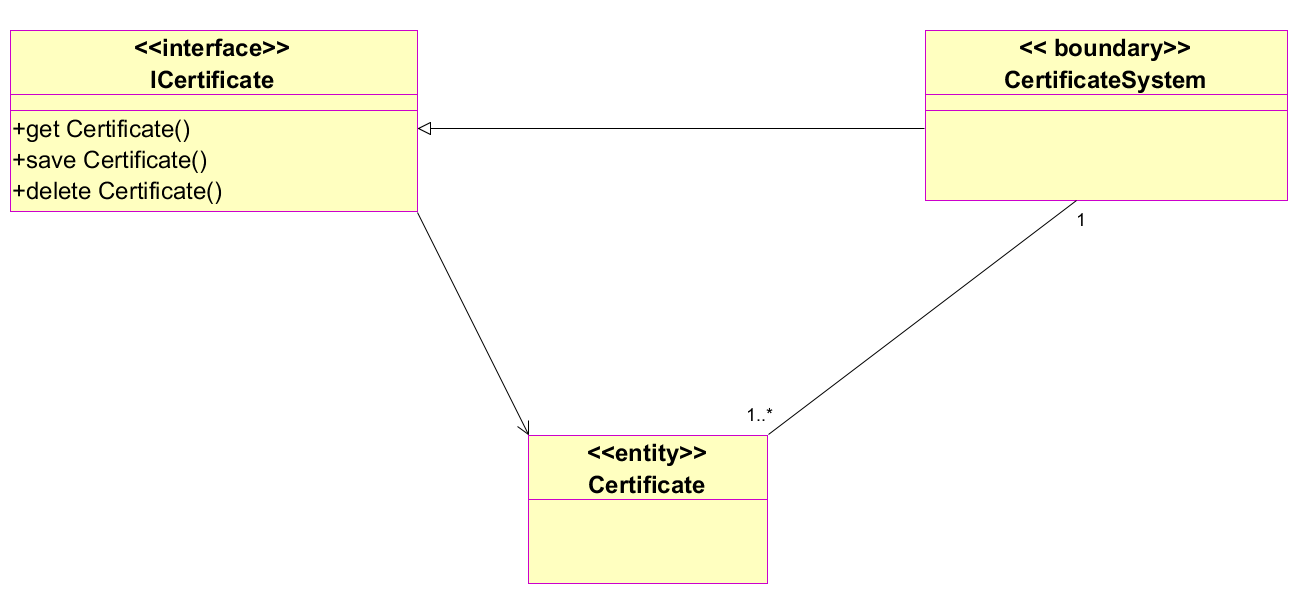
Item: Item may be users, courses, homework, or tutorials.

Add**:** add a new object.

Update: change information of an object.

Delete: delete the object.

#### 3.1.1.2 Student certificate confirming system:



System interface description:

ICertificate: encapsulates the communication with Certificate System.

