**EZ Comm**

EZ Comm LMS  
Software Design Document (SDD)

**Team members:**

Ribe Ninan

Hongquan Yu

Yue Wang

Zeqian Huang

Guangchu Shi

Xiangfeng Wang

[**Product Overview**](#_h7szp0gvnnbm) **3**

[**Architecture Drivers**](#_ngthtmeqzmoz) **3**

[**Architecture Key Decisions and Rationale**](#_fxn2j5e2pzxr) **3**

[**Architecture Logical View**](#_6j1d0ogm9xvc) **3**

[**Architecture Behavior**](#_fuk69713asoq) **3**

[**Architecture Deployment View**](#_wxa3ucn1qf3v) **4**

[**Information Design Model**](#_cf5c5twe2s8r) **4**

[**Other Architecture Views**](#_jfiwv1srb53) **4**

[**Requirements Allocation**](#_noo6y7hozsvf) **4**

[**Architecture Work Allocation**](#_z3xhy6eg46fe) **4**

[**Detailed Design Key Decisions and Rationale**](#_d20cgbo8s77e) **5**

[**Detailed Design Structure**](#_7vdetvwo7g9k) **5**

[**Detailed Design Behavior**](#_etjpzmns41y3) **5**

[**Data Structures**](#_3cft4zgq99za) **5**

[**Algorithms**](#_vwx15jq2ddwu) **5**

[**Architecture to Detailed Design Tracing**](#_93bvpbyreqx4) **5**

## 71.Product Overview

This Software Architecture Document provides the architectural overview of the Learning Management System, EZComm. This tool is to be used by educational institutions to help facilitate communication, and provide educational courses, and its materials to students. The two primary user roles in this LMS are instructors and students. The architectural design for this Learning Management System was developed to provide the instructors the capabilities such as posting files, grades and other features for students. Similarly, for students, this tool provides the capability to view these files and grades.

## Architecture Drivers

Although software architecture drivers are diverse which could affect the software architecture, such as requirement, constraints, best practices and organizational, the most important factors could be the functional requirement, quality attribute, and constraints.

For constraints for EZComm project, the available time, an academic semester, for EZComm is not very much. Some group members have full-time job during the day, thus available time for project is limited. As for other members, many are taking multiple courses during the semester, thereby limiting the amount of time they have to spend on the project. Thus time constraint is an import factor that affected the software architecture decisions.

In regards to technical constraints, most members in our team are inexperienced in software development, and do not have knowledge in industry software development. Many technologies used to design the software are new for the team members, therefore technical constraint is also a factor that affected the software architecture decisions.

Functional Requirement

The most important factor affecting the architecture design is the functional requirement, which are the tasks/features that we need to implement for this LMS system. The functional requirements for EZComm are stated as following:

For instructor:

1. instructor shall be able to see the list of all students enrolled in the classes she/he is teaching
2. instructor shall be able to manage (create, edit, delete, and publish/unpublish) course syllabus
3. instructor shall be able to manage (create, edit, delete, and publish/unpublish) course assignments
4. instructor shall be able to see students submissions
5. instructor shall be able to grade students submissions

For student:

1. student shall able to view syllabus
2. student shall able to view assignments
3. student shall able to submit assignments
4. student shall able to see her/his own submissions
5. student shall able to view her/his own grades

For system:

1. system shall able to automatically creates gradebook for each course
2. system shall able to maintains a grade book for each course

Quality Attribute/Non-functional Requirement

Another architecture driver is quality attribute, we represent it as the utility tree:

Usability：

* Operability.
* Learnability.
* User error protection.
* User interface aesthetics.

|  |  |
| --- | --- |
| Quality Attribute | Usability |
| Scenario | Operability |
| Source | End users |
| Stimulus | Users want to use the system efficiently |
| Artifact | System |
| Environment | At runtime |
| Response | * Reuses already entered data and/or commands * Supports efficient navigation within a screen * Provides a comprehensive searching within the website * Allows multiple simultaneous activities * Anticipate the functions user may need |
| Response Measure | * Less than 5 steps to realize a certain goal (M, M) * 80% of the users should be satisfied about the system’s efficiency (H, H) |

|  |  |
| --- | --- |
| Quality Attribute | Usability |
| Scenario | Learnability |
| Source | Students and instructors |
| Stimulus | Both students and instructors want to adapt to the system and learn system features without difficulties |
| Artifact | System |
| Environment | At runtime |
| Response | * Help topics are rich and sensitive to context * The user interface is similar to other online learning systems that users used before * In an unfamiliar context, the user interface is still easy to use |
| Response Measure | * Time users spend on learning system features is less than 15 minutes (M, M) * 80% of the new users are satisfied about transferring to the new system (M, H) |

|  |  |
| --- | --- |
| Quality Attribute | Usability |
| Scenario | User error protection |
| Source | End users |
| Stimulus | Users want to minimize the impact of their errors |
| Artifact | System |
| Environment | At runtime |
| Response | * Be able to retrieve forgotten usernames and passwords * Be able to undo, cancel, or recover from system failures * Be able to roll back to the initial stage if unsuccessful transactions occur * Be able to recognize and correct user errors * Record the error and report it to system |
| Response Measure | * Less than 5% errors occur during the whole process (H, H) * Password should be retrieved every time if it’s forgotten by user(H,L) |

|  |  |
| --- | --- |
| Quality Attribute | Usability |
| Scenario | User interface aesthetics |
| Source | End users |
| Stimulus | Users want to use a beautifully designed website and feel comfortable navigating through it |
| Artifact | System |
| Environment | At runtime |
| Response | * Layout of the website should be concise and intuitive * The design should be consistent across all subsystems and sections * The system should enable users to work at their own pace, without distractions |
| Response Measure | * 90% of the users feel comfortable and satisfied when operating the system (H, H) |

Modifiability (M, L)：

|  |  |
| --- | --- |
| Quality Attribute | Modifiability |
| Scenario | Some attributes of the system are to be modified/added/deleted |
| Source | Developers and system admins |
| Stimulus | Developers and admins wish to add/delete/modify/vary system functionalities and quality attributes |
| Artifact | System user interface; platform; environment |
| Environment | At design time; compile time; at build time; at run time |
| Response | * System should be able to locate the places that need to be modified * System should be able to allow developers to make the necessary changes, without affecting other functionalities * System should be able to test the modification * System should be able to deploy the modifications |
| Response Measure | * Cost of the modifications, in terms of money and effort, should be less than 10% of the total cost (H, M) * Zero of other attributes may be affected (H, H) |

Performance (H, H)：

|  |  |
| --- | --- |
| Quality Attribute | Performance |
| Scenario | A number of requests asking for a service arrive at the system |
| Source | Web requests |
| Stimulus | Web requests arrive at the system and ask for the response of a certain service |
| Artifact | System |
| Environment | Normal mode; overload mode; |
| Response | * System handles the requests without errors * System be able to protect itself against overwhelming requests |
| Response Measure | * Under normal amount of requests the data loss rate should be less than 1% (H, M) * Under overload mode the data loss should be less than 10% (M, H) * Latency under normal mode should be less than 75ms (H, M) * Latency under overload mode should be less than 150ms (M, H) |

Availability：

* Fault tolerance
* Recoverability
* Excess Data storage

|  |  |
| --- | --- |
| Quality Attribute | Availability |
| Scenario | Fault tolerance |
| Source | Internal to the system |
| Stimulus | Some process fault in the hardware or software |
| Artifact | System’s processor |
| Environment | Degraded mode |
| Response | * Report it； * Notify user/system admin/developer; * Some sources are temporarily unavailable; * Continue degraded mode until cause of the fault is found and fixed |
| Response Measure | * 60% part of the system can be used;(H,L) * Notify user/operator the fault within 3 minutes;(M,M) * The software repaired time will be in 2 days.(H,H) |

|  |  |
| --- | --- |
| Quality Attribute | Availability |
| Scenario | Recoverability |
| Source | External to the system or internal to the system |
| Stimulus | System crashed |
| Artifact | System‘s processor |
| Environment | Degraded mode |
| Response | * Report it; * Notify it to user/system admin/developer; * Developer check the system and fix the problem; * The data will be recovered; * Email user that system can be used. |
| Response Measure | * The time of checking and fixing the bug will be less than 2 days;(H,H) * 99.99% data should not be lost except the data which was editing when the system crashed.(H,H) |

|  |  |
| --- | --- |
| Quality Attribute | Availability |
| Scenario | Excess Data storage |
| Source | Internal to the system |
| Stimulus | User wants to upload the files bigger than the data size limitation |
| Artifact | Persistent storage |
| Environment | Normal operation |
| Response | * Notify user the size limitation of the uploaded files * Continue to operate when use upload files under the limitation. |
| Response Measure | Check the file size and notify user within 60 seconds;(M, L) |

Security (H,M)

|  |  |
| --- | --- |
| Quality Attribute | Security |
| Scenario | Student want to modify their grade. |
| Source | Hacker from inside the organization |
| Stimulus | Students attempt to modify grade data |
| Artifact | Grade data within system |
| Environment | Normal mode |
| Response | * Authenticates user； * blocks access to data and/or services； * records access/modifications or attempts to access/modify data by identity； * informs instructor |
| Response Measure | * The authenticate email will send to instructor after 5 times invalid login;(H,L) * Block access the data within 15 minutes.(H,L) * Block access for at least one hour.(H, L) * Data integrity is unaffected 98% of the time(H,M) |

Testability (H,L)

|  |  |
| --- | --- |
| Quality Attribute | Testability |
| Scenario | System test before delivered |
| Source | System developer and end user |
| Stimulus | Execution of tests due to completion of code increment |
| Artifact | Portion of the system; Full system |
| Environment | Compile time; Integration time; Deployment time; Run time |
| Response | * Controlled to perform the desired tests * Record the response to each test * Capture cause of fault, record it and fix the bug * Monitor the every state of the system |
| Response Measure | * 95% of executable statement executed;(H,L) * 3% of executable statement failed;(H,M) * Less than 2 day was needed to fix the bugs(H,H) |

## Architecture Key Decisions and Rationale

*<This section provides a rationale for the major design decisions embodied by the software architecture. It describes any design approaches applied to the software architecture, including the use of architectural styles or design patterns, when the scope of those approaches transcends any single architectural view. The section also provides a rationale for the selection of those approaches. It also describes any significant alternatives that were seriously considered and why they were ultimately rejected – if applicable. The section describes any relevant technology and COTS considerations, including any associated trade studies - if applicable.>*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| concerns | | | Technology to use in backend to build EZComm? | | | | |
|
| Alternatives | | | 1. Java  2. PHP  3. C#  4. Python | | | | |
|
|
|
|
| Ranking criteria | | | 1. Feasible to build the service in time;  2. Minimum time for team members to master the technologies;  3. It is free and open source; | | | | |
|
|
|
| Architecture decision | | description | Almost all of our team members are familiar with Java, We do not have to spend too much time to learn new technology to implement the backend. Java is enough to develop all the required features. | | | | |
|
|
|
| Rational | We will use Java to develop the backend structure and to interact with frontend.  We will use JDBC to connect the MySQL database. | | | | |
|
|
|

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| concerns | | | Database to use in backend to build EZComm? | | | | |
|
| Alternatives | | | 1. MySQL  2. SQL  3. Oracle | | | | |
|
|
|
|
| Ranking criteria | | | 1. It is free  2. Easy to connect to backend | | | | |
|
|
|
| Architecture decision | | description | MySQL is free for use and portable to Java backend | | | | |
|
|
|
| Rational | We will use MySQL to store our data | | | | |
|
|
|

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| concerns | | | Technology to use in front-end to build EZComm? | | | | |
|
| Alternatives | | | 1. Bootstrap  2. Foundation  3. Semantic UI | | | | |
|
|
|
|
| Ranking criteria | | | 1. Easy to get solution  2. Dynamic | | | | |
|
|
|
| Architecture decision | | description | Bootstrap are responsive web design support (can also be disabled if required),  it also has extensive documentation | | | | |
|
|
|
| Rational | We will use Bootstrap to build our frontend web service interacting with backend | | | | |
|
|
|

