Punjab University College of Information Technology

Project Supervisor:

Fareed Ul Hassan Baig

Group Members:

Syed Faateh Sultan Kazmi (BCSF18M022) ***Team Lead***
Muhammad Afaq Shuaib (BCSF18M004)
Muhammad Bilal (BCSF18M029)
Muhammad Azam (BCSF18M036)
Muhammad Omer Sharif Bhatti (BCSF18M055)

Campus Cube

Second Deliverable

Table Of Contents

1. *Introduction:*

Second deliverable represents use case modeling and software design. We will maneuver for a solution for the problem domain, discussed in first deliverable using object-oriented approach. Following artifacts are included in this deliverable document 'D2'.

1. Use case description
2. Use case diagram refined
3. Domain Model
4. Sequence Diagram
5. Collaboration Diagram
6. Operation Contracts
7. Design Class Diagram
8. Data Model

1.1. Usecase Description

While technically not part of UML, use case documents are closely related to UML use cases. A use case document is text that captures the detailed functionality of a use case. Description of all use cases are written down.

1.1.1. UC_Register

Brief Description:

To access the facilities provided by the Campus Cube. A user must have an account. So, in this use case the user will provide the required data (organization email and password) for the account. The system will perform validation checks where required if there is any invalid data the user will be asked to enter correct email provided by organization for registration.

After the submission of data, the system will enter the data into the database and a success message will be shown to the user. The user will automatically login into the system and user will redirect to the home screen.

Pre-Conditions:

The user must have internet connection.

Basic Flow

- The user will provide required data and request to register
- The system will validate the data and create an account for the user and display success message.

Alternate Flow

- The user may provide some invalid data in this case the user will be prompted to inset valid data and the normal flow of the user case will continue.

Post-Conditions:

An account for the user will be created and the user will be directed to home screen

1.1.2. UC_Login

Brief Description:

This use case describes how a user will log into the system. The actors starting this use case will be students and instructors.

Pre-Conditions:

The user must have an active internet connection and must have an account created earlier.

Basic Flow:

- The system will request user to enter the email and password.
- The actor will enter his email and password.
- The system will validate the actor's password and log's him/her into the system and the use case ends here.

Alternate Flow:

- Invalid Username/ Password
- Remains on the login screen

Post-Conditions:

If successful, the user will be log into the system and directed to the home screen. If not, the system state remains unchanged.

1.1.3. UC_Get_User_Posts

Breif Description:

The system provides the list of posts posted by the user on the news feed. The posts will appear on the news feed w.r.t some ranking algorithm.

Pre-Conditions:

The user must log into the system and have an active internet connection.

Basic Flow:

- As it's a business use case, so the user can view the posts on their news feed posted by the other users.

Alternate Flow:

None.

Post-Conditions:

- The system will provide all the posts, posted by the users followed by him/her and the use case ends.

1.1.4. UC_Post_Comments

Brief Description:
User will add comments to the user post.

Pre-Conditions:
User must log-in to the system.

Basic Flow

- User can add the comments to the post by pressing the comments button showing on the post.
- After typing the comment user will press the send button to add the comment on the post.

Alternate Flow:

- User can close comments section by repressing the comments button and move on to next post to view.

Post-Conditions:
Comments will be added to the post successfully.

1.1.5. UC_Post_Like_Share

Brief Description:
User will like, share the posts appears in their news feed, posted himself or by another user.

Pre-Conditions:
User must log-in to the system and for sharing the post, sharing privileges are allowed by the post owner.

Basic Flow:

- User can like the post by pressing the like button or by either double tap the picture.
- User can share the post by pressing the share button, after pressing the share button a pop-up modal will appear user simply add some caption and press the post button to share the post.

Alternate Flow:

- User simply view the post and doesn't perform any action.

Post-Conditions:

Post liked/ shared by the user.

1.1.6. UC_Post_Timeline_Show

Brief Description:

User (students/teacher) will create a post that will appear in its profile page along with news of other users, followed that user.

Pre-Conditions:

User must login into the system and have an active internet connection.

Basic Flow:

- Click on create post field/ button.
- Create Post modal will appear.
- Fill the create post details (caption, add image (optional))
- Press the post button and the use case ends here.

Alternate Flow:

- Some details left unfilled, prompt for filling in all fields.
- User end the use case by pressing close button appear in the modal.

Post-Conditions:

User's post will be uploaded successfully.

1.1.7. UC_User_Classrooms

Brief Description:

System shall provide a list of classrooms in which the logged-in user is enrolled in.

Pre-Conditions:

User should be logged in.

Basic Flow:

- User simply provided with a view of the classes in which it is enrolled if he/she is a student and the teacher will provide with a view of only those classes he is teaching.

Alternate Flow:

None

Post Conditions:

Specific view of classes will be provided to the user.

1.1.8. UC_Access_Classroom_Content

Brief Description:

In this use case, user can access the classroom post and resources provided by the system.

Pre-Conditions:

User should be logged in.
User must be enrolled in that class.

Basic Flow:

- User select the class from class view discussed in the previous use case, and after clicking the class tile, user will redirect to individual class page.
- The individual class page contains class post and resources posted by the instructor and students.

Alternate Flow:

None

Post-Conditions:

Individual Class View page is loaded having all the contents posted by the users enrolled in that course.

1.1.9. UC_User_Discussions

Brief Description:

This use case describes how a user Shall start/initiate and follow-up a discussion against some post or resource in classroom.

Pre-conditions:

User must be logged in and enrolled in that class.

Basic Flow:

- User can initiate and follow-up a discussion by clicking reply icon against the post or the comment respectively.

- User will type the message.
- Press the send button.
- The users of the class are notified that a message against a post/ comment is posted.

Alternate Flow:

- User not enrolled in that class and prompt to the main classrooms page.

Post-Conditions:

Message against a post/comment will be added successfully.

1.1.10. UC_Upload_Class_Resource

Brief Description:

This use case allows only instructor to upload the resource in the individual classroom room page.

Pre-Conditions:

User should be logged in and enrolled in that class as an instructor.

Basic Flow:

- Click on the resource button.
- Create Resource will be prompted to the user (instructor).
- User will add the resource content.
- Press the submit button.

Alternate Flow:

- User enrolled in the class as a student doesn't have the privileges to upload the resource.

Post-Conditions:

Resource will be uploaded successfully.

1.1.11. UC_User_Profiles

Brief Description:

It's a business use case, user shall access the individual user profile page provided by the system, populated from the data stored in the database.

Pre-Conditions:

User account must exist and user should be logged into the system.

Basic Flow:

- User can access his own profile page by clicking the avatar in the header section.
- To access the other user profile page the user can either search the profile from search user use case or by clicking the view profile of the users shown in the recommendation section in news feed.

Alternate Flow:

None

Post Condition:

User Profile page will be shown to the user where external links, user info and its post will be displayed.

1.1.12. UC_Add_Questions

Brief Description:

This use case describes how a user (teacher) can add questions to the question pool for a particular course.

Pre-Conditions:

User should be logged and have the instructor privileges assigned by the admin.

Basic Flow:

- There are two options either you can add a question to the existing question pool, or either you can add a new question pool.
- To add a new question pool, click on the add question pool button, and a file will be uploaded which contains a bunch of questions.
- To add a question to the existing pool, click on that specific question pool and select the add question option.
- User will be prompted to add the question.
- User (teacher) fill required fields and then click add question button.

Alternate Flow:

- User enrolled in the course as a student and doesn't have the privileges to add the questions pool.

Post-Conditions:

Question will be added to the question pool successfully.

1.1.13. UC_Post_Universal_Resource

Brief Description:
This use case describes how a user can upload a resource a resource in the universal resource center.

Pre-Conditions:
User should be logged into the system.

Basic Flow:
- User clicks on the add resource button.
- Create Resource will be prompted to the user.
- User will add the resource content and press the submit button.

Alternate Flow:
- Some requirements left unfilled, prompt for filling in all fields.

Post Conditions:
Resource will be posted successfully.

1.1.14. UC_Generate_Quiz

Brief Description:
This use case describes how a instructor will generate a quiz for the students enrolled in the course.

Pre-Conditions:
User should be logged into the system and must be an instructor.

Basic Flow:
For generating the quiz there are 4 steps
- In setup phase, the question pool for the quiz will be selected.
- In design phase, the question weightage and order will be selected.
- In assign phase, the pushed notification will be sent to all users enrolled in that course.
- In share phase, a link to the quiz have been generated.

Alternate Flow:
- User enrolled in the course as a student and didn't have any generate quiz privileges.

Post Conditions:

Resource will be generated successfully.

1.1.15. UC_Universal_Chat

Brief Description:
It is a functional use case. This will allow the users to chat with all the other users of the system globally.

Pre-Conditions:
User account must exist and user should be logged into the system.

Basic Flow:
- Click on the chat button.
- The user will be presented with a screen showcasing all the chats he had previously, and an option to switch from individual to group chats.
- Click on the global chat button.
- The user will be presented with a screen showcasing all the messages that the current, and all the other users have sent.
- User will also have an option to send new messages.

Alternate Flow:
- No alternate flow, as user must be logged in before messaging.

Post-Conditions:
Message will be sent/received successfully.

1.1.16. UC_P2P_Chat

Brief Description:
It is a functional use case. This will allow the users to have a personal conversation with their fellow colleagues and their instructors. This includes individual chats as well as group chats

Pre-Conditions:
User account must exist and user should be logged into the system.

Basic Flow:
- Click on the chat button.
- The user will be presented with a screen showcasing all the chats he had previously, and an option to switch from individual to group chats.
- User can click on the person he wants to have a chat with.

- User will be presented with a screen showcasing all the conversation they had with the person they clicked on previously, new messages that the recipient has sent, and also an option to send new messages.
- User can also search all the conversations he has had with people, or use it to initiate a new conversation.

Alternate Flow:
- No alternate flow, as user must be logged in before messaging.

Post-Conditions:
Message will be sent/received successfully.

1.1.17. UC_Report_Message

Brief Description:
It is a functional use case. This will allow the user to report a message for sensitive content.

Pre-Conditions:
- User account must exist and user should be logged into the system
- User must have received a message that contains sensitive content.

Basic Flow:
- Click on the chat button.
- The user will be presented with a screen showcasing all the chats he had previously, and an option to switch from individual to group chats.
- User can click on the person that has sent a suspicious message.
- User will be presented with a screen showcasing all the conversation they had with the person they clicked on previously, new messages that the recipient has sent, and also an option to send new messages.
- User will be able to report a message that contains sensitive content from this screen.
- Click on the message that contains sensitive content.
- Select a relative reason for reporting.
- Press report.
- Consistent reports against messages associated with the same user, will result in a temporary ban until reviewed by the admin.
- On reviewing the complaints, the admin will be able to impose a ban on the user in case of valid reports, otherwise remove the temporary ban imposed earlier.

Alternate Flow:

- No alternate flow, as user must be logged in before messaging and eventually reporting a message.

Post-Conditions:
Message will be reported successfully.

1.1.18. UC_Report_Post

Brief Description:
It is a functional use case. This will allow the user to report a post for sensitive content.

Pre-Conditions:
- User account must exist and user should be logged into the system
- User must have seen a post that contains sensitive content.

Basic Flow:
- Click on the news feed button.
- The user will be presented with a screen showcasing all the posts other users have made.
- User can click on the post that they think is suspicious.
- User will be able to report that post containing sensitive content.
- Select a relative reason for reporting.
- Press report.
- Consistent reports against posts associated with the same user, will result in a temporary ban until reviewed by the admin.
- On reviewing the complaints, the admin will be able to impose a ban on the user in case of valid reports, otherwise remove the temporary ban imposed earlier.

Alternate Flow:
- No alternate flow, as user must be logged in before viewing news feed and eventually reporting a post.

Post-Conditions:
Post will be reported successfully.

1.1.19. UC_Admin_Login

Brief Description:
It is a functional use case. This will allow the user to interact with the login screen of the admin and eventually access the admin functions.

Pre-Conditions:

- Admin account must exist on the system

Basic Flow:
- Go to the specified URL for Admin login.
- The admin user will be presented with a screen asking for email and password.
- On entering the credentials and then pressing the login button, the credentials will be verified by the system.
- On successful verification, the user will be presented with the admin menu.