**Get Certificate**: get the certificate.

**Save Certificate:** save the online certificate.

**Delete Certificate:** delete the online certificate.

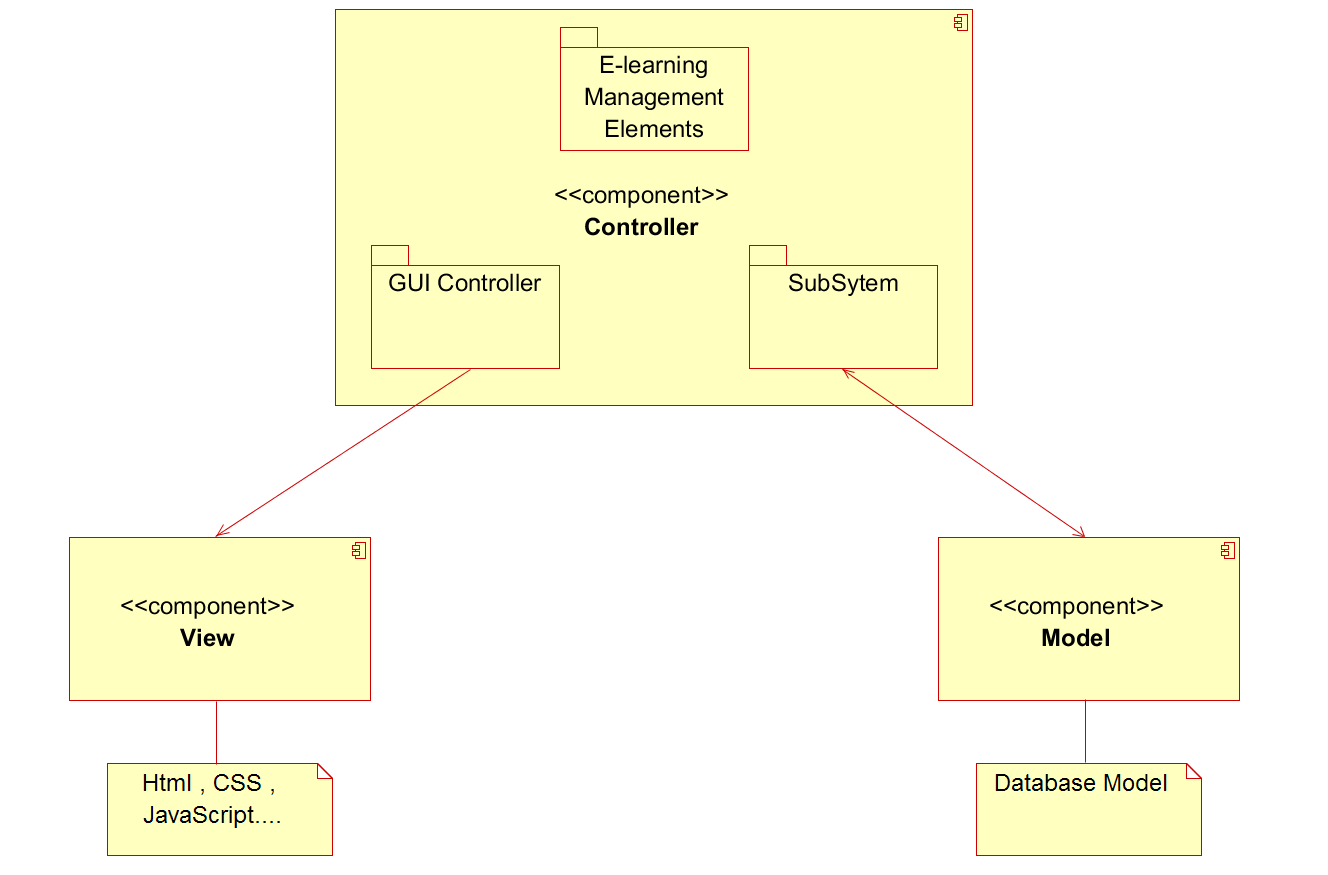
### 3.1.2 Analysis-Class-To-Design-Elements Map:

|  |  |
| --- | --- |
| **Analysis class** | **Design elements** |
| LoginForm | LoginForm |
| ChangeProfileForm | ChangeProfileForm |
| User | User |
| Item ( Abstract) |
| UserCatalog | UserCatalog Subsystem |
| ICatalog |
| SearchCourseForm | SearchCourseForm |
| CourseCatalog | CourseCatalog Subsystem |
| ICatalog |
| EnrollForm | EnrollForm |
| Course | Course |
| Item ( Abstract) |
| SubmitExerciseForm | SubmitExerciseForm |
| Exercise | Exercise |
| Item ( Abstract) |
| ExerciseCatalog | ExerciseCatalog Subsystem |
| ICatalog |
| Document | Document |
| Item ( Abstract) |
| DocumentCatalog | DocumentCatalog Subsystem |
| ICatalog |
| DownloadDocumentForm | DownloadDocumentForm |
| CertificateForm | CertificateForm |
| Certificate | Certificate |
| CertificateSystem | CertificateSystem Subsystem |
| ICertificate |
| RatingForm | RatingForm |
| CreateCourseForm | CreateCourseForm |
| AssessExerciseForm | AssessExerciseForm |
| RegisterForm | RegisterForm |
| AccountVerificationForm | AccountVerificationForm |
| ManageAccountForm | AccountForm |
| AddNewUserForm |
| EditUserForm |
| DeleteUserForm |
| ManageCourseForm | CourseForm |
| AddNewCourseForm |
| EditCourseForm |
| DeleteCourseForm |