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| concerns | | | Where to deploy EZComm? | | | | |
|
| Alternatives | | | 1. local Server  2.AWS  3. Google cloud | | | | |
|
|
|
|
| Ranking criteria | | | 1. Free and easy to use  2. Accessible from anywhere | | | | |
|
|
|
| Architecture decision | | description | AWS is free for light use in EC2. it is also the largest cloud provider in this industry. | | | | |
|
|
|
| Rational | We will use AWS to host the web service and database to make it accessible from anywhere | | | | |
|
|
|

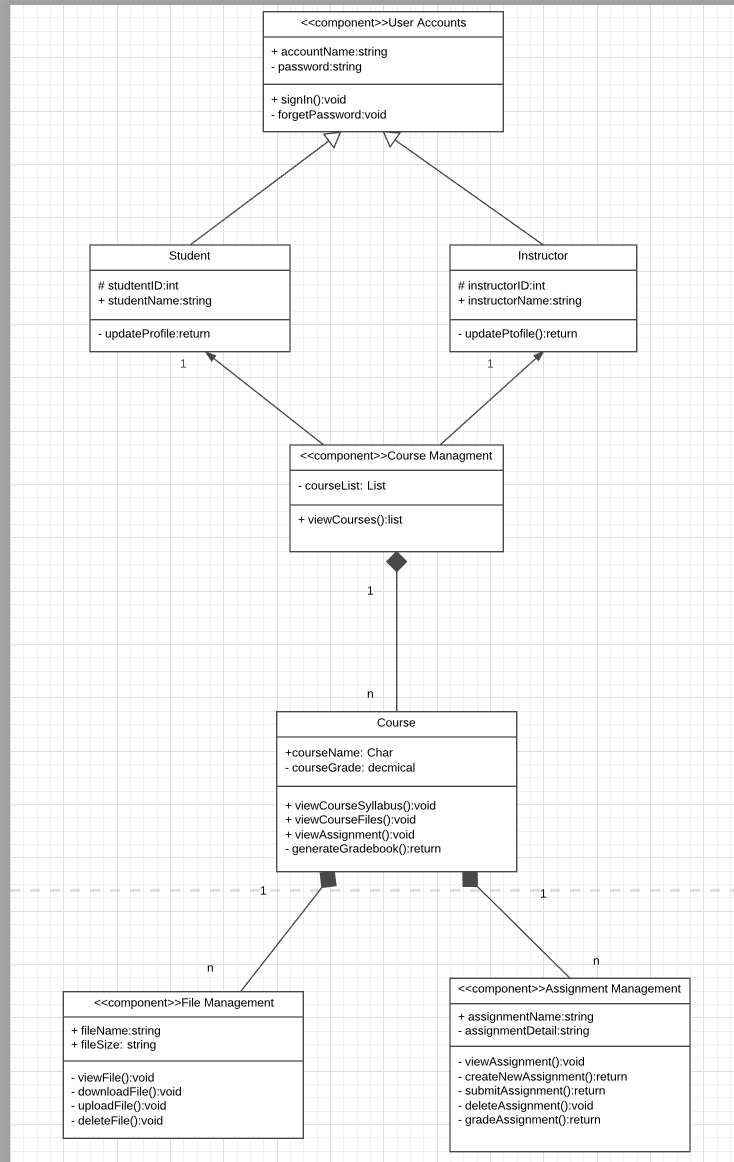
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| concerns | | | Design pattern for EZComm | | | | |
|
| Alternatives | | | 1.MVC  2.Pipe-filter | | | | |
|
|
|
|
| Ranking criteria | | | 1. Maintainability  2. Reusability | | | | |
|
|
|
| Architecture decision | | description | MVC has following features:  Easily Maintainable  High Re-usability  Readable because of abstractions | | | | |
|
|
|
| Rational | we use MVC to divide EZComm into three interconnected parts | | | | |
|
|
|

## Architecture Logical View

*<Static, structural view of the architecture. Must include:*

*a) models that represent architectural components and their relations and interfaces. UML diagrams must be used for this view, such as UML package diagrams to show sub-systems, and either UML component diagram(s) or UML class diagram(s) to show components, relations and interfaces* ***AND***

*b) descriptions/specifications of all components and interfaces. Use the templates provided in class – see lecture material >*



Component Description:

User account------------manage user account information

Course Management-------- manage all the course related task

Assignment Management------manage all the assignment related task like create new assignment, submit the assignment, etc.

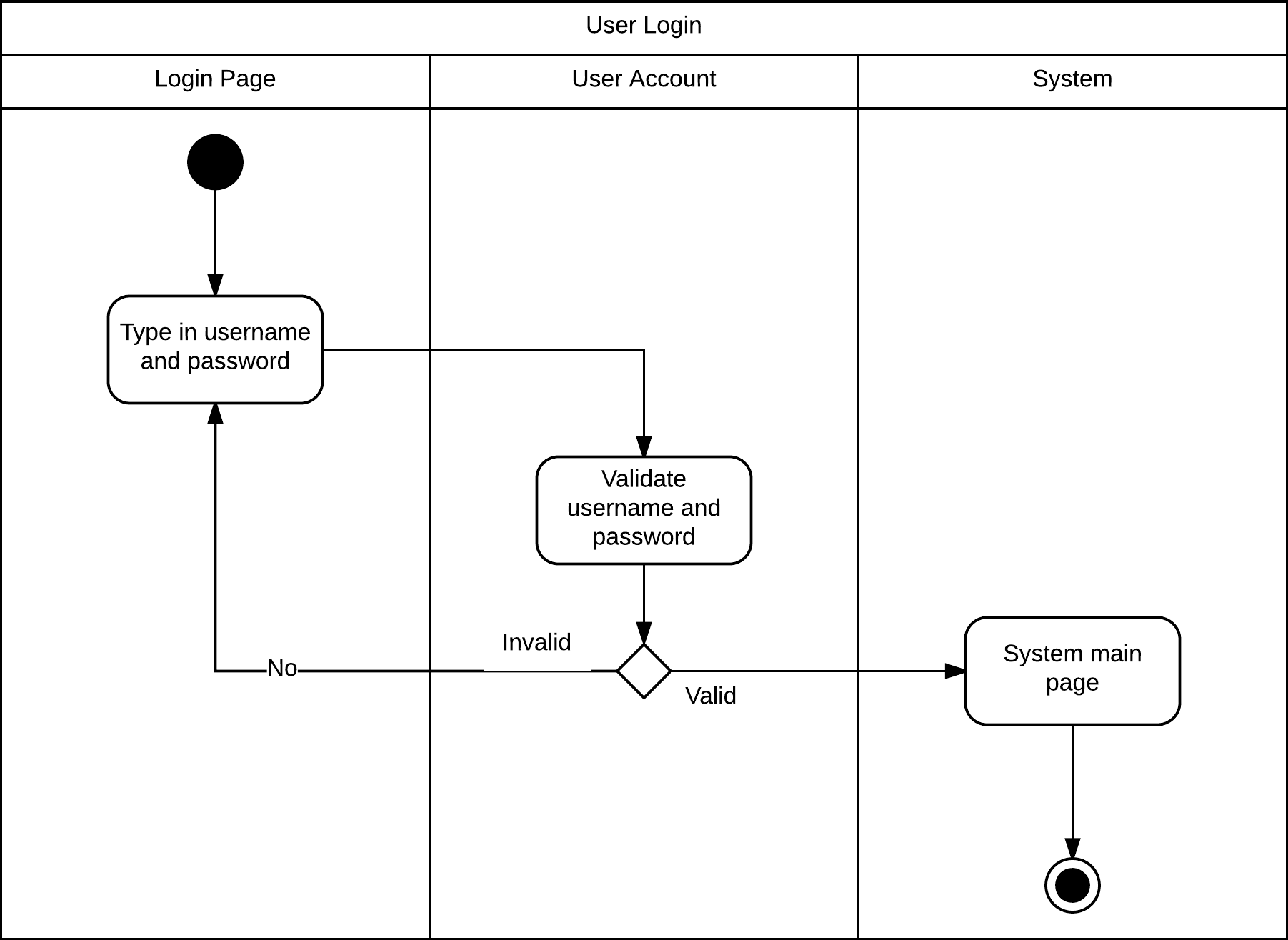
File Management------manage all the file related task like syllabus and class materials, etc.

User Interface s

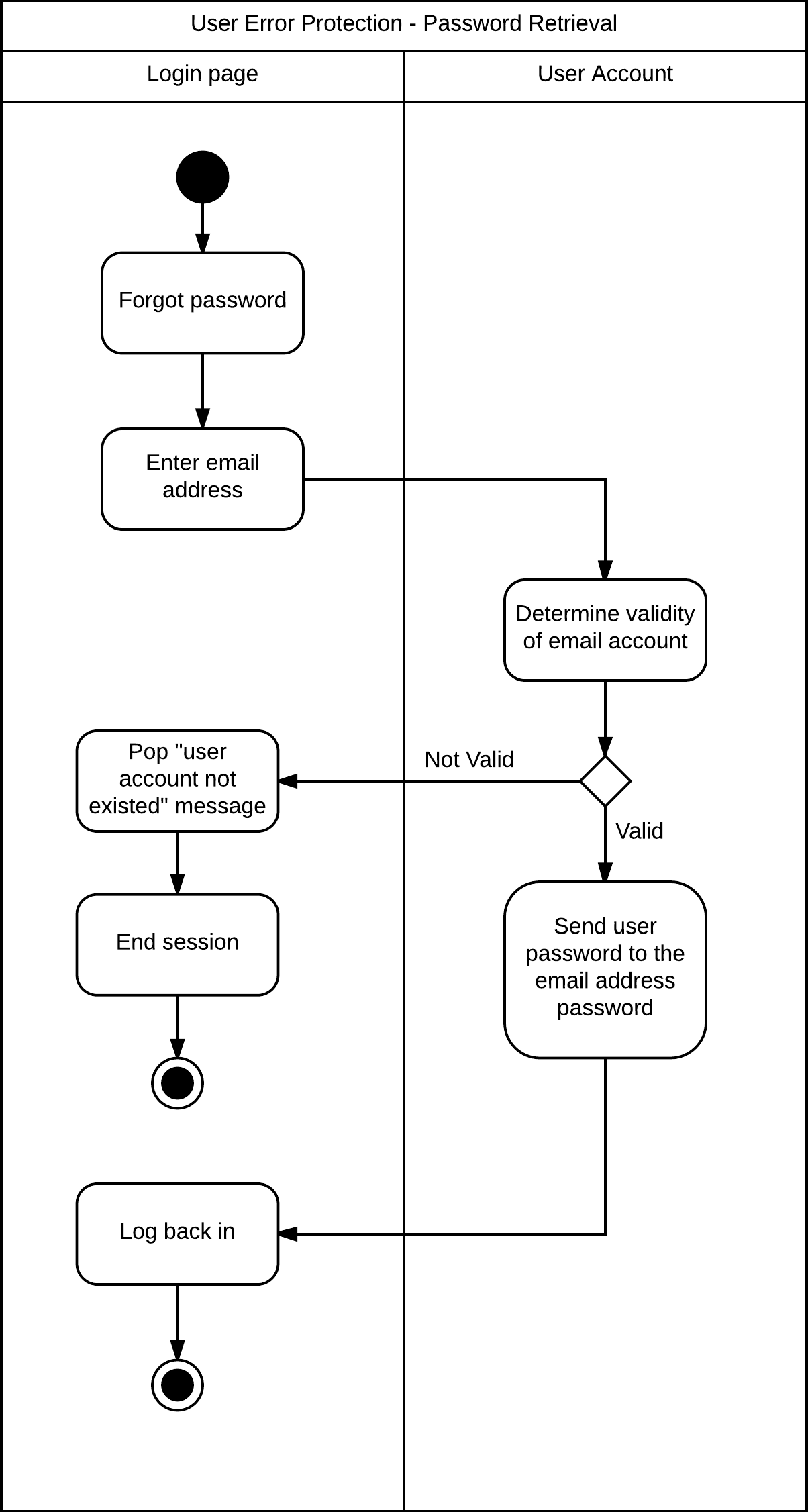
## Architecture Behavior

*<Show how architectural components work together to realize scenarios. Include at least four high priority usage scenarios and/or quality scenarios; at least one scenario must be a quality scenario. Use UML activity diagrams and/ or UML sequence diagrams for behavior representation>*

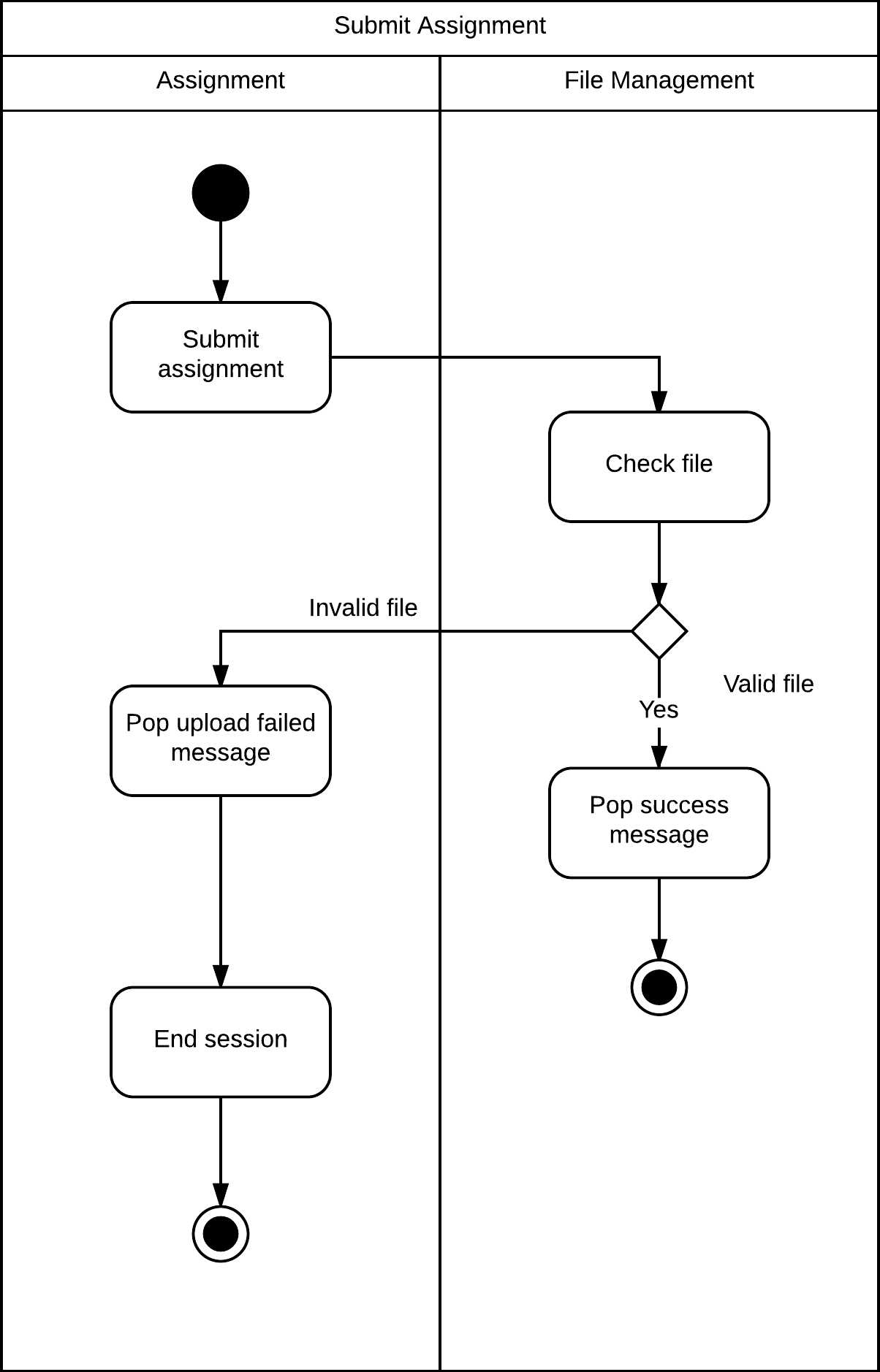
**Scenario 1:** Login



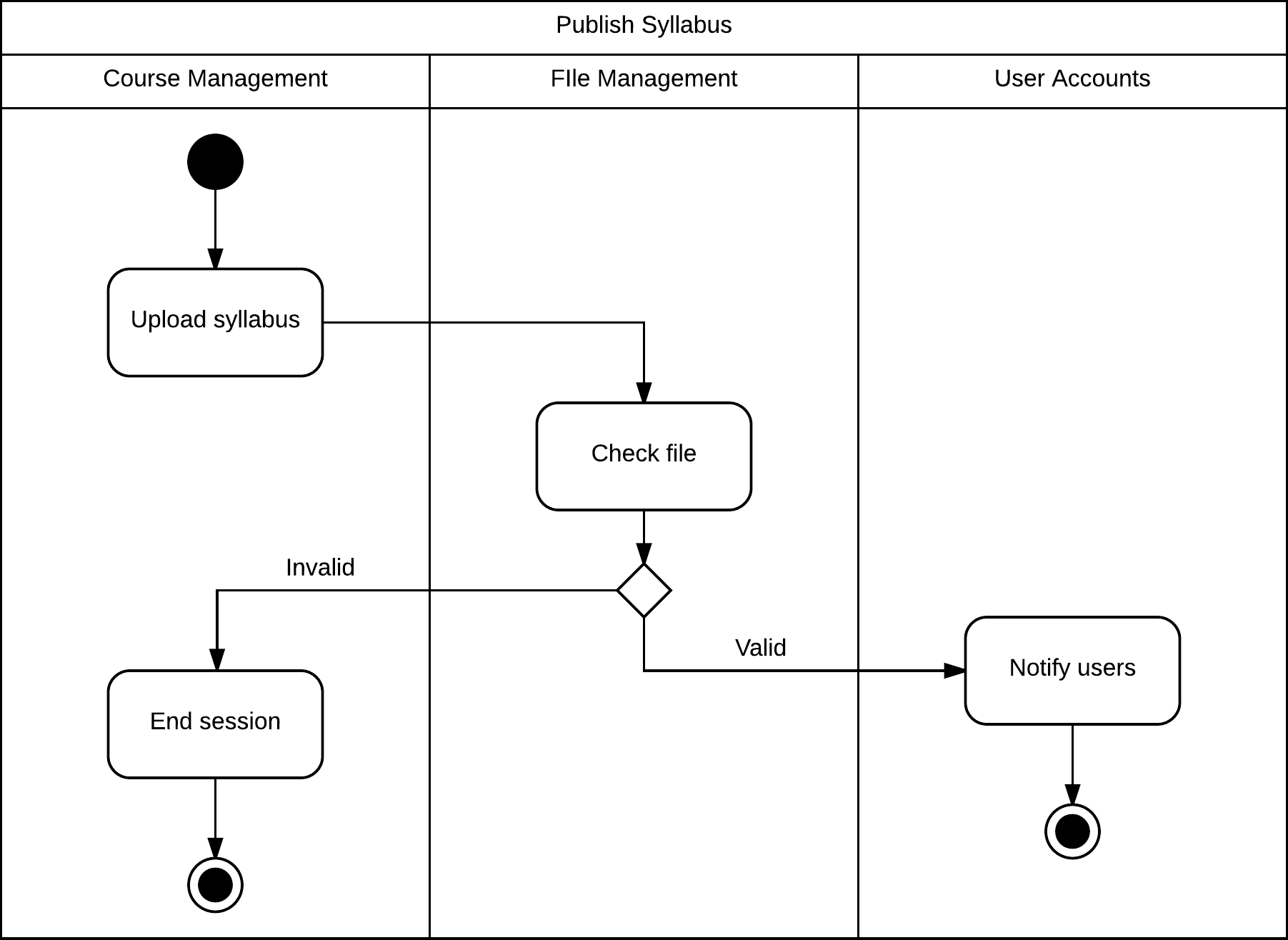
**Scenario 2:** User error protection - password retrieval



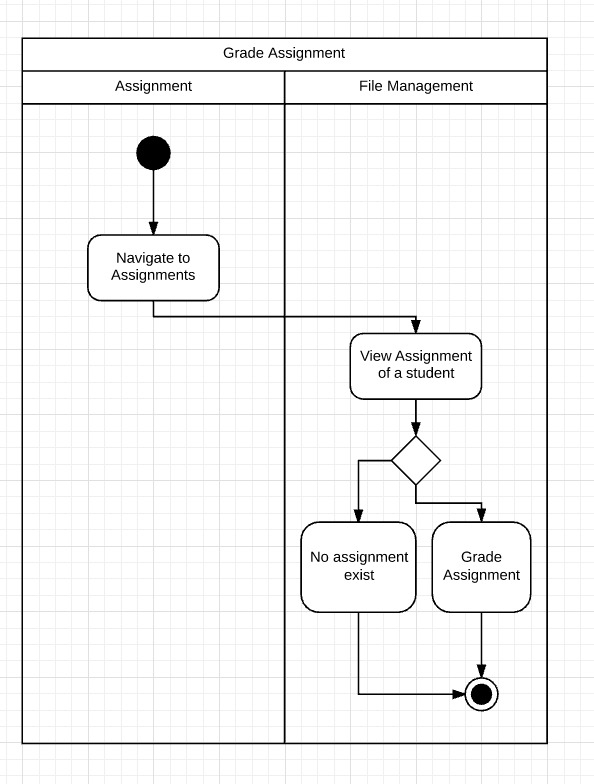
**Scenario 3:** Student Submit assignment



**Scenario 4:** Instructor publish/edit syllabus

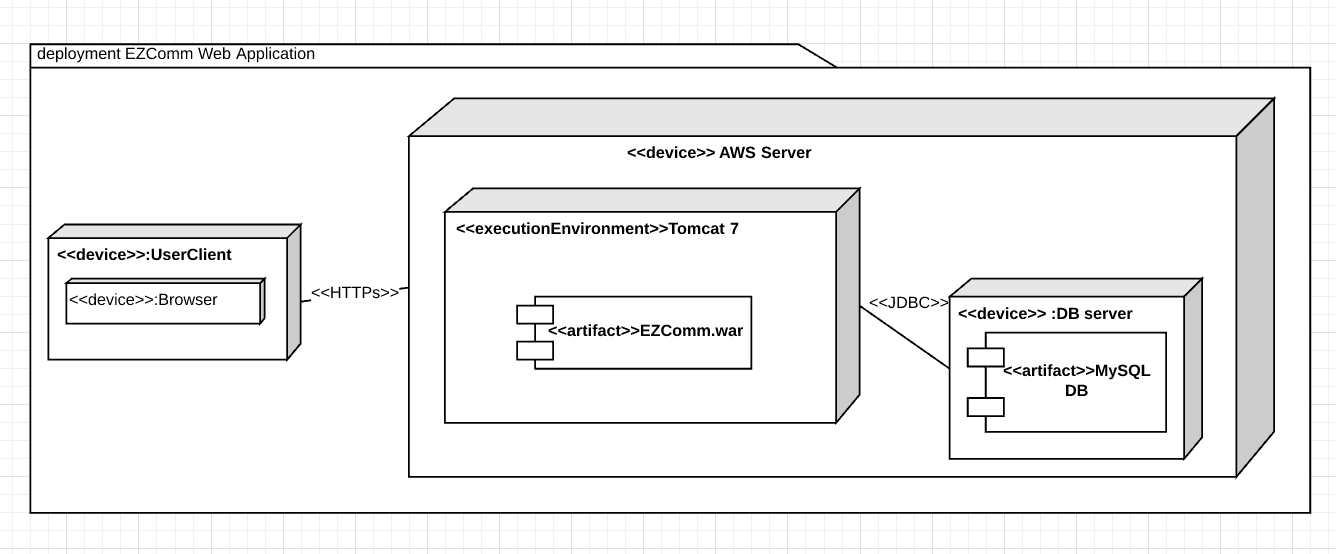


**Scenario 5:** Instructor grade the assignment wrote by student



## Architecture Deployment View

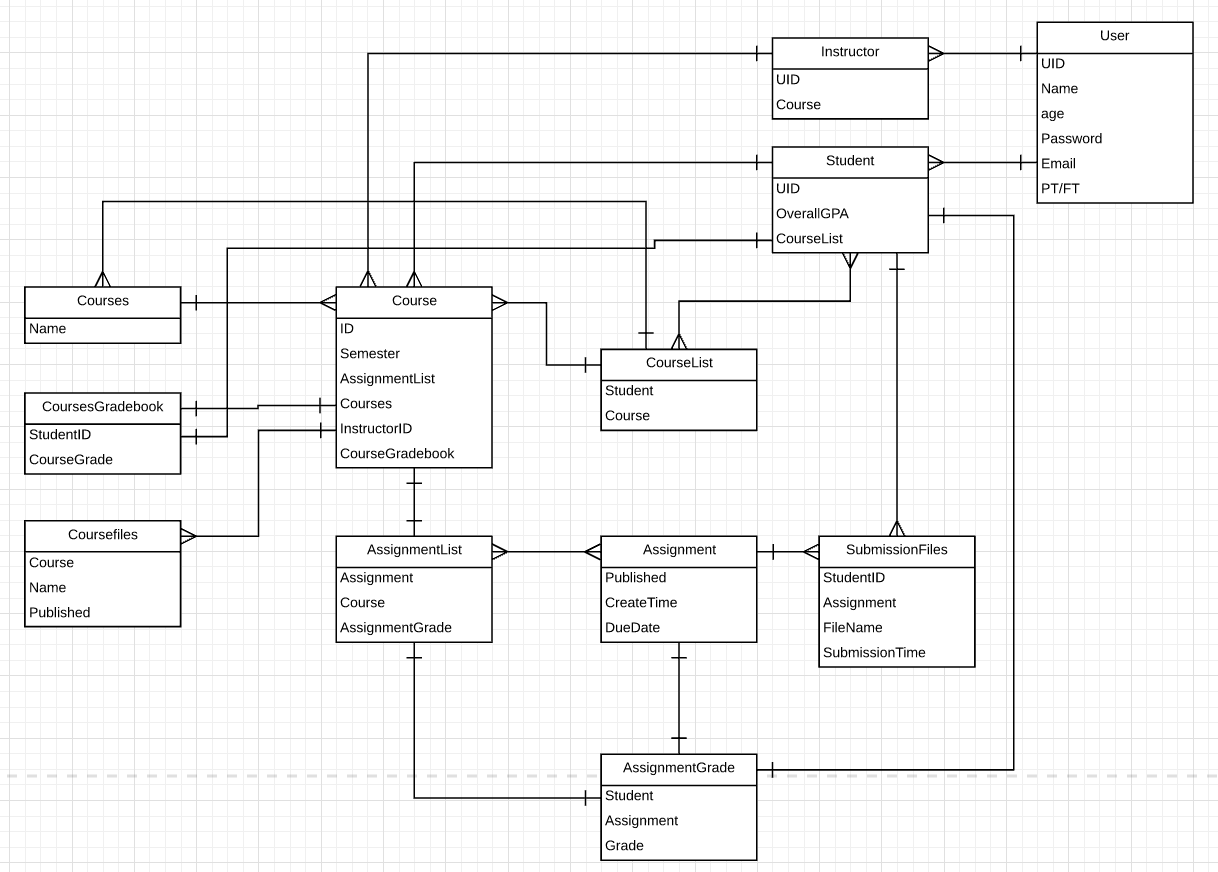
*<Show how architecture components will be deployed on hardware nodes and environment. Use UML deployment diagram to represent this view>*

**

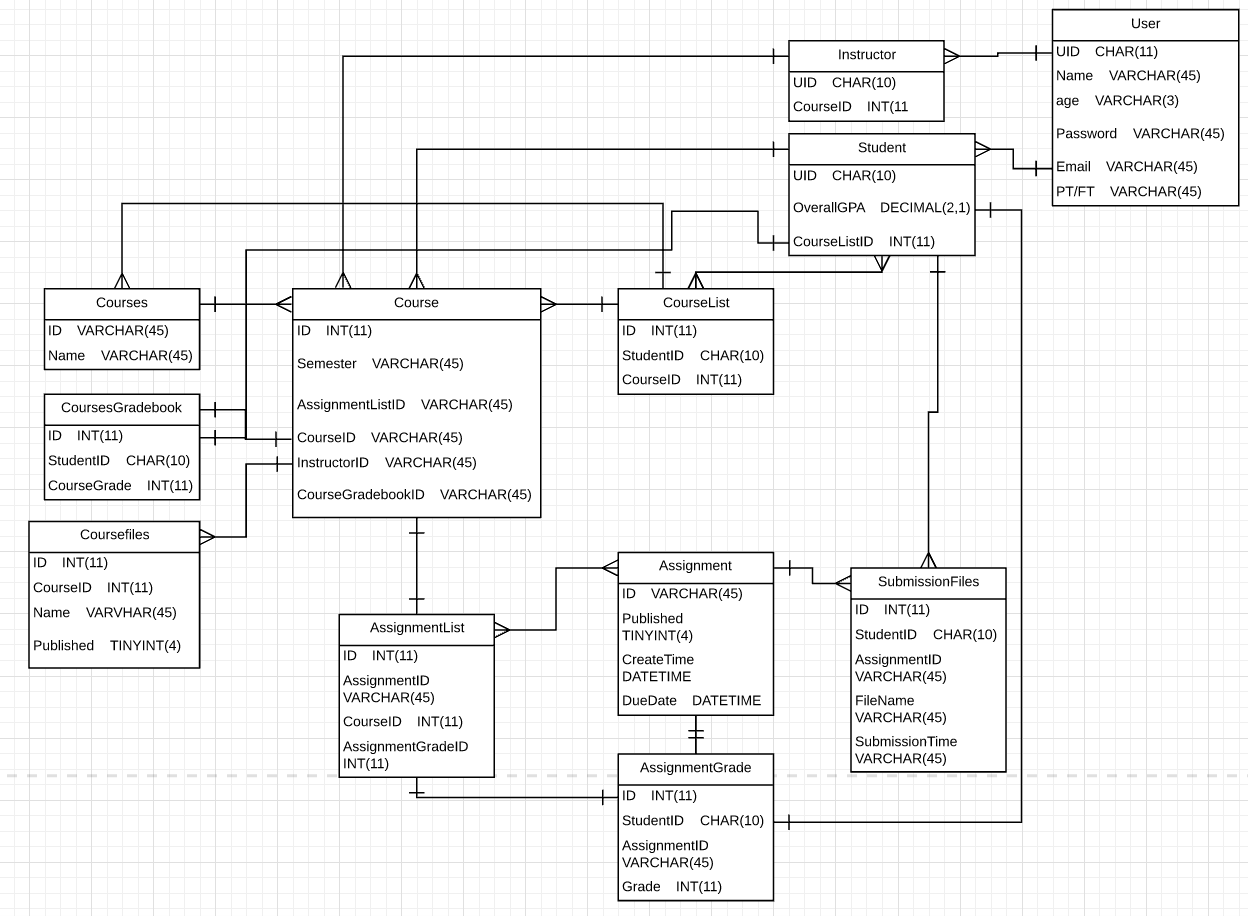
## Information Design Model

*<Insert here the conceptual, logical and physical data models>*

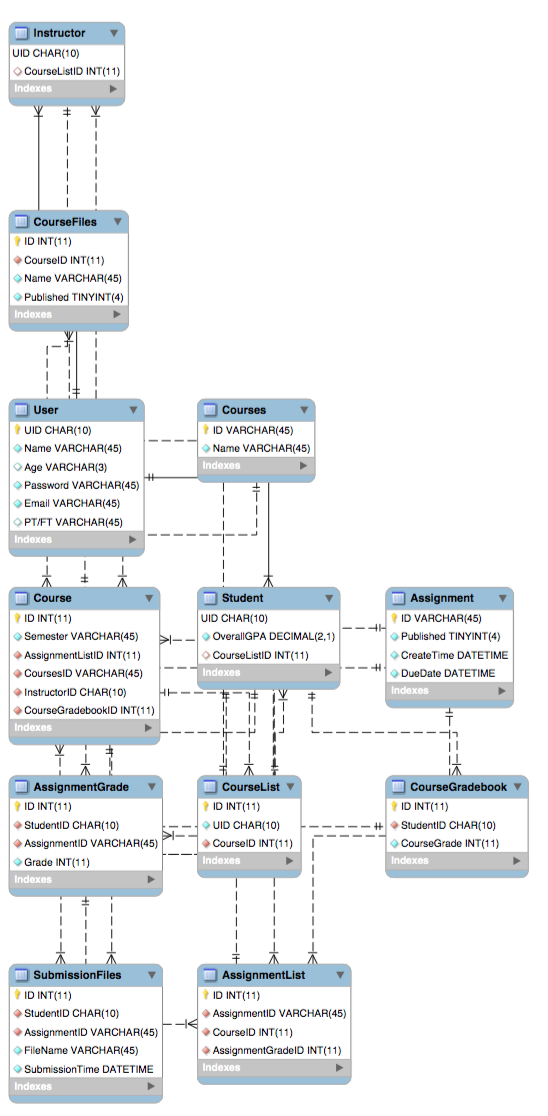
conceptual data models



logical data models



physical data models



## Other Architecture Views

*<Any other views that are relevant to this architecture, if applicable.>*

## Requirements Allocation

*<Trace matrix, showing correspondence between requirements and architecture elements. For this project, as requirements are represented as use case, a table like the one below should be used, showing which of the architecture components realize which use case. Requirements coverage must be ensured.>*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Use Cases/ Architecture Components | Component 1 File Management | Component 2  Assignment Management | Component 3 User Accounts | Component 4 Course Management | Component 5 User Interface |
| Students view syllabus | X |  | X |  | X |
| Students view assignment |  | X | X |  | X |
| Student submit assignment and view submission |  | X | X |  | X |
| View her/his own grades |  | X | X | X | X |
| Instructor see enrolled student list |  |  | X | X | X |
| Instructor create/delete/publish/unpublish syllabus | X |  | X |  | X |
| Instructor create/delete/publish/unpublish assignment |  | X | X |  | X |
| Instructor see and grade student assignment |  | X | X |  | X |
| Student view grade book for Certain course |  |  | X | X | X |
| Instructor view grades for all student in his class |  |  | X | X | X |

**Abuse Use Case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Abuse Use Cases/ Architecture Components | Component 1 File Management | Component 2  Assignment management | Component 3 User Accounts | Component 4 Course Management | Component 5 User Interface |
| Students delete syllabus | X |  | X |  |  |
| Students Change course schedule | X |  | X |  |  |
| Student upload malicious documents | X |  |  |  |  |
| Student modify grades |  | X | X | X |  |
| Plagiarize from other students | X |  | X |  |  |
| Instructor modify/delete student assignment |  | X |  |  |  |
| instructor modify student grade arbitrary |  | X |  | X |  |

## Architecture Work Allocation

*<Show allocation of architecture components to team members, for specification and further development and/or testing. Recommended to use a table, see example below>*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Architecture component name/ Team member** | **Ribe Ninan** | **Zeqian Huang** | **Xiangfeng Wang** | **Hongquan Yu** | **Yue Wang** | **Guangchu Shi** |
| **Course Management** | Development | Testing | Testing | Development | Development | Testing |
| **User Accounts** | Testing | Development | Development | Testing | Testing | Development |
| **File Management** | Development | Development | Testing | Testing | Testing | Development |
| **Assignment Management** | Testing | Testing | Development | Development | Development | Testing |
| **User Interface** | Testing | Development | Testing | Testing | Development | Development |