Alternate Flow:
- Invalid Username/ Password
- Remains on the login screen

Post-Conditions:
The admin will be logged in successfully and be able to access all admin options.

1.1.20. UC_Add_Courses

Brief Description:
It is a business use case. The admin will have the option to create a new course, which will be added to the course list.

Pre-Conditions:
- Admin account must exist on the system.
- Admin must be logged in.

Basic Flow:
- The admin will click on the course management option.
- Here the admin will be able to create a new course, and delete a course.
- On pressing the create course option, the admin will be prompted to enter the course details:
  - Course Code
  - Course Name
- A new course will then be created.

Alternate Flow:
- The specified course already exists; hence admin will not be able to create a new course.

Post-Conditions:
The course will be created successfully.

1.1.21. UC_Generate_Class

Brief Description:
It is a business use case. The admin will have the option to create a new class, and assign to it a course, an instructor, instructor assistants, and students.

Pre-Conditions:
- Admin account must exist on the system.
- Admin must be logged in.

Basic Flow:
- The admin will click on the class management option.
- Here the admin will be able to create a new class, add users to existing class, and delete a class.
- On pressing the create class option, the admin will be prompted to enter the class details:
    - Term (Session)
    - Course code of the course to be assigned
    - Instructor
    - Instructor assistants
    - Timeslot
- The admin will then be prompted to select students, from the session previously selected, that they want to add the class.
- A new class will be created.

Alternate Flow:
- The specified class already exists; hence admin will not be able to create a new class.
- A class against the specified timeslot already exists.

Post-Conditions:
The class will be created successfully.

1.1.22. UC_Review_Reports

Brief Description:
It is a functional use case. The admin will have the option to review all the reports made by users.

Pre-Conditions:

- Admin account must exist on the system.
- Admin must be logged in.
- Unreviewed complaints must exist.

Basic Flow:
- The admin will click on the complaints option.
- Here the admin will be able to view all the complaints that users have made.
- The admin will then be able to impose a ban on the user tied with the complaint, in case of a valid report, or remove the temporary ban previously imposed on the user in case of invalid report.
- The admin will have successfully reviewed a report/ complaint

Alternate Flow:
- No unreviewed complaints exist.

Post-Conditions:
The admin will have successfully reviewed a report/ complaint.

1.1.23. UC_Campus_Announcements

Brief Description:
It is a functional use case. The admin will have the option to create announcements.

Pre-Conditions:
- Admin account must exist on the system.
- Admin must be logged in.

Basic Flow:
- The admin will click on the announce option.
- Here the admin will be able to add an announcement message, and the users that will receive the announcement message.

Alternate Flow:
- No alternate flow.

Post-Conditions:
The admin will have successfully created an announcement.

1.1.24. UC_Logout

Brief Description:
This use case describes how a user will logout from the system.

Pre-Conditions:
User must have an account and logged into the system.

Basic Flow:
- Click on the hamburger shown on the side of the header.
- Select the logout option.
- Session will be abandoned.

Post Conditions:
- The user will logout from the system.

1.1.25. UC_Edit_Profile

Brief Description:
User can edit their profile

Pre-conditions:
User must be logged in.

Basic Flow:
- Click on the profile button
- Click on the edit profile button
- Here, the user will be provided with options to change their:
    - Name
    - Profile picture
    - Cover picture
    - Bio
    - External links
    - password
- After updating desired fields, user needs to click on the update button

Alternate Flow:
User clicks on the cancel button after updating all the fields, which results into changes being discarded.

Post-Conditions:

The profile will be updated successfully.

1.1.26. UC_Create_Post

Brief Description:
User can create a post from their profile

Pre-conditions:
User must be logged in.

Basic Flow:

- Click on the create post button
- Add a caption
- Add an option picture
- Click on create post button

Alternate Flow:
No alternate flow as user only needs to be logged in.

Post-Conditions:
The post will be created successfully.

1.1.27. UC_Delete_Post

Brief Description:
User can delete their own post from their profile or newsfeed when they desire.

Pre-conditions:
User must be logged in.

Basic Flow:
There are two ways to delete a post.
1st

- User can delete a post from its profile where all of his posts are shown. User click the post to see it as single post view.
- User will click the delete button.
- Confirmation notification will be shown. User will confirm deletion.

2nd

- User can view the post from the news feed.
- User will click the delete button.
- Confirmation notification will be shown.
- User will confirm deletion.

Alternate Flow:

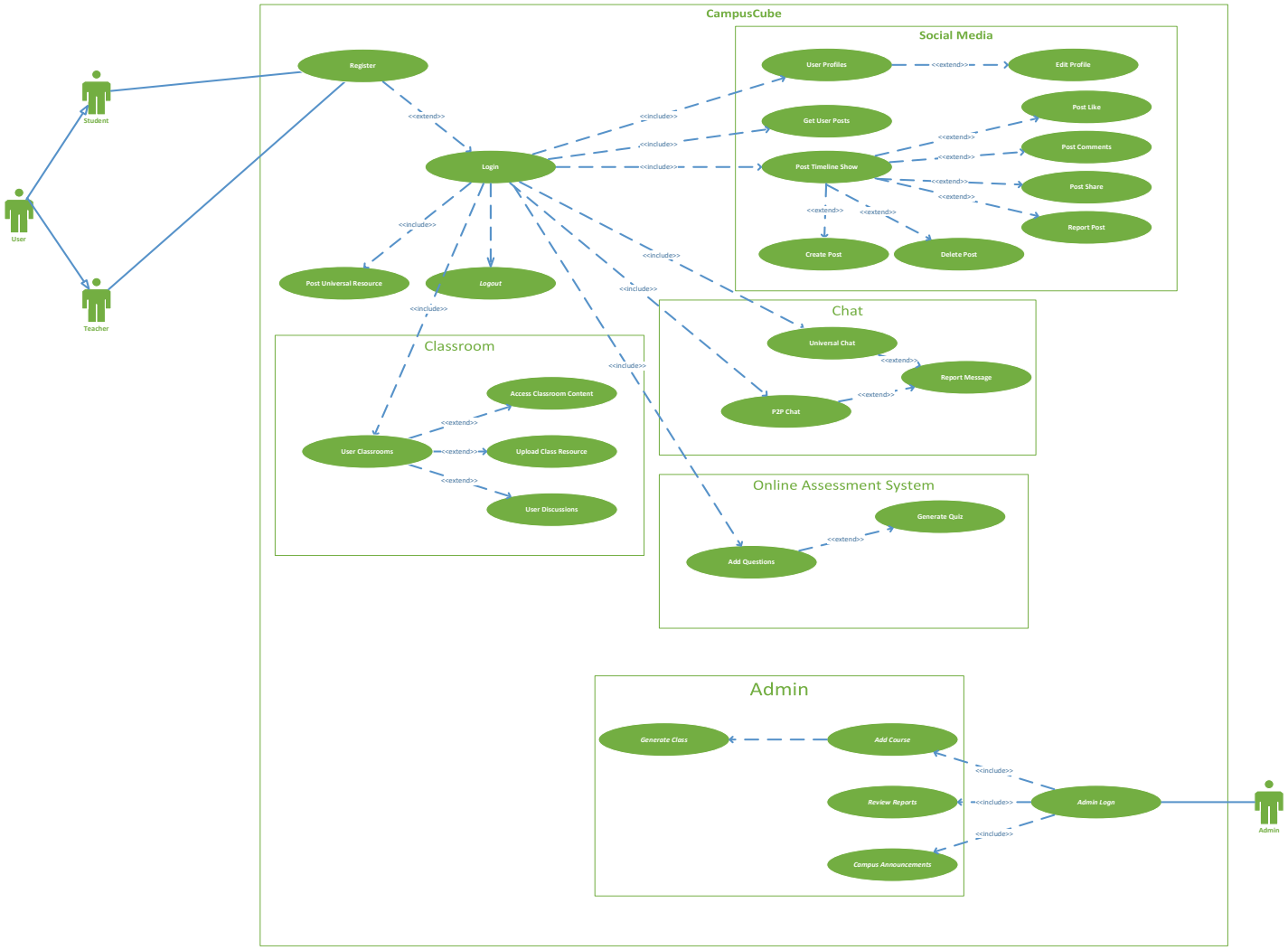
Upon confirmation user doesn't delete the post.

Post-Conditions:

The post will be deleted successfully.

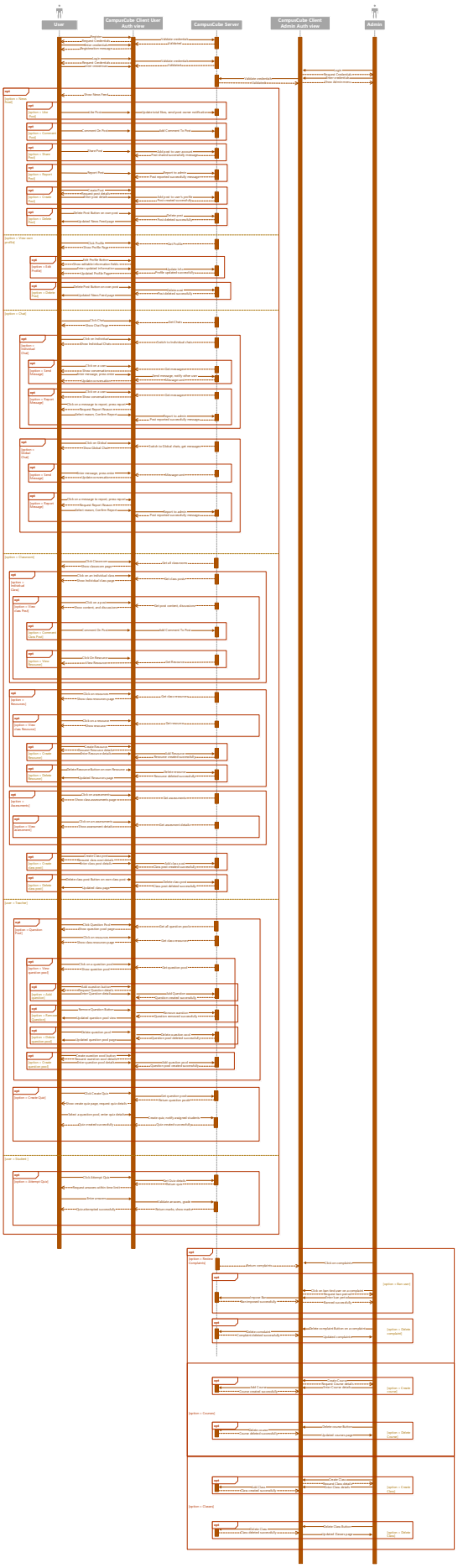
1.2 Usecase Diagram (refined and updated)

The refined and updated use cases have been listed in the previous (1.1) section. The updated diagram, extended from the previous deliverable, including the extend and include is shown as follows:



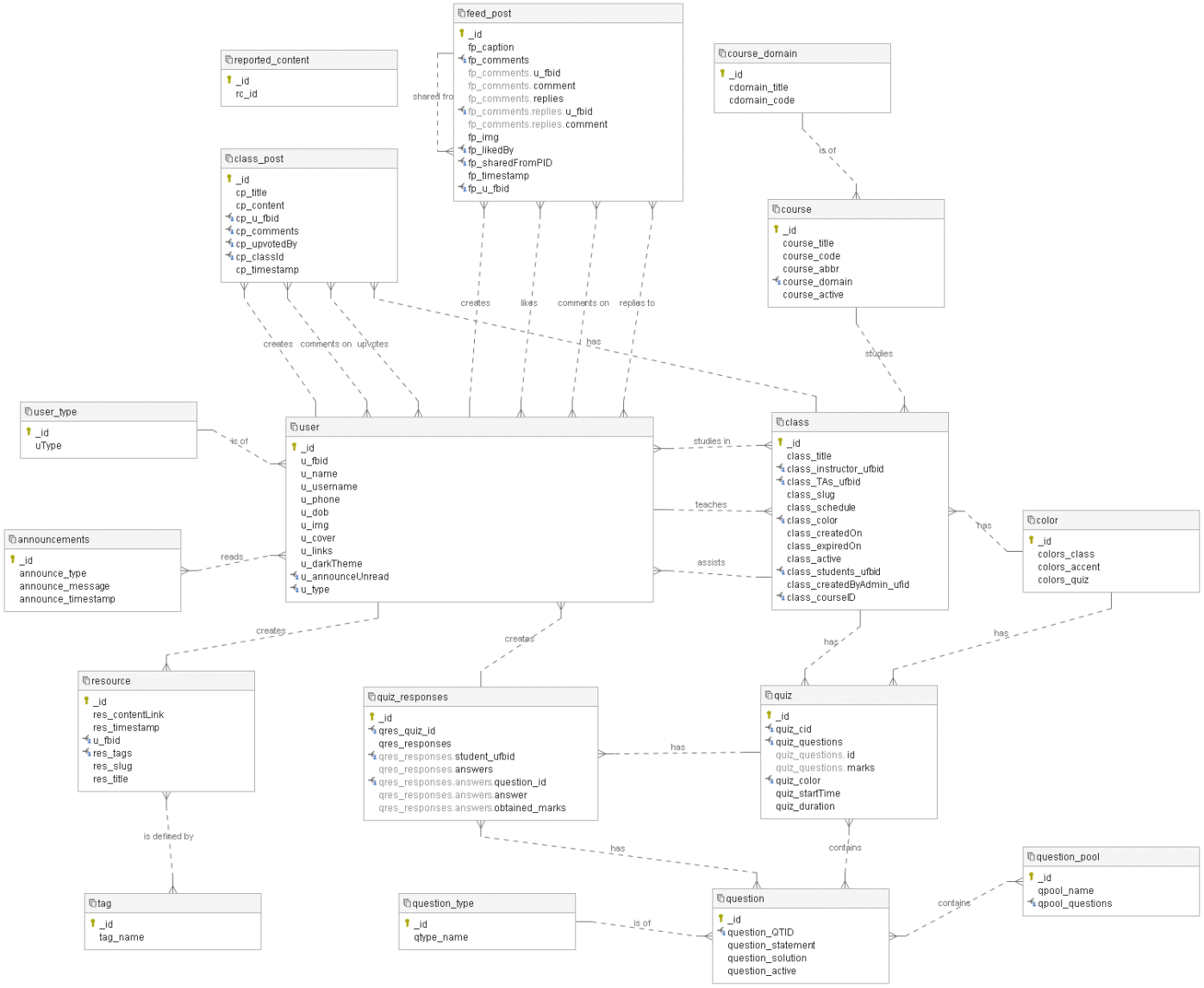
1.3 Sequence Diagram

Sequence diagram is as follows:


User
CampusCube Client User Auth view
CampusCube Server
CampusCube Client Admin Auth view
Admin

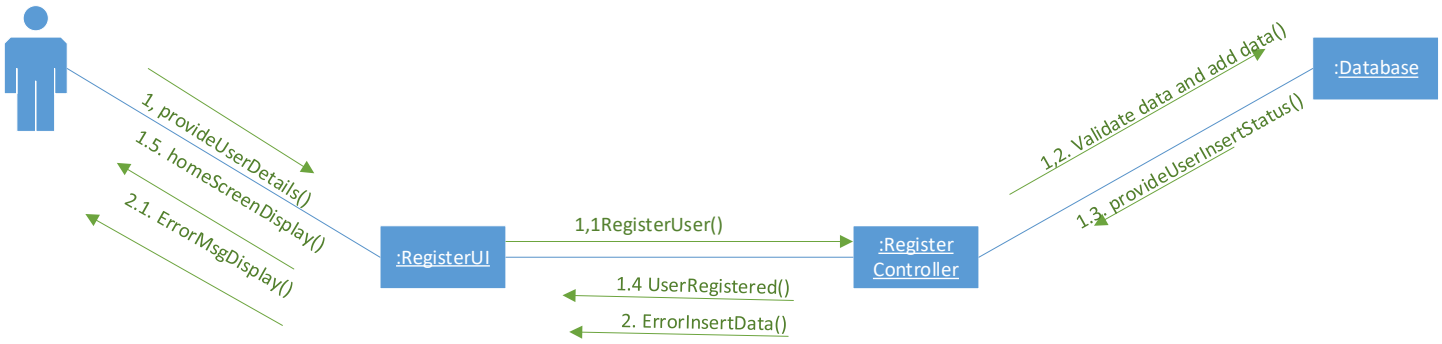
1.4 *Domain Model Diagram*

Domain Model Diagram is as follows:

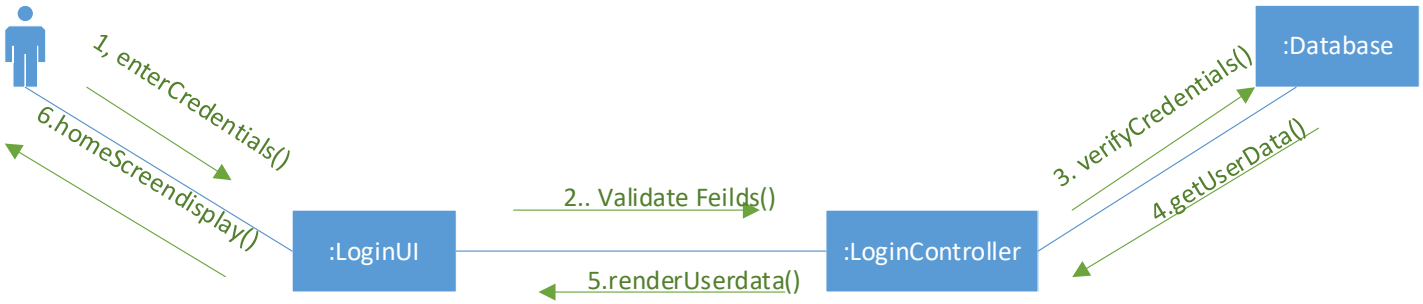


1.5 Collaboration Diagram:

1.5.1 UC_Register

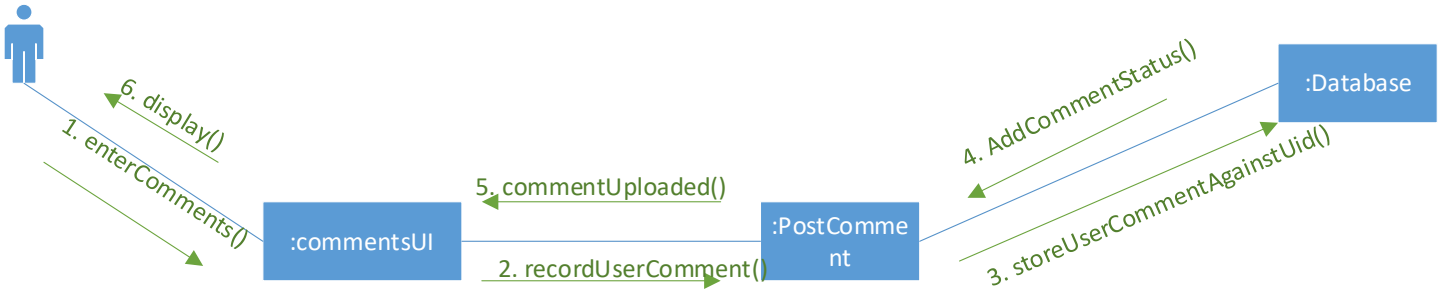


1.5.2 UC_Login

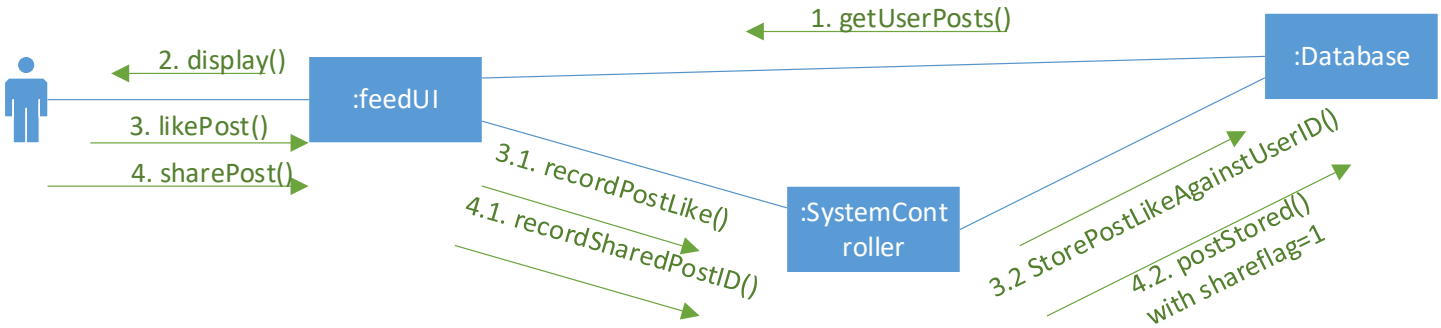


1.5.3 UC_Get_User_Posts

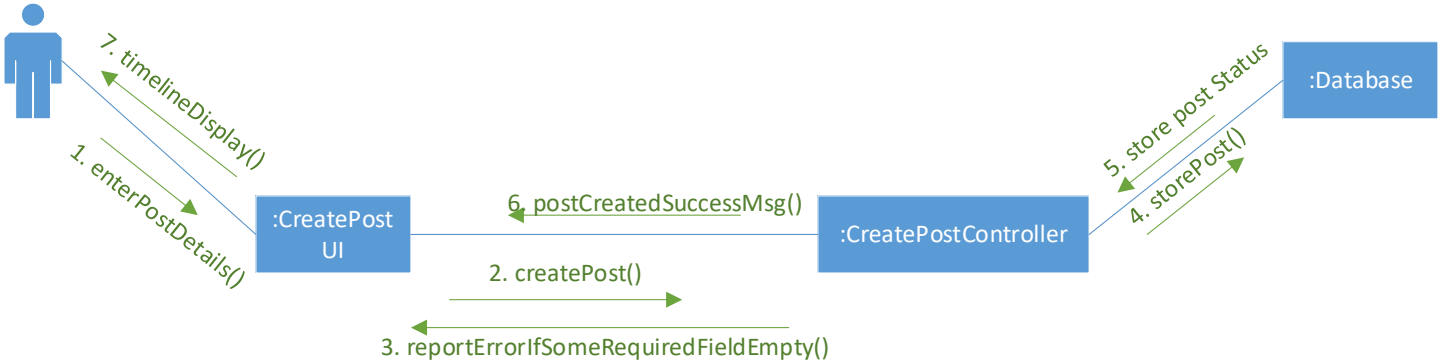
1.5.4 UC_Post_Comments



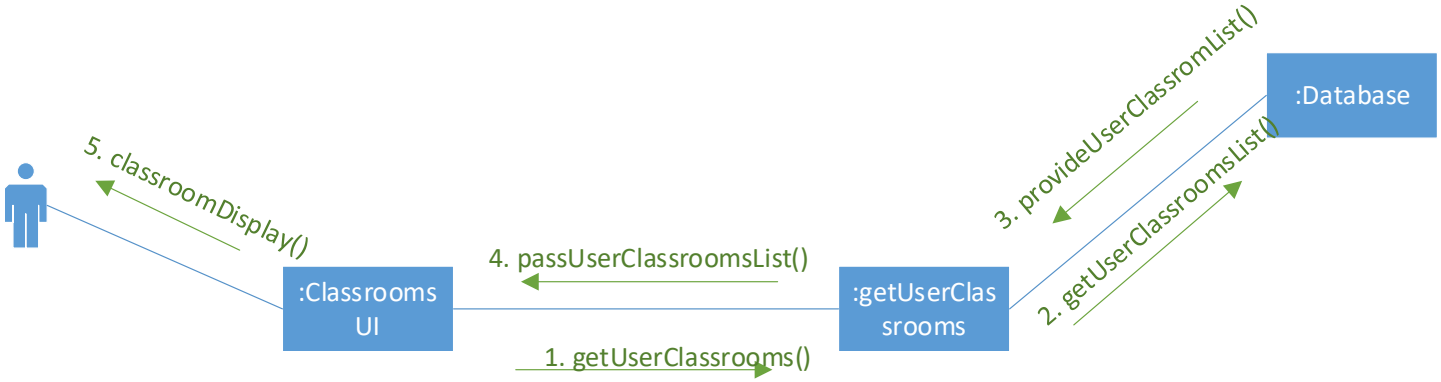
1.5.5 UC_Post_Like_Share



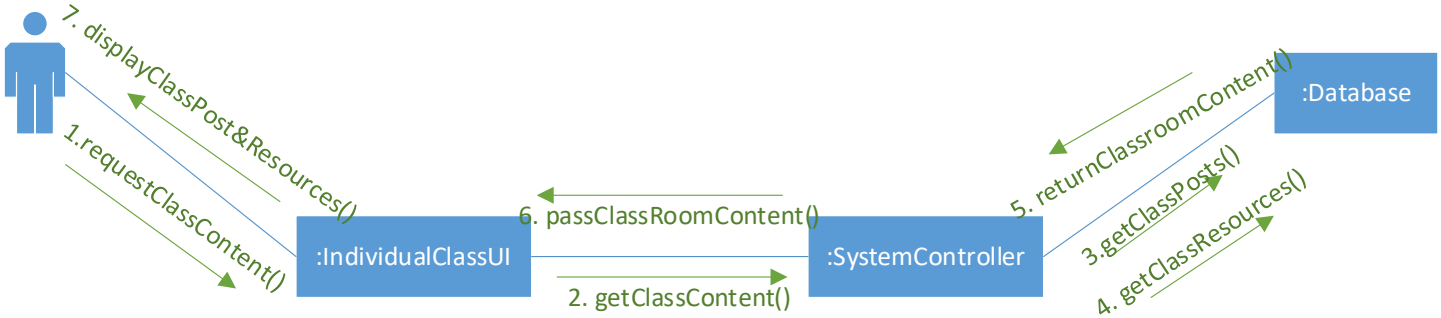
1.5.6 UC_Post_Timeline_Show



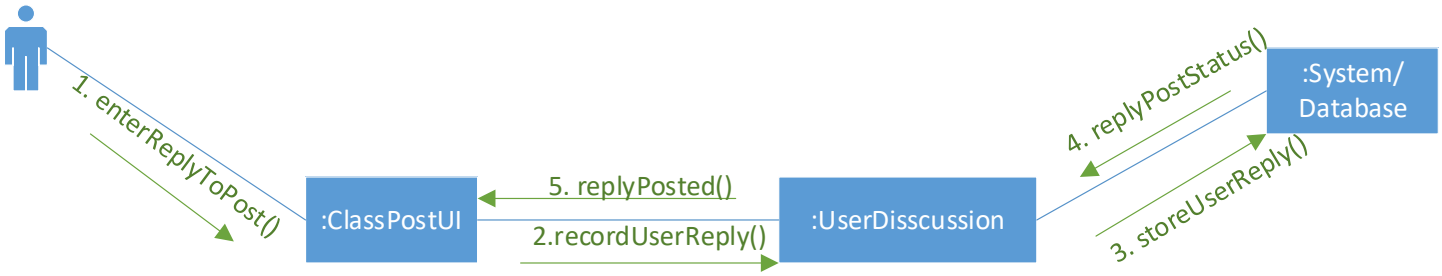
1.5.7 UC_User_Classrooms



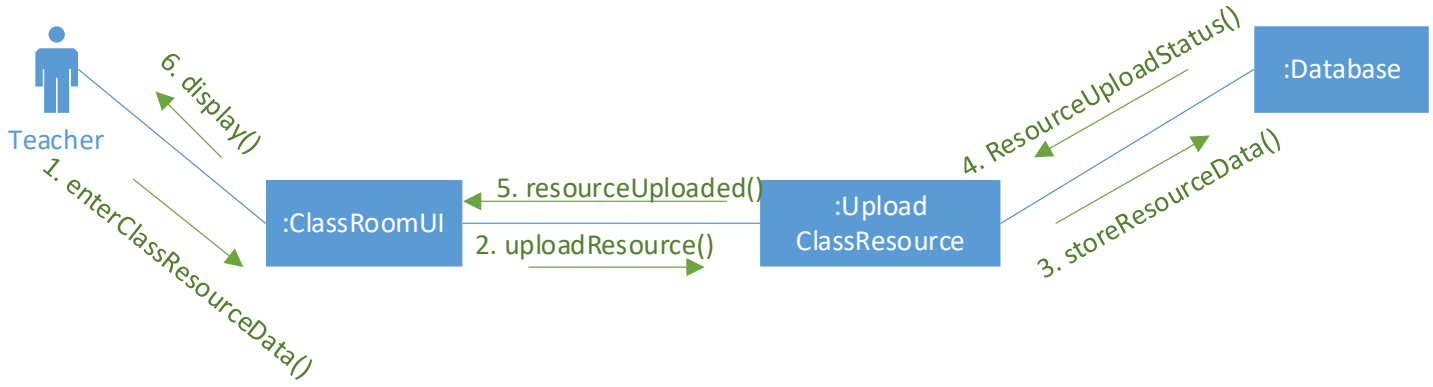
1.5.8 UC_Access_Classroom_Content



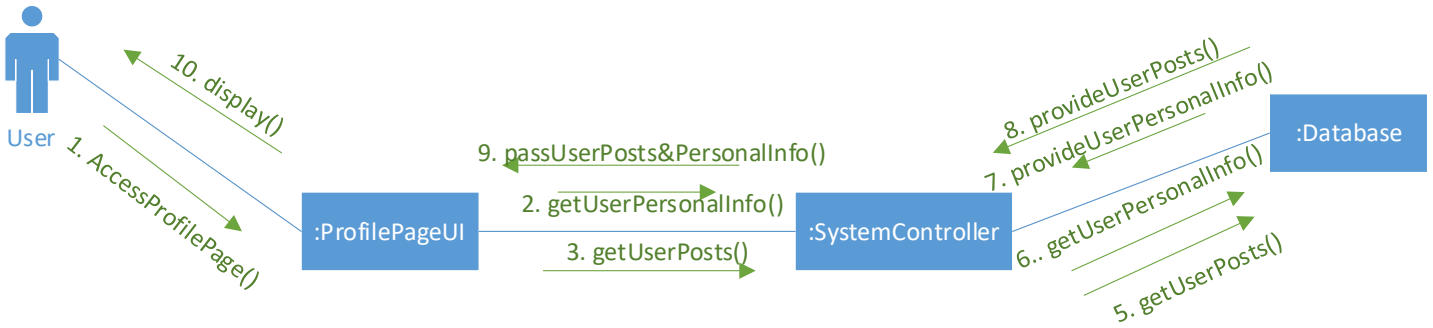
1.5.9 UC_User_Discussions



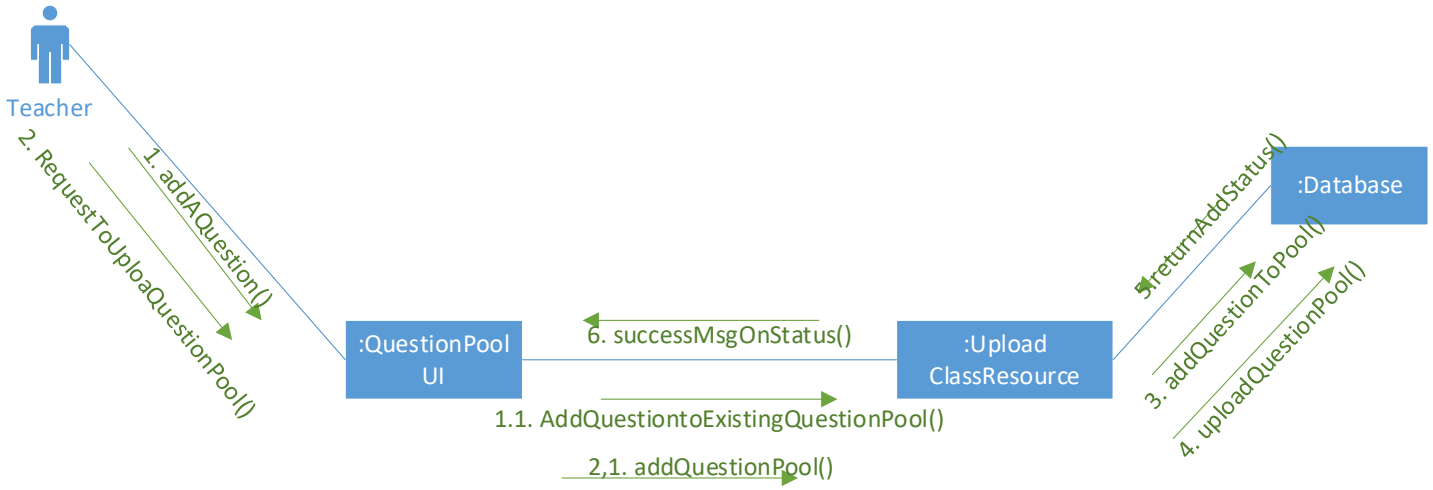
1.5.10 UC_Upload_Class_Resource



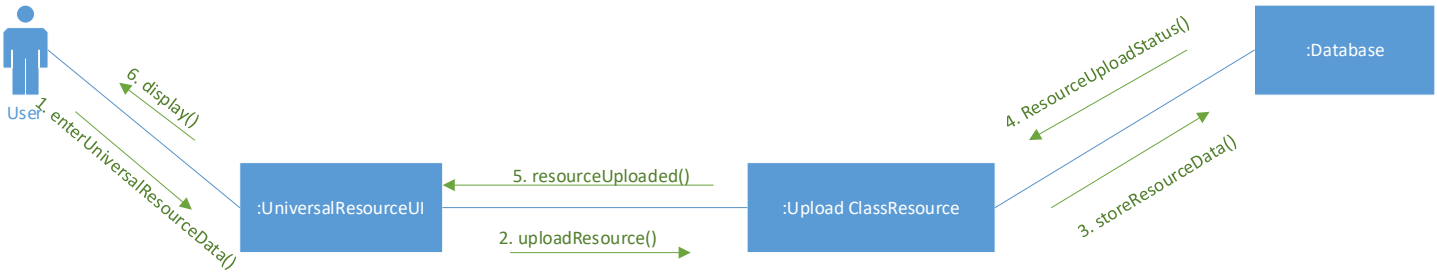
1.5.11 UC_User_Profiles



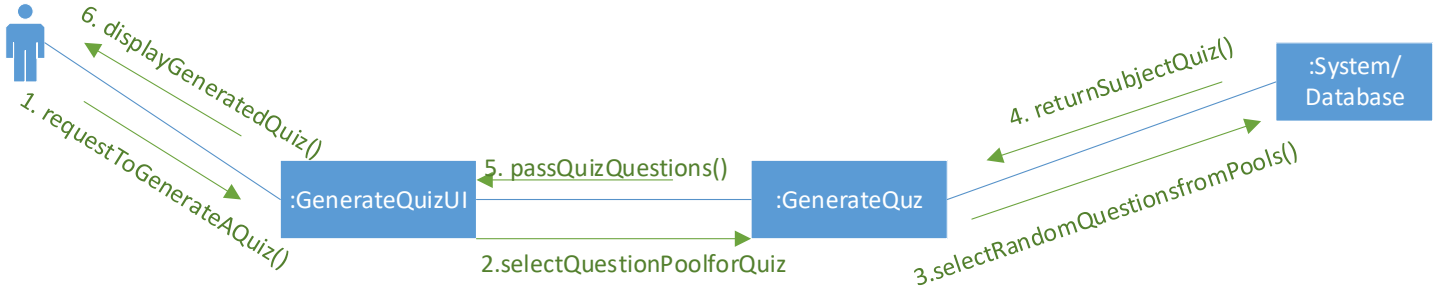
1.5.12 UC_Add_Questions



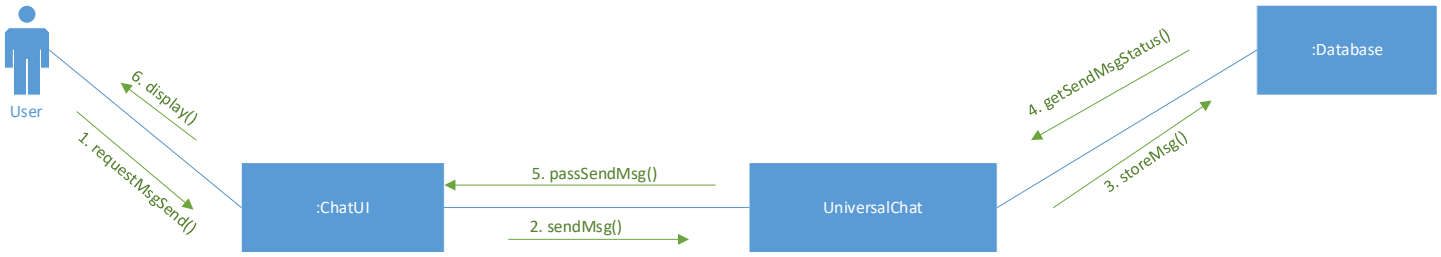
1.5.13 UC_Post_Universal_Resource



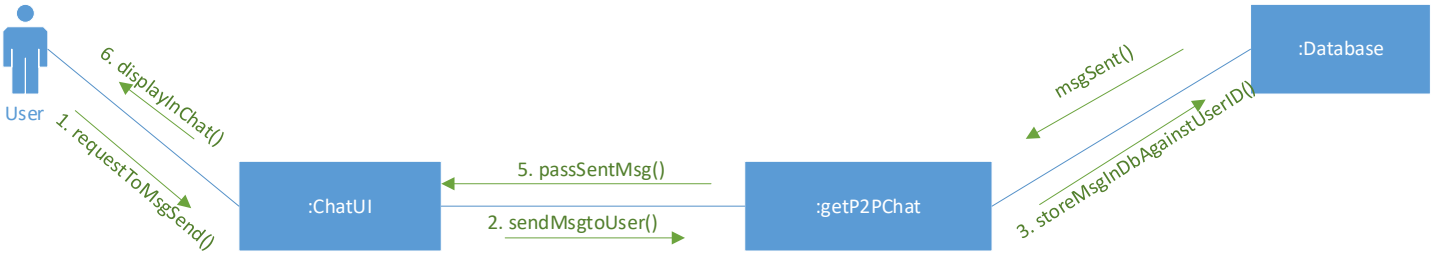
1.5.14 UC_Generate_Quiz



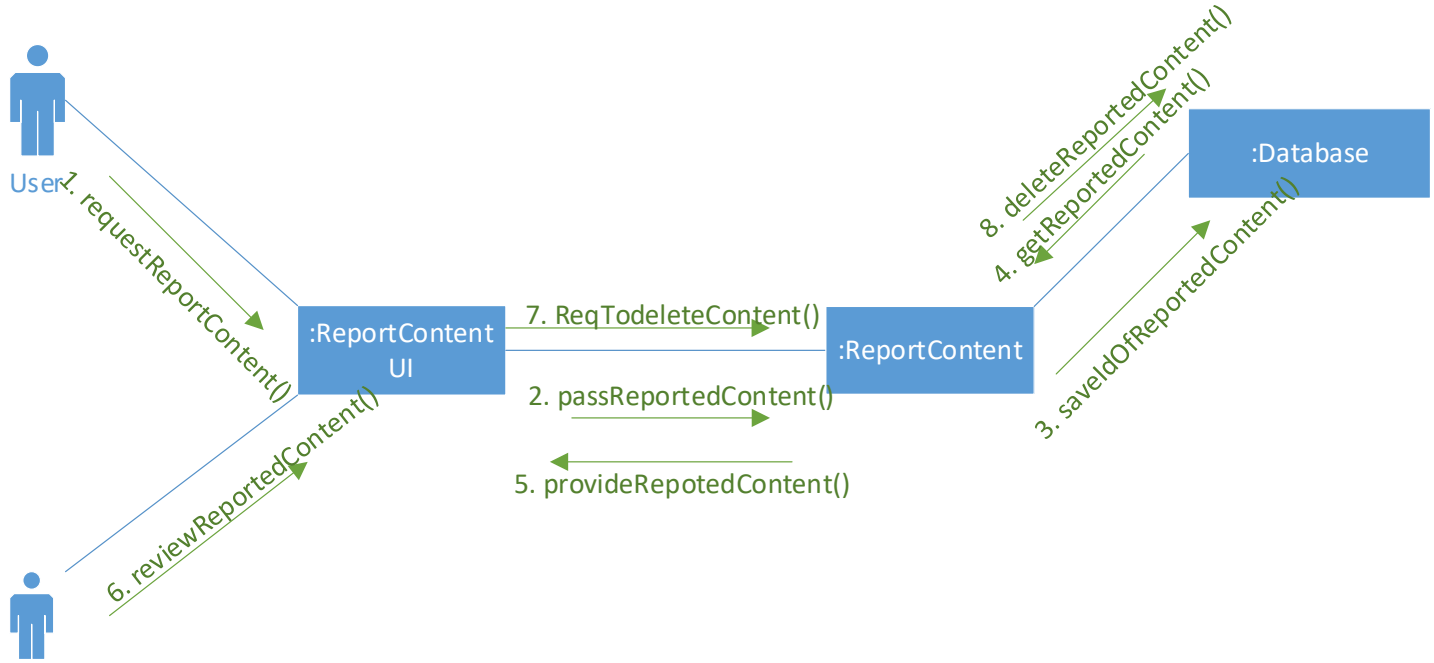
1.5.15 UC_Universal_Chat



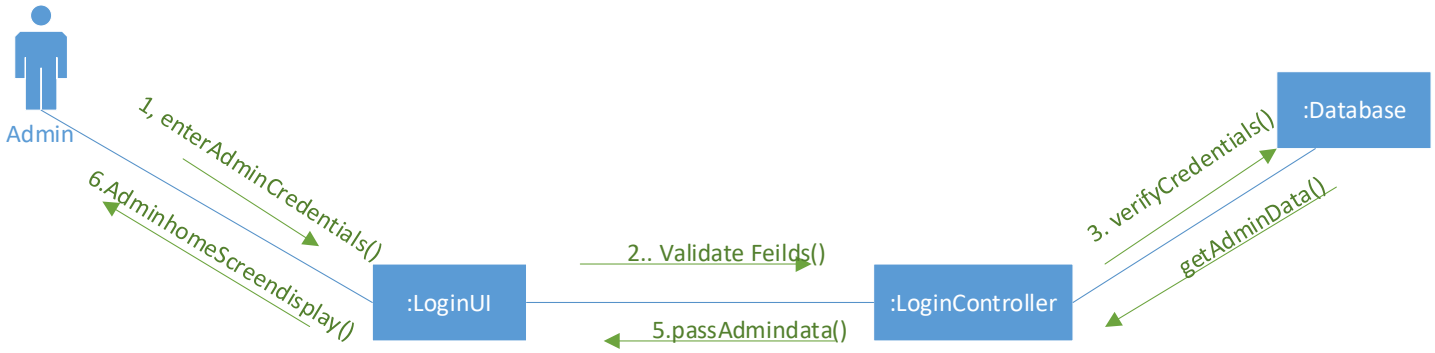
1.5.16 UC_P2P_Chat



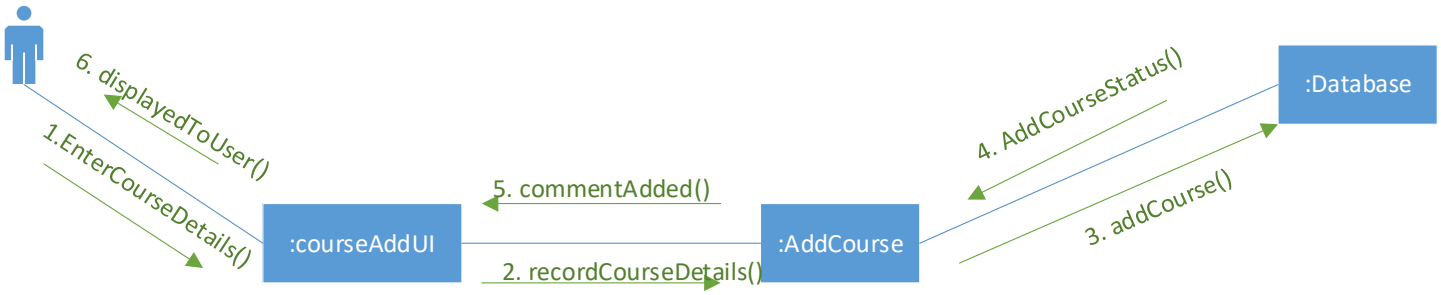
1.5.17 UC_Report_Message



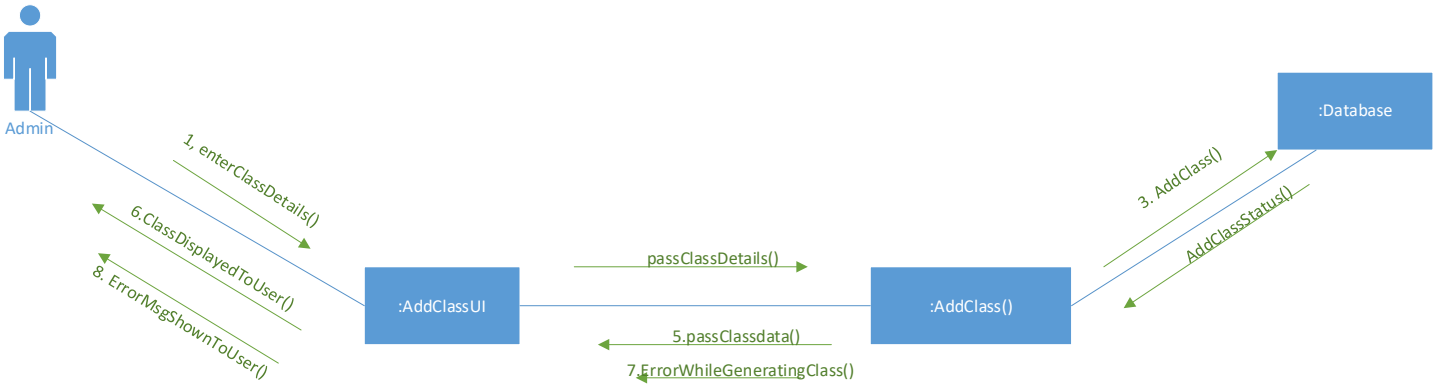
1.5.18 UC_Admin_Login



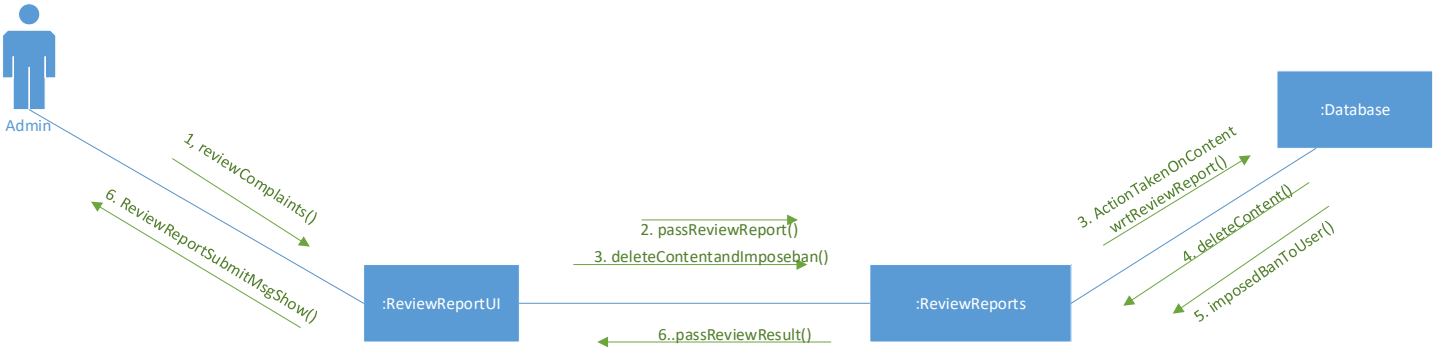
1.5.19 UC_Add_Courses



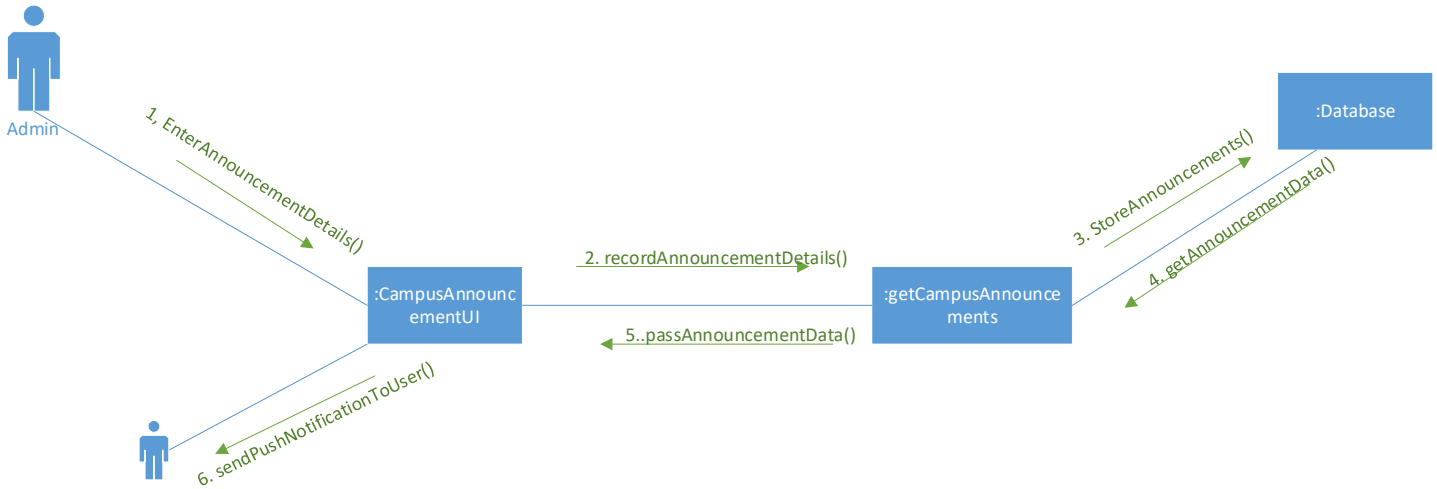
1.5.20 UC_Generate_Class



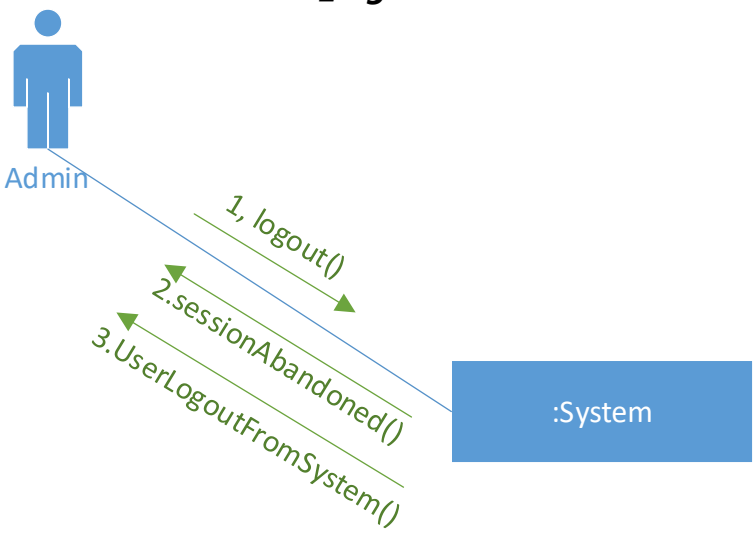
1.5.21 UC_Review_Reports



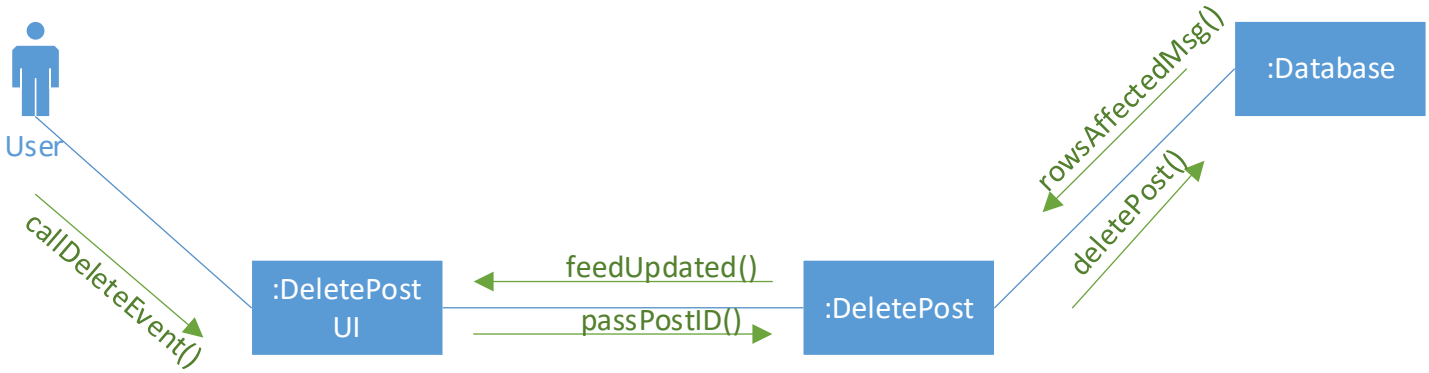
1.5.22 UC_Campus_Announcements



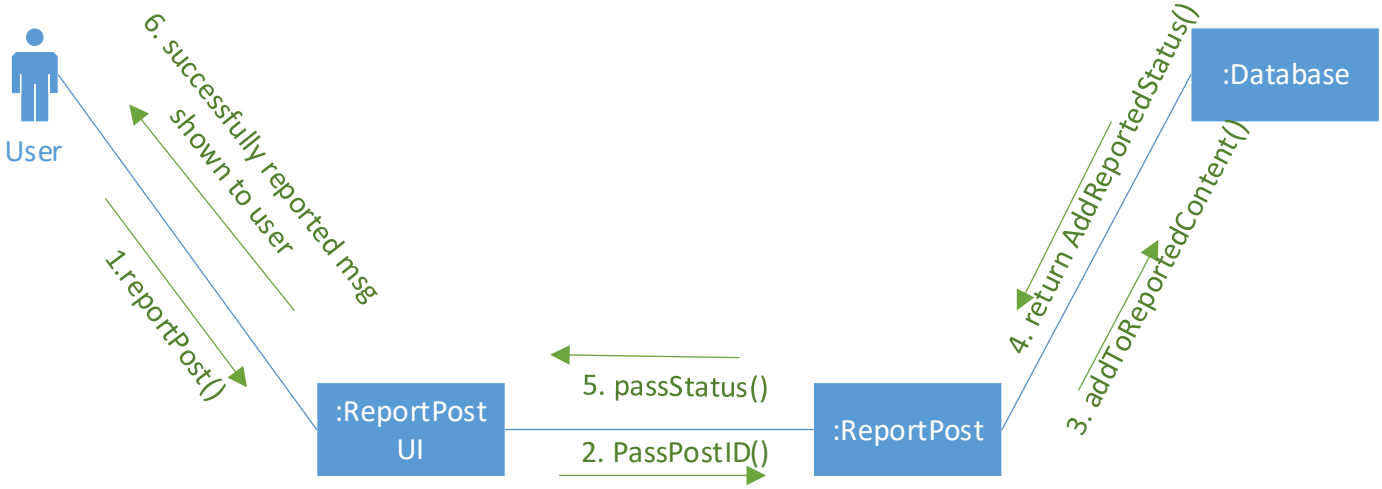
1.5.23 UC_logout



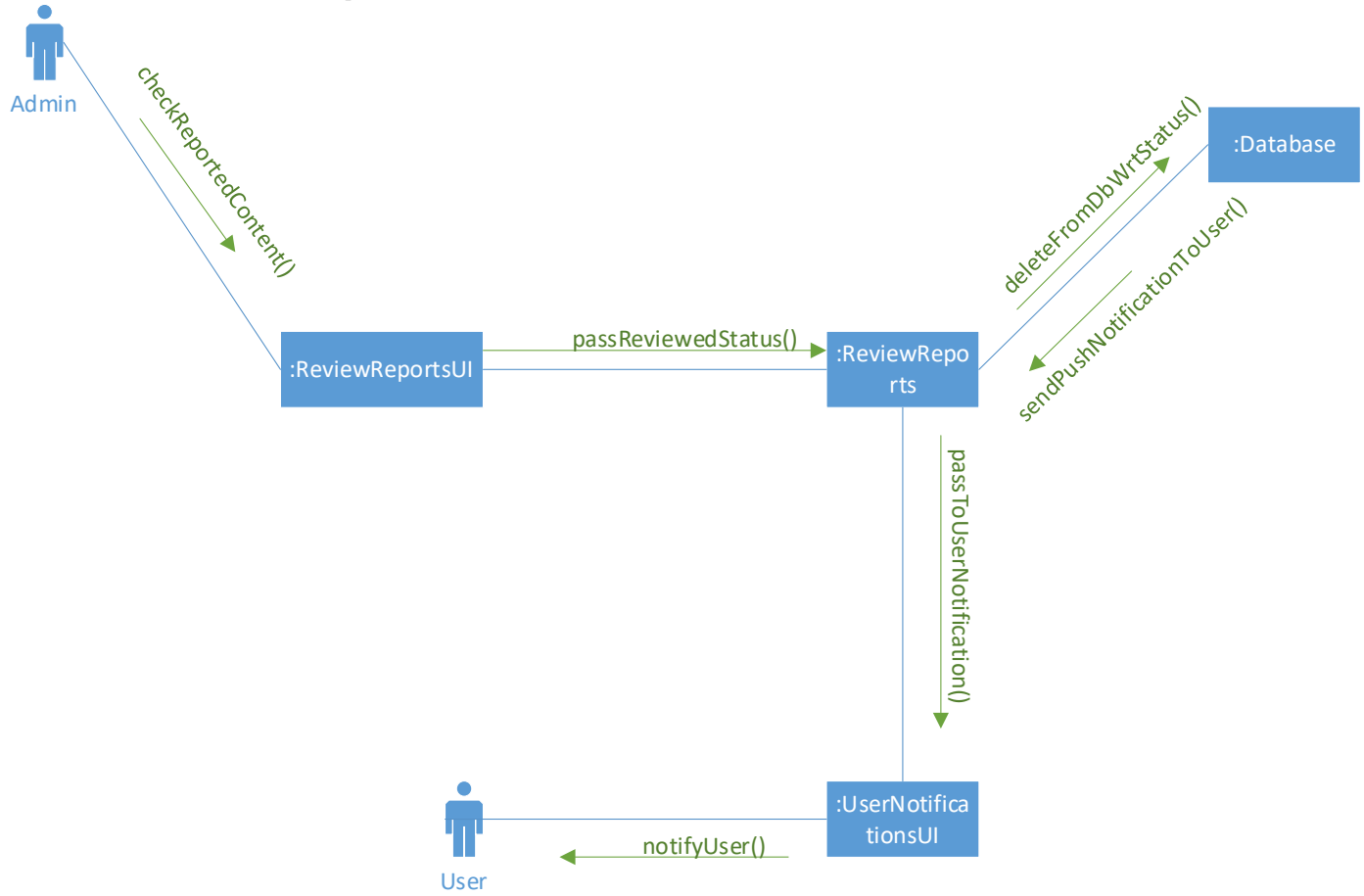
1.5.24 UC_Delete_Post



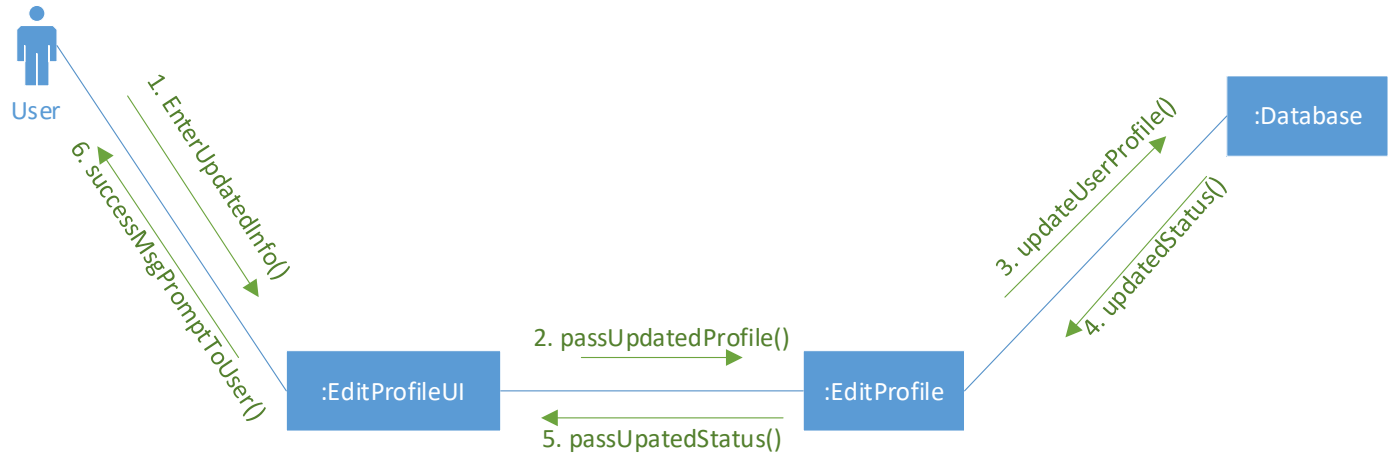
1.5.25 UC_Report_Post



1.5.26. UC_Review_Reports



1.5.27. UC_Edit_Profile



1.6 Operation Contracts

A UML Operation contract identifies system state changes when an operation happens. Effectively, it will define what each system operation does. An operation is taken from a system sequence diagram. It is a single event from that diagram. A domain model can be used to help generate an operation contract.

1.6.1 Contract CO_1

Name: User Registration
Responsibilities: Successful registration of the user.
Cross References: Use Cases: Register
Exceptions: User Already Exists
Preconditions: The user must have active internet connection
Postconditions: An account for the user will be created and the user will be directed to home screen.

1.6.2 Contract CO_2

Name: User Login
Responsibilities: It lets a registered user to sign into the application
Cross References: Use Cases: Registration, Login
Exceptions: User is not registered or wrong credentials are entered.
Preconditions: The user must have an account registered on the system and the user must have internet connection.
Postconditions: If the use case will be successful, the user will be logged into the system and directed to the home screen If not the system state will remain unchanged.

1.6.3 Contract CO_3

Name: News Feed Timeline
Responsibilities: It provides the post of users followed by the logged user in the news feed.