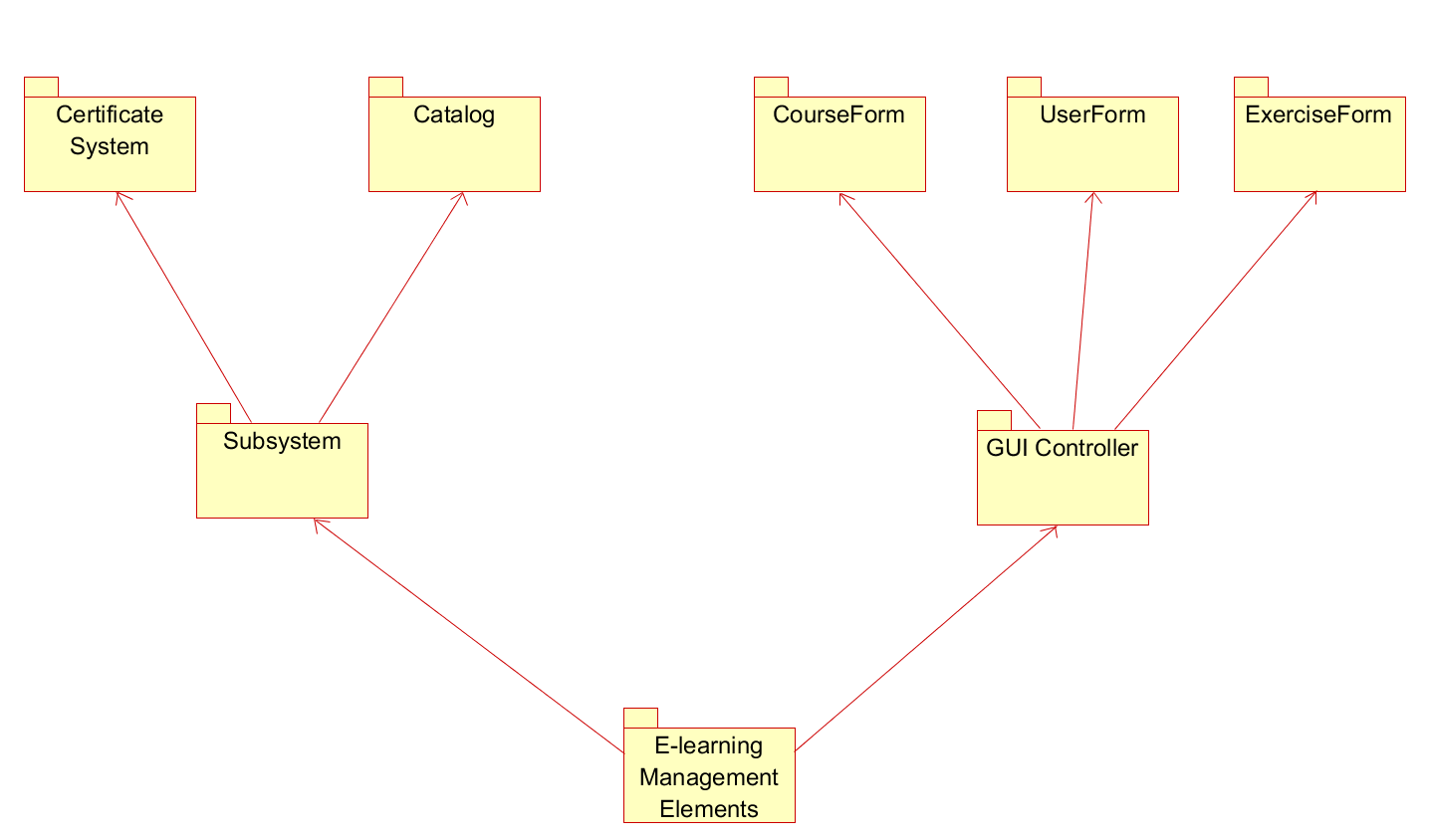
### 3.1.3 Design-Element-To-Owning-Package Map:

|  |  |
| --- | --- |
| **Design element** | **“Owning” package** |
| LoginForm | Controller:: GUI controller. |
| ChangeProfileForm |
| SearchCourseForm |
| EnrollForm |
| SubmitExerciseForm |
| DownloadDocumentForm |
| CertificateForm |
| RatingForm |
| CreateCourseForm |
| AssessExerciseForm |
| RegisterForm |
| AccountVerificationForm |
| AccountForm |
| AddNewUserForm |
| EditUserForm |
| DeleteUserForm |
| CourseForm |
| AddNewCourseForm |
| EditCourseForm |
| DeleteCourseForm |
| Item ( Abstract) | Controller: System elements. |
| User |
| Course |
| Document |
| Certificate |
| Exercise |
| UserCatalog Subsystem | Controller:: subsystem. |
| CourseCatalog Subsystem |
| ExerciseCatalog Subsystem |
| DocumentCatalog Subsystem |
| CertificateSystem Subsystem |
| ICatalog |
| ICertificate |