**Component 1: File Management (Ribe)**

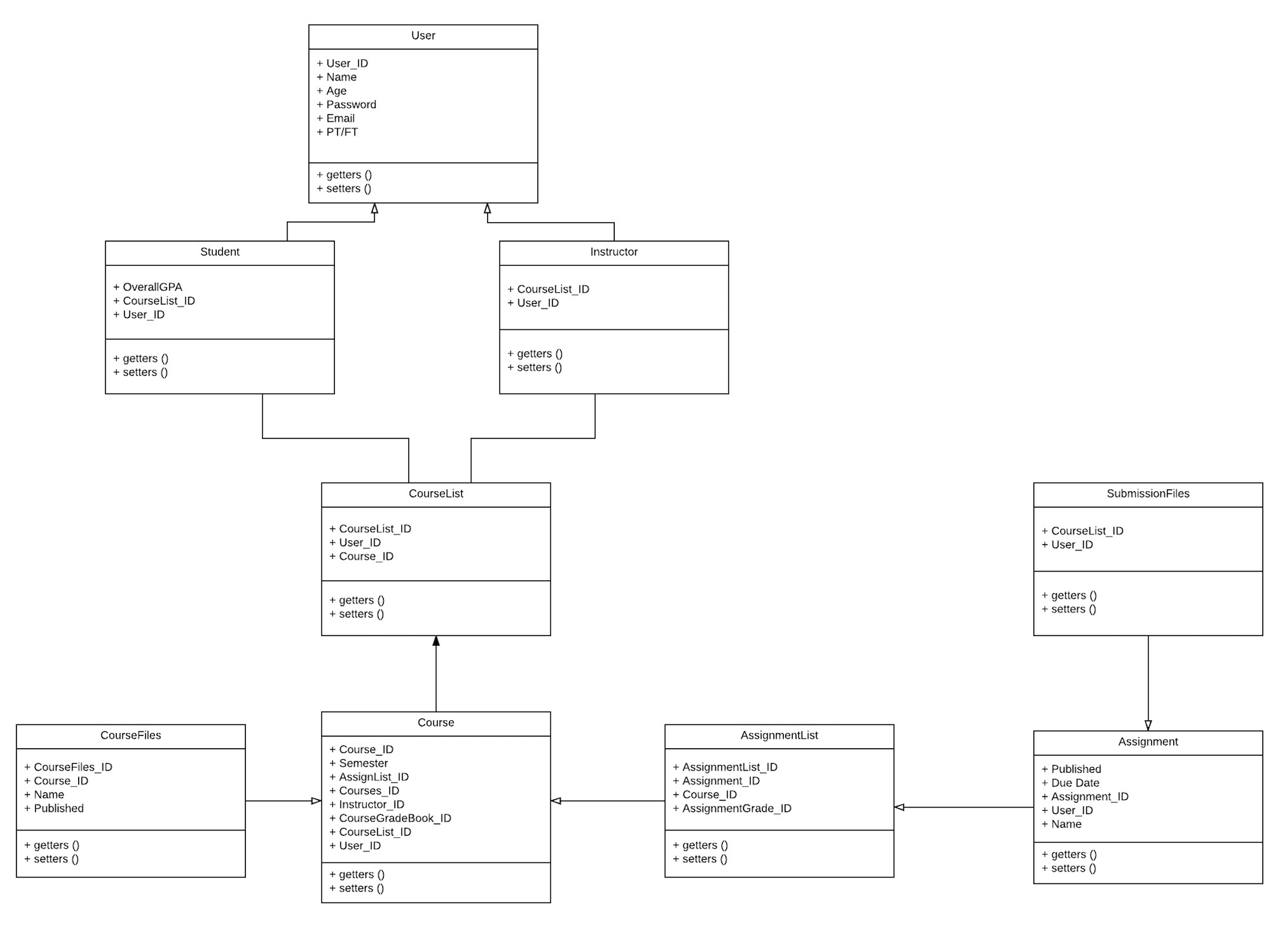
## 1.1 Detailed Design Key Decisions and Rationale

The File Management component is the only component that has access to the files stored by the application, and it is used to store/retrieve files for the system.

Follow the Dependency Inversion Principle (ISP) to ensure that the high level modules do not depend on the implementation of the file management component. The high level modules will use the interfaces defined by the file management component to save and retrieve file.

|  |  |  |
| --- | --- | --- |
| **Key decisions** | **Rationale** | **Use of design patterns** |
| Use a facade design pattern for the File Manager. Provide an interface for the File Manager. | The File Manager provides the interface for the application to store and retrieve a file. The way this is done/modifications to the logic of how this is done should not in any way affect the users of this class. | Facade Pattern |

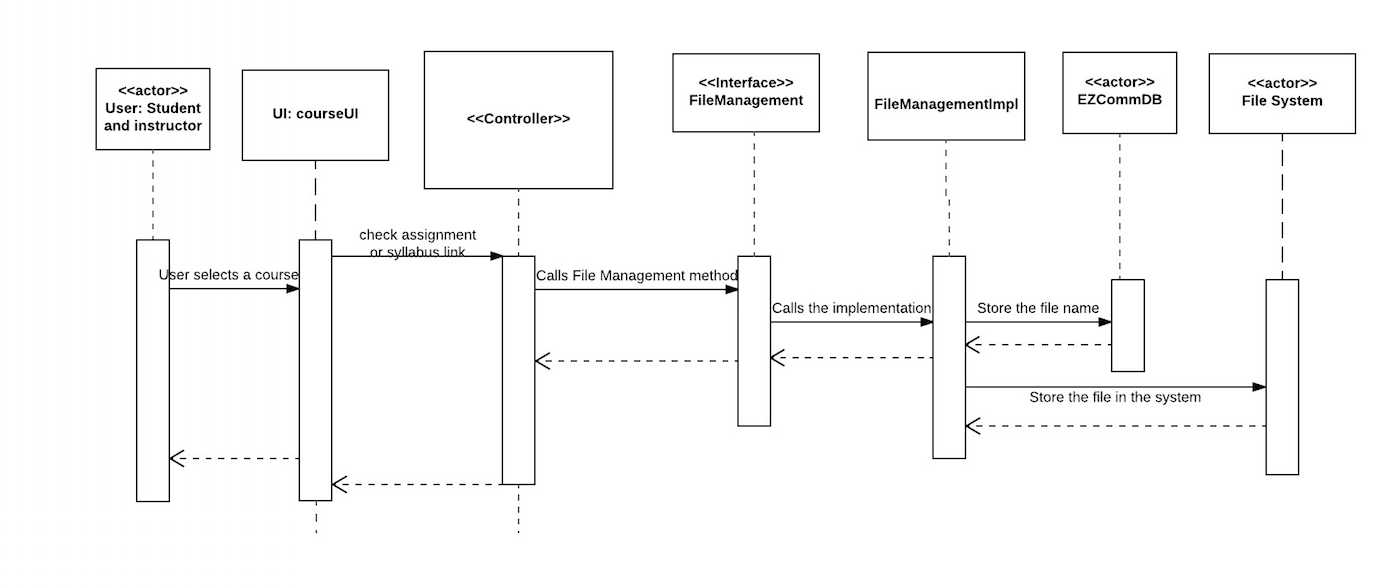
## 1.2 Detailed Design Structure

**

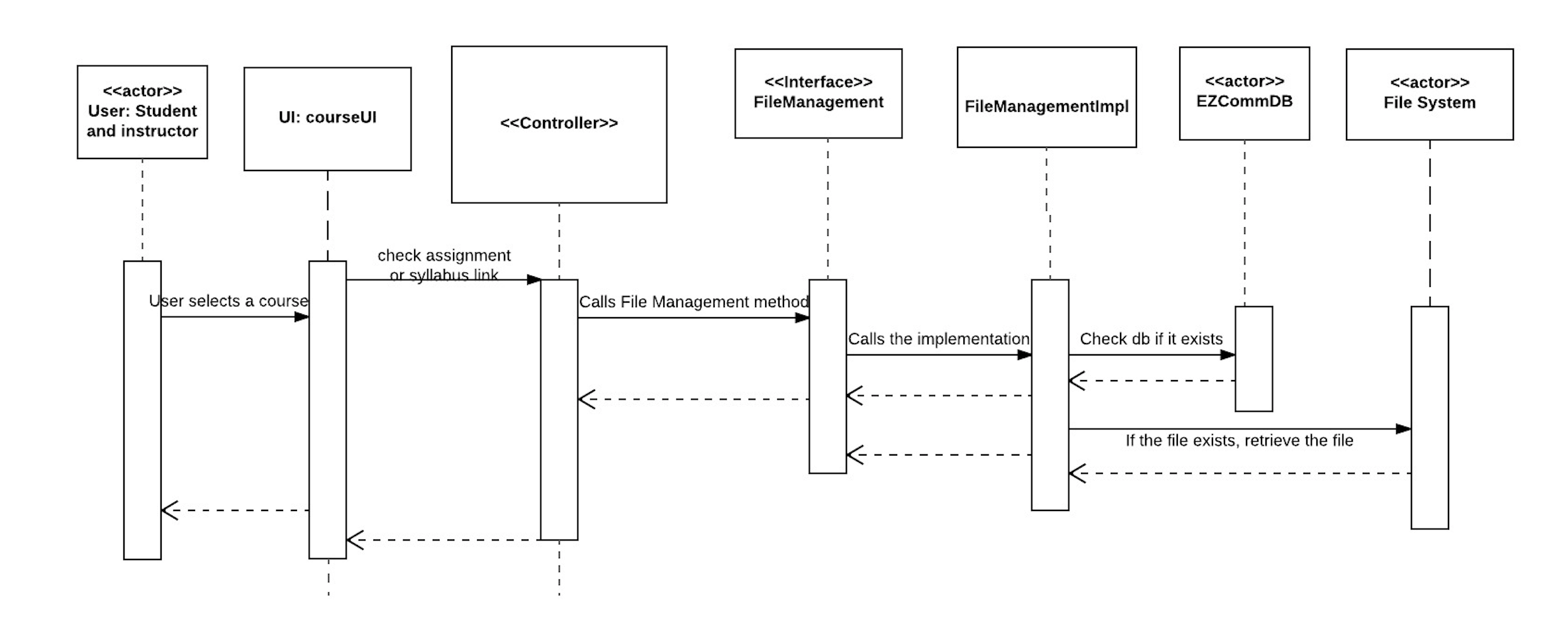
## 

## 1.3 Detailed Design Behavior

**Store File**

****

**Retrieve File**

****

## 1.4 Data Structures

List is used to store values/objects when a collection is required.

## 1.5 Algorithms

No algorithm is used in this component.

## 1.6 Architecture to Detailed Design Tracing

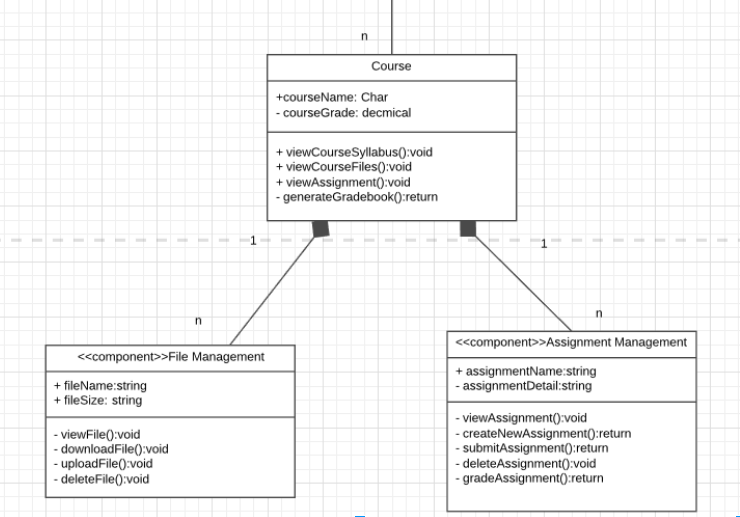
This component does not use any other components in the system. It provides an interface that other components can use.

**Component 2: Assignment Management (Emily+Guangchu)**

## 2.1 Detailed Design Key Decisions and Rationale

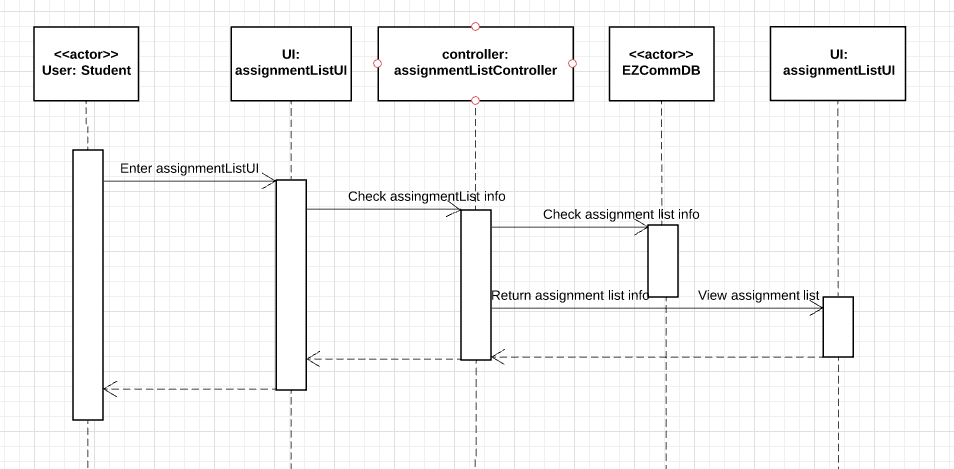
|  |  |  |
| --- | --- | --- |
| **Key decisions** | **Rationale** | **Use of design patterns** |
| View assignment list and assignment details | This provides a way for the students to view the assignment list as well as the detailed information (including files) of each assignment | Broker |
| Submit assignment | Students can submit files on the assignment page and files will be stored in database |  |
| Instructor grades assignment | Instructor is able to have access to all submission files by all students. Instructor views each submission file and grades it. |  |
| Instructor creates and deletes assignments | Instructor has the ability to create as well as delete assignments |  |

## 2.2 Detailed Design Structure

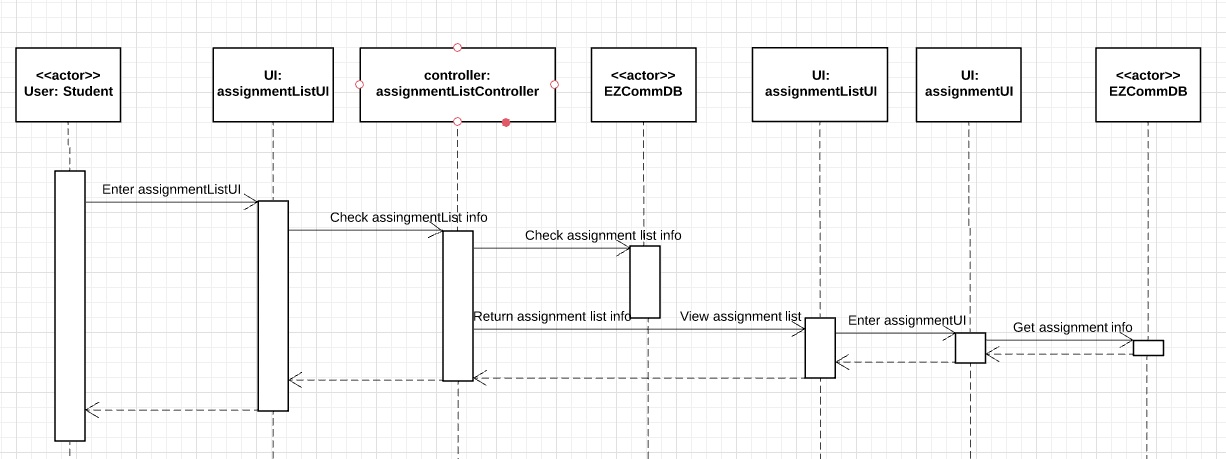
**

## 2.3 Detailed Design Behavior

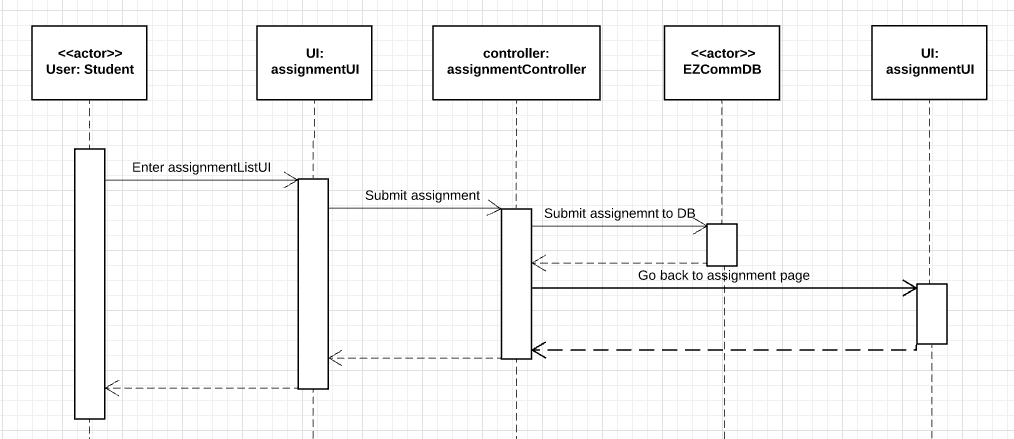
View assignment list:



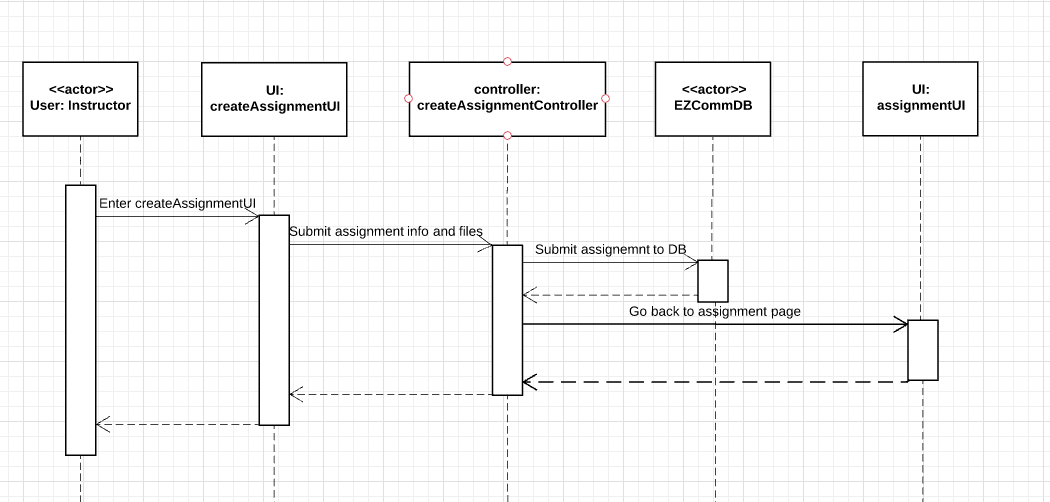
View assignment:



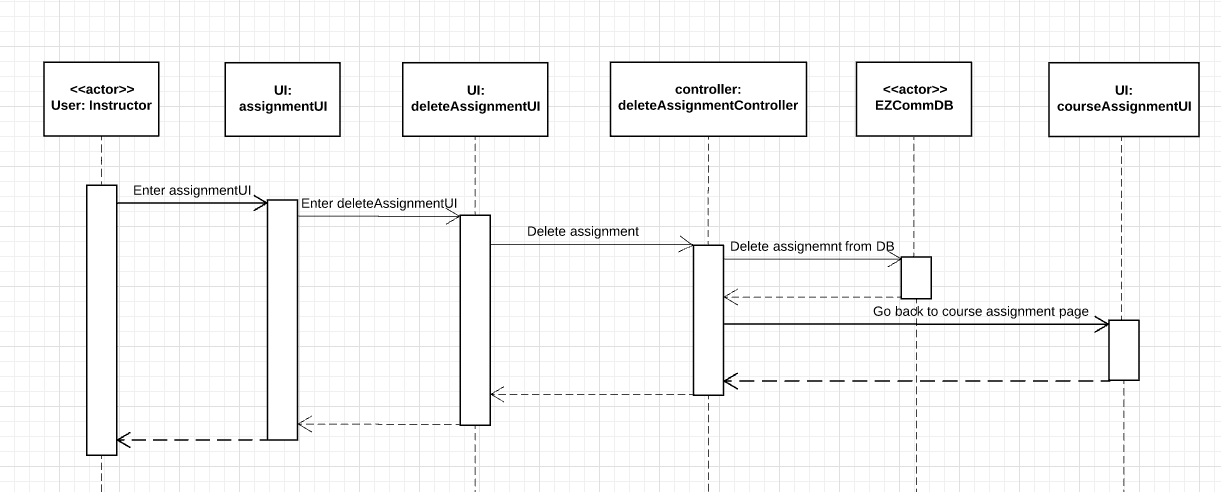
Submit assignment:



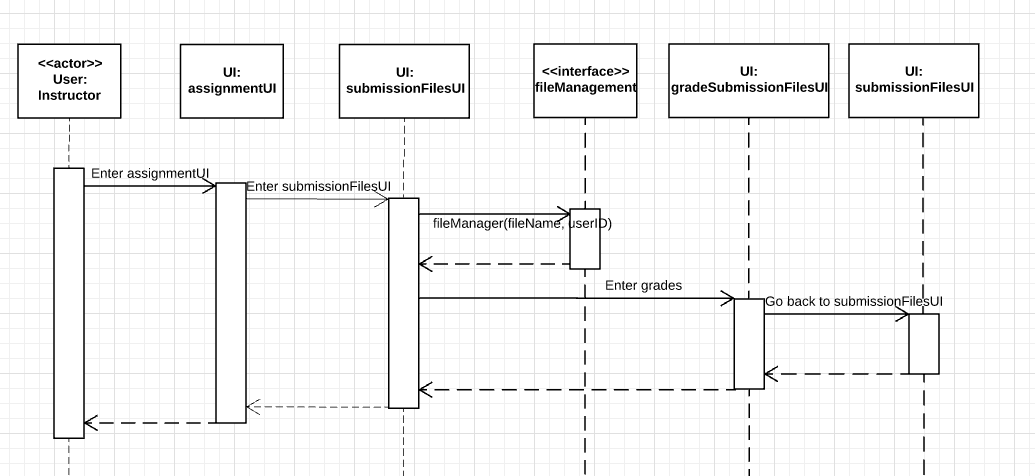
Create new assignment:



Delete assignment:



Grade assignment:



## 2.4 Data Structures

A List object is used to store the fields that have a list of objects.

## 2.5 Algorithms

No algorithm is used in this component.

## 2.6 Architecture to Detailed Design Tracing

|  |  |  |  |
| --- | --- | --- | --- |
| **Architecture Components/ Detailed Design components** | **Assignment List Viewer** | **Assignment Viewer** | **Submission Files Viewer** |
| **User Account** | *x* | *x* | *x* |
| **Assignment Management** | *x* | *x* | *x* |
| **File Management** |  |  | *x* |
| **Course Management** |  |  | *x* |
| **User interface Management** | *x* | *x* | *x* |
| **Arch Interface** |  |  | *x* |

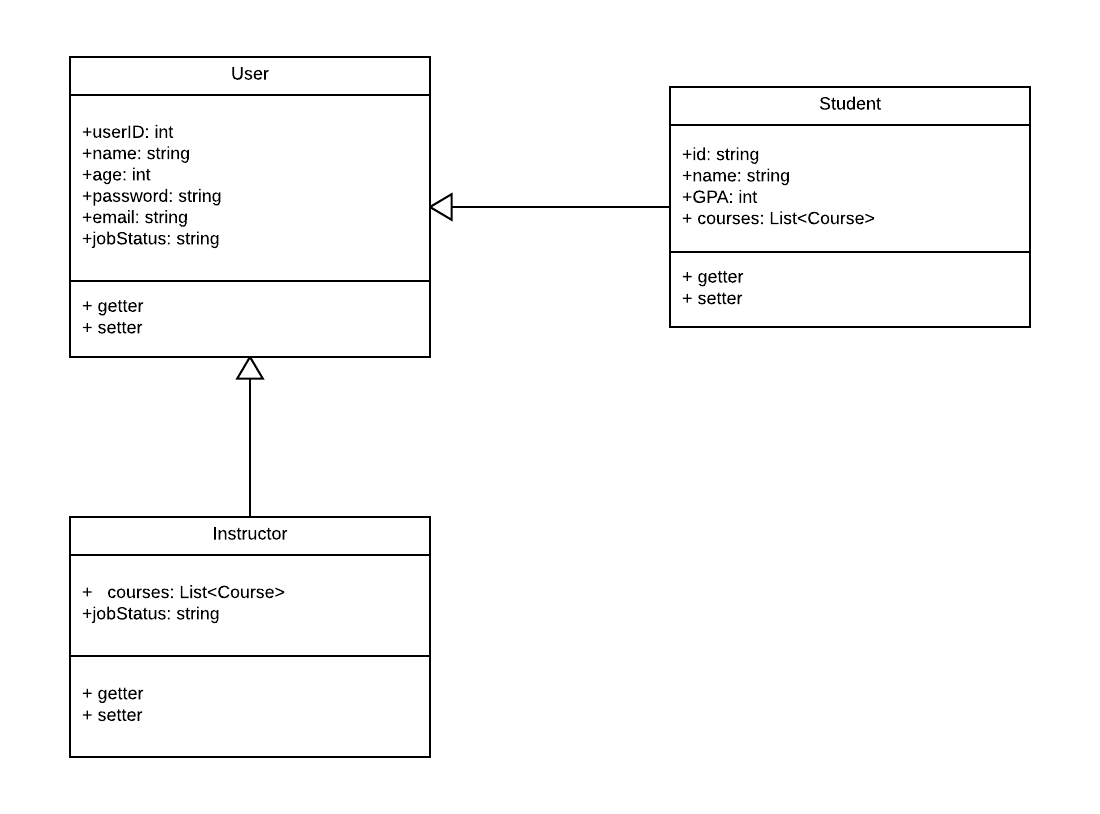
**Component 3: User account (Xiangfeng)**

## 3.1 Detailed Design Key Decisions and Rationale

|  |  |
| --- | --- |
| Key decisions | Rationale |
| login account with username and password (student and instructor) | Other alternative like multi factor authentication increases the security but significantly reduces usability |
| No sign up option(student and instructor) | Only students and instructors in university database should be granted the privilege to login. |
| Homepage will display after login | LMS only provide information to the registered students and instructors, not to general public. |

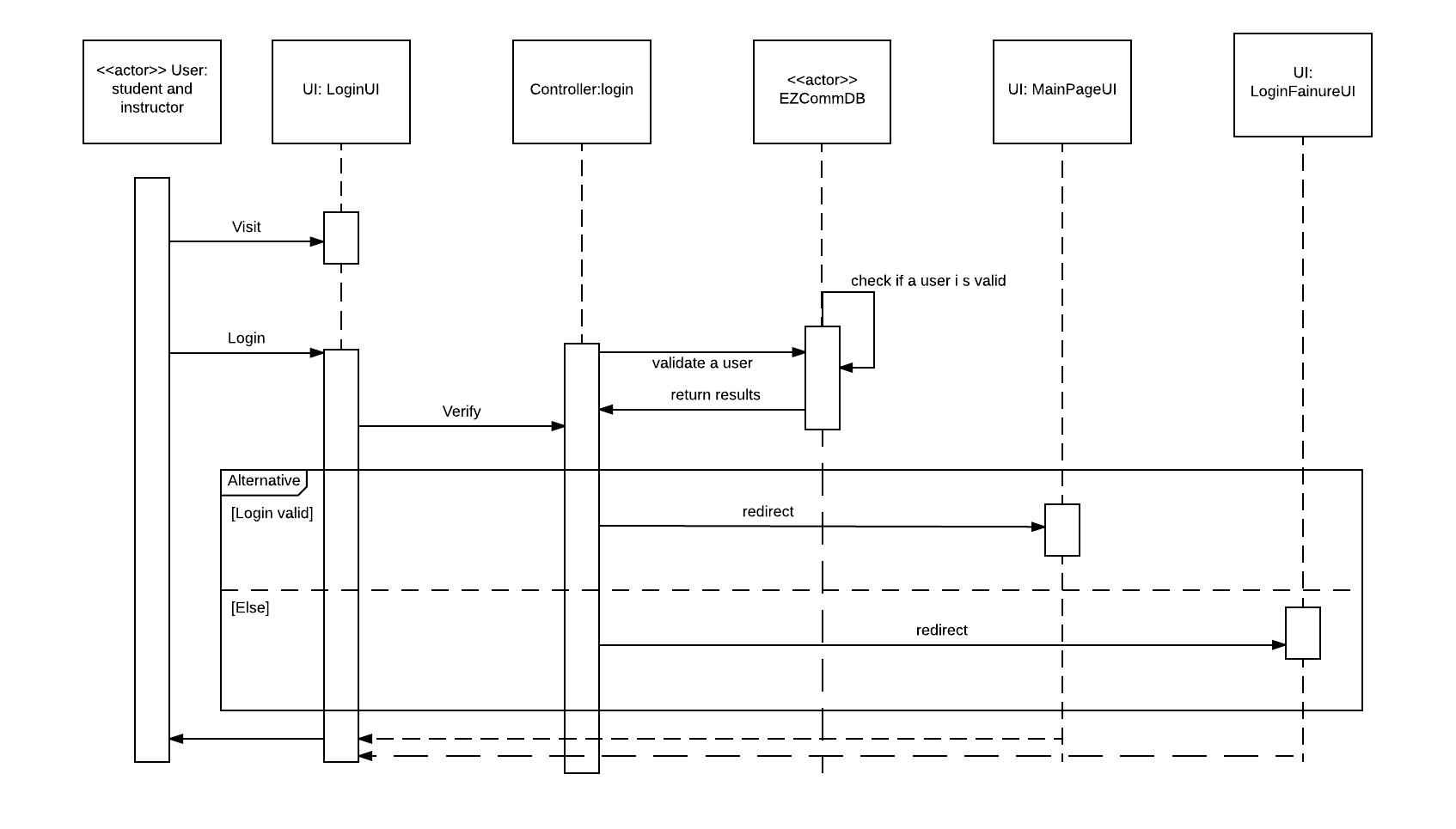
## 

## 3.2 Detailed Design Structure

**

## 3.3 Detailed Design Behavior

*<Insert here the detailed design behavior models, using* ***UML sequence diagrams****>*



## 3.4 Data Structures

Object of student and instructors are used to encapsulate the data and behavior

## 3.5 Algorithms

Hash encryption is used to increase the security of password.

## 3.6 Architecture to Detailed Design Tracing

|  |  |
| --- | --- |
| **Architecture Components/ Detailed Design components** | **Login** |
| **User Account** | X |
| **Assignment Management** | X |
| **File Management** | X |
| **Course Management** | X |
| **User interface Management** | X |

**Component 4: Course Management (Zeqian & Hongquan)**

## 4.1 Detailed Design Key Decisions and Rationale

*<Describe key decisions and their rationale, for example use of design patterns>*

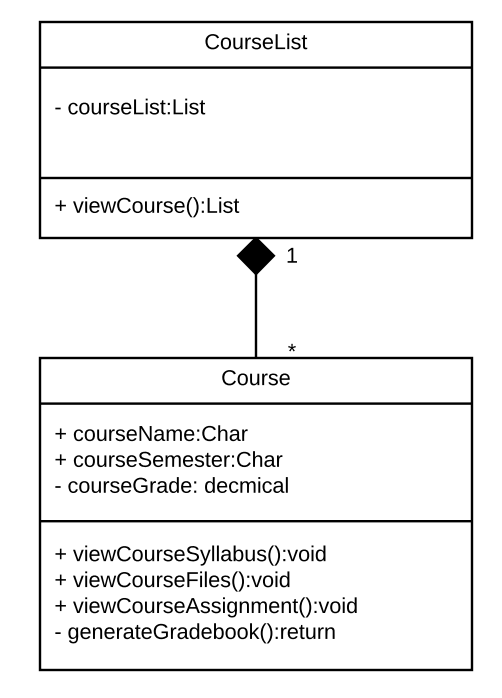
The course management is used for manage all the course related task.

There are several key decisions here and rationale, also the use of design patterns are corresponding to it:

|  |  |  |
| --- | --- | --- |
| **Key decisions** | **Rationale** | **Use of design patterns** |
| View course list by browsing card layouts | Card layout is easy to represent some details of the course | Broker patterns |
| View course syllabus by clicking the card layout to enter the course syllabus or clicking the course dropdown on the navigation bar | the card layout is a bigger rectangle which is easier than a table to click  the navigation bar a quick search for course, which is similar to user behavior in other systems. |  |
| View course files by clicking the card layout to enter the course syllabus or clicking the course dropdown on the navigation bar | the navigation bar a quick search for course files, which is similar to user behavior in other systems. |  |
| View assignment by clicking the assignment item the navigation bar | the navigation bar is a quick search for assignment, which is similar to user behavior in other systems. |  |
| View semester grade for courses by clicking the assignment item the navigation bar | the navigation bar is a quick search for semester grade, which is similar to user behavior in other systems. |  |

## 4.2 Detailed Design Structure

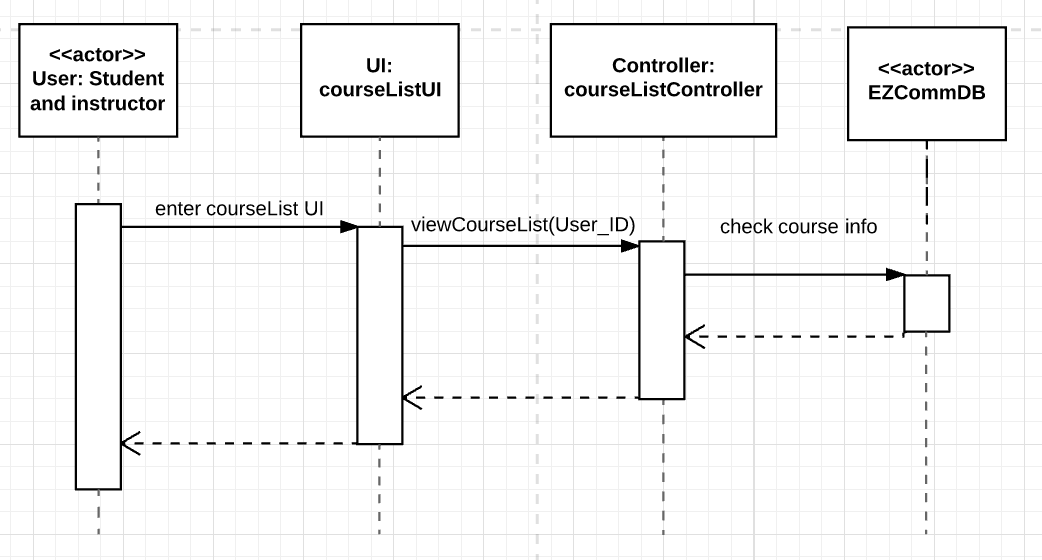
*<Insert here the detailed design static/structural model, using* ***UML class diagrams****>*

**

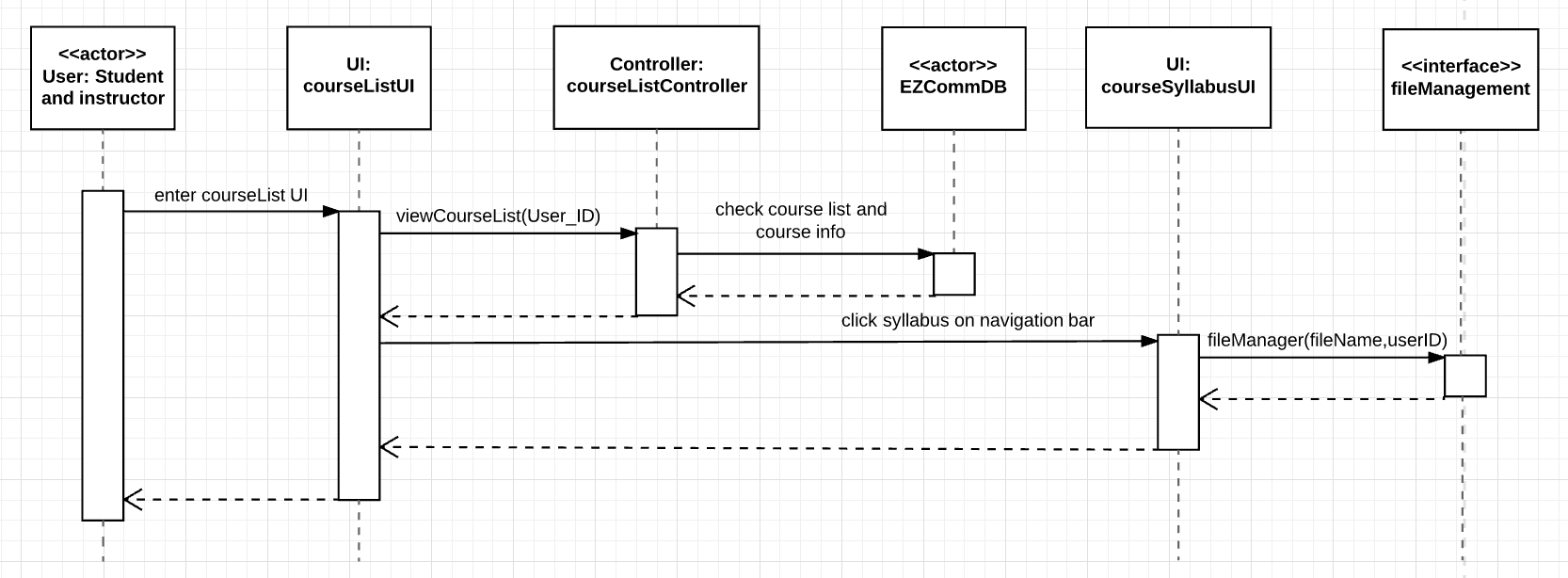
## 4.3 Detailed Design Behavior

*<Insert here the detailed design behavior models, using* ***UML sequence diagrams****>*

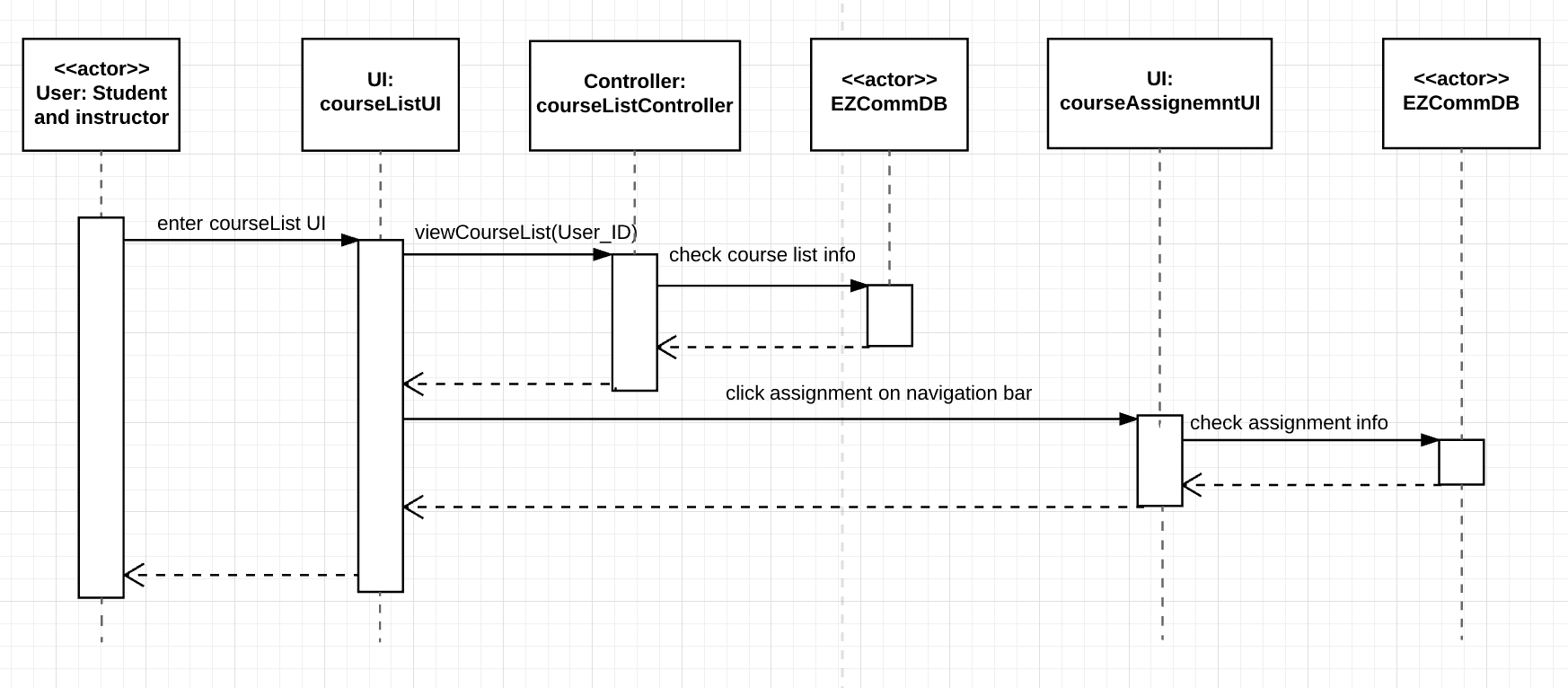
View course list



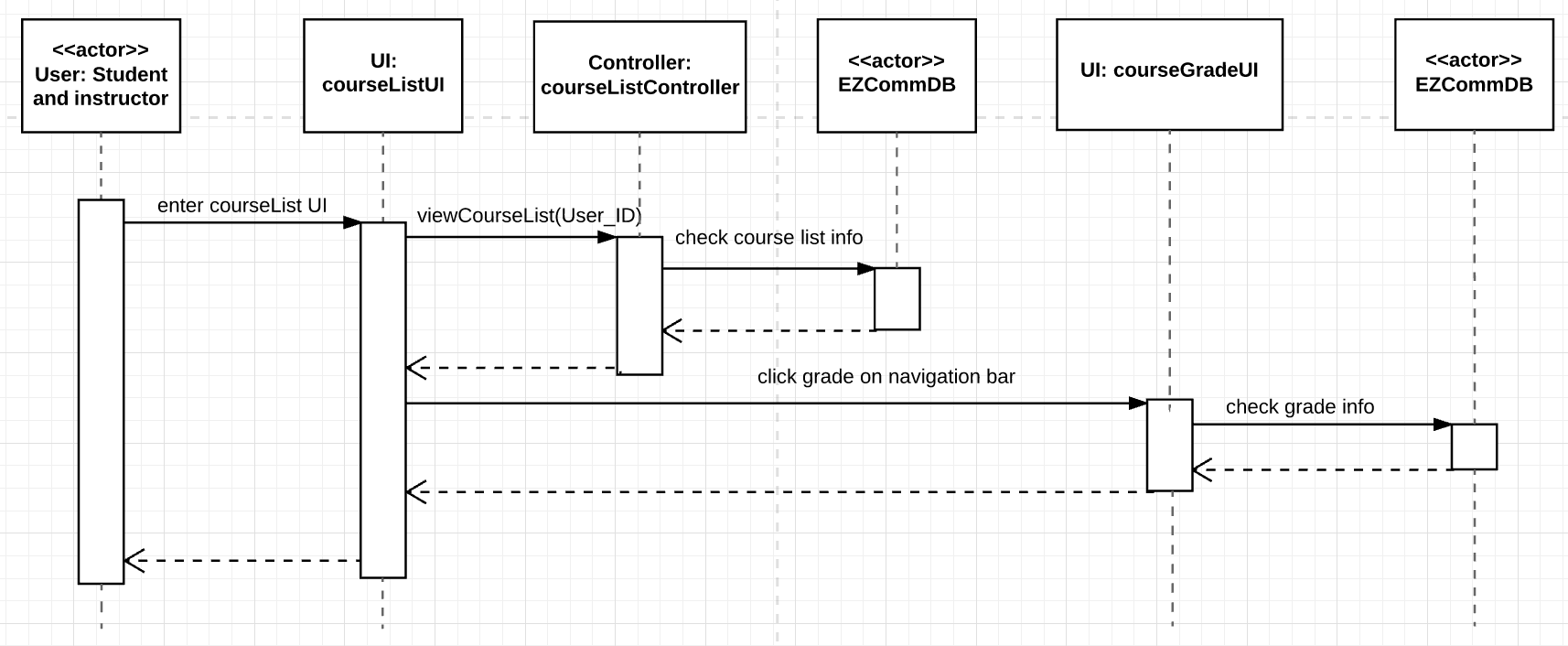
View course syllabus



View course assignment



view course grades



## 4.4 Data Structures

*<Description of data structures, if needed>*

List is used to contain the courses in the course management component.

## 4.5 Algorithms

*<Description of key algorithms, as needed>*

No algorithm is used in this component.

## 4.6 Architecture to Detailed Design Tracing

*<Trace architecture components and interfaces to detailed design components that realize them. Use a table format, such as the template below>*

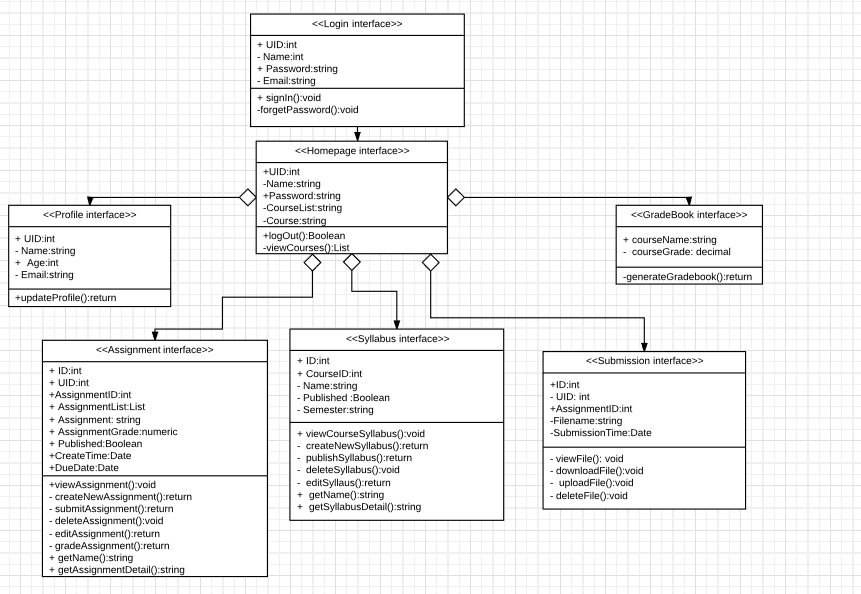
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Architecture Components/ Detailed Design components** | **Course Syllabus Viewer** | **Course Assignment viewer** | **Course Grade Viewer** | **Course List Viewer** |
| **User Account** | *x* | *x* | *x* | *x* |
| **Assignment Management** |  | *x* |  |  |
| **File Management** | *x* |  |  |  |
| **Course Management** | *x* | *x* | *x* | *x* |
| **User interface Management** | *x* | *x* | *x* | *x* |

**Component 5: User Interface (Emily)**

## 5.1 Detailed Design Key Decisions and Rationale

|  |  |  |
| --- | --- | --- |
| **Key decisions** | **Rationale** | **Use of design patterns** |
| Using a single login page to deal with security problem | The login page not only manages the legal accessibility of user to the system but also prevent the illegal access to the information related to courses since no course related information is presented without legal username and password. | Broker pattern; |
| The navigation bar on the top for navigation through different sections | Following the logic of the system. Courses are the main objects ,then there are several courses we use dropdown to provide several courses options, followed with profile and gradebook for student for information review which is not directly under the course. | Façade Pattern; |
| Use tab as a secondary navigation to manage the course related | objects like syllabus and assignment and grades or student list are listed under each course, so use this helps make it clear that all these objects are under certain courses. | Façade Pattern; |
| Model is used for the further actions under certain actions | For example, the submission button for assignment submission will guide you to a model where student will be able to upload documents which reduce the webpage number and more process more consistent. | Façade Pattern; |
| Table are commonly used for information display | Table helps to display different kinds of information in different columns which button and other can't achieve. It also makes the actions like inserting a new row or deleting or editing this row possible. | Façade Pattern; |

## 5.2 Detailed Design Structure

**

## 5.3 Data Structures

Basically for all the assignment and course list. linked list is used.

## 5.4 Algorithms

no algorithm is used.

## 5.5 Architecture to Detailed Design Tracing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Architecture Components/ Detailed Design components** | **Course Syllabus Viewer** | **Course Assignment viewer** | **Course Grade Viewer** | **Course List Viewer** |
| **Architecture interface** | x | x | x | x |