Cross References: Use Cases: Get User Posts
Exceptions: None
Preconditions: The user must log into the system and have an active internet connection
Postconditions: The system will provide all the posts, posted by the users followed by him/her.

1.6.4 Contract CO_4

Name: Add Comments to Post
Responsibilities: It let the user to add the comment to the user post in replies/comment section.
Exceptions: None
Cross References: Use Cases: Add Comments

Preconditions: User should be logged into the system
Postconditions: The comment will be posted.

1.6.5 Contract CO_5

Name: Like and Share the Post
Responsibilities: It let the user to like and share the post of the user from his/her profile and Newsfeed.
Exceptions: None
Cross References: Use Cases: Post Like & Share
Preconditions: User must log-in to the system and for sharing the post, sharing privileges are allowed by the post owner.
Postconditions: The post will be liked/ shared by the user.

1.6.6 Contract CO_6

Name: Create Post
Responsibilities: It let the user to create a post by uploading a pic and a caption or simply a text-based post.
Exceptions: Some details left unfilled, prompt for filling in all fields.
Cross References: Use Cases: Post Timeline Show
Preconditions: User must login into the system and have an active internet connection
Postconditions: The post will be uploaded successfully.

1.6.7 Contract CO_7

Name: User Classrooms
Responsibilities: It provides you all the classrooms in which the logged in user is enrolled.
Exceptions: No Class Enrolled by a User
Cross References: Use Cases: Get User Classrooms
Preconditions: User should be logged in.
Postconditions: Specific view of classes will be provided to the user.

1.6.8 Contract CO_8

Name: Get Classroom Posts and Resources
Responsibilities: It let you to access the posts and resources of the selected class.
Exceptions: None
Cross References: Use Cases: Get Classroom Content
Preconditions: User should be logged in and the user must be enrolled in that class.
Postconditions: Individual Class Page is loaded containing all Course Relevant Posts and Resources.

1.6.9 Contract CO_9

Name: Classroom Post Discussions
Responsibilities: It let the user to initiate or follow-up a discussion against the classroom post.
Exceptions: None
Cross References: Use Cases: User Discussions
Preconditions: User must be logged in and enrolled in that class.
Postconditions: Message against a classroom post or comment will be added successfully.

1.6.10 Contract CO_10

Name: Upload Class Resource
Responsibilities: It let the course instructor to upload the course resources in the classroom.
Exceptions: None
Cross References: Use Cases: Upload Class Resource
Preconditions: User should be logged in and enrolled in that class as an instructor.
Postconditions: Class Resource will be added/uploaded successfully.

1.6.11 Contract CO_11

Name: User Profiles
Responsibilities: It let user to access the content of user profile containing its personal info, external links for the other social sites and user post
Exceptions: None
