

## Project Part 2

### 1. Team:

- a. Richard Noronha
- b. Jayakrishnan HJ

### 2. Title: CUConnect

### 3. Project Summary:

CUConnect is an online portal for CU students and past pupils to connect with each other. It is a social media prototype for such a portal. In this portal, the users can connect and share content such as photos and status. The other users will have the ability to view these photos and status. They will also be able to make comments and Thumbsup a post.

Actors: User, Admin

The backend of the system will be implemented using Java. We have tried to be as detailed as we can foresee as of now. As and when modifications are needed, the documents and diagrams will be updated. A database will be implemented too

### 4. Project Requirements:

User Requirements			
ID	DESCRIPTION	USER	PRIORITY
UR-01	As a new user I should be able to register and create an account	User	Low
UR-02	As an existing user I should be able to login to the system	All actors	Low
UR-03	As an existing user I should be able to add comments on other user's profile contents	All actors	Low
UR-04	As an existing user who is an admin, I should be able to review and delete objectionable comments	Admin	Low
UR-05	As an existing user I should be able to add content to my account	All actors	Low
UR-06	As an existing user as an admin I should be able to delete any content of another user	Admin	Low
UR-07	As an existing user I should be able to search user based on filters	All actors	Low
UR-08	As an existing user I should be able to add other users as friends	User	Low
UR-09	All actors should be able to delete their accounts	All actors	Low
UR-10	As an existing user I should be able to delete another user's account	Admin	Low
UR-11	As an existing user I should be able to view another user's full content if he's added as a friend	User	Low
UR-12	As an existing user I should be able to thumbs up/down to another friend's content	User	Low
UR-13	As a user, I should be able to edit my content	User	Low
UR-14	As a user, I should be able to change privacy settings	All actors	Low

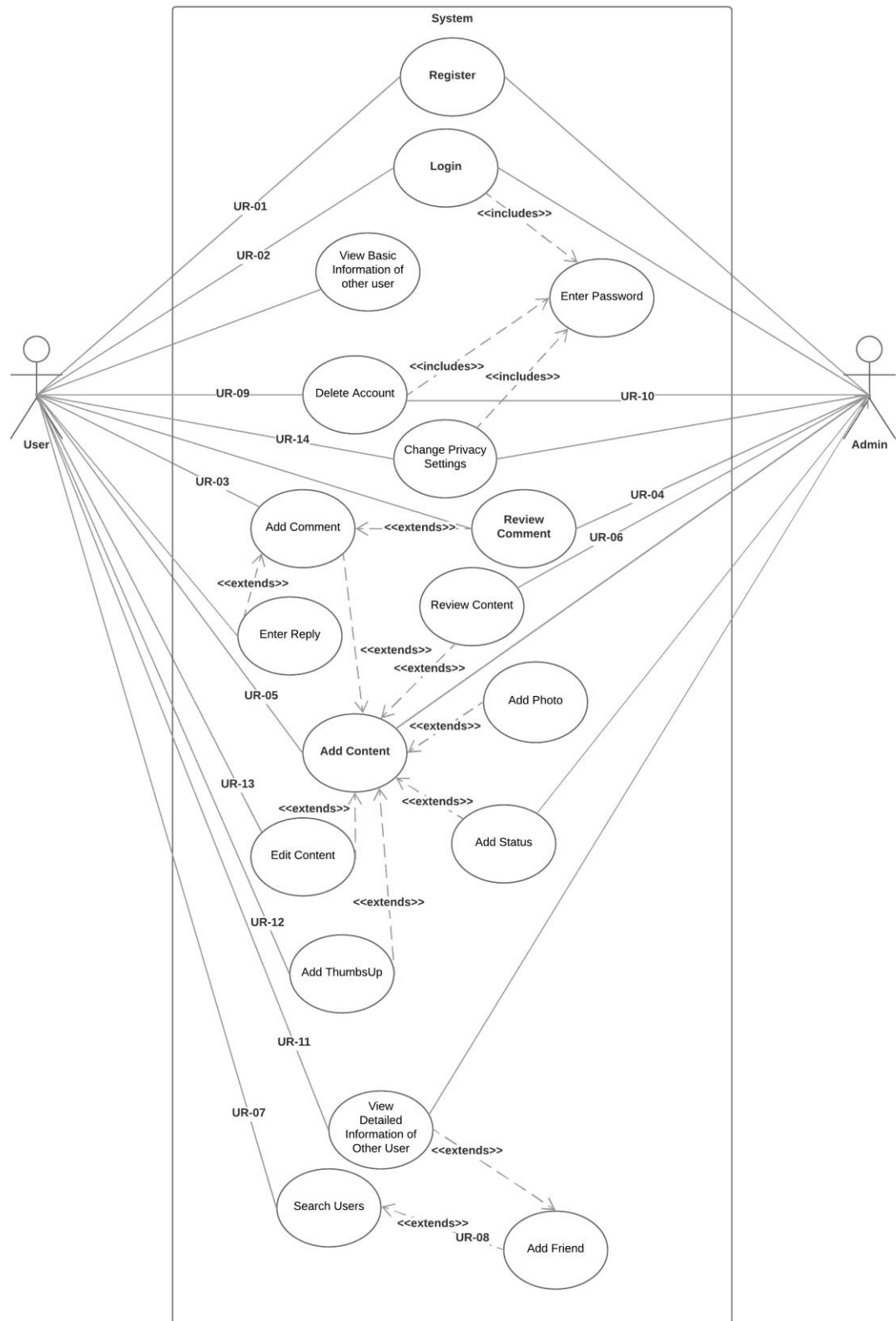
ID	DESCRIPTION	TOPIC AREA
BR-01	Project must be implemented in Java using Object Oriented Design practices	Implementation
BR-02	It should be a standalone application	Implementation

