***Gadgeteers***   
 *TechnoTidbits*  
Software Architecture Document (SAD)

Team members:

KrishnaSai Reddy Patil

Pranav Jaipurkar

Aman Geetey

Tigist Shibeshi

Yili Zhang

Vijayadhaarani Vijayaarunachalam

**Techno-Tidbits**

Contents

[**Product Overview**](#_heading=h.4dc9pjcxa6nq)3

[**Architecture Drivers**](#_heading=h.f3b6z12xtnd)3

[2.1 Functional requirements](#_heading=h.is4lux2fppq3) 3

[2.2 Design Constraints:](#_heading=h.bzzwf3l37czf) 4

[2.3 Quality Attributes:](#_heading=h.l1hiszntolxg) 5

[**Architecture Key Decisions and Rationale**](#_heading=h.4d3f313yzujm)5

[3.1 Architecture Style](#_heading=h.9iwnzhj44j2q) 5

[3.2 Architecture tactics:](#_heading=h.tw3ovpl7etd) 7

[**Architecture Logical View**](#_heading=h.z5n8ochp1dez)8

[Subsystem Diagram](#_heading=h.ygrzv4z40wo5) 8

[Subsystem Description](#_heading=h.vf430yfz2000) 8

[Logical view using UML component diagram](#_heading=h.4wmaqrcau7vo) 9

[Component Description](#_heading=h.9f9rma5j827t) 9

[Interface Description](#_heading=h.12zq6x7hhf1t) 14

[**Architecture Behavior View**](#_heading=h.w5tr7xqnygod)29

[**Architecture Deployment View**](#_heading=h.3wbu35n2mzov)35

[**Information View**](#_heading=h.kag72u7hxh5w)39

[**Requirements Allocation**](#_heading=h.g6iw10tla13s)40

[**Architecture Work Allocation**](#_heading=h.l39r1eathsmy)41

# 

## Product Overview

Techno Tidbits is an online learning platform that offers free courses for students and professionals to upskill and improve job-related skills. This platform allows users to sign up as instructors to tutor and help others in their area of expertise as well as sign up as students to learn new skills and technologies.

**Stakeholders**

1.     Students

2.     Instructors

3.     System Administrator

## Architecture Drivers

### 2.1 Functional requirements

**2.1.1 Student Functionality:**

1) The system should record the course(s) each student takes.

2) The system should record the progress of each course each student takes.

3) The student should be able to view course content in the course (s)he takes.

4) The student should be able to take an assignment in the course (s)he takes.

5) The student should be able to take a quiz in the course (s)he takes.

6) The student should be able to view the grades of taken quiz and assignment.

7) The student should be able to seek help from administrators to deal with system issues.

8) The student should be able to take/drop a course.

**2.1.2 Instructor Functionality:**

1) The system should record the course each instructor tutors.

2) The system should enable each instructor to add/remove/modify the course content, quizzes and assignments for the courses s/he tutors.

3) The system should enable each instructor to modify the syllabus for the courses s/he tutors.

4) The system should enable each instructor to modify the course content shown to each student in the course s/he tutors.

5) The instructor should be able to seek help from administrators to deal with system issues.

6) The instructor should be able to apply to/cancel instruct a course.

**2.1.3 Common User Requirements:**

1) Each user should be able to sign up for the first time, and sign in after then.

2) The system should display up-coming event for each user.

3) Each user should be able to read the syllabus for each course.

4) Each user should be able to read the manual of the system.

5) Each user should be able to browse help for operations s/he is able to do in a system.

6) Each user should be able to communicate with other students or instructor registered in the same course.

7) Each user should be modify his/her profile.

8) Each user should be able to search for a course.

**2.1.4 Administrator Functionality:**

1) The system should enable an administrator to modify the syllabus for each course.

2) The administrator should be able to add/remove/modify course content, quiz and assignment for each course.

3) The administrator should be able to modify the course content shown to each student in the course s/he tutors.

4) The system should display the help request raised by each user to an administrator.

5) The administrator should be able to add/remove student(s) from a course.

6) The administrator should be able to approve/decline application of a user to be an instructor of a course.

**2.1.5 Other system interface:**

1) Once a student finishes a quiz, the system should be able to auto-grade it.

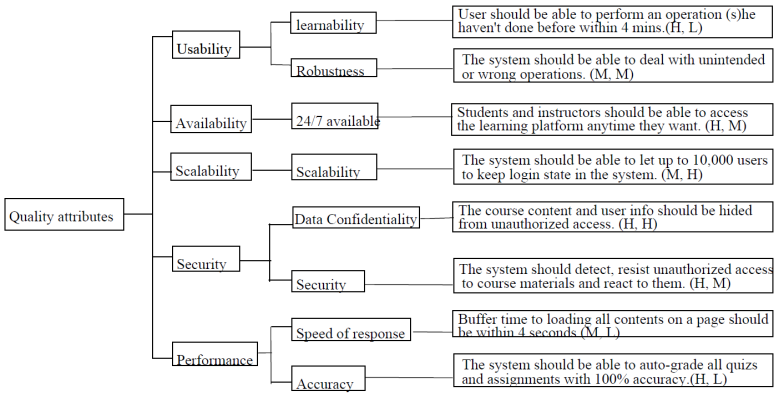
2) The system should only show operations that user is allowed to do within the system.

### 2.2 Design Constraints:

1) The course content; including syllabus, quiz and assignment should be stored in course database.

2) The authentication and profile information of users should be stored in a user database.

### 2.3 Quality Attributes:



## Architecture Key Decisions and Rationale

### 3.1 Architecture Style

3.1.1 Architecture style Rationale:

The system provides lots of clients with finite and independent services, including video lectures, discussion board, and modify/add/remove/take assignments/quizzes. Due to the independence of these services, the style structure has to be with high cohesion, flexible to modify and easy to exchange data between modules. Therefore, we should not consider styles like layered style, which has a hierarchical structure, or pipe-and-filter style, which is hard to share data and mainly deal with input/output data streams. On the other hand, the client-server style perfectly fits the needs of "multiple services and lots of clients".

We have considered using shared-data structure. However, considering that our system is not that large, so that we do not have the need to let that much modules to share some data. If using this style, it may lead to performance degrade. We hope that the system response is fast for the users. We have decided not to use shared-data style. Besides, because of the fact that data in the system needs to be modified quite frequently, once some error happens to the shared-data, the system might crash and we want to avoid that circumstance.

The system may have different types of users. To show them different views of the same module, we decided to use MVC style. On the other hand, if using publish-subscribe style to meet this requirement, if there is unreliable message bus, error may be generated. Besides, MVC style has a high cohesion for independent modules.

Thus we have chosen MVC and Client Server Style for the representation of our Architecture.

The logical view shows the MVC architecture and Client Server Style,

MVC

**Model** - Data Access Model

**View** - User Interface

**Controller** - Controller

Client Server Style

**Client Side Component -** User Interface

**Server Side Component** - Controller and Data Access Model

**3.1.2 Advantages of architecture style choice:**

1. Clients can share one server for different services.
2. Responsibilities are clear separated among the client and server, modules can be with high cohesion and low coupling.
3. Client and server can run concurrently.
4. Different type of clients can have different views for the same module.
5. Easy to maintain the system with high security.
6. Our team can develop simultaneously.

**3.1.3 Disadvantages of architecture style choice:**

1. The framework navigation may be complex.
2. Thread management may be complex.
3. It may not be easy to decompose every feature into three artifacts.
4. The server may handle heavy traffic, so we will need load balancing.

### 3.2 Architecture tactics:

**Data Confidentiality:**

The software should detect, resist scenarios if unauthorized user is trying to get access of course materials and react to them accordingly.

**a)** Security tactics **(Resist Attack):**

The software should detect, resist scenarios if unauthorized user is trying to get access of course materials and react to them accordingly. The Authorization Component handles the “resist attack” tactics

b) Security Tactics (**Detect Attack**):

The system should detect any unauthorized user attempting to login to our website. The Issue Management Component handles this “detect attack” tactics

**Advantages of using Security tactics:**

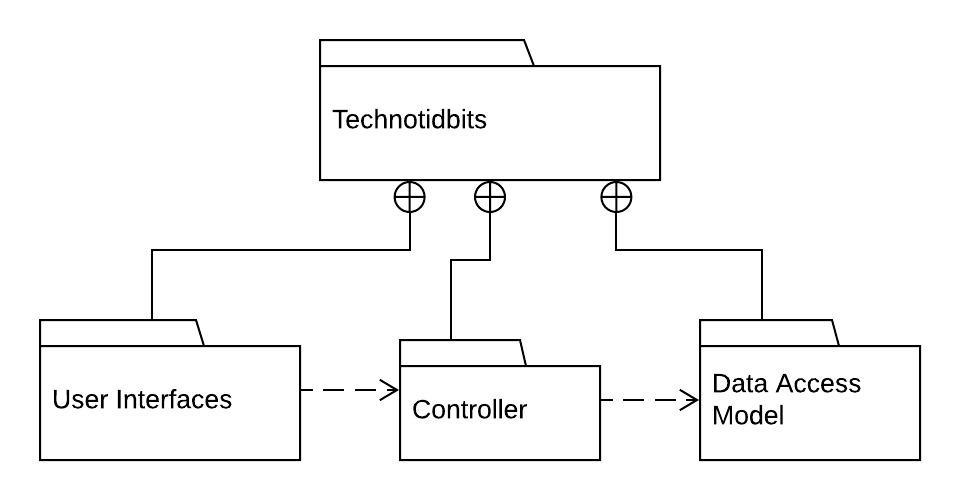
1. Robustness
2. Reliable

**Disadvantages of using Security tactics:**

1. Slow (due to constraints of different tactics overlapping in each other’s performance).
2. Difficult to implement.

## Architecture Logical View

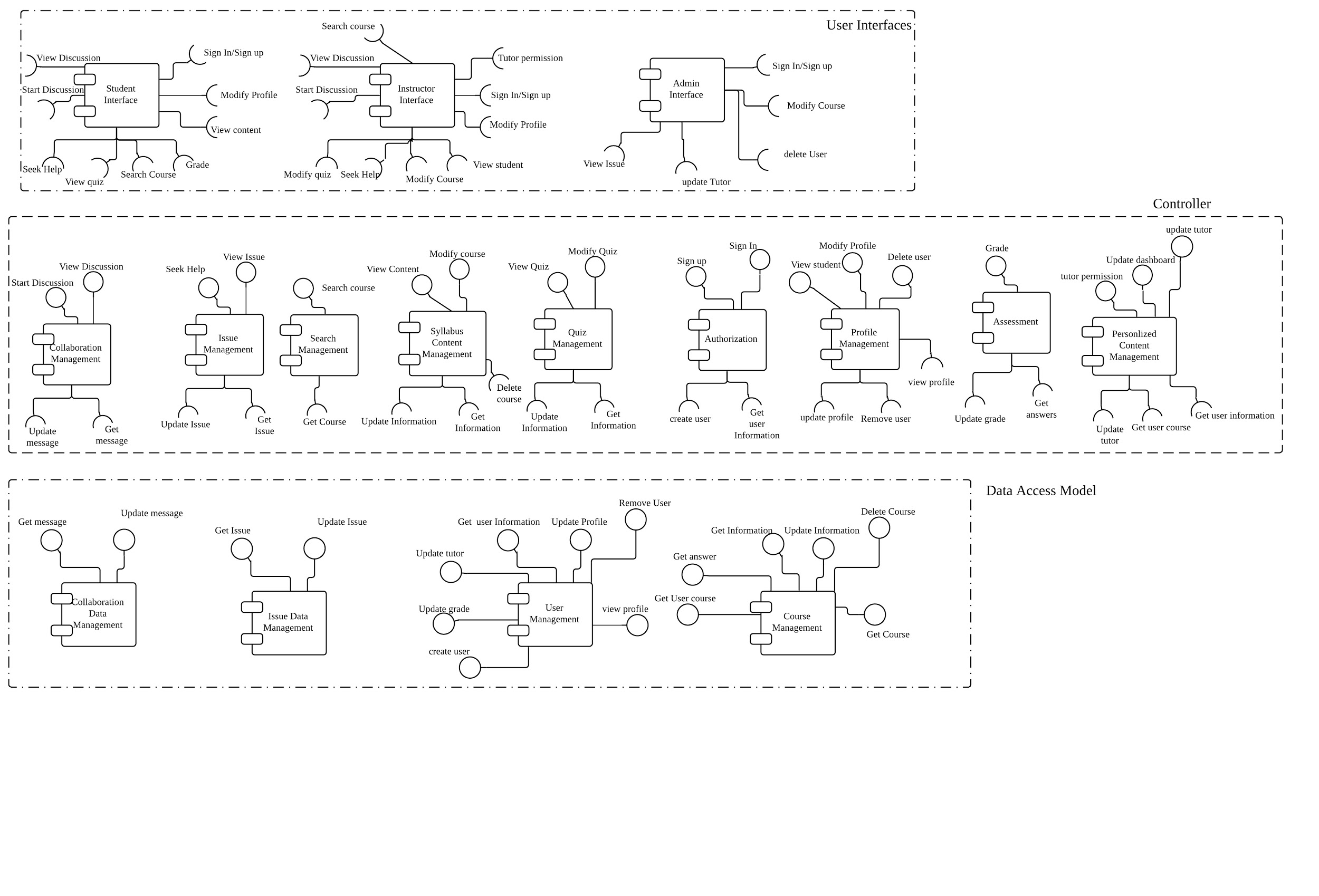
### Subsystem Diagram



### Subsystem Description

|  |  |
| --- | --- |
| **Subsystem Name** | **Subsystem Responsibilities** |
| User Interfaces | Reads the input from the user and invokes the corresponding services and displays the results obtained from the controllers to the users. |
| Controller | Provides all the services requested by the user and interacts with the data access model to obtain information from the database. |
| Data Access Model | Interacts with the database and performs read,write and update on the records that store course information and user information. |

### Logical view using UML component diagram

**

### Component Description

* + 1. Student Interface Component

|  |  |
| --- | --- |
| Component Name | Student Interface |
| Component Responsibilities | Enable the student to view and do operations they are authorized to |
| Provided Interface | None |
| Required Interface | View Discussion, Sign in/sign up, Start Discussion, Modify Profile, View Content, Seek Help,View Quiz, Search Course,Grade |

* + 1. Instructor Interface

|  |  |
| --- | --- |
| Component Name | Instructor Interface |
| Component Responsibilities | Enable the instructor to view and do operations they are authorized to |
| Provided Interface | None |
| Required Interface | View Discussion, Sign in/sign up, Start Discussion, Modify Profile, View Student, Seek Help, Modify Quiz, Modify Course, Tutor Permission, Search Course |

* + 1. Admin Interface

|  |  |
| --- | --- |
| Component Name | Admin Interface |
| Component Responsibilities | Enable the admin to view and do operations they are authorized to |
| Provided Interface | None |
| Required Interface | Sign in/Sign up, Modify Course, Delete User,  View Issue, Update Tutor |

* + 1. Personalized Content Management Component

|  |  |
| --- | --- |
| Component Name | Personalized Content Management |
| Component Responsibilities | Enables the dashboard view with data corresponding to the user. Handles the tutor permission for the courses. |
| Provided Interface | Update tutor, update dashboard, tutor permission |
| Required Interface | Update tutor, Get user course, Get user information |

* + 1. Assessment Component

|  |  |
| --- | --- |
| Component Name | Assessment |
| Component Responsibilities | Grade the student based on his/her performance in the quiz |
| Provided Interface | Grade |
| Required Interface | Get answers, Update grade |

* + 1. Profile Management Component

|  |  |
| --- | --- |
| Component Name | Profile Management |
| Component Responsibilities | Manages the profile information of the user and enables deletion of user and viewing other profiles |
| Provided Interface | Delete User, Modify Profile, View Student |
| Required Interface | view profile, update profile, remove user |

* + 1. Authorization Component

|  |  |
| --- | --- |
| Component Name | Authorization |
| Component Responsibilities | Records a user's identification as a student, an instructor, or an administrator and allows only a user to do operations that he/she is authorized to do in the system. |
| Provided Interface | Sign In, Sign up |
| Required Interface | Get user Information, create user |

* + 1. Quiz Management Component

|  |  |
| --- | --- |
| Component Name | Quiz Management |
| Component Responsibilities | Enables the student to view, and take a quiz for the course he/she is enrolled. Enables the instructor to update and view the quiz for the course he/she tutors. |
| Provided Interface | Update Quiz, View Quiz |
| Required Interface | Get Information, Update Information |

* + 1. Syllabus Content Management Component

|  |  |
| --- | --- |
| Component Name | Syllabus Content Management |
| Component Responsibilities | Enables the student to view course and enables tutor and admin to modify course |
| Provided Interface | Modify Course, View content |
| Required Interface | Get Information, Update Information, Delete course |

* + 1. Search Management Component

|  |  |
| --- | --- |
| Component Name | Search Management |
| Component Responsibilities | Enables the user to search for courses |
| Provided Interface | Search course |
| Required Interface | Get Course |

* + 1. Issue Management Component

|  |  |
| --- | --- |
| Component Name | Issue Management |
| Component Responsibilities | Allows user to seek help to the administrator to cope up with system issues. |
| Provided Interface | View Issue, seek help |
| Required Interface | Get Issue, Update Issue |

* + 1. Collaboration Management Component

|  |  |
| --- | --- |
| Component Name | Collaboration Management |
| Component Responsibilities | Enables to users to start and view discussions |
| Provided Interface | Start Discussion, View Discussion |
| Required Interface | Get Message, Update Message |

* + 1. Collaboration Data Management Component

|  |  |
| --- | --- |
| Component Name | Collaboration Data Management |
| Component Responsibilities | Updates and gets discussion information from the database |
| Provided Interface | Get Message, Update Message |
| Required Interface | None |

* + 1. Issue Data Management Component

|  |  |
| --- | --- |
| Component Name | Issue Data Management |
| Component Responsibilities | Gets, updates issue data from the database |
| Provided Interface | Get Issue, Update Issue |
| Required Interface | None |

* + 1. User Management Component

|  |  |
| --- | --- |
| Component Name | User Management |
| Component Responsibilities | Gets, updates user data from the database |
| Provided Interface | Create User, Update Grade, Update Tutor, Get user Information, Update Profile, Remove User, View Profile |
| Required Interface | None |

* + 1. Course Management Component

|  |  |
| --- | --- |
| Component Name | Course Management |
| Component Responsibilities | Gets, updates course information from the database |
| Provided Interface | Get Course, Delete Course, Update Information, Get Information, Get answer, Get user course |
| Required Interface | None |

### Interface Description

**1. Interface name: Start Discussion**

**Interface responsibilities**: Interface triggers a request to generate a new discussion.

**Interface operations:**

**Generate Discussion**

Operation Signature : GenerateDiscussion(UserName: String, Course: String, Topic: String, Content: String): Strings

Operation Description : Get discussion topic and content.