Cross References: Use Cases: UC_User_Profiles
Preconditions: User account must exist and user should be logged into the system.
Postconditions: User Profile page will be shown to the user where external links, user info and its post will be displayed.

1.6.12 Contract CO_12

Name: Add Questions and Questions Pool
Responsibilities: It let instructor of the course to add questions to the existing pool for the course or either add a new question pool to generate a quiz for the course by picking the random questions from the quiz
Exceptions: User is a student and doesn't have the rights to add a question to the pool.
Cross References: Use Cases: Add Questions
Preconditions: User should be logged and have the instructor privileges for that course assigned by the admin
Postconditions: Questions Pool or a question to the existing pool added successfully.

1.6.13 Contract CO_13

Name: Universal Resource Post
Responsibilities: It let any of the user to post a resource related to anything by adding the tags in the universal resource center.
Exceptions: Some required fields left unfilled, prompt for filling in all fields.
Cross References: Use Cases: Post Universal Resource
Preconditions: User should be logged into the system.
Postconditions: Universal Resource will be posted successfully.

1.6.14 Contract CO_14

Name: Quiz Generation
Responsibilities: It let instructor of the course to generate the quiz from the existing question pools by selecting the random questions from the pool and assign them weightage manually.
Exceptions: None
Cross References: Use Cases: Generate Quiz
Preconditions: User should be logged into the system and must be an instructor.
Postconditions: Quiz will be generated successfully and push notification for the quiz send to all enrolled students

1.6.15 Contract CO_15

Name: Universal Chat
Responsibilities: It let any of the user to participate in the universal chat and collaborate and start the conversation with each other.
Exceptions: User is banned for temporary span of time because of sharing inappropriate content.
Cross References: Use Cases: UC_Universal_Chat
Preconditions: User account must exist and user should be logged into the system.
Postconditions: Message will be sent/received successfully.

1.6.16 Contract CO_16

Name: P2P Chat
Responsibilities: It will let any of the user to have a personal conversation with their fellow colleagues and their instructors. This includes individual chats as well as group chats
Exceptions: User is banned for temporary span of time because of sharing inappropriate content.
Cross References: Use Cases: UC_P2P_Chat
Preconditions: User account must exist and user should be logged into the system.

Postconditions: Message will be sent/received successfully.

1.6.17 Contract CO_17

Name: Report Message to Admin
Responsibilities: It will let the user to report the violent/inappropriate message to the admin . Admin review the message, if he found something unethical from the message, he will impose a ban on user for a temporary span of time.
Exceptions: None
Cross References: Use Cases: Report Message