<b>BR-03</b>	It should have a central repository in Github and it should be versioned controlled through git	Implementation
<b>BR-04</b>	The code should be refactored to follow industry recognized design patterns	Implementation

Functional Requirements			
ID	DESCRIPTION	TOPIC AREA	PRIORITY
<b>FR-01</b>	Authentication of all users is performed using login and password	Authentication	Low
<b>FR-02</b>	All users should be able to modify their privacy settings	Authentication	Low
<b>FR-03</b>	Only CU-Boulder students, staff and alumni are allowed to set up an account	Accounts	High
<b>FR-04</b>	Users should be able to search other users based on various filter settings such as graduation year, location and major	Search	High

Non-Functional Requirements			
ID	DESCRIPTION	USER	PRIORITY
<b>NFR-01</b>	Passwords can be stored after encrypting	Authentication	High
<b>NFR-02</b>	The system should be available all day	Reliability	High
<b>NFR-03</b>	The system should be able to support n users simultaneously	Performance	High
<b>NFR-04</b>	The system should generate periodic logs about new users and the amount of data added to the database	Stats	Medium

## 5. Use Cases:



## Use Case Documents

<b>Use Case ID:</b>	UC-01
<b>Use Case Name:</b>	Register a User
<b>Description:</b>	Register a new user. Create a new login for this user when a guest signs up

<b>Actors:</b>	All Actors		
<b>Pre-conditions:</b>	User does not have an account		
<b>Post-Conditions:</b>	User will have an account and can login		
<b>Frequency of Use:</b>	Every time a guest user wants to be registered on the system		
<b>Flow of Events:</b>		<b>Actor Action</b>	<b>System Response</b>
	1	Select Sign-Up	Show Sign-Up Screen
	2	Enter user information: username, password, Name, Address, Date of Birth	Store the entered information into a database
	3		Authenticate the email id.
	4	User can login	
<b>Variations:</b>			
<b>Exceptions:</b>	New account creation would fail if the account exists. If username exists, ask for new username		
<b>Developer Notes:</b>			

<b>Use Case ID:</b>	UC-02
<b>Use Case Name:</b>	Login
<b>Description:</b>	Registered user can login to the system

<b>Actors:</b>	All actors		
<b>Pre-conditions:</b>	User is not currently logged into the system		
<b>Post-Conditions:</b>	User is logged into the system		
<b>Frequency of Use:</b>	Every time a user wants to log into the system		
<b>Flow of Events:</b>		<b>Actor Action</b>	<b>System Response</b>
	1	User enters username and password	Authenticates the credentials
	2		If authentication succeeds, open homepage
	3		If authentication fails, display error message to the user
	4		Ask user to reenter details or Signup
<b>Variations:</b>	Provide a means of handling forgotten password System requests user to enter email address to send password reset instructions User enters email address		