Operation Preconditions: User is registered in the course.

Operation Postconditions : User's discussion content is obtained.

**Post Discussion**

Operation Signature : UpdateMessage(DiscussionData: Strings): Boolean

Operation Description : Post discussion content, storing to database

Operation Preconditions : Discussion content is obtained

Operation Postconditions : Discussion content is posted to database

**2. Interface name: View Discussion**

**Interface responsibilities**: Interface updates the view with the details of the discussion

**Interface operations**:

**Search discussion**

Operation Signature : SearchDiscussion(DiscussionTopic: String): Boolean

Operation Description : Triggers a search request for the discussion

Operation Preconditions : Discussion data is present in the database

Operation Postconditions : Discussion is located

**Get discussion**

Operation Signature : GetDiscussion(DiscussionTopic: String):Strings

Operation Description : Obtains the discussion data to be displayed

Operation Preconditions : Discussion data is located

Operation Postconditions : Discussion data is obtained

**Update view**

Operation Signature : UpdateView(DiscussionData: Strings): Boolean

Operation Description : Discussion data is formatted & view updated

Operation Preconditions : Discussion data is obtained

Operation Postconditions : Interface view is updated with the obtained data

**3. Interface name: Get Message**

**Interface responsibilities**: Get message from database

**Interface operations**:

**Search Topic**

Operation Signature : SearchTopic(Topic: String): Boolean

Operation Description : Search message topic in database

Operation Preconditions : Message topic is in database

Operation Postconditions : Message topic is located

**Get message**

Operation Signature : GetMessage(Topic: String): Strings

Operation Description : Get messages related to one topic.

Operation Preconditions : Message topic is located

Operation Postconditions : Messages within that topic is obtained

**4. Interface name: Update Message**

**Interface responsibilities**: Store new messages to the database

**Interface operations**:

**Update message**

Operation Signature : UpdateMessage(Topic: String, MessageFormat: Strings): Boolean

Operation Description : Store messages to database related to one topic

Operation Preconditions : Topic and message are obtained

Operation Postconditions : New message is stored into database with the topic

**5. Interface name: Seek Help**

**Interface responsibilities**: Forward Help Request from a user to an admin

**Interface operations**:

**Get Help Info**

Operation Signature : GetHelpInfo(Username: String, HelpTopic: String, HelpContent: String): Strings

Operation Description : Get help information from user input

Operation Preconditions : Help info is input by a user

Operation Postconditions : Help info is got

**Post Help Info**

Operation Signature : PostHelpInfo(HelpInfo: Strings): Boolean

Operation Description : Post help to database

Operation Preconditions : Help info is got

Operation Postconditions : Help info is stored into database

**6. Interface name: View Issue**

**Interface responsibilities**: Interface updates the view with the details of the issue

**Interface operations**:

**Search Issue**

Operation Signature : SearchIssue(IssueId: String): Boolean

Operation Description : Triggers a search request for the issue

Operation Preconditions : Issue data is present in the database

Operation Postconditions : Issue is located

**Get Issue**

Operation Signature : GetIssue(IssueId: String): String

Operation Description : Obtains the issue data to be displayed

Operation Preconditions : Issue data is located

Operation Postconditions : Issue data is obtained

**Update view**

Operation Signature : UpdateView(IssueData: Strings): String

Operation Description : Issue data is formatted & view updated

Operation Preconditions : Issue data is obtained

Operation Postconditions : Issue view is updated with the obtained data

**7. Interface name: Get Issue**

**Interface responsibilities**: Get issue from database

**Interface operations**:

**Get issue**

Operation Signature : GetIssue(IssueId: String): Strings

Operation Description : Get issue data from database

Operation Preconditions : Issue identifier is in database

Operation Postconditions : Issue data is located and obtained

**8. Interface name: Update Issue**

**Interface responsibilities**: Store new issue to the database

**Interface operations**:

**Update issue**

Operation Signature : UpdateIssue(IssueData: Strings): Boolean

Operation Description : Store issue into database

Operation Preconditions : Issue content is obtained

Operation Postconditions : New issue is stored into database.

**9. Interface name: Search Course**

**Interface responsibilities**: Get related course information with a input course identifier

**Interface operations**:

**Search course**

Operation Signature : SearchCourse(CourseId: String): Boolean

Operation Description : Trigger a search request for the course

Operation Preconditions : Course data is present in the database

Operation Postconditions : Course is located

**Get course**

Operation Signature : GetCourse(CourseId: String): Strings

Operation Description : Obtain the course data to be displayed

Operation Preconditions : Course is located

Operation Postconditions : Course data is obtained

**10. Interface name: Get Course**

**Interface responsibilities**: Get related course information from database

**Interface operations:**

**Search course**

Operation Signature : SearchCourse(CourseId: String): Boolean

Operation Description : Trigger a search request for the course

Operation Preconditions : Course data is present in the database

Operation Postconditions : Course is located

**Get course**

Operation Signature : GetCourse(CourseId: String): Strings

Operation Description : Obtain the course data from database

Operation Preconditions : Course is located

Operation Postconditions : Course data is obtained

**11. Interface name: View Content**

**Interface responsibilities**: Interface updates the view with the course syllabus content

**Interface operations**:

**Search Content**

Operation Signature : SearchContent(CourseId: String): Boolean

Operation Description : Trigger a search request for the course syllabus

Operation Preconditions : Course is located

Operation Postconditions : Course syllabus is located

**Get Content**

Operation Signature : GetContent(CourseId: String): String

Operation Description : Obtain the course content to be displayed

Operation Preconditions : Course syllabus is located

Operation Postconditions : Course syllabus content is obtained

**Update view**

Operation Signature : UpdateView(CourseContent: String): Boolean

Operation Description : Course syllabus content is formatted & view updated

Operation Preconditions : Course syllabus content is obtained

Operation Postconditions : Course syllabus view is updated with the obtained data

**12. Interface name: Modify Course**

**Interface responsibilities**: Interface updates the course syllabus content with the obtained input

**Interface operations**:

**Get Course Update**

Operation Signature : GetCourseUpdate(UpdatedContent: String): String

Operation Description : Obtain the updated course content

Operation Preconditions : Updated course content is input

Operation Postconditions : Updated course content is obtained

**Search Course**

Operation Signature : SearchCourse(CourseId: String): Boolean

Operation Description : Locate course data in database

Operation Preconditions : Course data in database

Operation Postconditions : Course data is located

**Update Course**

Operation Signature : UpdateCourse(UpdatedContent: String, CourseId: String): Boolean

Operation Description : Updated Course Content overwrites the original one for the specific course

Operation Preconditions : Updated course content obtained and course data is located

Operation Postconditions : Updated course content overwrites the old one in database

**13. Interface name: Delete Course**

**Interface responsibilities**: Remove course data from database

**Interface operations**:

**Search Course**

Operation Signature : SearchCourse(CourseId: String): Boolean

Operation Description : Locate course data in database

Operation Preconditions : Course data in database

Operation Postconditions : Course data is located

**Remove Course**

Operation Signature : RemoveCourse(CourseId: String): Boolean

Operation Description : Remove course data in database

Operation Preconditions : Course data is located

Operation Postconditions : Course data is removed from database

**14. Interface name: Get Information**

**Interface responsibilities**: Get course information from database

**Interface operations**:

**Search Course**

Operation Signature : SearchCourse(CourseId: String): Boolean

Operation Description : Locate course data in database

Operation Preconditions : Course data in database

Operation Postconditions : Course data is located

**Get information**

Operation Signature : GetInfo(CourseId: String, InfoId: String): String

Operation Description : Get specific course information for the course

Operation Preconditions : Course data is located

Operation Postconditions : Course information is obtained

**15. Interface name: Update Information**

**Interface responsibilities**: Update course information into database

**Interface operations**:

**Update Information**

Operation Signature : UpdateInfo(CourseId: String, InfoId: String, Info: String): Boolean

Operation Description : Update specific course information into database

Operation Preconditions : Updated information is obtained and course is located

Operation Postconditions : Course information is updated into database

**16. Interface name: View Quiz**

**Interface responsibilities**: Interface updates the view with the details of the quiz

**Interface operations**:

**Search quiz**

Operation Signature : SearchQuiz(QuizId: String): Boolean

Operation Description : Triggers a search request for the quiz

Operation Preconditions : Quiz data is present in the database

Operation Postconditions :Quiz is located

**Get Quiz**

Operation Signature : GetQuiz(QuizId: String): Strings

Operation Description : Obtains the quiz data to be displayed

Operation Preconditions : Quiz data is present in the database

Operation Postconditions : Quiz is located and data is obtained

**Update view**

Operation Signature : UpdateView(QuizId: String): Boolean

Operation Description : Quiz data is formatted & view updated

Operation Preconditions : Quiz data is obtained

Operation Postconditions : Interface view is updated with quiz data

**17. Interface name: Modify Quiz**

**Interface responsibilities**: Interface updates the quiz with the obtained input

**Interface operations**:

**Get Quiz Update**

Operation Signature : GetQuizUpdate(Quiz: String): String

Operation Description : Obtain the updated quiz

Operation Preconditions : Updated quiz is input

Operation Postconditions : Updated quiz is obtained

**Search Quiz**

Operation Signature : SearchQuiz(QuizId: String): Boolean

Operation Description : Locate quiz in database

Operation Preconditions : Quiz in database

Operation Postconditions : Quiz data is located

**Update Quiz**

Operation Signature : UpdateQuiz(QuizId: String, Quiz: String): Boolean

Operation Description : Updated quiz overwrites the original one for the specific quiz

Operation Preconditions : Updated quiz obtained and quiz data is located

Operation Postconditions : Updated quiz overwrites the old one in database

**18. Interface name: Sign Up**

**Interface responsibilities**: Trigger a request to create a new user with input information

**Interface operations**:

**Search User**

Operation Signature : SearchUser(UserId: String): Boolean

Operation Description : Trigger a search request to locate user information

Operation Preconditions : User not in database

Operation Postconditions : Fail to locate user information, return user not exist

**Create User**

Operation Signature : CreateUser(UserId: String, Passwd: String, Infos: Strings): Boolean

Operation Description : Trigger a request to create a new user

Operation Preconditions : UserId not exist

Operation Postconditions : User account created

**19. Interface name: Sign In**

**Interface responsibilities**: Trigger a request to authorize a user as an existing user

**Interface operations**:

**Search User**

Operation Signature : SearchUser(UserId: String): Boolean

Operation Description : Trigger a search request to locate user information

Operation Preconditions : User in database

Operation Postconditions : User data is located

**Authorize User**

Operation Signature : AuthorizeUser(UserId: String, Passwd: String): Boolean

Operation Description : Authorize the user account if password is correct for the UserId

Operation Preconditions : User data is located and password is correct

Operation Postconditions : User is authorized to the account

**Get User Information**

Operation Signature : GetUserInfo(UserId: String, State: Boolean): String

Operation Description : Load user information for the user account

Operation Preconditions : User data is located and user is identified as logged in

Operation Postconditions : User information is obtained

**Update view**

Operation Signature : UpdateView(UserInfo: Strings): Boolean

Operation Description : User information is formatted & view updated

Operation Preconditions : User information is obtained

Operation Postconditions : Interface view is updated with the obtained data

**20. Interface name: Create User**

**Interface responsibilities**: Create a new user account

**Interface operations**:

**Create User**

Operation Signature : CreateUser(UserId: String, Passwd: String, UserInfo: Strings): Boolean

Operation Description : Create a new user account with set data

Operation Preconditions : User account not in database

Operation Postconditions : User account created in database

**21. Interface name: Get User Information**

**Interface responsibilities**: Trigger a request to obtain user information in database

**Interface operations**:

**Search User**

Operation Signature : SearchUser(UserId: String): String

Operation Description : Trigger a search request to locate user information

Operation Preconditions : User in database

Operation Postconditions : User is located

**Get user information**

Operation Signature : GetUserInfo(UserId: String, State: Boolean): String

Operation Description : Obtain user information from database

Operation Preconditions : User is located and user is authorized

Operation Postconditions : User information is obtained

**22. Interface name: Modify Profile**

**Interface responsibilities**: Interface updates the user profile

**Interface operations**:

**Get Profile Info**

Operation Signature : GetProfileInfo(UpdatedInfo: Strings): String

Operation Description : Obtain the updated profile

Operation Preconditions : Updated profile is input

Operation Postconditions : Updated profile is obtained

**Search User**

Operation Signature : SearchUser(UserId: String): Boolean

Operation Description : Locate user in database

Operation Preconditions : User in database

Operation Postconditions : User data is located

**Update Profile**

Operation Signature : UpdateProfile(UpdatedInfo: Strings, UserId: String, State: Boolean): Boolean

Operation Description : Updated profile overwrites the original one for the user

Operation Preconditions : Updated profile obtained, user data is located, and user is authorized

Operation Postconditions : Updated profile overwrites the old one in database

**23. Interface name: View Profile**

Interface responsibilities: Interface updates the view with the user profile

Interface operations:

**Search User**

Operation Signature : SearchUser(UserId: String): Boolean

Operation Description : Locate user in database

Operation Preconditions : User in database

Operation Postconditions : User data is located

**Get Profile**

Operation Signature : Get Profile(UserId: String): Strings

Operation Description : Obtains the user profile data to be displayed

Operation Preconditions : User data is located

Operation Postconditions : Profile data is obtained

**Update view**

Operation Signature : UpdateView(ProfileData: Strings): Boolean

Operation Description : Profile data is formatted & view updated

Operation Preconditions : Profile data is obtained

Operation Postconditions : Interface view is updated with the obtained data

**24. Interface name: Update Profile**

**Interface responsibilities**: Store updated profile to the database

**Interface operations**:

**Update Profile**

Operation Signature : UpdateProfile(UpdatedInfo: Strings, UserId: String, State: Boolean): Boolean

Operation Description : Updated profile overwrites the original one for the user

Operation Preconditions : Updated profile obtained, user data is located, and user is authorized

Operation Postconditions : Updated profile overwrites the old one in database

**25. Interface name: Delete User**

**Interface responsibilities**: Trigger a request to remove user data from database

**Interface operations**:

**Search User**

Operation Signature : SearchUser(UserId: String): Boolean

Operation Description : Locate user data in database

Operation Preconditions : User data in database

Operation Postconditions : User data is located

**Remove User**

Operation Signature : RemoveUser(UserId: String): Boolean

Operation Description : Remove user data in database

Operation Preconditions : User data is located

Operation Postconditions : User data is removed from database

**26. Interface name: Remove User**

**Interface responsibilities**: Remove user account from database

**Interface operations:**

**Remove User**

Operation Signature : RemoveUser(UserId: String): Boolean

Operation Description : Remove user data in database

Operation Preconditions : User data is located

Operation Postconditions : User data is removed from database

**27. Interface name: View Student**

**Interface responsibilities**: Interface triggers a request to update the view of the student user profile

**Interface operations:**

**Search Student**

Operation Signature : SearchStudent(StudentId: String): Boolean

Operation Description : Locate student data in database

Operation Preconditions : Student data in database

Operation Postconditions : Student data is located

**Get Profile**

Operation Signature : GetProfile(StudentId: String): Strings

Operation Description : Obtains the student profile data to be displayed

Operation Preconditions : Student data is located

Operation Postconditions : Student profile data is obtained

**Update view**

Operation Signature : UpdateView(ProfileData: Strings): Boolean

Operation Description : Profile data is formatted & view updated

Operation Preconditions : Student profile data is obtained

Operation Postconditions : Interface view is updated with the obtained data

28. Interface name: Grade

**Interface responsibilities**: Trigger a request to do grading for a file

**Interface operations**:

**Get Answer**

Operation Signature : GetAnswer(QuizId: String): Strings

Operation Description : Locate answers of the quiz

Operation Preconditions : Quiz data is located

Operation Postconditions : Quiz answer is located and obtained

**Grade**

Operation Signature : Grade(Input: Strings, Target: Strings): Strings

Operation Description : Grade the quiz with user answer and standard answer

Operation Preconditions : Quiz answer is obtained and quiz is submitted

Operation Postconditions : Quiz is graded

**Update Grade**

Operation Signature : UpdateGrade(QuizId: String, UserId: String, Grade: String): Boolean

Operation Description : Trigger a request to save the grade of the quiz into the database related to the student

Operation Preconditions : Quiz is graded

Operation Postconditions : Student's quiz grade is saved into database

**29. Interface name: Update Grade**

**Interface responsibilities:**

**Interface operations**:

**Update Grade**

Operation Signature : UpdateGrade(QuizId: String, UserId: String, Grade: String): Boolean

Operation Description : Save the grade of the quiz into the database related to the student

Operation Preconditions : Quiz is graded

Operation Postconditions : Student's quiz grade is saved into database

**30. Interface name: Get answers**

**Interface responsibilities**: Obtain the answer of a specific quiz

**Interface operations**:

**Get Answer**

Operation Signature : GetAnswer(QuizId: String): Strings

Operation Description : Locate answers of the quiz

Operation Preconditions : Quiz data is located

Operation Postconditions : Quiz answer is obtained

**31. Update Tutor**

**Interface responsibilities**: Trigger a request to authorize a user to be tutor of a course

**Interface operations**:

**Search course**

Operation Signature : SearchCourse(CourseId: String): Boolean

Operation Description : Trigger a search request for the course

Operation Preconditions : Course data is present in the database

Operation Postconditions : Course is located

**Search User**

Operation Signature : SearchUser(UserId: String): Boolean

Operation Description : Trigger a search request to locate user information

Operation Preconditions : User in database

Operation Postconditions : User data is located

**Update Tutor**

Operation Signature : UpdateTutor(CourseId: String, UserId: String, AdminState: Boolean): Boolean

Operation Description : Set the user as new tutor for the course

Operation Preconditions : Authorize as an admin, user in database and course in database

Operation Postconditions : The user is set as the new tutor for the course

**32. Interface name: Tutor Permission**

**Interface responsibilities**: Permit/decline a user's request to be tutor of a course

**Interface operations**:

**Search Request**

Operation Signature : SearchRequest(RequestId: String): Boolean

Operation Description : Triggers a search for the permission request

Operation Preconditions : Permission request data is present in the database

Operation Postconditions : Permission request is located

**Get Request**

Operation Signature : GetRequest(RequestId: String): String

Operation Description : Obtains the request data to be displayed

Operation Preconditions : Permission request data is located

Operation Postconditions : Permission request data is obtained

**Update view**

Operation Signature : UpdateView(RequestData: Strings): String

Operation Description : Request data is formatted & view updated

Operation Preconditions : Request data is obtained

Operation Postconditions : Issue view is updated with the obtained data

**Update Tutor**

Operation Signature : UpdateTutor(CourseId: String, TutorId: String, AdminState: Boolean, PermissionState: Boolean): Boolean

Operation Description : Set new tutor for a course

Operation Preconditions : Authorize as an admin, permission is set as true, tutor in database and course in database

Operation Postconditions : The user is set as the new tutor for the course

**33. Interface name: Update Dashboard**

**Interface responsibilities**: Interface updates the dashboard view according to the user account information

**Interface operations**:

**Search User**

Operation Signature : SearchUser(UserId: String): Boolean

Operation Description : Trigger a search request to locate user information

Operation Preconditions : User in database

Operation Postconditions : User data is located

**Get User Course**

Operation Signature : GetUserCourse(UserId: String): Strings

Operation Description : Obtains the user course data to be displayed

Operation Preconditions : User data is located

Operation Postconditions : User course information is obtained

**Update view**

Operation Signature : UpdateView(UserCourse: Strings): Boolean

Operation Description : User dashboard view updated with user course data

Operation Preconditions : User course data is obtained

Operation Postconditions : User dashboard view is updated with the obtained data

**34. Interface name: Get User Course**

**Interface responsibilities**: Obtain user course information from database

**Interface operations**:

**Get User Course**

Operation Signature : GetUserCourse(UserId: String, State: Boolean)

Operation Description : Obtain user course information from database

Operation Preconditions : Request is from an authorized access, user is in database

Operation Postconditions : User course data is obtained

**35. Interface name: Update Tutor**

**Interface responsibilities**: Set new tutor for a specific course

**Interface operations**:

**Update tutor**

Operation Signature : UpdateTutor(CourseId: String, TutorId: String): Boolean

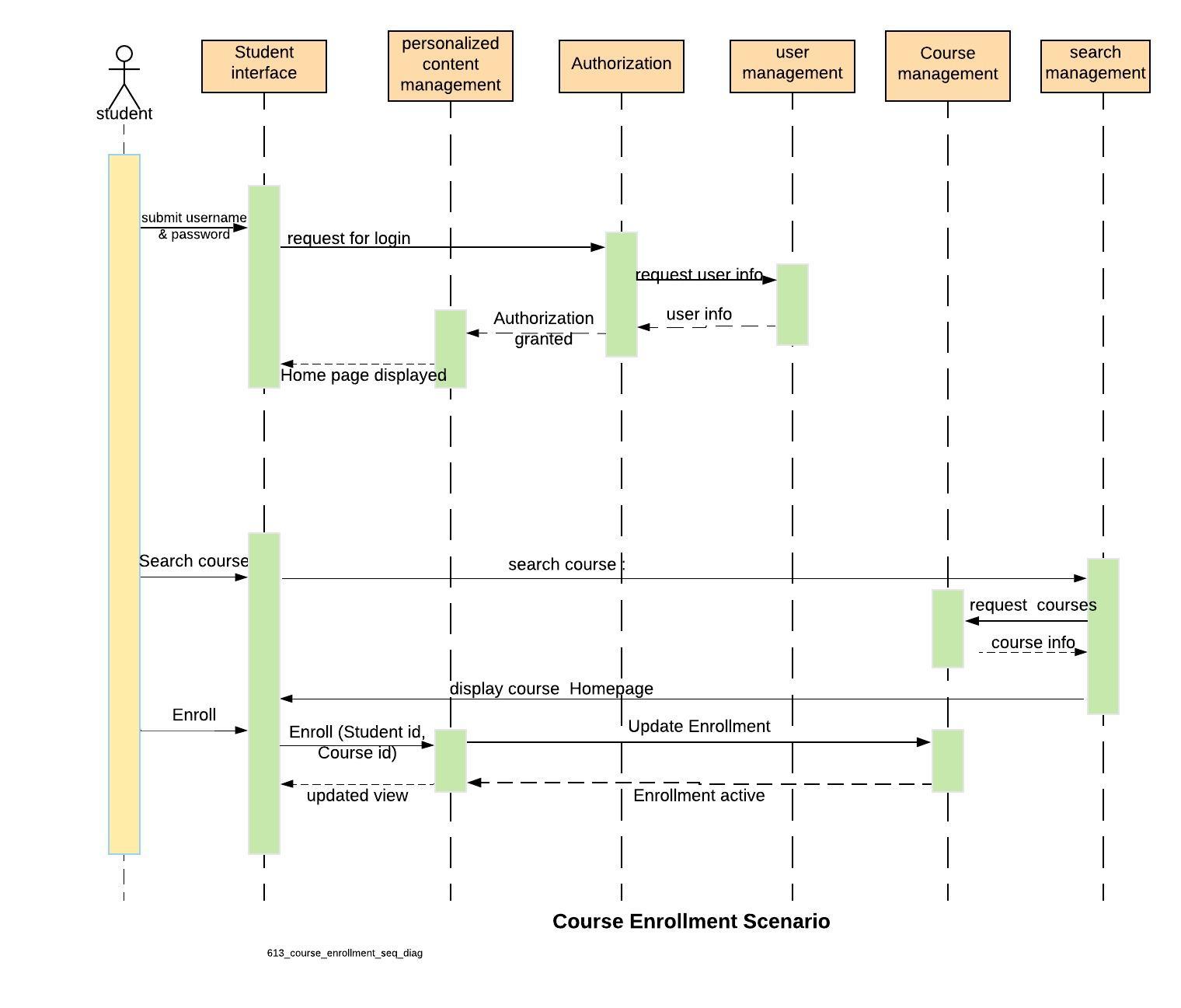
Operation Description : Set new tutor for a course

Operation Preconditions : Tutor in database and course in database

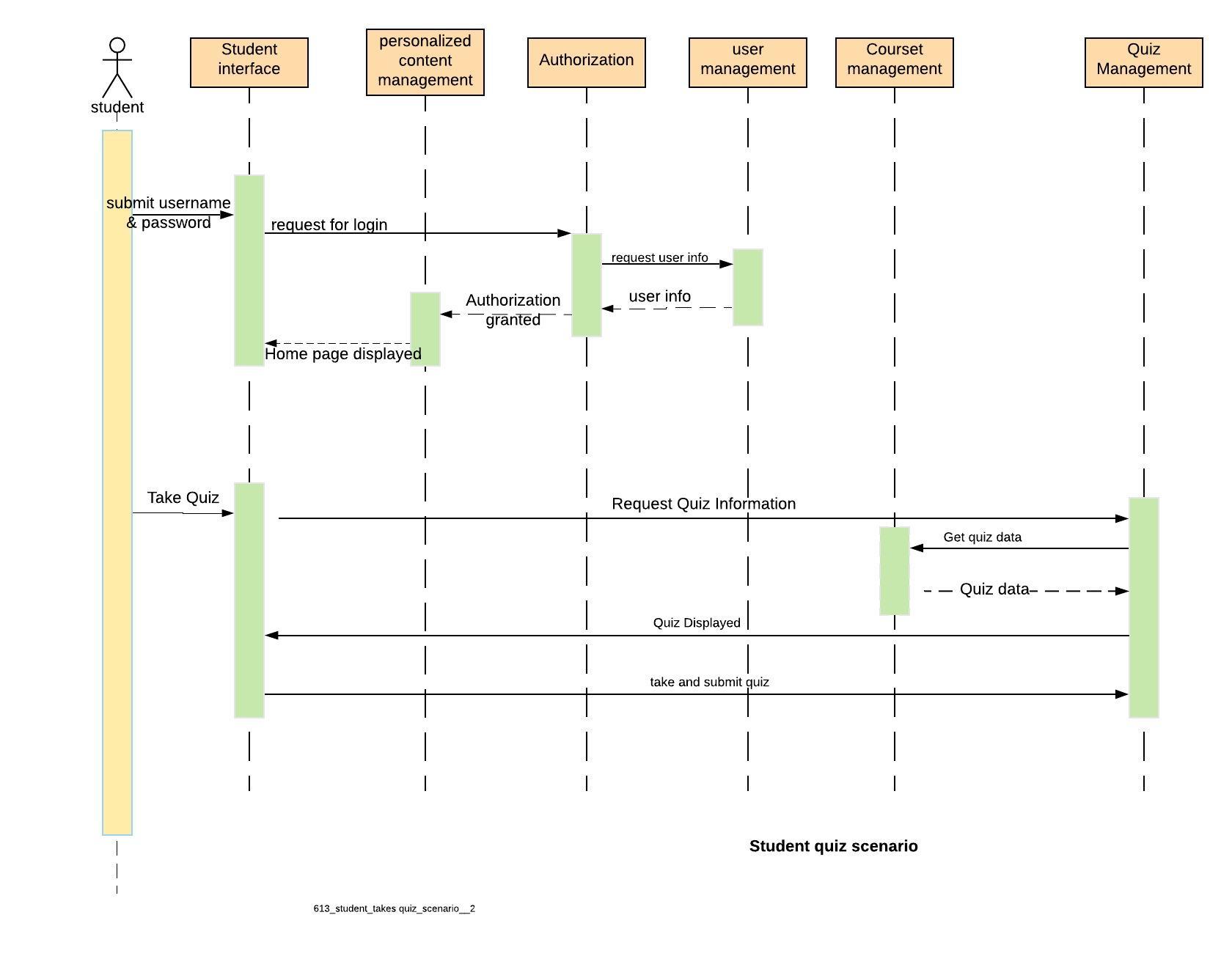
Operation Postconditions : Tutor user is set as the new tutor for the course

## Architecture Behavior View

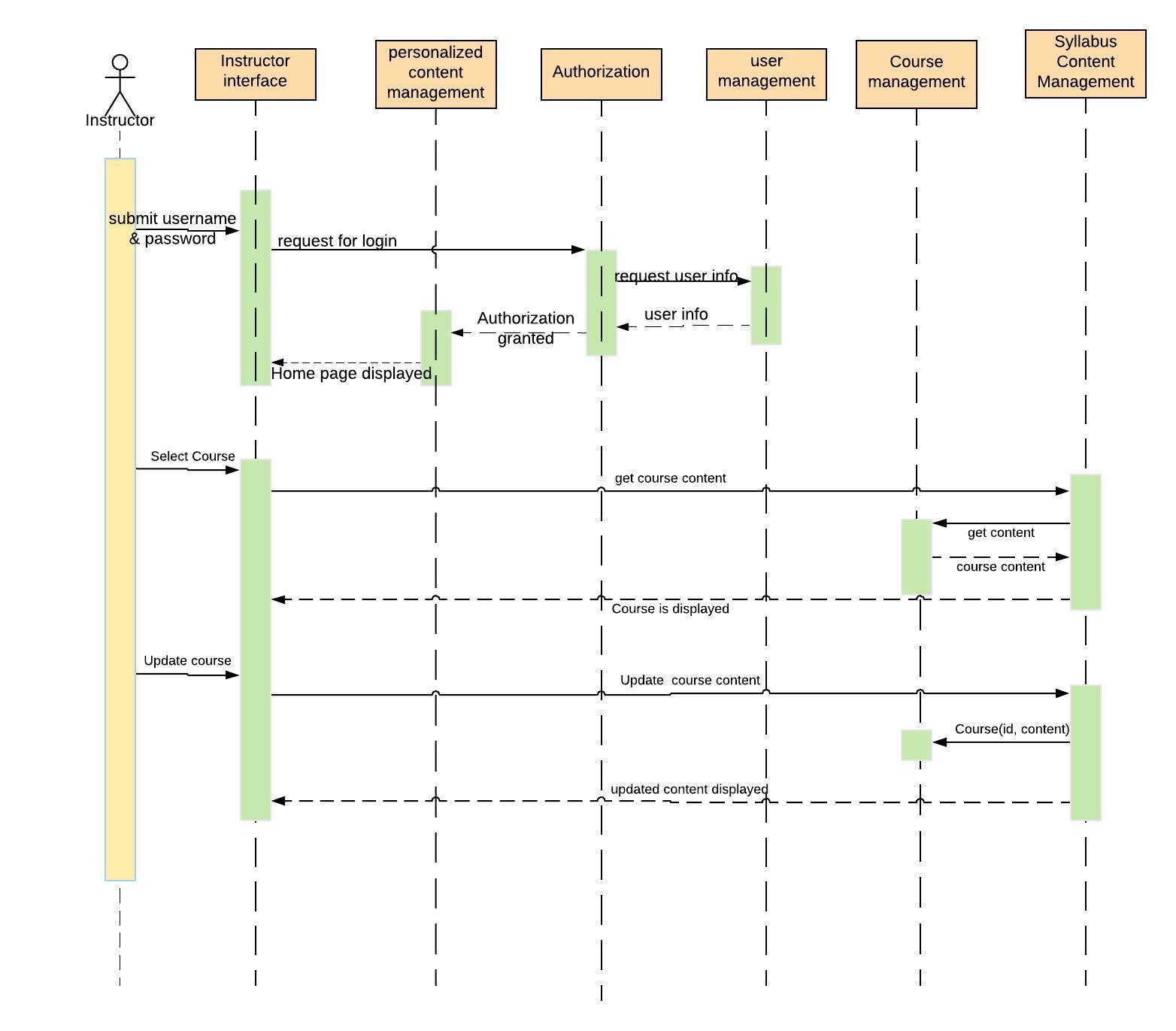
1. Course Enrollment Scenario

****

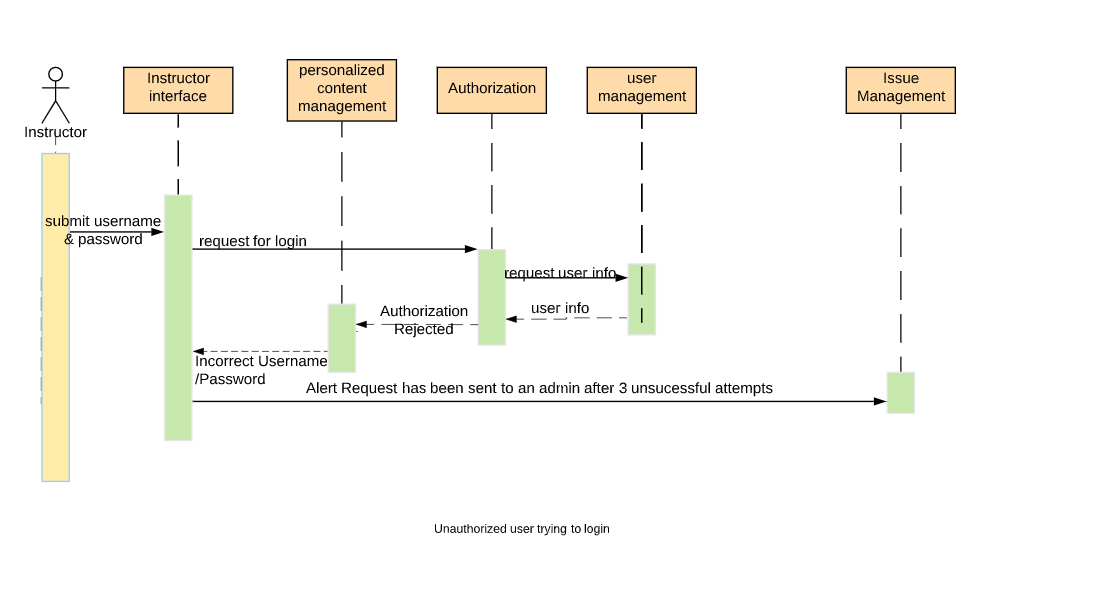
1. Student taking quiz Scenario.

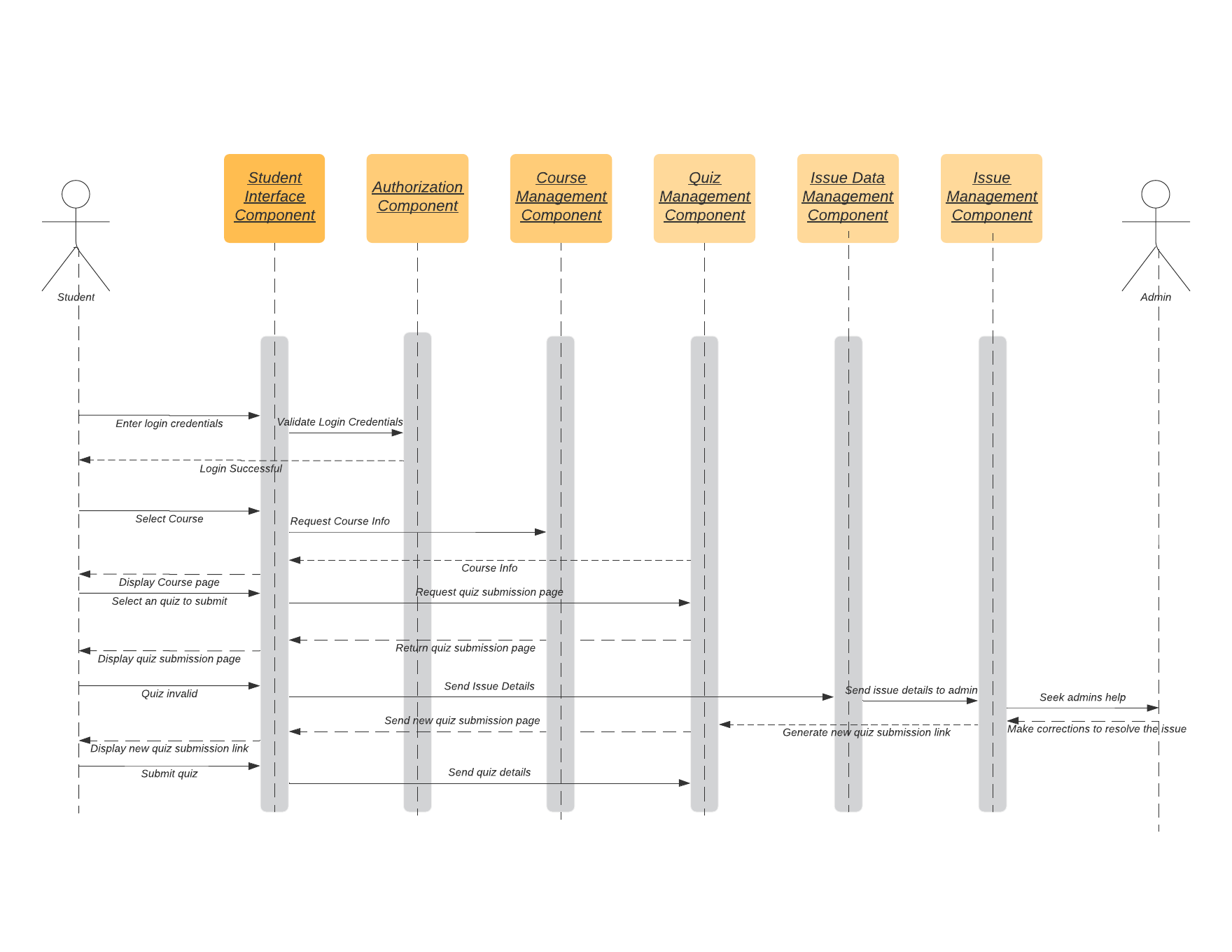
****

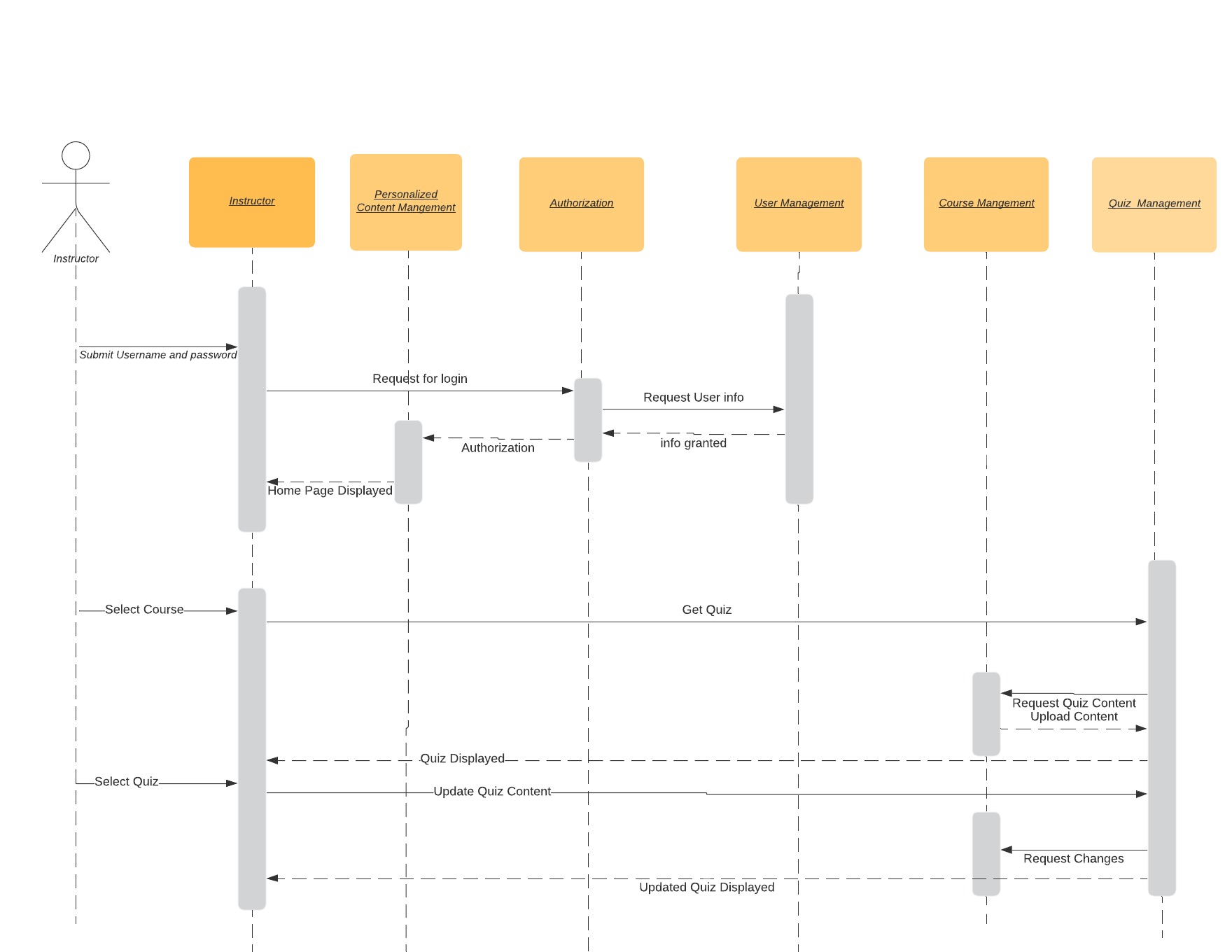
1. **Instructor updating the content of the course**

****

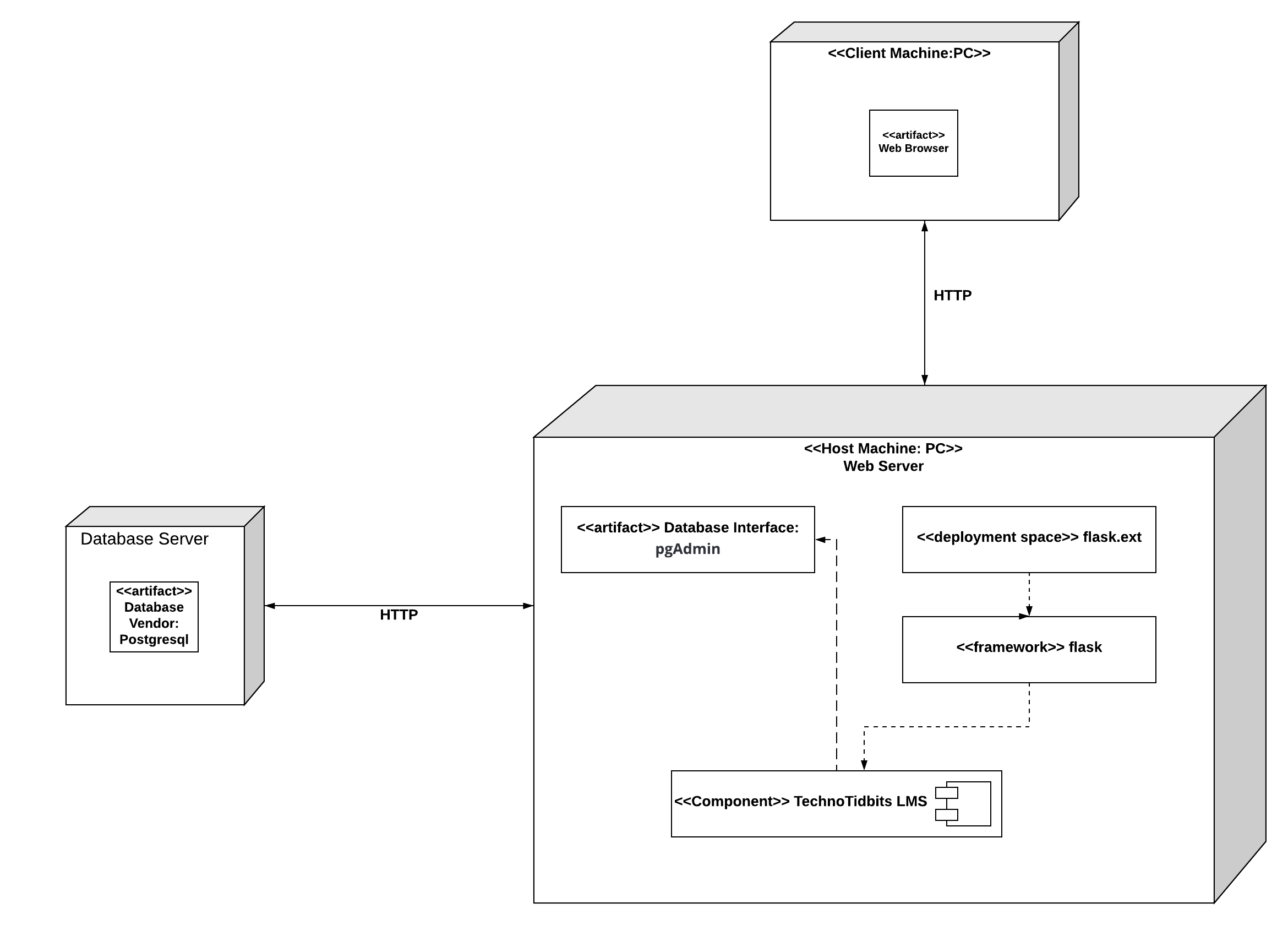
1. **Unauthorized user trying to login as instructor (Quality Scenario)**

****

1. Assist with issues: Student faces unexpected problems in submitting his quiz.
2. **Instructor Uploads/Updates Quiz content**

****

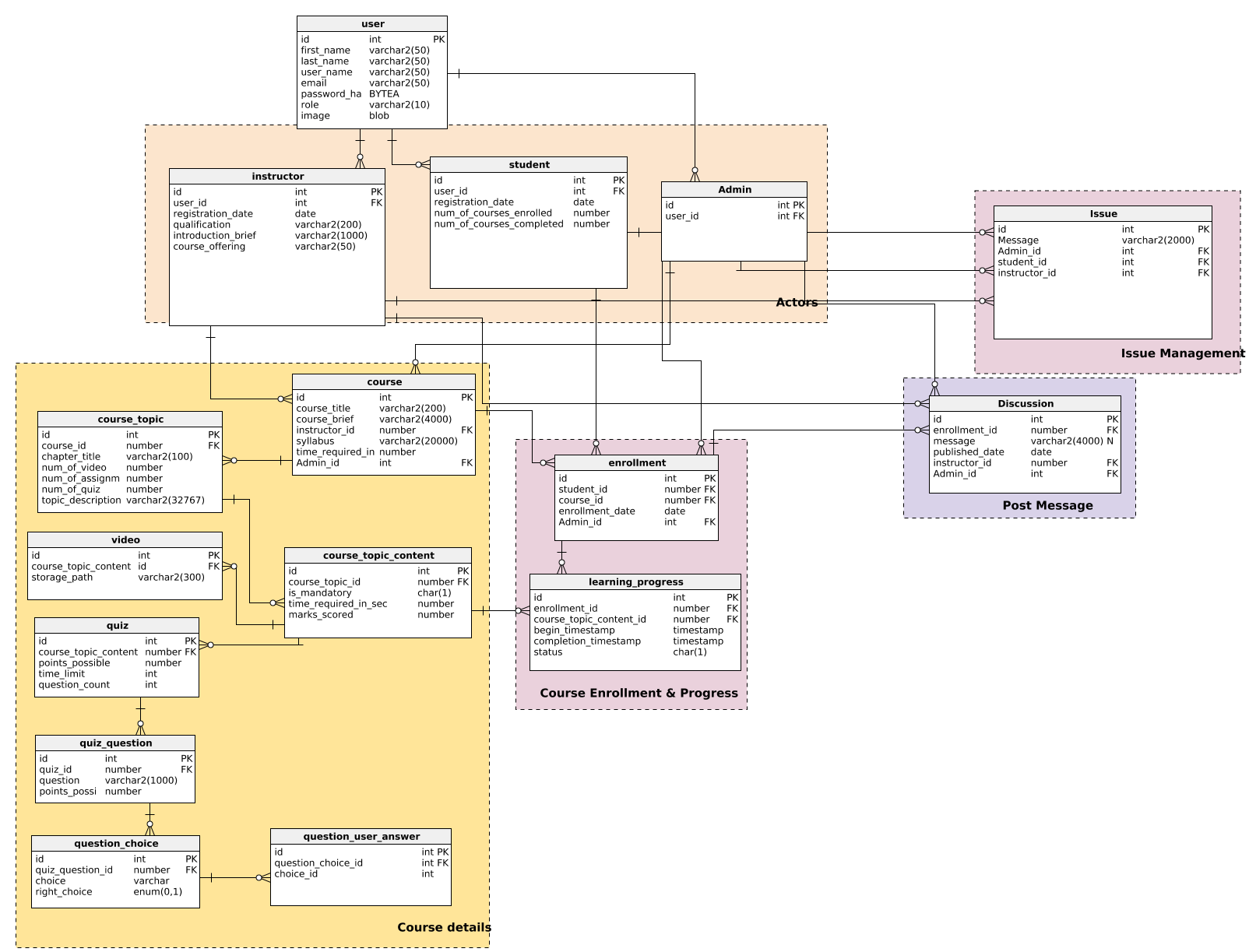
## Architecture Deployment View



**Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Component Name | Component Type | Description |
| 1 | Personalized Content Management | Logical  (TechnoTidbits LMS) | This module enables to view and update user specific courses and its progress. |
| 2 | Course Management | Logical  (TechnoTidbits LMS) | This model handles the update, delete and view request from syllabus, quiz, assignment and authentication management. |
| 3 | Issue Management | Logical  (TechnoTidbits LMS) | This module should show a user manual of the system. |
| 4 | Collaboration Management | Logical  (TechnoTidbits LMS) | This component should enable students and instructors to communicate through the system. |
| 5 | User Management | Logical  (TechnoTidbits LMS) | This module interacts with the user database and provides information for authentication and profile management. |
| 6 | Quiz Management | Logical  (TechnoTidbits LMS) | Enables the student to view and take a quiz for the course enrolled. Enables the instructor to update and view the quiz for the course tutored. |
| 7 | Assessment Management | Logical  (TechnoTidbits LMS) | Grade the student based on his/her performance in the quiz. |
| 8 | Profile Management | Logical  (TechnoTidbits LMS) | Manages the profile information of the user and enables deletion of user and viewing other profiles. |
| 9 | Authorization Management | Logical  (TechnoTidbits LMS) | Records a user's identification as a student, an instructor, or an administrator and allows only a user to do operations that he/she is authorized to do in the system. |
| 10 | Quiz Management | Logical  (TechnoTidbits LMS) | Enables the student to view, and take a quiz for the course he/she is enrolled. Enables the instructor to update and view the quiz for the course he/she tutors. |
| 11 | Syllabus Content Management | Logical  (TechnoTidbits LMS) | Enables the student to view course and enables tutor and admin to modify course. |
| 12 | Search Management | Logical  (TechnoTidbits LMS) | Enables the user to search for courses. |
| 13 | Collaboration Data Management | Logical  (TechnoTidbits LMS) | Updates and gets discussion information from the database. |
| 14 | Issue Data Management | Logical  (TechnoTidbits LMS) | Gets and updates issue data from the database. |
| 15 | PostgreSQL | Database Server | Performs tasks such as data analysis, storage, data manipulation, archiving, and other non-user specific tasks. |
| 16 | Flask | Deployment Space | Provides platform for an application to run and operate in a specific environment. It involves installation, configuration, testing and making changes to optimize the performance of the software. |
| 17 | flask.ext | Deployment Framework | Provides platform for code that establishes an ideal complete application that knows “what” to do, but it doesn't know “how” to do it. |
| 18 | Student Interface | UI | Enable the student to view and do tasks. |
| 19 | Instructor Interface | UI | Enable the instructor to view and do tasks. |
| 20 | Admin Interface | UI | Enable the admin to view and do tasks. |

## Information View

**

## Requirements Allocation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Features Allocation |  |  |  |  |  |
| **Features** | Personalized Content management | Collaboration Management | Course Management | User Management | Issue Management |
| Sign in/ Sign up |  |  |  | X |  |
| Modify Profile |  |  |  | X |  |
| See Courses details | X |  | X |  |  |
| Add, drop courses | X |  |  |  |  |
| View dashboard. | X |  |  |  |  |
| Take quiz, exams |  |  | X |  |  |
| View other profiles |  |  |  | X |  |
| Send or receive messages to/from instructor or students. |  | X |  |  |  |
| Seek help from the admin |  |  |  |  | X |
| Seek permission to teach courses | X |  | X |  |  |
| Modify the course content |  |  | X |  |  |
| See all students enrolled in the course. |  |  |  | X |  |
| Create, edit, delete and quiz. |  |  | X |  |  |
| Add, Modify, remove course content, quiz |  |  | X |  |  |
| Grant, Decline permission to teach courses. | X |  | X |  |  |
| Add, delete students. |  |  |  | X |  |
| Add,delete instructors (admin) |  |  |  | X |  |
| Revoke permission to Teach |  |  | X |  | X |

## Architecture Work Allocation

## 

|  |  |  |  |
| --- | --- | --- | --- |
| Work Allocation |  |  |  |
| **Components** | **Detailed Design** | **Implementation** | **Testing** |
| Personalized Content management | Dhaarani,pranav | Sai,Aman | Tigist, Yilli |
| Collaboration Management and Collaboration Data Management | Pranav,Aman | Dhaarani,Yilli | Tigist, Sai |
| Course Management, Search Management, Syllabus Content Management, Quiz Management and Assessment | Dhaarani,Yilli | Pranav, Yilli,Sai | Tigist,Aman |
| User Management, Authorization and Profile Management | Tigist, Dhaarani | Yilli,Aman | Sai,Pranav |
| Issue Management and Issue Data Management | Sai,Pranav | Tigist,Aman | Yilli,Dhaarani |
| User Interface | Dhaarani,Pranav | Sai,Aman | Yilli,Tigists |