Preconditions: User account must exist and user should be logged into the system and User must have received a message that contains sensitive content.
Postconditions: Message will be reported to admin successfully.

1.6.18 Contract CO_18

Name: Report Post to Admin
Responsibilities: It will let the user to report the violent/inappropriate post to the admin
Cross References: Use Cases: Report Post
Exceptions: None
Preconditions: User account must exist and user should be logged into the system and user must have seen a post that contains sensitive content.
Postconditions: Post will be reported to admin successfully.

1.6.19 Contract CO_19

Name: Admin Login
Responsibilities: It will let the admin user to login into the system and eventually access the admin functions.
Exceptions: None
Cross References: Use Cases: UC_Admin_Login
Preconditions: Admin account must exist on the system
Postconditions: The admin will be logged in successfully and be able to access all admin options.

1.6.20 Contract CO_20

Name: Add Course
Responsibilities: It will let the admin user to add new course to the existing road map.
Exceptions: None
Cross References: Use Cases: UC_Add_Class
Preconditions: Admin account must exist on the system and Admin must be logged in.

Postconditions: The course will be created successfully.

1.6.21 Contract CO_21

Name: Generate Classroom
Responsibilities: It will let the admin user to generate a new class and assign the course, instructor and instructor assistant to it.
Cross References: Use Cases: Generate Class
Exceptions: Instructor or Course doesn't exist
Preconditions: Admin account must exist on the system and Admin must be logged in.
Postconditions: The class will be created successfully.
.

1.6.22 Contract CO_22

Name: Review Report Content
Responsibilities: It will let the admin user to review all the reports made by users.
Cross References: Use Cases: Generate Class
Exception: None
Preconditions: Admin account must exist on the system and admin must be logged in, plus unreviewed complaints must exist.
Postconditions: The admin will have successfully reviewed a report/ complaint.

1.6.23 Contract CO_23

Name: Get Campus Announcements
Responsibilities: It will let the admin user to create campus announcements.
Cross References: Use Cases: Campus Announcements
Exception: None
Preconditions: Admin account must exist on the system and Admin must be logged in.
Postconditions: The admin will have successfully created an announcement.

1.6.24 Contract CO_24

Name: Logout User From System
Responsibilities: It will let a user to logout from the system.
Cross References: Use Cases: Logout
Exception: None
Preconditions: User must have an account and logged into the system.
Postconditions: The user will logout from the system.

1.6.25 Contract CO_25

Name: Edit User Profile
Responsibilities: It will let the user to edit his/her profile on the campuscube platform.
Cross References: Use Cases: Edit Profile
Exception: None
Preconditions: User must be logged in
Postconditions: The profile will be updated successfully.