	System sends the password reset instructions to the email address User resets password New password is stored in system System asks users to login with new credentials
<b>Exceptions:</b>	Lock user account temporarily if the wrong credentials are entered more than 5 times
<b>Developer Notes:</b>	

<b>Use Case ID:</b>	UC-03
<b>Use Case Name:</b>	Adding a friend
<b>Description:</b>	The logged in user wants to add a friend who is already registered in the system

<b>Actors:</b>	User		
<b>Pre-conditions:</b>	User is logged in		
<b>Post-Conditions:</b>	The other user is added as a friend		
<b>Frequency of Use:</b>	Anytime the user requests		
<b>Flow of Events:</b>		<b>Actor Action</b>	<b>System Response</b>
	1	Enter other user's name in search box	Search database
	2		If user not found, display error finding user
	3		If user found, check friendship between users.
	4	Adds other user as friend	If users are not friends, ask the user to add the other user as friend
			Display detailed content of other user
<b>Variations:</b>	If users are already friends display the content of the other user		
<b>Exceptions:</b>	User can choose to not add the person as friend and just view basic information		
<b>Developer Notes:</b>			

<b>Use Case ID:</b>	UC-04
<b>Use Case Name:</b>	Adding content
<b>Description:</b>	The user can add personal content

<b>Actors:</b>	User, Admin		
<b>Pre-conditions:</b>	User is logged in		
<b>Post-Conditions:</b>	Content available to all friends		
<b>Frequency of Use:</b>	Anytime		
<b>Flow of Events:</b>		<b>Actor Action</b>	<b>System Response</b>
	1		Displays homepage with button to add content
	2	Clicks type of content to add	Confirm the visibility of content as basic or detailed
	3	Chooses type of content	Enters content into the database
<b>Variations:</b>	User can choose to add content of the form status or photo		
<b>Exceptions:</b>	Admin can only add status content		
<b>Developer Notes:</b>			

<b>Use Case ID:</b>	UC-05
<b>Use Case Name:</b>	Making comment and editing comments
<b>Description:</b>	Comments can be added to the content posted

<b>Actors:</b>	User, Admin		
<b>Pre-conditions:</b>	Content is present		
<b>Post-Conditions:</b>	Comment is posted		
<b>Frequency of Use:</b>	When the user wants to post a comment		
<b>Flow of Events:</b>		<b>Actor Action</b>	<b>System Response</b>
	1	The user searches for other user's content	Checks if user is friends with other user
	2		Display Basic Content if not friend.
	3		Displays Detailed Content if friend
	4	User adds comment	Checks if content is still present
	5		System adds comment if content is still present
	6	User edits comment	Check if user owns comment
	7		If user owns comment save comment
	8	View error message	If user doesn't own comment, display error message
<b>Variations:</b>	2. The User cannot comment if not a friend. Go to UC-03 to add user as friend.		
<b>Exceptions:</b>	4. If content is deleted by the time comment is made, the system responds with error message 6. User may not own comment to edit		

<b>Developer Notes:</b>	
-------------------------	--

<b>Use Case ID:</b>	UC-06
<b>Use Case Name:</b>	Delete content
<b>Description:</b>	The content can be removed

<b>Actors:</b>	User who owns content, admin		
<b>Pre-conditions:</b>	Content is present		
<b>Post-Conditions:</b>	The Content is removed		
<b>Frequency of Use:</b>			
<b>Flow of Events:</b>		<b>Actor Action</b>	<b>System Response</b>
	1	Content owner marks the content for deletion	Confirm whether content is to be removed.
	2	Approves removal of content	Removes content from the system
	3		Update the content of the User
	4		
<b>Variations:</b>	Admin can delete any users content that is found objectionable		
<b>Exceptions:</b>			
<b>Developer Notes:</b>			

<b>Use Case ID:</b>	UC-07
<b>Use Case Name:</b>	Delete comments
<b>Description:</b>	The users and admin have the ability to delete content they own

<b>Actors:</b>	User who owns comment		
<b>Pre-conditions:</b>	Comment is present		
<b>Post-Conditions:</b>	Comment is deleted		
<b>Frequency of Use:</b>			
<b>Flow of Events:</b>		<b>Actor Action</b>	<b>System Response</b>
	1	User marks the comment for deletion	The system checks if user owns the comment
	2	User confirms deletion of comment	If user owns comment, ask for confirmation
	3		The system deletes the comment
	4		The system checks for replies to comment and deletes the string of replies
<b>Variations:</b>			
<b>Exceptions:</b>			
<b>Developer Notes:</b>			

<b>Use Case ID:</b>	UC-08
<b>Use Case Name:</b>	Change privacy settings of user
<b>Description:</b>	

<b>Actors:</b>	User, Admin		
<b>Pre-conditions:</b>	User is registered and logged in		
<b>Post-Conditions:</b>	Settings modified		
<b>Frequency of Use:</b>			
<b>Flow of Events:</b>		<b>Actor Action</b>	<b>System Response</b>
	1	User requests to change settings	Request for password again
	2	Enters password	Checks credentials
	3		Display settings page
	4	Makes necessary changes to settings	Saves changes to settings
<b>Variations:</b>	Admin can change all users settings simultaneously		
<b>Exceptions:</b>			
<b>Developer Notes:</b>			

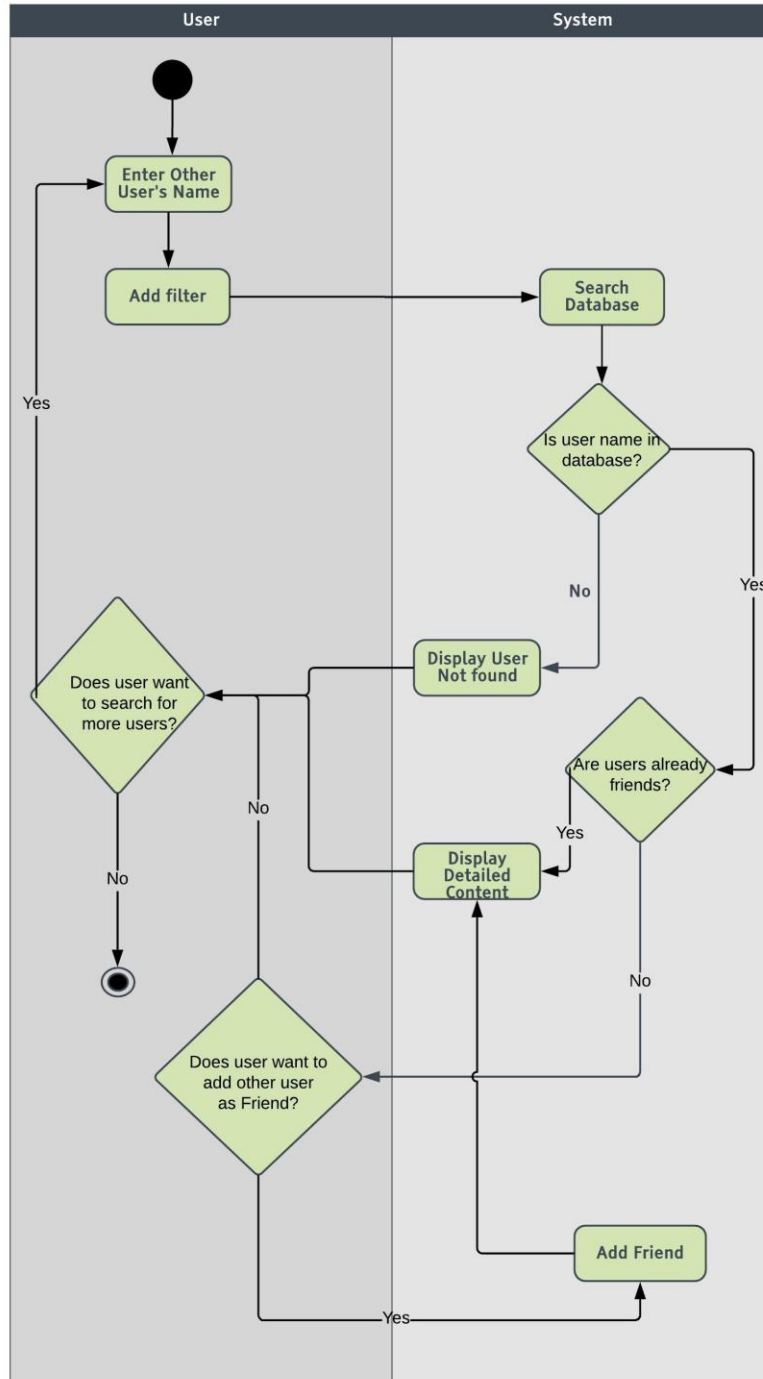


6. Activity Diagrams:

**Use Case Name: Adding a friend**

**RequirementID: UR-08**

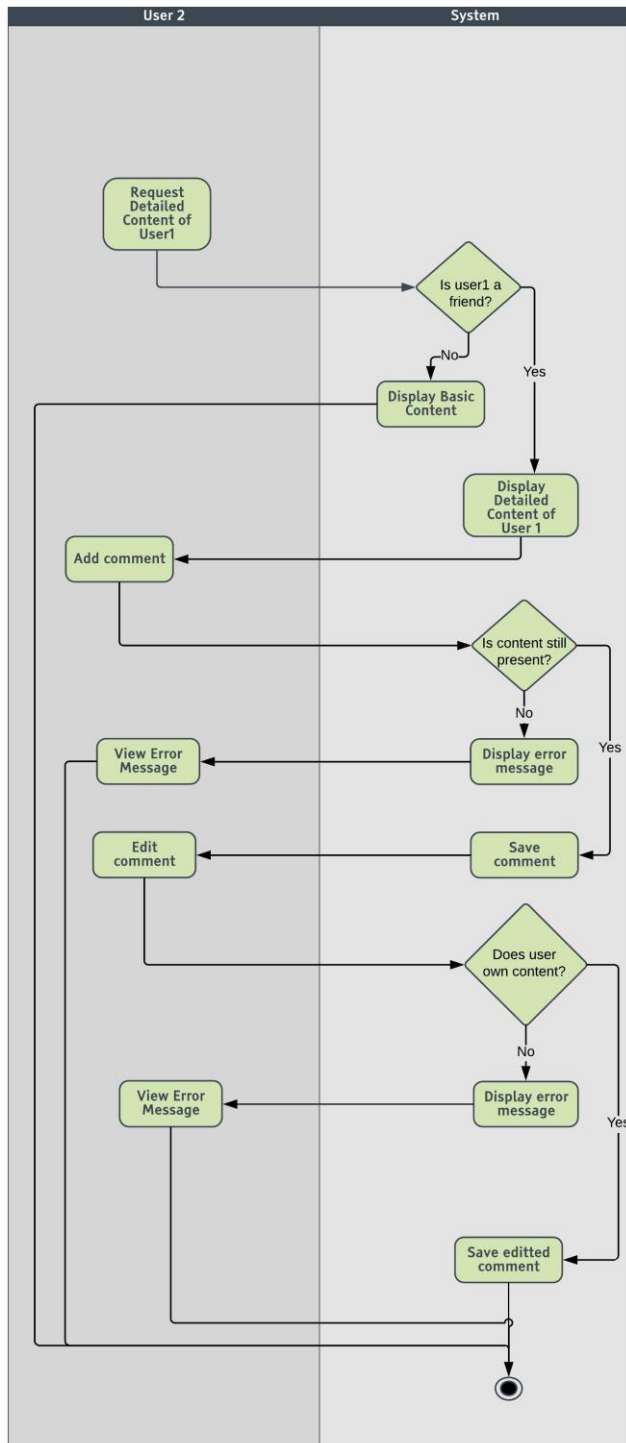
**UseCaseID: UC-03**



Use Case Name: Making comment and editing comments

RequirementID: UR-03

UseCaseID: UC-03

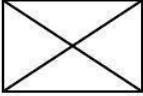


## UI Mockups

Title

[←](#) [→](#) [↻](#) [🏠](#)

Wecolme to



CUConnect

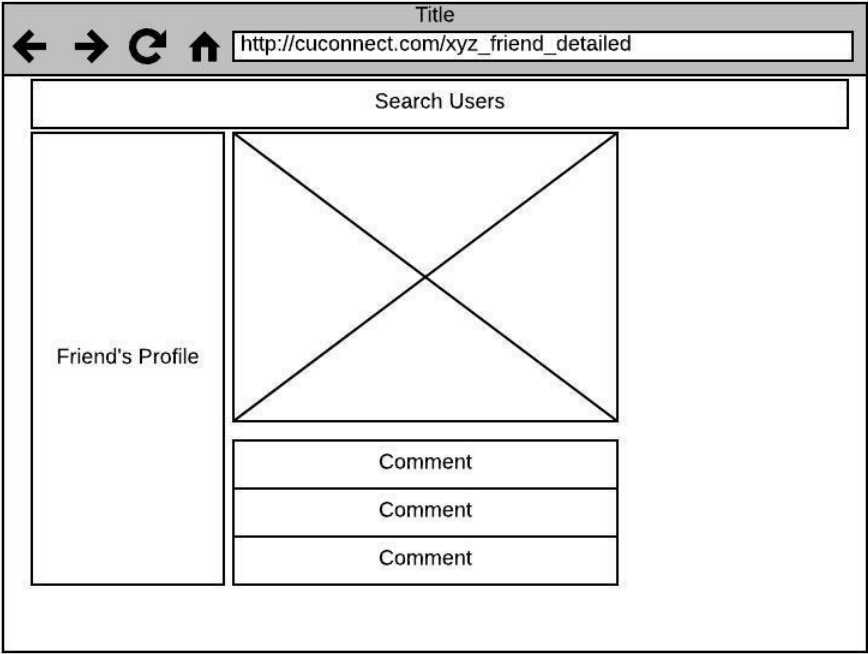
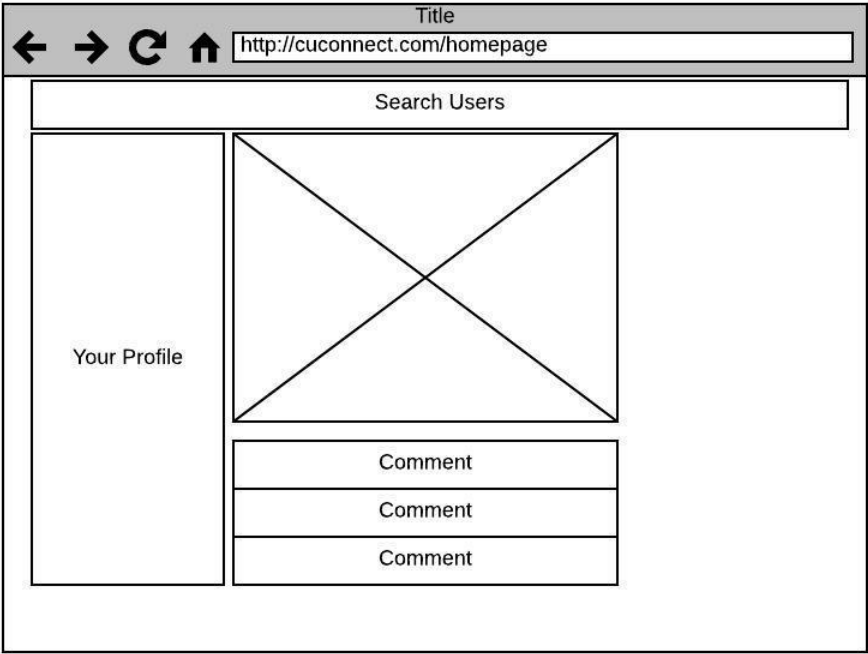
Enter Username

Enter Password

[Forgot Password?](#)

[Sign In](#)

[Register](#)

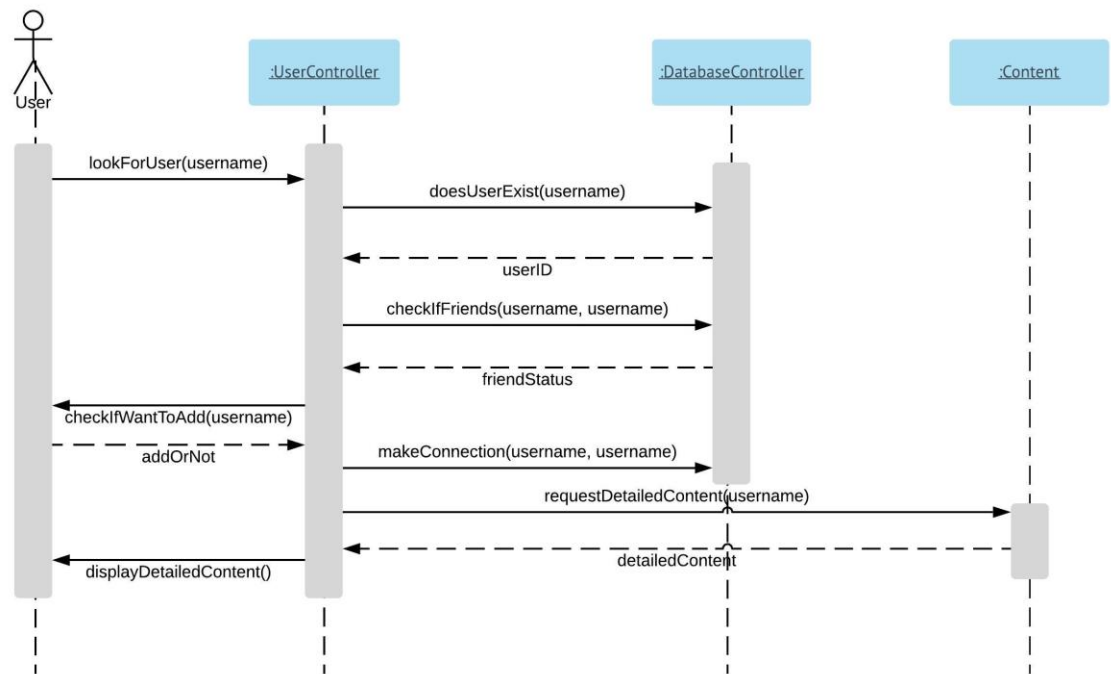


Sequence Diagram for adding friend:

**Use Case Name: Adding a friend**

**RequirementID: UR-08**

**UseCaseID: UC-03**

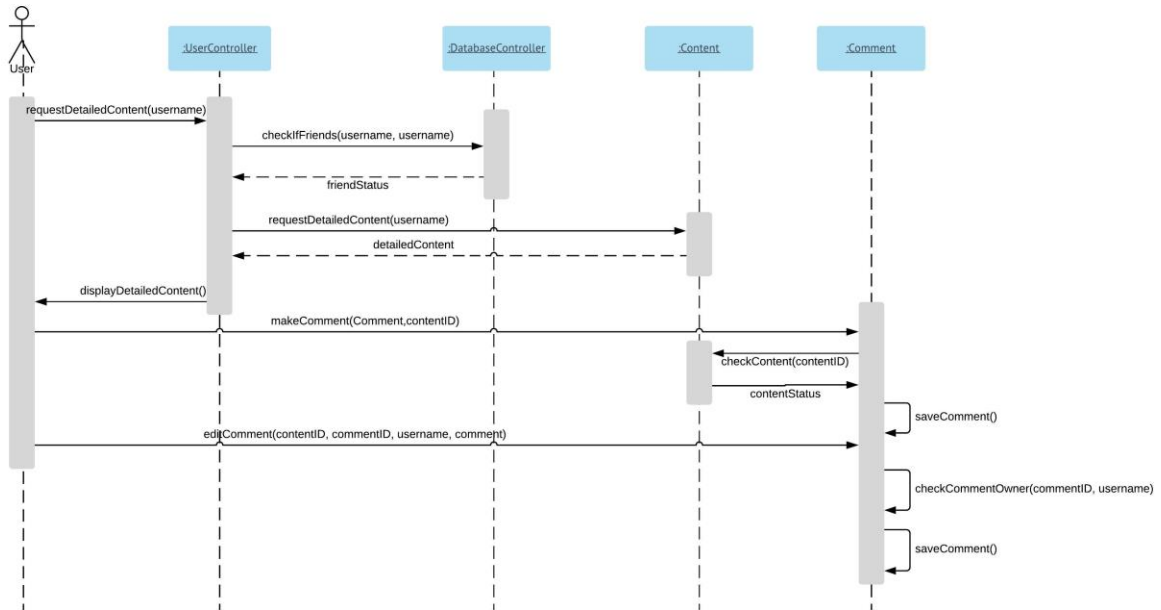


Sequence Diagram for making comments:

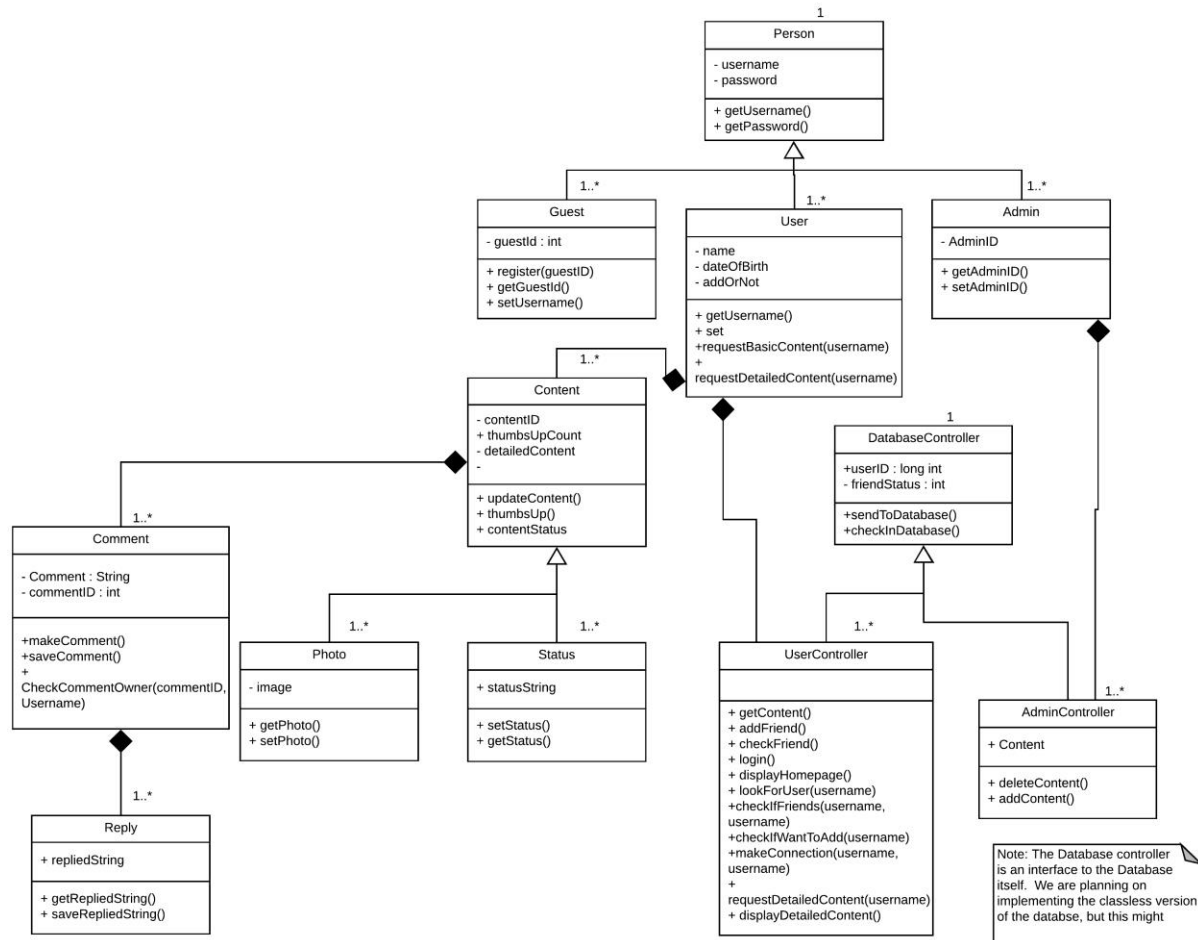
**Use Case Name: Making comment and editing comments**

**RequirementID: UR-03**

**UseCaseID: UC-03**



## Class Diagram:



For Datastorage, we plan to use Hibernate with MySQL. This will be changed/Modified based on the requirements for the project.