### 3.1.4 Architectural components and their dependencies:



### 3.1.5 Packages and their dependency:



Package description :

Catalog: contains interfaces : ICatalog, UserCatalog, CourseCatalog, ExerciseCatalog, DocumentCatalog.

Subsystem: import Catalog and Certificate System packages.

CourseForm: contains forms interact with course.

UserForm: contains forms interact with user.

ExerciseForm: contains forms interact with exercise.

GUI Controller: contains interfaces: CourseForm, UserForm, ExerciseForm.

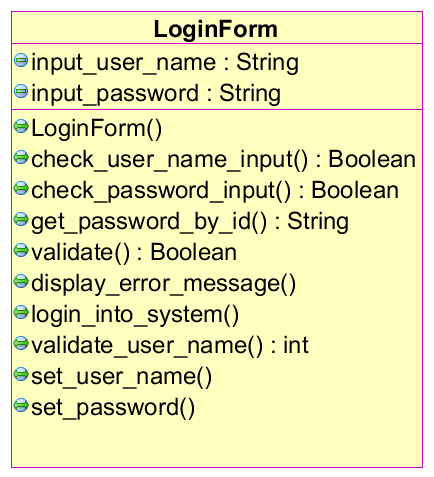
E-learningManagementElements: contains above package and other components of the system.

## 3.2 Class design:

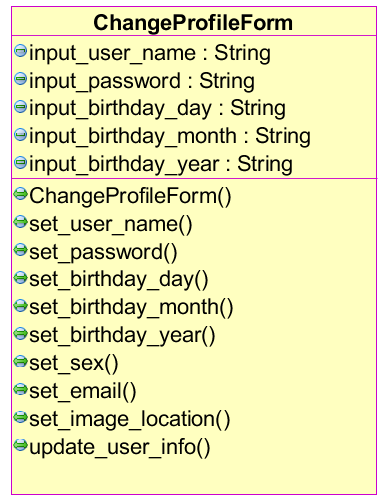
This class describes the all class structure and their description also the total class diagram and their dependency.

### 3.2.1 Describe each class and interface:

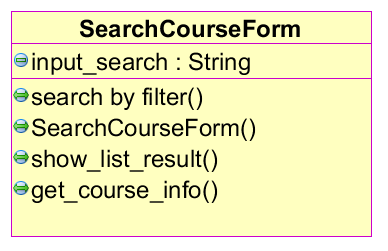
#### 3.2.1.1 LoginForm:



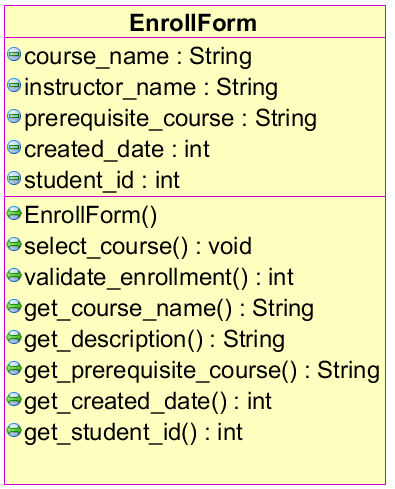
#### 3.2.1.2 ChangeProfileForm:



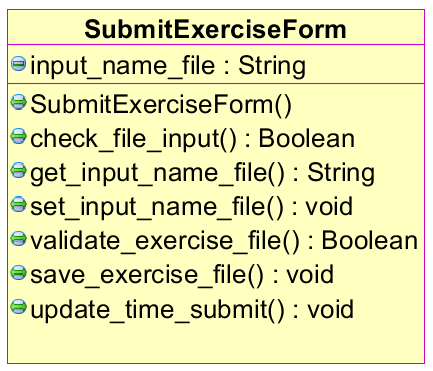
#### 3.2.1.3 SearchCourseForm:



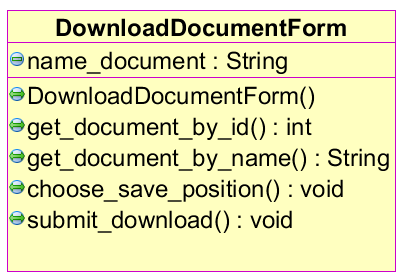
#### 3.2.1.4 EnrollForm:



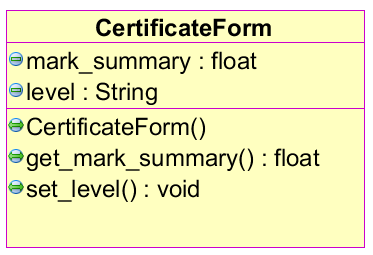
#### 3.2.1.5 SubmitExerciseForm:



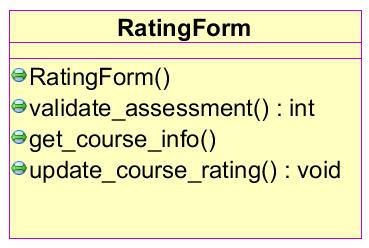
#### 3.2.1.6 DownloadDocumentForm:



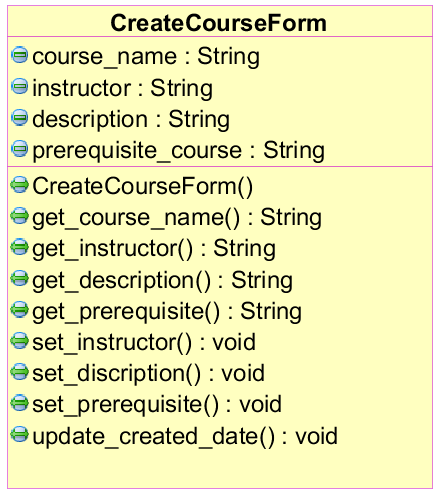
#### 3.2.1.7 CertificateForm:



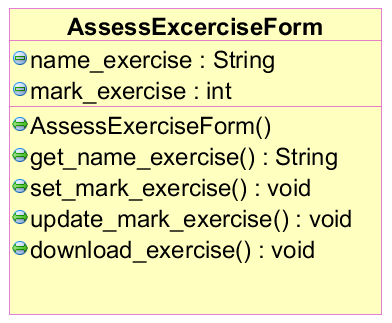
#### 3.2.1.8 RatingForm:



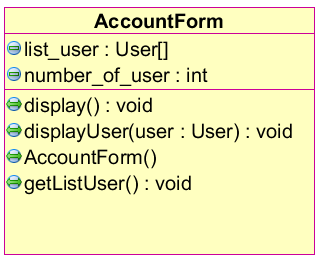
#### 3.2.1.9 CreateCourseForm:



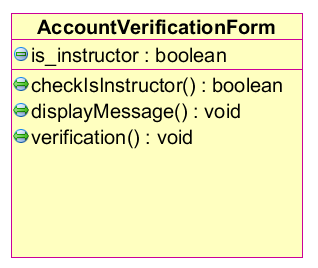
#### 3.2.1.10 AssessExerciseForm:



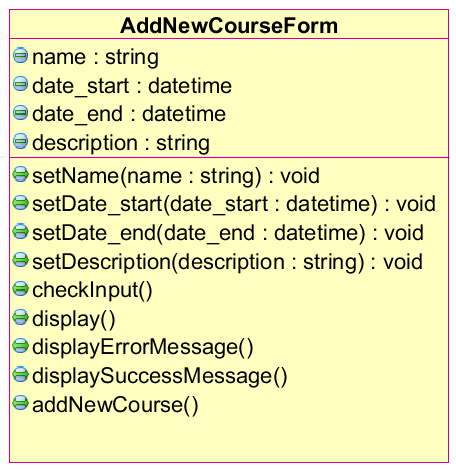
#### 3.2.1.11 AccountForm:



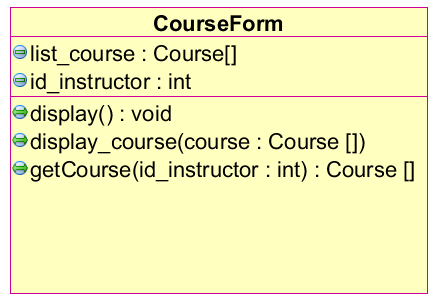
#### 3.2.1.12 AccountVerificationForm:



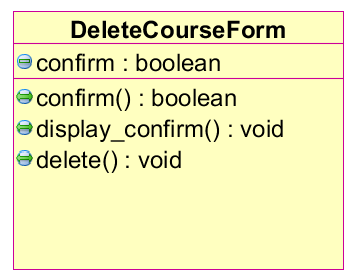
#### 3.2.1.13 AddNewCourseForm:



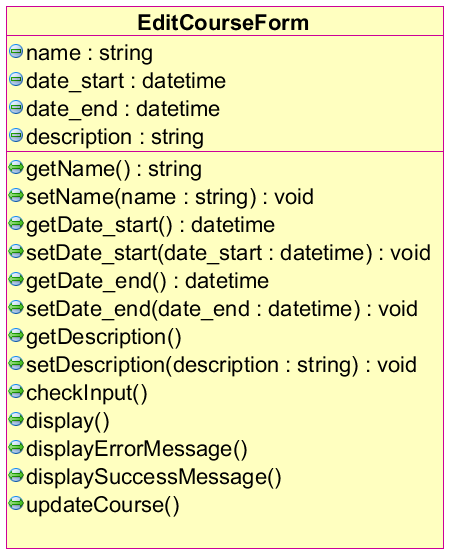
#### 3.2.1.14 CourseForm:



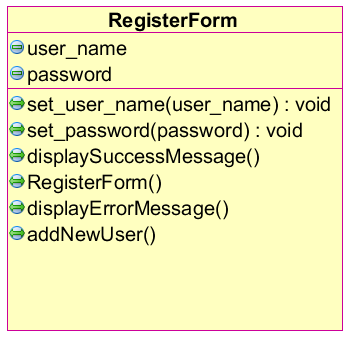
#### 3.2.1.15 DeleteCourseForm:



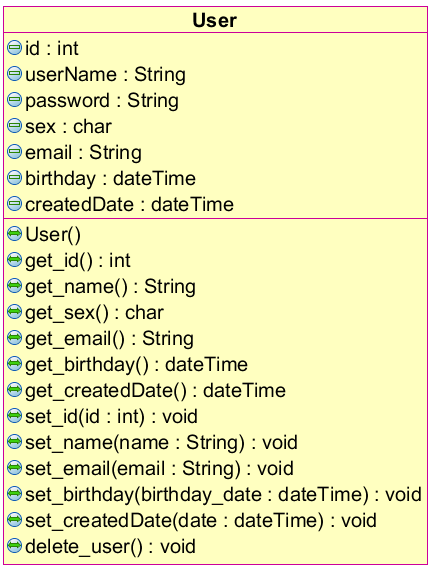
#### 3.2.1.16 EditCourseForm:



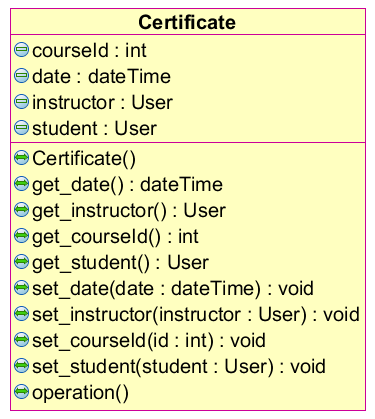
#### 3.2.1.17 RegisterForm:



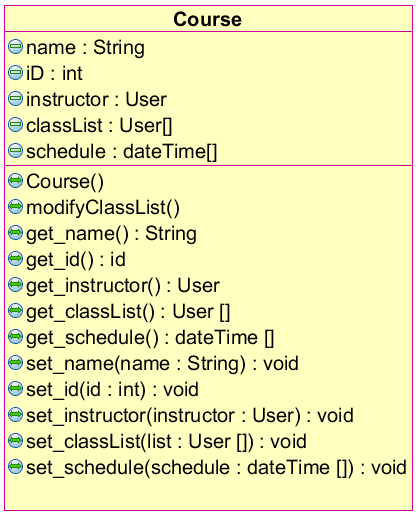
#### 3.2.1.18 User:



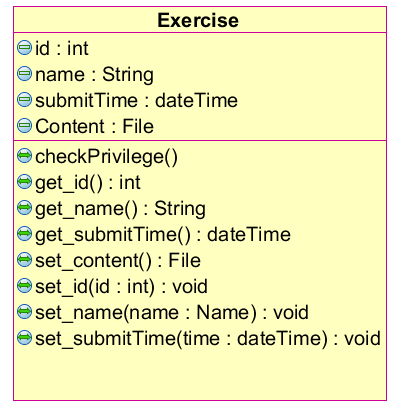
#### 3.2.1.19 Certificate:



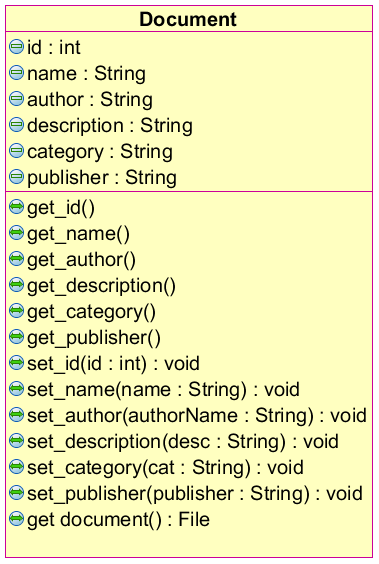
#### 3.2.1.20 Course



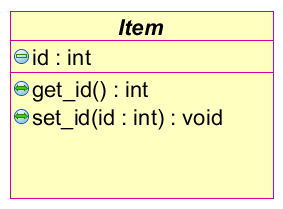
#### 3.2.1.21 Exercise:



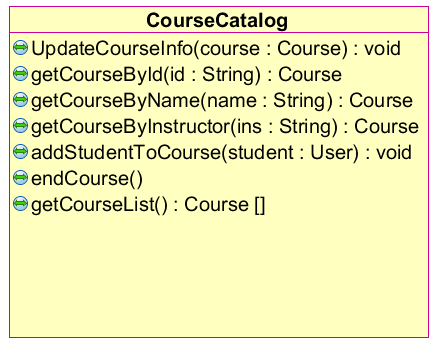
#### 3.2.1.22 Document:



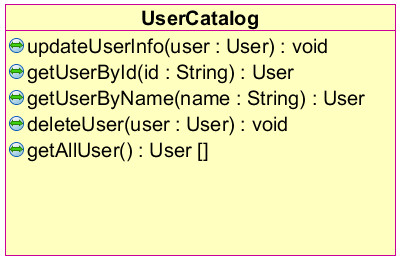
#### 3.2.1.23 Item:



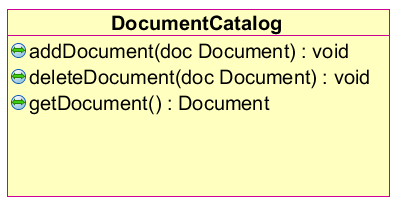
#### 3.2.1.24 CourseCatalog:



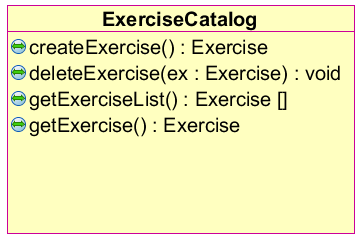
#### 3.2.1.25 UserCatalog:



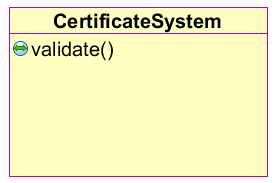
#### 3.2.1.26 DocumentCatalog:



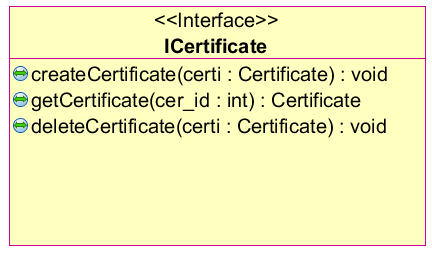
#### 3.2.1.27 ExerciseCatalog:



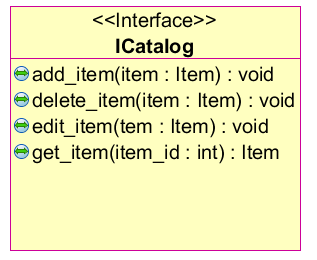
#### 3.2.1.28 CertificateSystem:



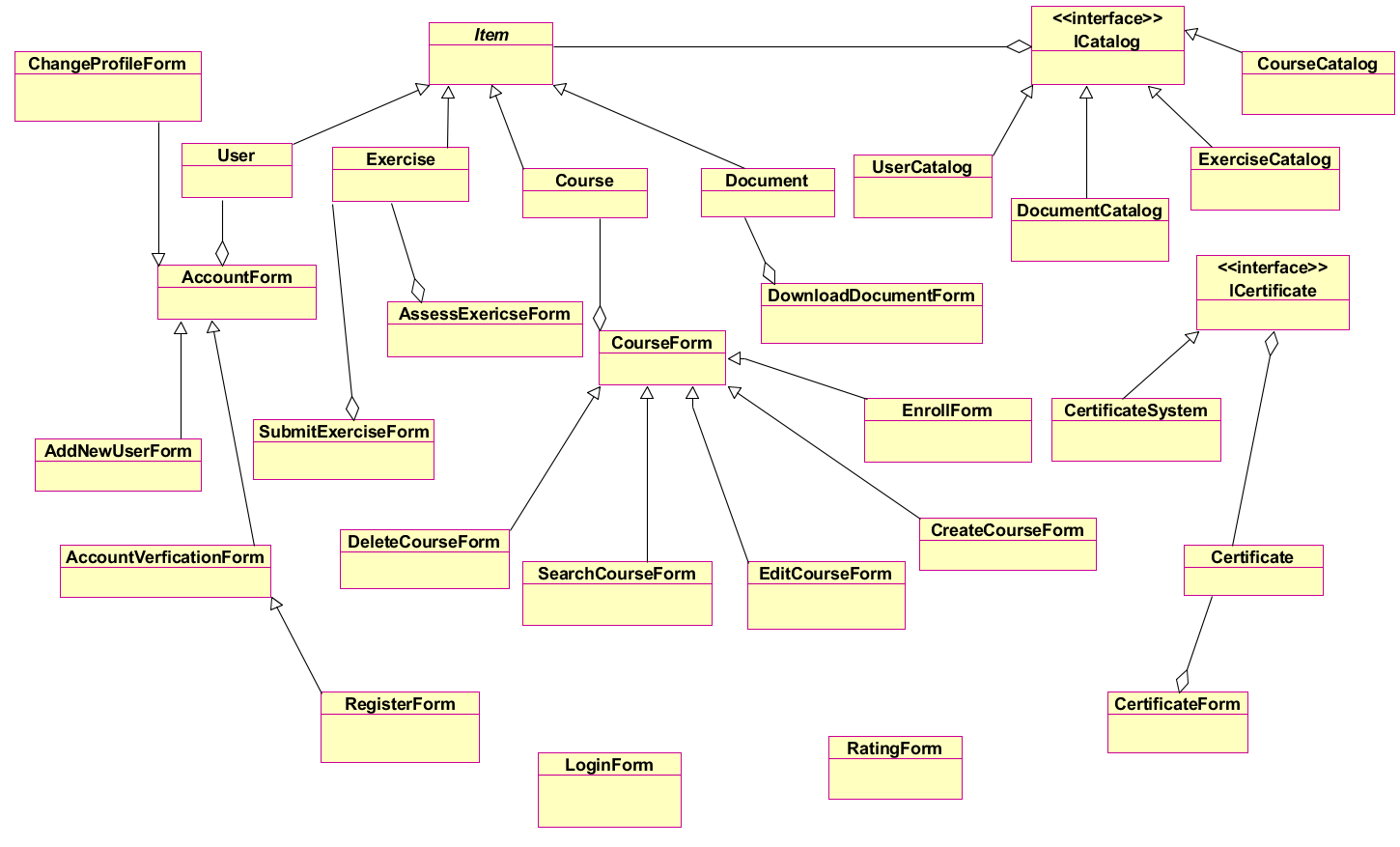
#### 3.2.1.29 ICertificate:



#### 3.2.1.30 ICatalog:



### 3.2.2 Class diagram in total:

****