1.6.26 Contract CO_26

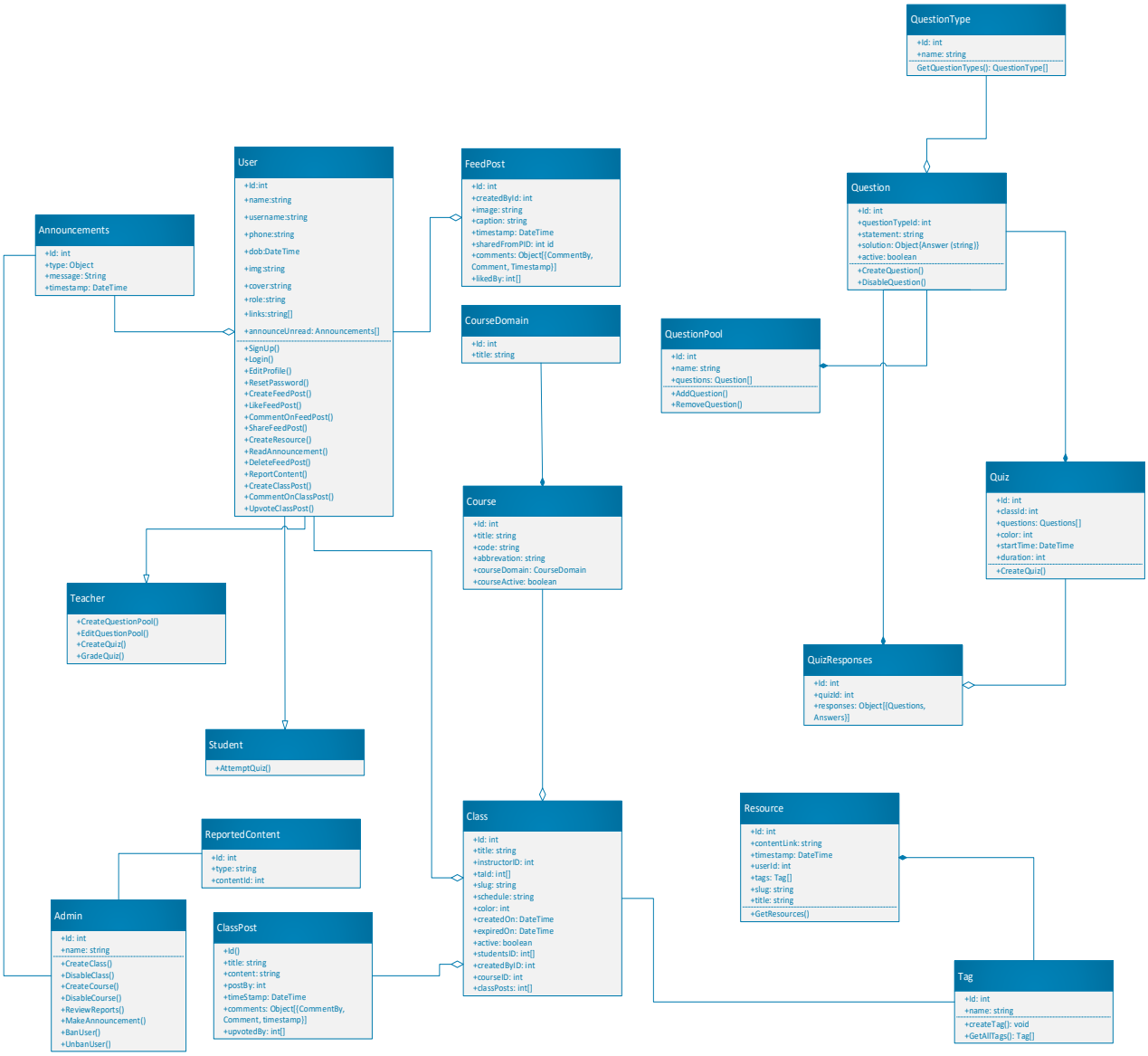
Name: Delete User Post
Responsibilities: It will let the user to delete his/her own post.
Cross References: Use Cases: Delete User Post
Exception: The post doesn't belong to the user who wants to delete it.
Preconditions: User must be logged in
Postconditions: The post will be deleted successfully.

1.6.27 Contract CO_27

Name: Search User Profile
Responsibilities: It will let the user to search other user profiles from the search bar shown in the header.
Cross References: Use Cases: Search User
Exception: Searched User Doesn't exist
Preconditions: User must be logged in
Postconditions: User found successfully.

.

1.7 Design Class Diagram



1.8 Data Model

Each of the entities were first identified, and a list of them was created:

Sr.	Name
1	announcements
2	class
3	class_post
4	color
5	course
6	course_domain
7	feed_post
8	question
9	question_pool
10	question_type
11	quiz
12	quiz_responses
13	reported_content
14	resource
15	tag
16	user
17	user_type

A detailed information of each of the entities listed above is as follows:

1.8.1 Announcements

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	announce_type		Document		
2.1	announce_type.type		String		
2.2	announce_type.postID		Id		
3	announce_message		String		

4	announce_timestamp	DateTime

Relations

Foreign table	Primary table	Join
user	announcements	user.u_announceUnread = **announcements**._id

1.8.2 Class

Columns

Sr.	Name	Key	Data type	Null	References
1	_id		Id		
2	class_title		String		
3	class_instructor_ufbid		String		user
4	class_TAs_ufbid		String[]		user
5	class_slug		String		
6	class_schedule		String		
7	class_color		Id		color
8	class_createdOn		DateTime		
9	class_expiredOn		DateTime		
10	class_active		Boolean		
11	class_students_ufbid		String[]		user
12	class_createdByAdmin_ufid		String		
13	class_courseID		Id		course

Relations

Foreign table	Primary table	Join
class	color	**class**.class_color = color.colors_class
class	course	**class**.class_courseID = course._id
class	user	**class**.class_students_ufbid = user.u_fbid
class	user	**class**.class_instructor_ufbid = user.u_fbid

1.8.3 class_post

Columns

	Name	Key	Data type	Null	References
1	_id		Id		
2	cp_title		String		
3	cp_content		String		
4	cp_u_fbid		String		user
5	cp_comments		Document[]		user
5.1	cp_comments.u_fbid		String		
5.2	cp_comments.comment		String		
5.3	cp_comments.replies		Document[]		
5.3.1	cp_comments.replies.u_fbid		String		
5.3.2	cp_comments.replies.comment		String		
5.3.3	cp_comments.replies.replies		Document[]		
6	cp_upvotedBy		String[]		user
7	cp_classId		Id		class
8	cp_timestamp		DateTime		

Relations

Foreign table	Primary table	Join
class_post	class	**class_post**.cp_classId = class._id
class_post	user	**class_post**.cp_u_fbid = user.u_fbid
class_post	user	**class_post**.cp_comments = user.u_fbid
class_post	user	**class_post**.cp_upvotedBy = user.u_fbid

1.8.4 color

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	colors_class		Document[]		
2.1	colors_class.color_code		String		
2.2	colors_class.color_slug		String		
2.3	colors_class.color_name		String		
3	colors_accent		Document[]		
3.1	colors_accent.color_code		String		
3.2	colors_accent.color_slug		String		
3.3	colors_accent.color_name		String		
4	colors_quiz		Document[]		
4.1	colors_quiz.color_code		String		
4.2	colors_quiz.color_slug		String		
4.3	colors_quiz.color_name		String		

Relations

Foreign table	Primary table	Join
class	color	class.class_color = color.colors_class
quiz	color	quiz.quiz_color = color.colors_quiz

1.8.5 course

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	course_title		String		
3	course_code		String		
4	course_abbr		String		
5	course_domain	FK	Id		course_domain
6	course_active		Boolean		

Relations

Foreign table	Primary table	Join
course	course_domain	course.course_domain = course_domain._id
class	course	class.class_courseID = course._id

1.8.6 course_domain

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	cdomain_title		String		
3	cdomain_code		String		

Relations

Foreign table	Primary table	Join
course	course_domain	course.course_domain = course_domain._id

1.8.7 *feed_post*

Columns

	Name	Key	Data type	Null	References
1	_id		Id		
2	fp_caption		String		
3	fp_comments		Document[]		user
1	fp_comments.u_fbid		String		
2	fp_comments.comment		String		
3	fp_comments.replies		Document[]		
1	fp_comments.replies.u_fbid		String		user
2	fp_comments.replies.comment		String		
3	fp_comments.replies.replies		Document[]		
4	fp_img		String		
5	fp_likedBy		String[]		user
6	fp_sharedFromPID		Id		feed_post
7	fp_timestamp		DateTime		
8	fp_u_fbid		String		user

Relations

Foreign table	Primary table	Join
feed_post	feed_post	**feed_post**.fp_sharedFromPID = **feed_post**._id
feed_post	user	**feed_post**.fp_comments = user.u_fbid
feed_post	user	**feed_post**.fp_u_fbid = user.u_fbid
feed_post	user	**feed_post**.fp_likedBy = user.u_fbid
feed_post	user	**feed_post**.fp_comments.replies.u_fbid = user.u_fbid

1.8.8 *question*

Columns

Sr.	Name	Key	Data type	Null	References

1	_id	PK	Id	
2	question_QTID	FK	String	question_type
3	question_statement		String	
4	question_solution		Document	
5	question_active		Boolean	

Relations

Foreign table	Primary table	Join
question	question_type	question.question_QTID = question_type._id
question_pool	question	question_pool.qpool_questions = question._id
quiz	question	quiz.quiz_questions = question._id
quiz_responses	question	quiz_responses.qres_responses.answers.question_id = question._id

1.8.9 question_pool

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	qpool_name		String		
3	qpool_questions	FK	String[]		question

Relations

Foreign table	Primary table	Join
question_pool	question	question_pool.qpool_questions = question._id

1.8.10 question_type

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	qtype_name		String		

Relations

Foreign table	Primary table	Join
question	question_type	question.question_QTID = **question_type**._id

1.8.11 *quiz*

Columns

Sr.	Name	Key	Data type	Null	References
1	_id		Id		
2	quiz_cid		Id		class
3	quiz_questions		Document[]		question
3.1	quiz_questions.id		Id		
3.2	quiz_questions.marks		Int32		
4	quiz_color		Id		color
5	quiz_startTime		DateTime		
6	quiz_duration		Int64		

Relations

Foreign table	Primary table	Join
quiz	class	quiz.quiz_cid = class._id
quiz	color	quiz.quiz_color = color.colors_quiz
quiz	question	quiz.quiz_questions = question._id
quiz_responses	quiz	quiz_responses.qres_quiz_id = quiz._id

1.8.12 *quiz_responses*

Columns

Sr.	Name	Key	Data type	Null	References
1	_id		Id		
2	qres_quiz_id		Id		quiz
3	qres_responses		Document[]		

3.1	qres_responses.student_ufbid	String	user
3.2	qres_responses.answers	Document[]	
3.2.1	qres_responses.answers.question_id	Id	question
3.2.2	qres_responses.answers.answer	String	
3.2.3	qres_responses.answers.obtained_marks	Int32	

Relations

Foreign table	Primary table	Join
quiz_responses	question	quiz_responses.qres_responses.answers.question_id = question._id
quiz_responses	quiz	quiz_responses.qres_quiz_id = quiz._id
quiz_responses	user	quiz_responses.qres_responses.student_ufbid = user.u_fbid

1.8.13 reported_content

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	rc_type		String		
3	rc_id	FK	Id		post

Relations

Foreign table	Primary table	Join
reported_content	post	reported_content.rc_id = post._id

1.8.14 resource

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	res_contentLink		String		
3	res_timestamp		DateTime		
4	u_fbid	FK	String		user
5	res_tags	FK	String[]		tag

6	res_slug	String
7	res_title	String

Relations

Foreign table	Primary table	Join
resource	tag	resource.res_tags = tag._id
resource	user	resource.u_fbid = user.u_fbid

1.8.15 tag

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	tag_name		String		

Relations

Foreign table	Primary table	Join
resource	tag	resource.res_tags = tag._id

1.8.16 user

Columns

Sr.	Name	Key	Data type	Null	References
1	_id	PK	Id		
2	u_fbid		String		
3	u_name		String		
4	u_username		String		
5	u_phone		String		
6	u_dob		DateTime		
7	u_img		String		
8	u_cover		String		
9	u_type		Id		user_type
10	u_links		Document		

11	u_darkTheme		Boolean	
12	u_announceUnread	FK	ObjectID[]	announcements

Relations

Foreign table	Primary table	Join
user	announcements	**user**.u_announceUnread = announcements._id
user	user_type	**user**.u_type = user_type._id
class	user	class.class_students_ufbid = **user**.u_fbid
class	user	class.class_instructor_ufbid = **user**.u_fbid
class	user	class.class_TAs_ufbid = **user**.u_fbid
class_post	user	class_post.cp_u_fbid = **user**.u_fbid
class_post	user	class_post.cp_comments = **user**.u_fbid
class_post	user	class_post.cp_upvotedBy = **user**.u_fbid

1.8.17 user_type

Columns

Sr.	Name	Key	Data type	Null	References
1	_id		Id		
2	uType		String		

Relations

Foreign table	Primary table	Join
user	user_type	user.u_type = **user_type**._id

All the entities and their relations defined above were then used in formulating a data model diagram as follows: