EAST WEST UNIVERSITY

CSE411: Software Engineering & Information System Design

Sec-1

Project Report

Submitted to:

Dr. Mohammad Salah Uddin

Assistant Professor

Department of Computer Science & Engineering

East West University

Submitted by:

(2018-1-60-063) Md. Habibur Rahman

(2018-1-60-076) Abdur Rahman Tumzied

(2018-1-60-042) Md. Mania Ahmed Joy

Submission Date: 05/15/2021

Problem:

There has never been a virtual space exclusively for the students of EWU where they can connect with their faculties, clubs, communities and the university as a whole. Also, the faculties do not have any personalized virtual space where they can share their thoughts and motivate the students of EWU. Now a days, most of the students have to connect with their desired EWU communities via Facebook or other platforms. Now these platforms are not designed solely for students, so the students often get distracted and waste a lot of time. Also, there may be few students who do not want to use Facebook as Facebook collects user data.

Solution:

To solve the aforementioned problem, we are going to design a website called EWU CONNECT.
EWU CONNECT is a virtual community platform designed exclusively for the students of East West University.

Description:

Only EWU members can be registered and registered members can submit links, text posts, images. which are then voted up or down by other members in the same community. Posts are arranged by tops into user-created group called "community" which cover up many topics like news, EWU club activity, events, movies, video games, music, books and image-sharing. Post that has more up-votes appears at the top of their community, if the post receives enough up-votes, eventually on the site's front page. Even with strict rules prohibiting harassment, Community administrators can moderate the site.

Feasibility Study of EWU Connect:

Technical feasibility:

Our project is technically feasible as we are using PHP8, HTML, CSS, Bootstrap, AND MySQL. All these current technologies will go a long way to ensure the satisfaction of our users and our implementation process is time efficient and cheap.

Operational Feasibility:

Our project will go a long way to ensure a private virtual space for the students and teachers of EWU. Our website will make it easier and more efficient for the students to connect with their EWU Communities and their Faculties. Moreover, the faculties will have an exclusive space to share their thoughts and ideas. We also aim to ensure security for our users and no data will be used elsewhere. Also, our upvote system will make sure our user gets to know about all the top posts.

Economic Feasibility:

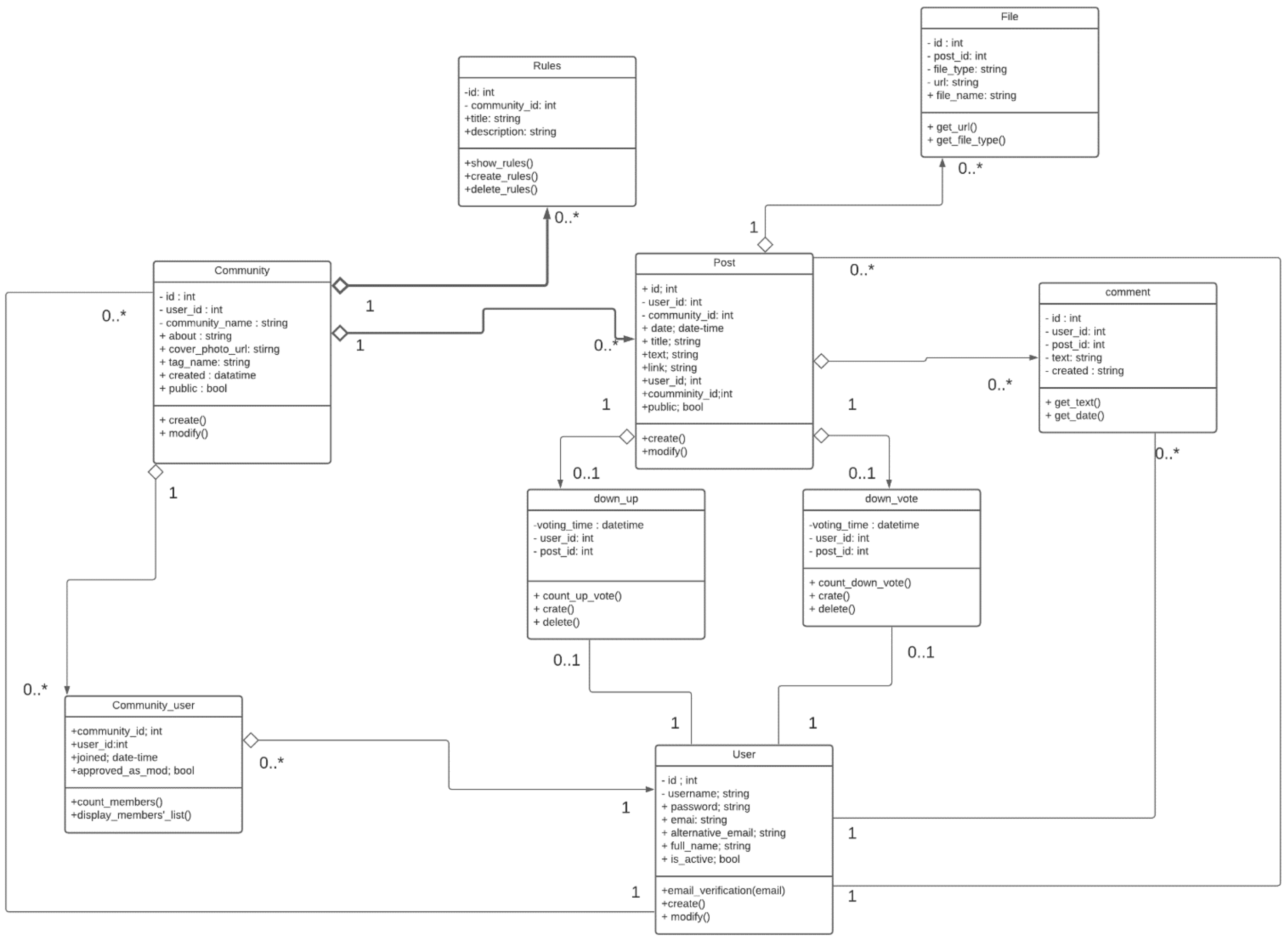
We have no goal to earn from this as our project is non-profit. Hence, we have no economic feasibility.

Requirement Analysis of EWU Connect:

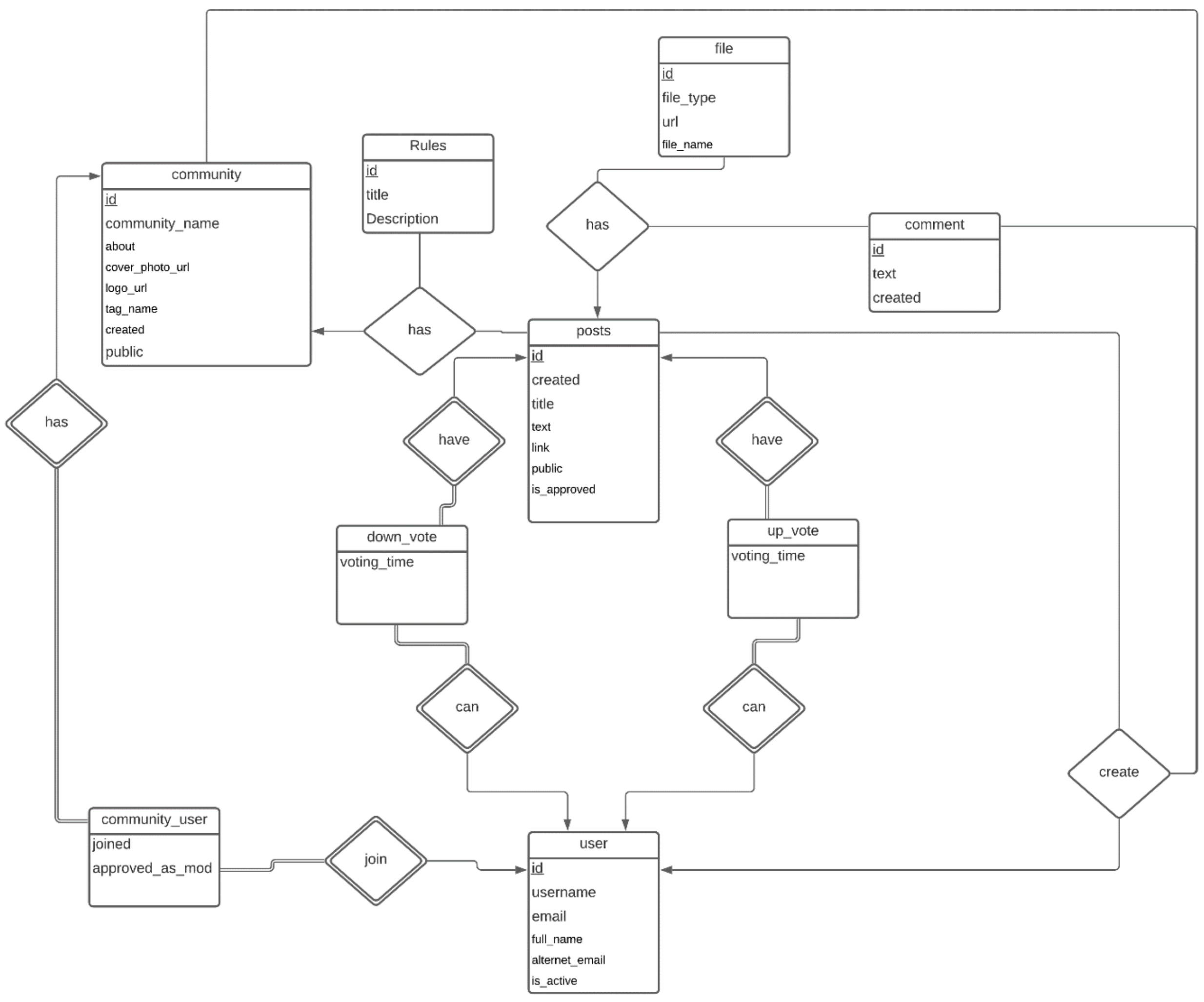
Requirement analysis:

1. User creation with email verification.
2. Users can login.
3. The user can visit and modify his profile.
4. The user can update their password.
5. The user can create communities.
6. The user can Search for community.
7. The user can see Community Details.
8. The user can create posts.
9. The user can add comments to a post.
10. There is a voting system for the posts.
11. The user will have a personal timeline where he can see all his posts.
12. Users can modify their own posts.
13. Admin can approve or delete all posts.
14. The Admin can modify community details.
15. Community moderator (Admin) can add rules and delete rules
16. The system will nominate the highest voted post of the day and display it on the home page.
17. User can logout

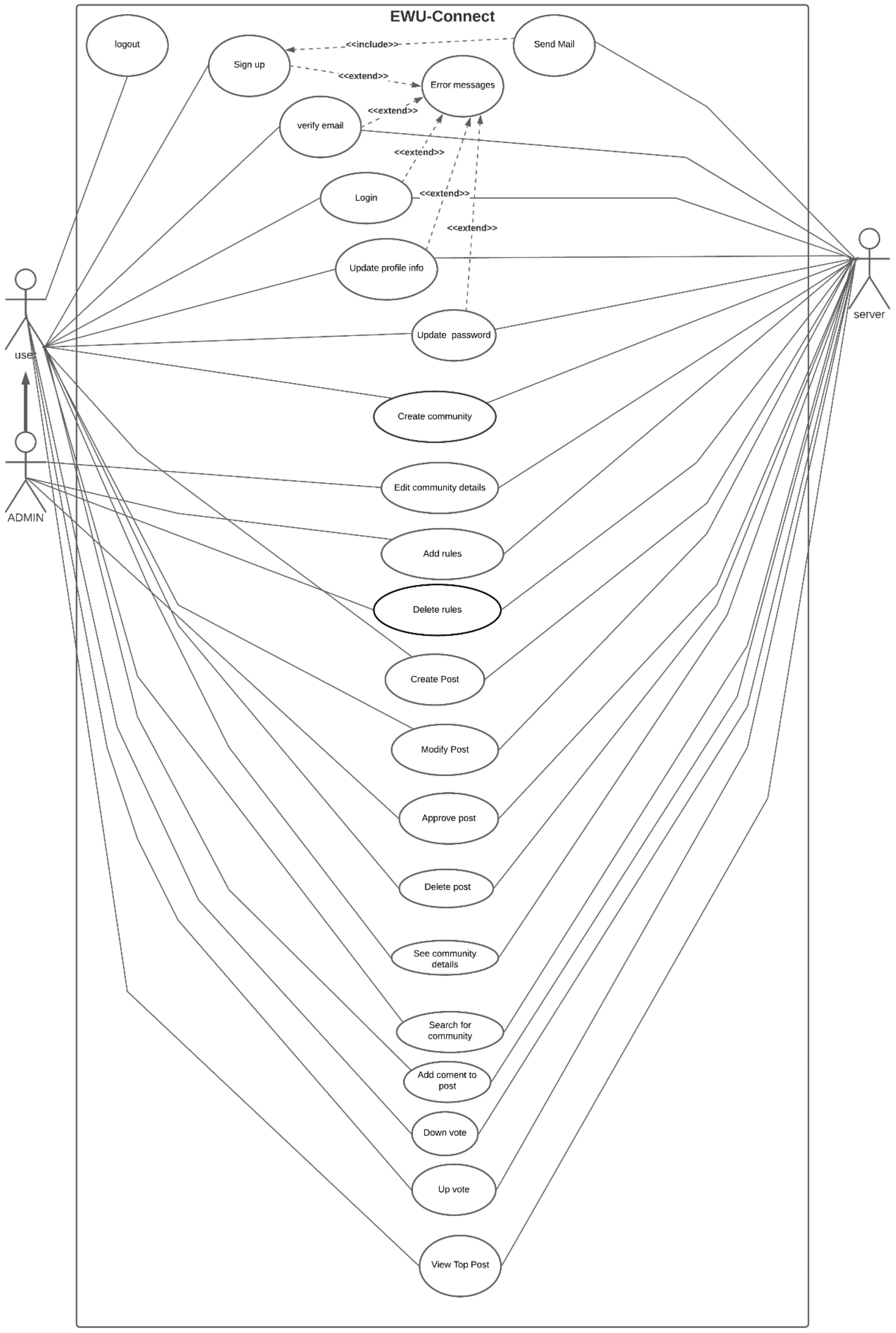
EWU Connect Class Diagram:



EWU Connect ER Diagram:



EWU Connect Use Case Diagram:



<u>EWU Connect Use Case Description:</u>

USE Case# 1	Sign up	
Goal in Context	This use case allowed user to register to EWU-connect	
Precondition	Anyone who is member of East West university can register with their EWU domain email.	
Success End Condition	User will receive a message to check their email.	
Failed End Condition	User won’t be receiving a message to check there email rather than a error message will be shown.	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user navigate to signup page	
Description	Step	Description
	1	User request for signup page
	2	User have to provide there EWU mail address and password submit it to the server
	3	system quarry the mail and password.
	4	The system shows an message that confirm the user is register.
Extension or Variations	step	Branching Action
	1	Invalid or existing email lead to error message.
	2	Send mail is an extension or sub use case

USE Case #2	Send mail	
Goal in Context	This use case allowed to send an email to the register user.	
Precondition	User have to complete signup process.	
Success End Condition	User will receive an email on their EWU mail for verification.	
Failed End Condition	User won’t get an email but a message will be shown.	
Primary Actor	User	
Secondary Actor	server	
Trigger	When signup process is complete.	
Description	Step	Description
	1	After signup process, email get extracted.
	2	A token is generated by encoding expired date and user identifier.
	3	System sends a mail to the extracted email address
Extension or Variations	step	Branching Action

USE Case #3	Verify email	
Goal in Context	This use case allowed the register user to verify and active their account.	
Precondition	User must be signup and a verification mail should be sent to his email.	
Success End Condition	User will automatically login and redirect to his profile page.	
Failed End Condition	User won’t get to redirect to profile page.	
Primary Actor	User	
Secondary Actor	server	
Trigger	User must click on a link that was email to him from the server.	
Description	Step	Description
	1	System decode the token form url and extracted user identifier and expired date of the token.
	2	System validates the expired date.
	3	System query the user identifier
	4	User gain access as login user.
	5	Redirect to user profile page
Extension or Variations	step	Branching Action
	1	Invalid token lead to an error message.

USE Case #4	Login	
Goal in Context	This use case allows the user to login to the website.	
Precondition	The EWU Connect server should be up and running	
Success End Condition	The user will successfully login to the website.	
Failed End Condition	The user will fail to login to the website	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user click login button after entering required information in the login page.	
Description	Step	Action
	1	User will go to the login page
	2	User will input required information.
	3	The user will click on the login button.
	4	The system will run queries in the database to see if the user exists and if the password matches.
	5	The user will be taken to index page signifying successful login.
Extension or Variations	Step	Branching Action
	1.	Error message will be shown

USE Case #5	Update Profile Information	
Goal in Context	This use case allows the user to update his personal profile information	
Precondition	User will have to login to the system	
Success End Condition	Profile information will be successfully updated	
Failed End Condition	Profile information will not be updated	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user click submit button after entering text in the edit profile section	
Description	Step	Action
	1	User will click on “Edit my info” section in the profile page
	2	User will input text in the Edit profile section.
	3	The user will click on the submit button.
	4	The information will be saved in the database
	5	The new information will be used for further requirements.
Extension or Variations	Step	Branching Action
	1.	Error message will be shown

USE Case #6	Update password	
Goal in Context	This use case allows the user to update his password.	
Precondition	User will have to login to the system	
Success End Condition	Profile password will be successfully updated	
Failed End Condition	Profile password will not be updated	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user click submit button after entering texts in the “Password Change” section.	
Description	Step	Action
	1	User will click on “Edit my info” section in the profile page
	2	User will input texts in the “Password Change” section.
	3	User will click Change Password button
	4	The system will check if the old password matches with the corresponding user information in the database.
	5	The system will check if the new password matches with the confirm password
	6	If all the previous steps are successful, new password will be saved in the database.
Extension or Variations	Step	Branching Action
	1.	Error message will be shown

USE Case #7	Create community	
Goal in Context	This use case allows the user to create a new community.	
Precondition	User will have to login to the system	
Success End Condition	A new community will be created successfully.	
Failed End Condition	New community will not be created.	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user clicks “create a new community” button in the index page.	
Description	Step	Action
	1	User will click “create a new community” button in the index page.
	2	User will be taken to a create community page.
	3	The user will provide required information and upload media files.
	4	The user will click on the create button
	5	The system will check if the new community has the same name as any existing community.
	6	If the name is unique, new community will be created and all information will be saved in the data base.
	7	The new community will be available for other users.
	8	The creator will become admin of the community
Extension or Variations	Step	Branching Action

USE Case #8	Edit Community Details	
Goal in Context	This use case allows the admin to edit community details.	
Precondition	User(admin) will have to login to the system	
Success End Condition	Community details will be edited and updated successfully.	
Failed End Condition	Community details will not be edited.	
Primary Actor	Admin	
Secondary Actor	Server	
Trigger	When admin clicks "edit" button in the corresponding community page.	
Description	Step	Action
	1	The admin will click "edit" button in the corresponding community page.
	2	The admin will be taken to an update community page.
	3	The admin will be allowed to update required information and upload media files.
	4	The user will click on the create button
	5	The system will save the new information in the database
	6	The community will display the new information for further use.
Extension or Variations	Step	Branching Action

USE Case #9	Add rules	
Goal in Context	This use case allows the admin to add rules to the community.	
Precondition	User(admin) will have to login to the system	
Success End Condition	Rules will be added successfully.	
Failed End Condition	New rules will not be added.	
Primary Actor	Admin	
Secondary Actor	Server	
Trigger	When admin clicks “Add rules” button in the corresponding community page.	
Description	Step	Action
	1	The admin will click “edit” button in the corresponding community page.
	2	The admin will be taken to an “add rules” page.
	3	The admin will input rules tile and rule details. He can add multiple rules at a time
	4	The admin will click on the add button
	5	The system will save the new rules in the database
	6	The community will display the new rules along with the existing ones in the rules section.
Extension or Variations	Step	Branching Action

USE Case #10	Delete rules	
Goal in Context	This use case allows the admin to delete rules from the community.	
Precondition	User(admin) will have to login to the system	
Success End Condition	Rules will be deleted successfully.	
Failed End Condition	Rules will not be deleted.	
Primary Actor	Admin	
Secondary Actor	Server	
Trigger	When admin clicks “Delete” button in the corresponding rule in the rules page.	
Description	Step	Action
	1	The admin will click “rules” button in the corresponding community page.
	2	The admin will be taken to an “rules” page.
	3	The admin will select the rule he wants to delete
	4	The admin will click on the corresponding delete button.
	5	The system will delete the specific rule from the database.
	6	The community will display the remaining rules in the rules section.
Extension or Variations	Step	Branching Action

USE Case #11	Create post	
Goal in Context	This use case allows users to create post	
Precondition	User will have to login to the system.	
Success End Condition	A new post will be created	
Failed End Condition	New post will not be created	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user clicks on “Create Post” in the index page or in any community page.	
Description	Step	Action
	1	The user will click on “Create Post” in the index page or in any community page
	2	The user will be taken to the “create post here” page.
	3	The user will input required information and upload media file.
	4	The user will click on Submit button
	5	A new post will be created in the specified community
	6	The post will appear both in the community and in the timeline of the user who posted it.
Extension or Variations	Step	Branching Action

USE Case #12	Modify post	
Goal in Context	Everything in the post will be able to be modified	
Precondition	Users who have created post previously will be able to update all the element	
Success End Condition	Post will be updated	
Failed End Condition	Old post content will remain	
Primary Actor	Users who have created the post	
Secondary Actor	Server	
Trigger	When user will update the information of the post	
Description	Step	Action
	1	User who have created the post will click on edit button
	2	Will add new items and click on updated
	3	New items will be added in the database and pervious item will be cleared
Extension or Variations	Step	Branching Action

USE Case #13	Approve post	
Goal in Context	The post has been made by user will be approved in a community by the admin	
Precondition	User has admin access of the group	
Success End Condition	Post will be posted in the group	
Failed End Condition	Post will not be published	
Primary Actor	Admin	
Secondary Actor	Server, User	
Trigger	When user click on approve button	
Description	Step	Action
	1	Admin will click on approve button of the post made by the general user
	2	System will check if the person has clicked for approval has admin access
	3	The post will be available on the timeline of the group
Extension or Variations	Step	Branching Action

USE Case #14	Delete Post	
Goal in Context	This use case allows the user to delete a post made by him	
Precondition	User will have to login to the system and the post he wants to delete has to be created by him	
Success End Condition	User will be able to delete the post successfully	
Failed End Condition	Post will remain in the community	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user click on delete button	
Description	Step	Action
	1	User will click on delete button
	2	System will check if the post was created by him
	3	All post related information will be deleted from the system
Extension or Variations	Step	Branching Action

USE Case #15	See community detail	
Goal in Context	This use case allows the user to see community details including the community member	
Precondition	User will have to login to the system	
Success End Condition	See community details and member information	
Failed End Condition	No information will be showed	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user visit on a community	
Description	Step	Action
	1	User will see information if he enters in a community page
	2	System will also show the number of members in the group on a button
	3	When user click on the button, they will get the list of all the member
Extension or Variations	Step	Branching Action

USE Case #16	Search for community	
Goal in Context	This use case allows the user to search community by name	
Precondition	User will have to login to the system	
Success End Condition	User will be able to find out the community he is looking for	
Failed End Condition	Searching will not work	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user click on search button	
Description	Step	Action
	1	User will write a key word in the search bar
	2	User will click on search button
	3	System will search in the database for groups that match with the key word
	4	The system shows the match result with full group name
Extension or Variations	Step	Branching Action

USE Case #17	Add Comment to post	
Goal in Context	This use case allows the user to comment in a post of a community	
Precondition	User will have to login to the system	
Success End Condition	New comment will be added with post	
Failed End Condition	No new comment with post	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user click submit button after entering text	
Description	Step	Action
	1	User will write a comment in a post
	2	User will click on submit button
	3	The comment will be saved on the database
	4	All users will be able to see the comment with the post
Extension or Variations	Step	Branching Action

USE Case #18	Down Vote	
Goal in Context	This use case allows the user to give down vote in a post of a community	
Precondition	User will have to login to the system	
Success End Condition	Number of total down vote of a post will increase	
Failed End Condition	Total down vote of a post will be unchanged	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user click on down vote button	
Description	Step	Action
	1	User will click on down vote
	2	System will check login status of user
	3	System will cancel the previous up or down vote in the post if user have already voted
	4	The system will increase the total down vote once
	5	New total vote will be shown in the post
Extension or Variations	Step	Branching Action

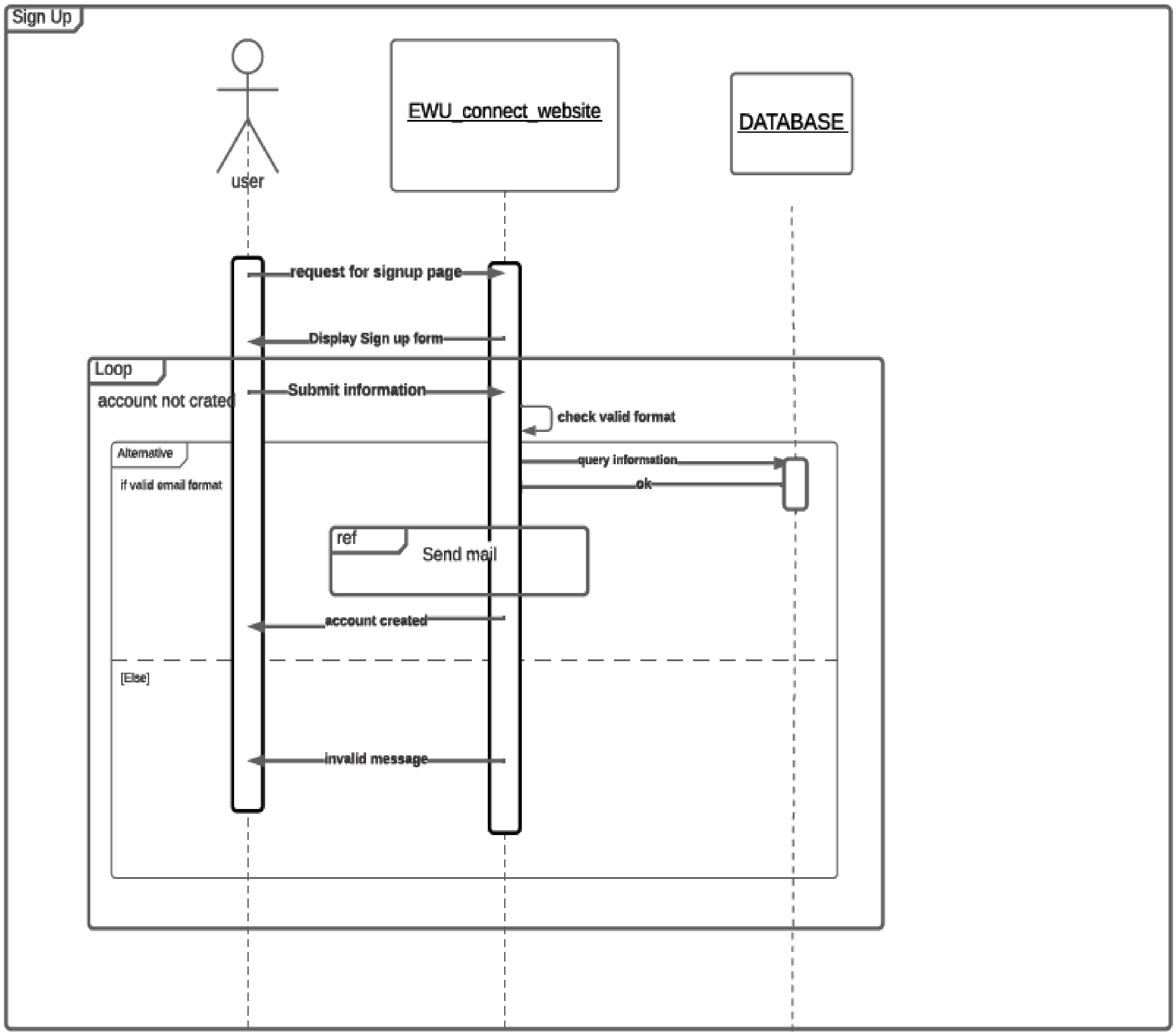
USE Case #19	Up Vote	
Goal in Context	This use case allows the user to give up vote in a post of a community	
Precondition	User will have to login to the system	
Success End Condition	Number of total up vote of a post will increase	
Failed End Condition	Total up vote of a post will unchanged	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user click on up vote button	
Description	Step	Action
	1	User will click on up vote
	2	System will check login status of user
	3	System will cancel the previous up or down vote in the post if user hve already voted
	4	The system will increase the total up vote once
	5	New total vote will be shown in the post
Extension or Variations	Step	Branching Action

USE Case #20	View top post	
Goal in Context	This use case allows the user to view the post with most upvotes the website.	
Precondition	The EWU Connect server should be up and running	
Success End Condition	The user will successfully view the top post.	
Failed End Condition	The top post will not be displayed.	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user clicks the “EWU Connect” button on the navigation bar.	
Description	Step	Action
	1	User will click on the “EWU Connect” button
	2	The user will be taken to the website’s index page
	3	The system will display the top post in a dedicated section on the page.
Extension or Variations	Step	Branching Action

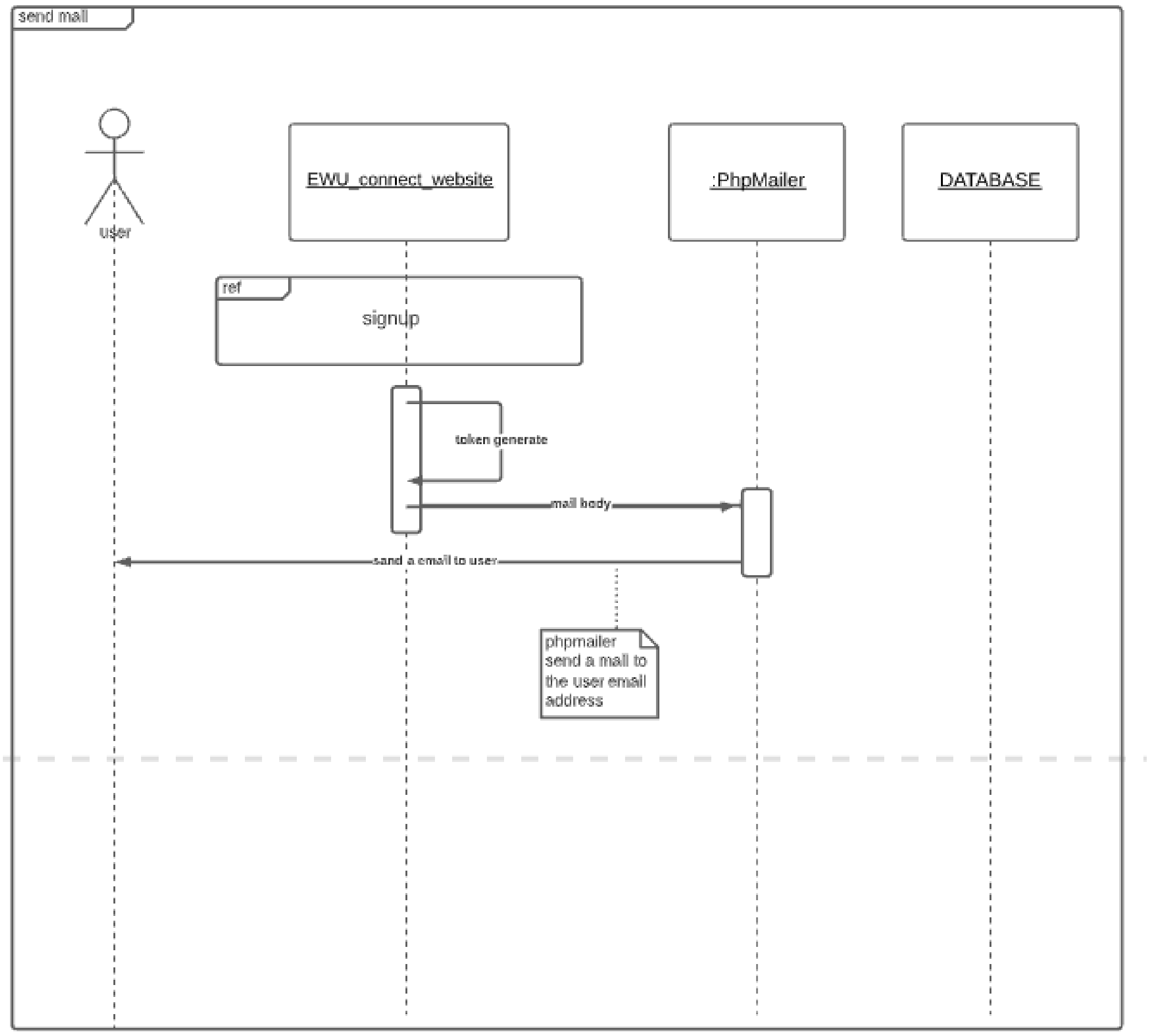
USE Case #21	Logout	
Goal in Context	This use case allows the user to logout from the website.	
Precondition	The EWU Connect server should be up and running	
Success End Condition	The user will successfully logout from the website.	
Failed End Condition	The user will fail to logout from the website	
Primary Actor	User	
Secondary Actor	Server	
Trigger	When user clicks logout button on the navigation bar.	
Description	Step	Action
	1	User will click on the logout button
	2	The system will end the user’s session
	3	The user will be taken to the login page
Extension or Variations	Step	Branching Action

Ewu Connect Sequence Diagram:

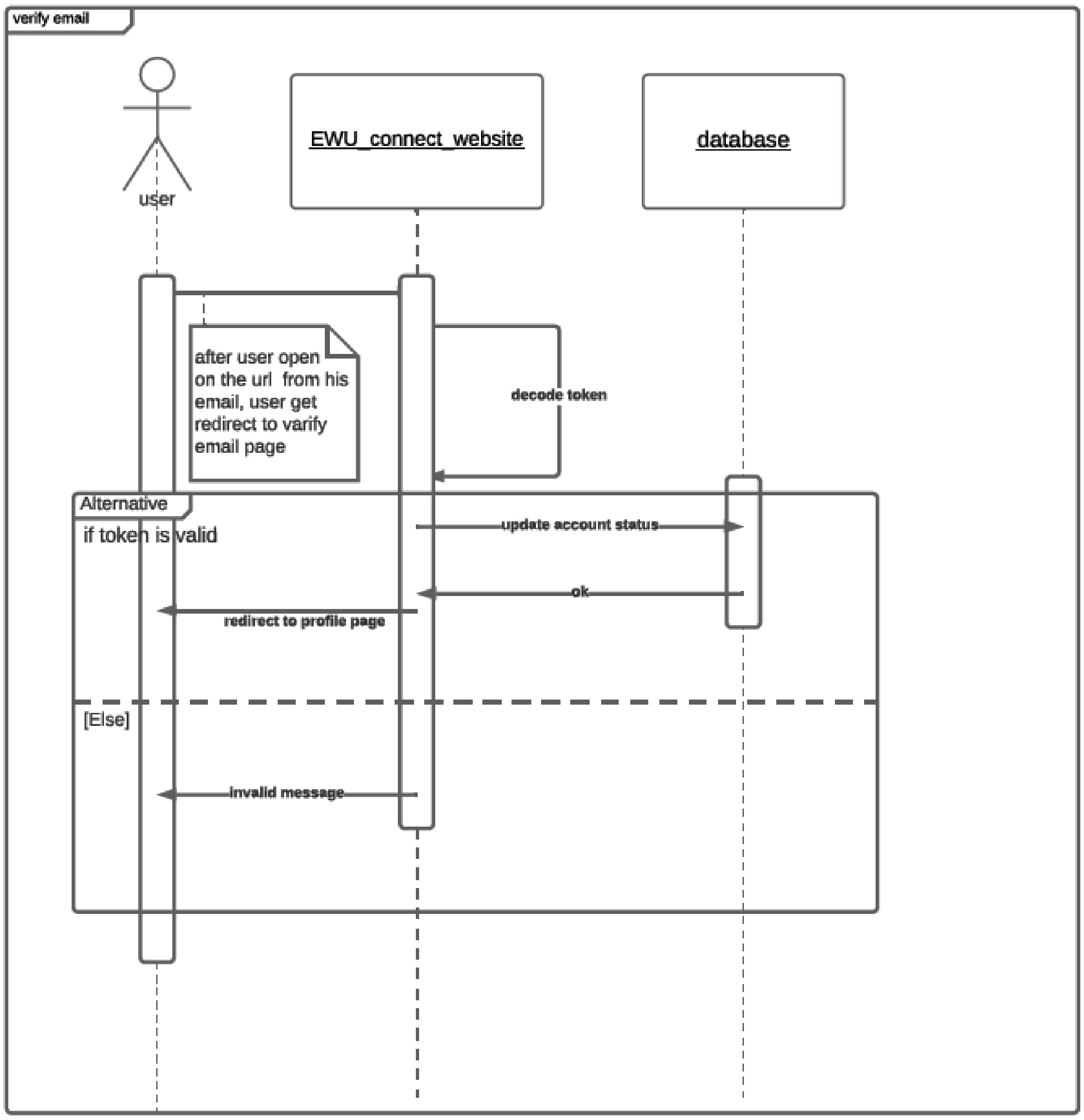
1) **Signup:**



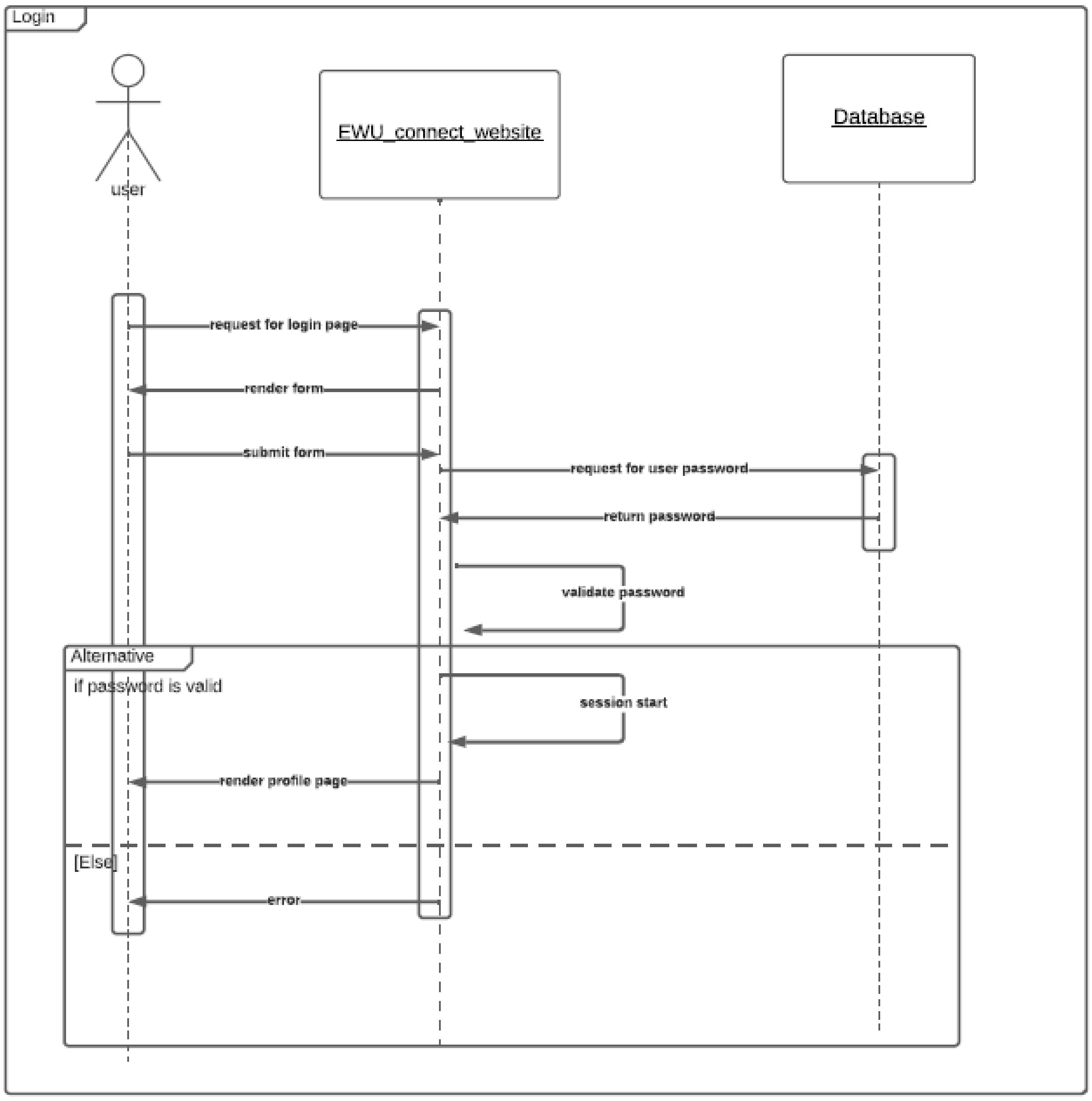
2) Send Mail


send mail
user
EWU_connect_website
:PhpMailer
DATABASE
ref
signup
token generate
mail body
send a email to user
phpmailer send a mail to the user email address

3) Verify Email:

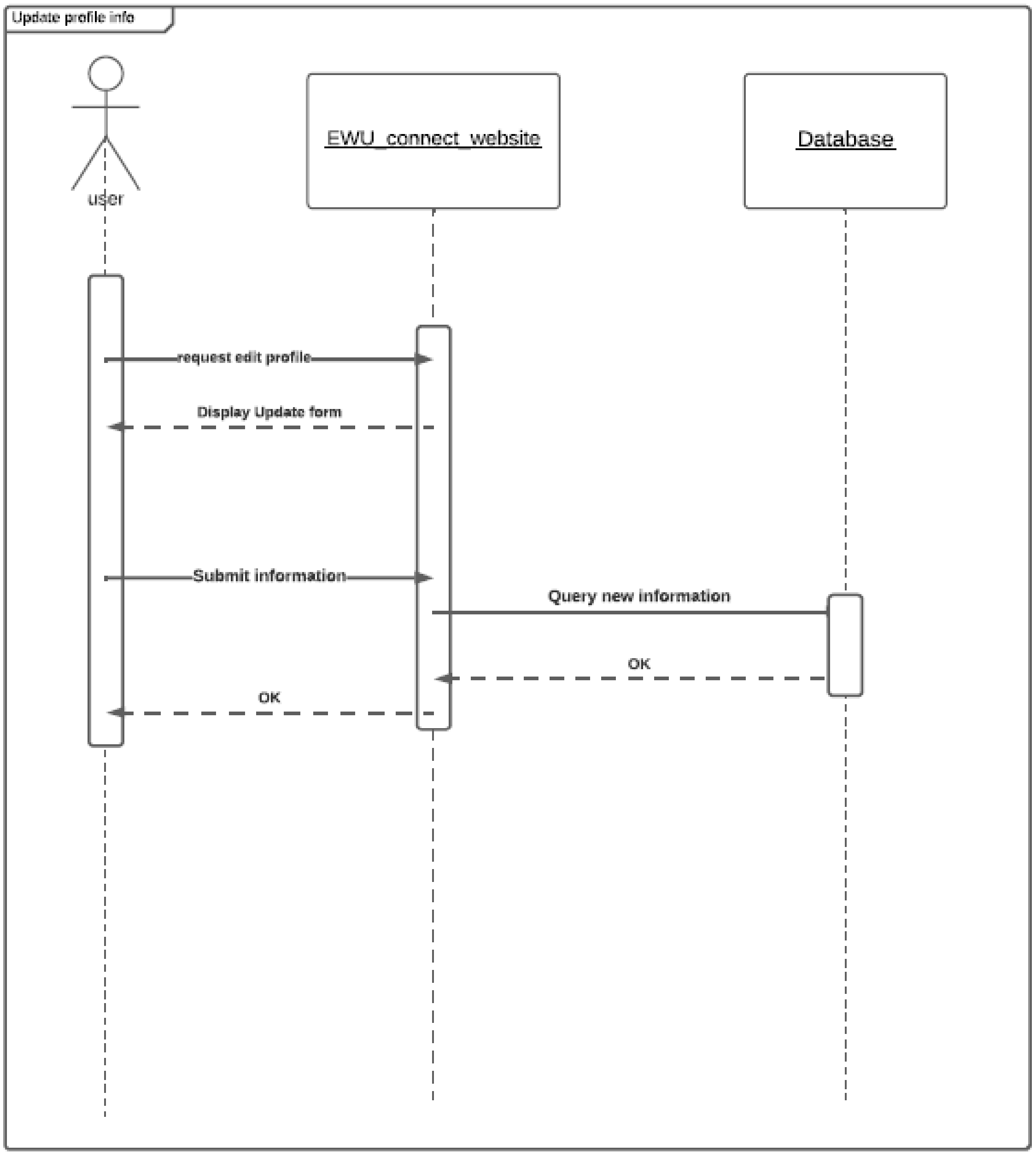


4) Login:

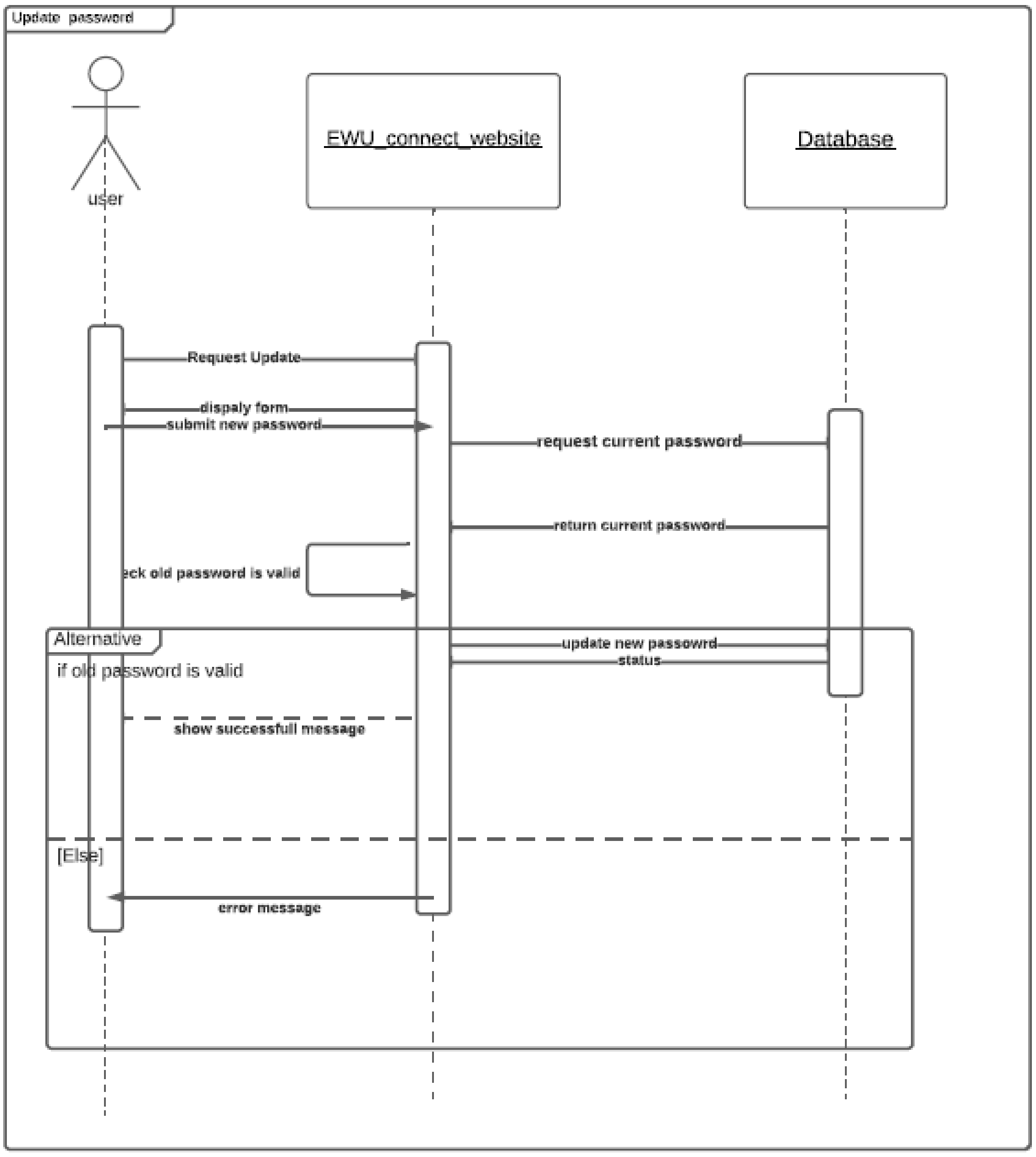


Login
user
EWU_connect_website
Database
request for login page
render form
submit form
request for user password
return password
validate password
Alternative
if password is valid
session start
render profile page
[Else]
error

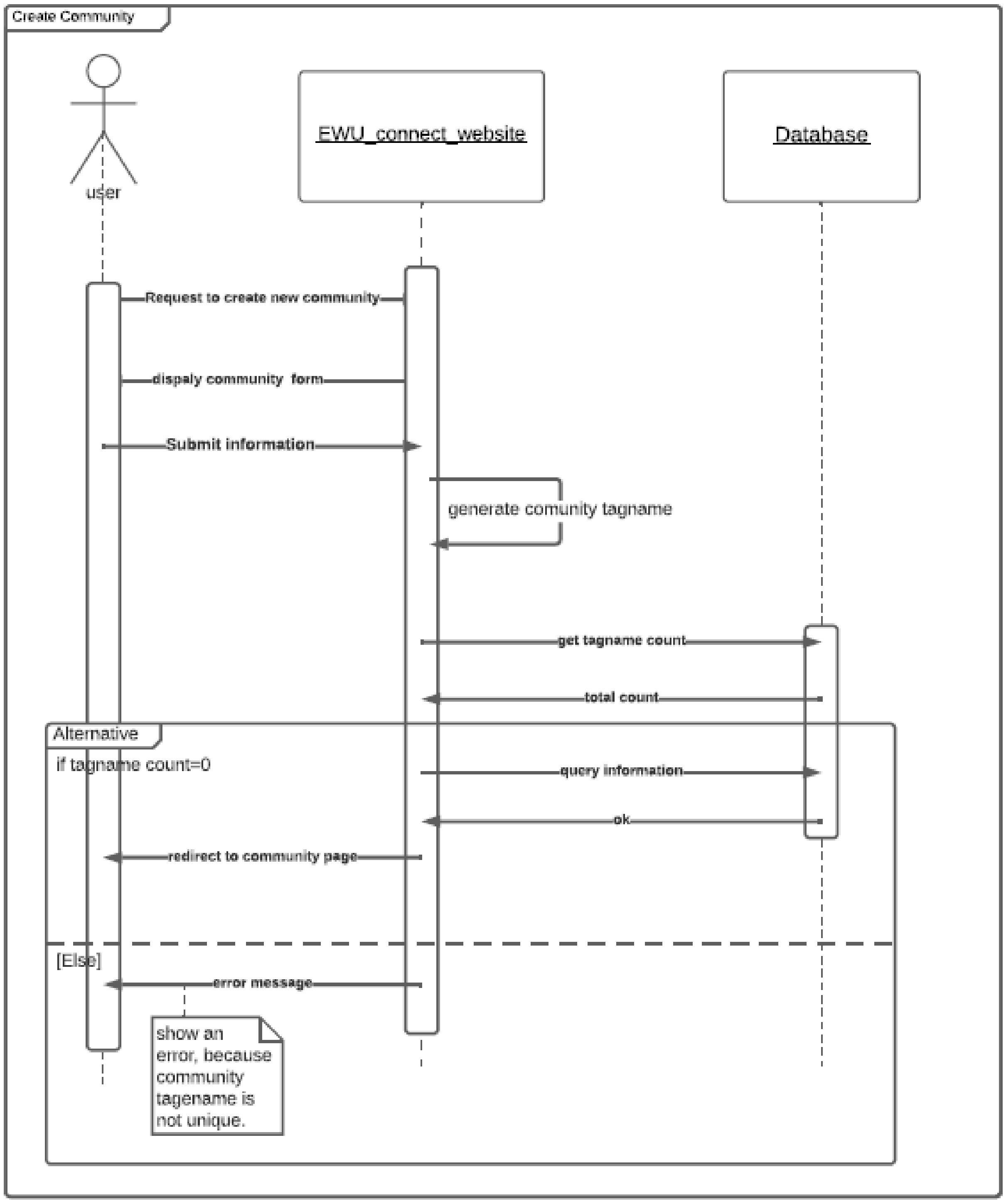
5) Update Profile Information:



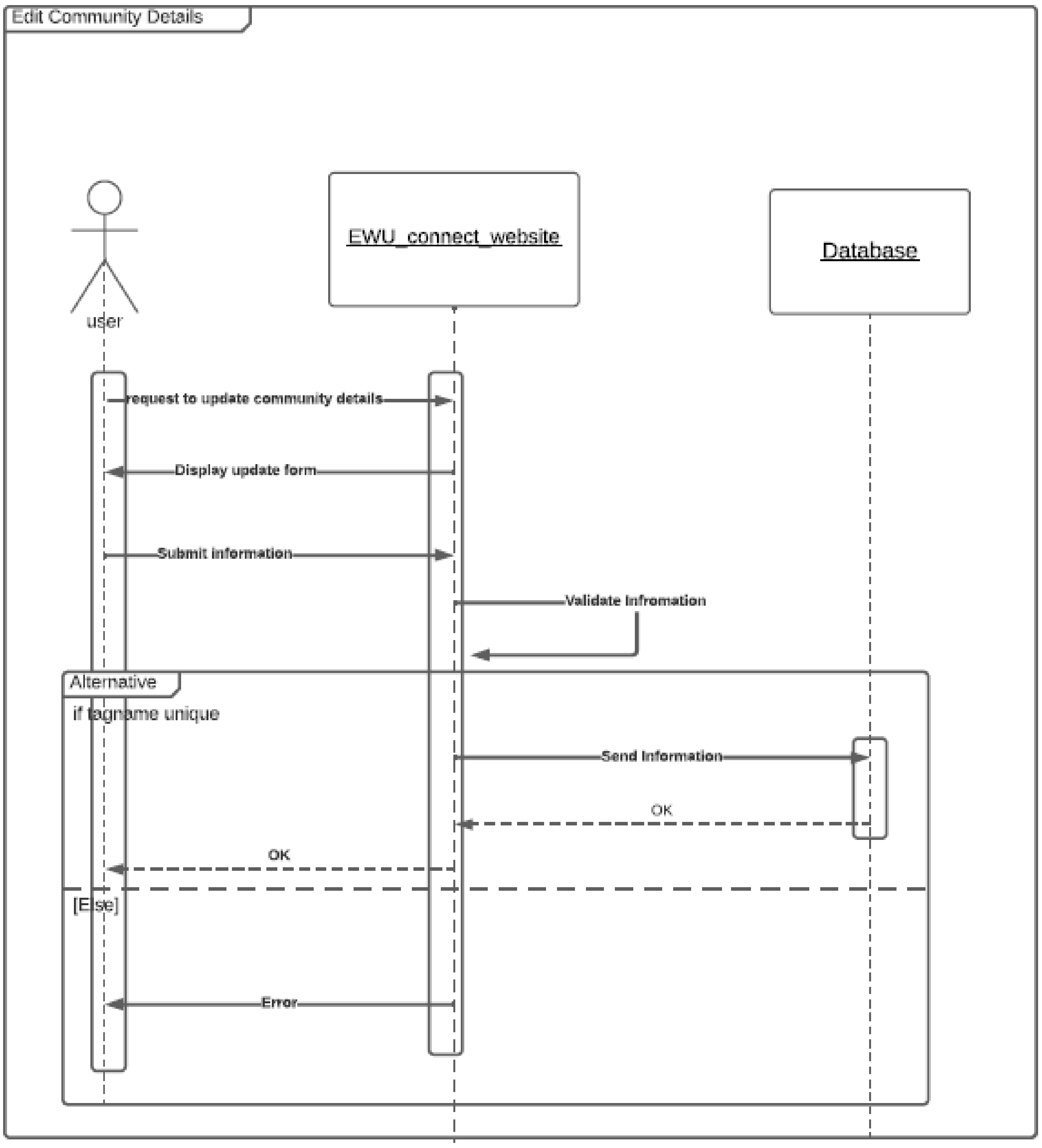
6) Update password:



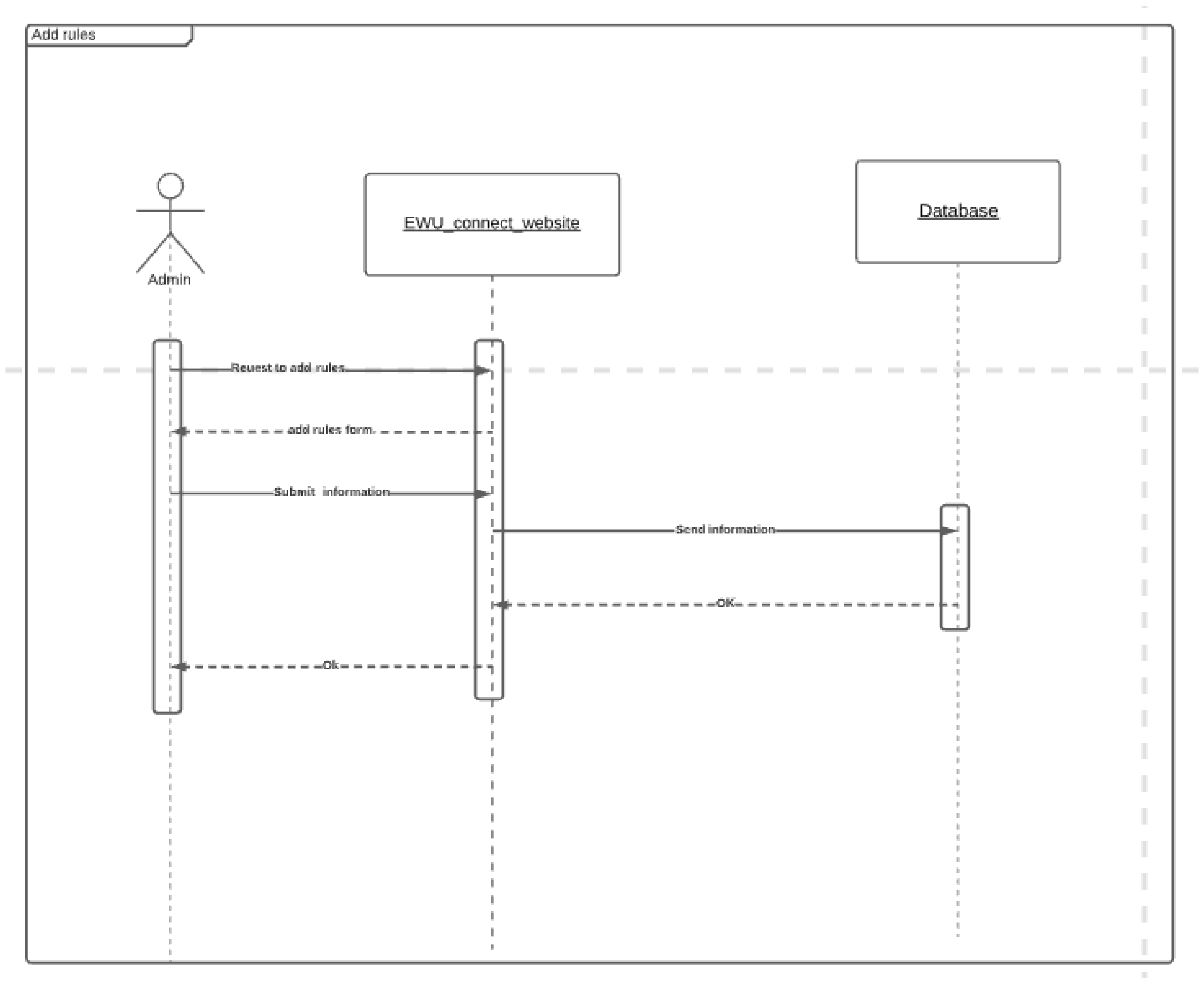
7) Create community



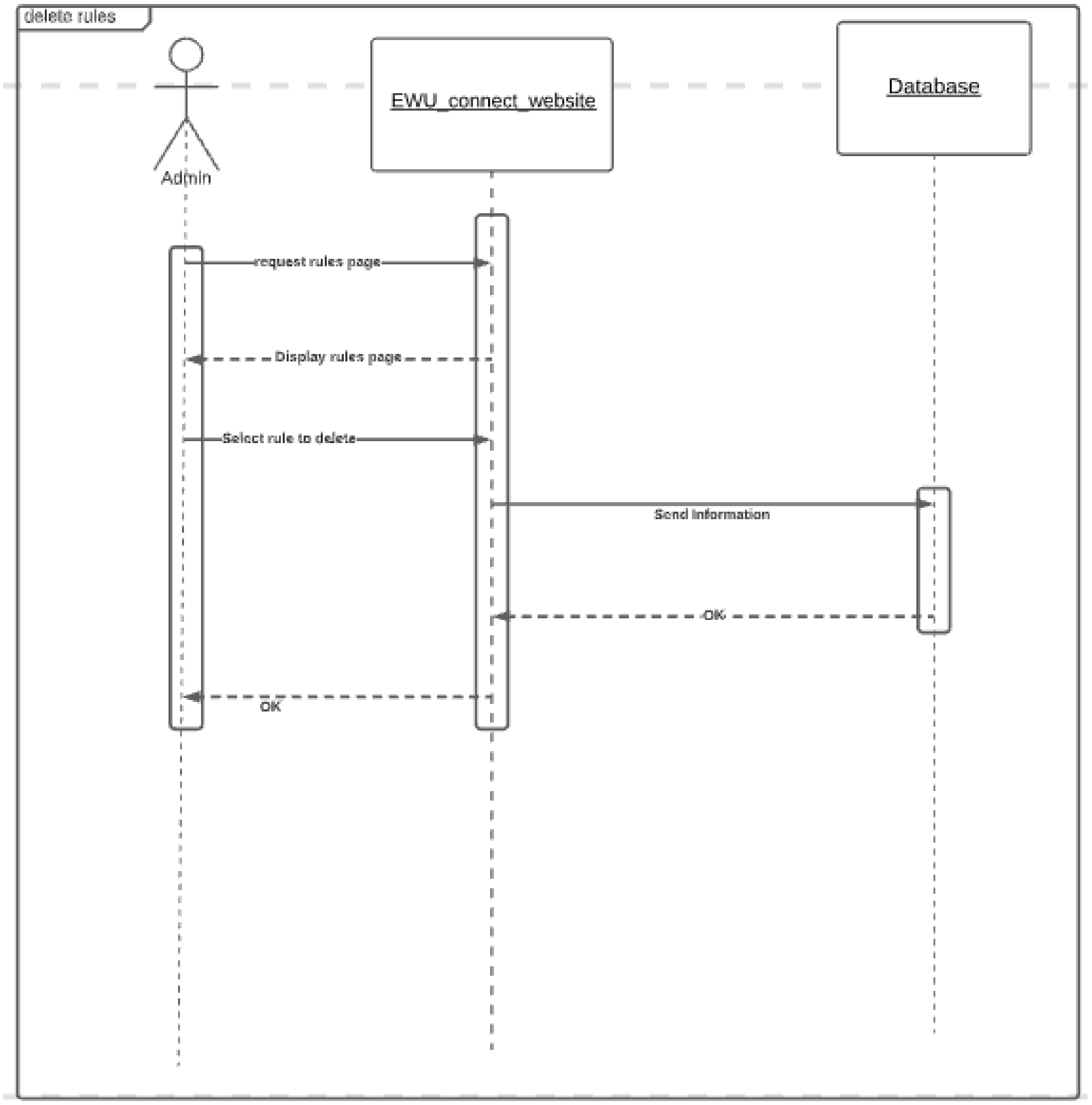
8) Edit Community Details:



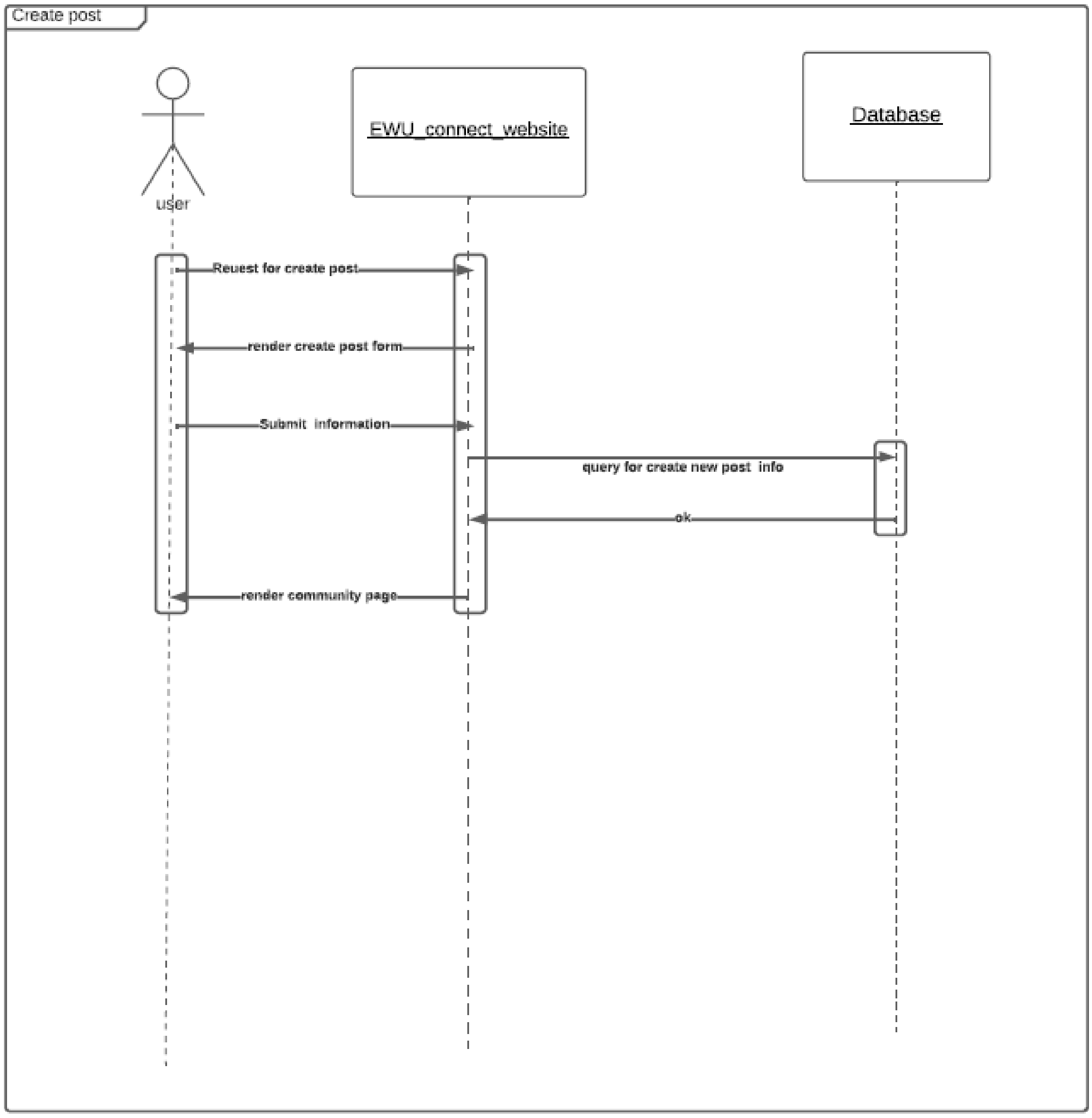
9) Add rules:


Add rules
Admin
EWU_connect_website
Database
Reuest to add rules
add rules form
Submit information
Send information
OK
Ok

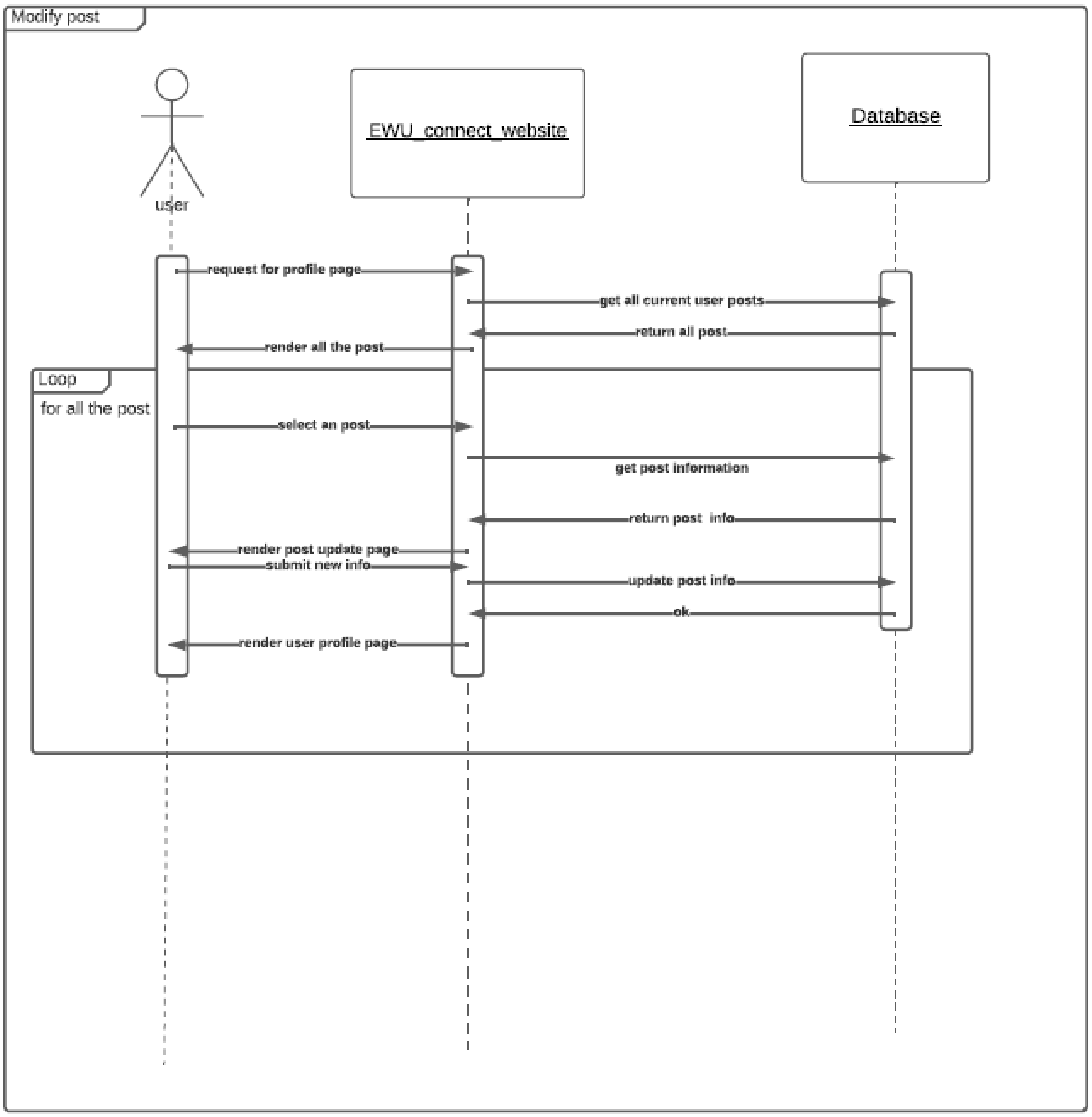
10) **Delete rules:**



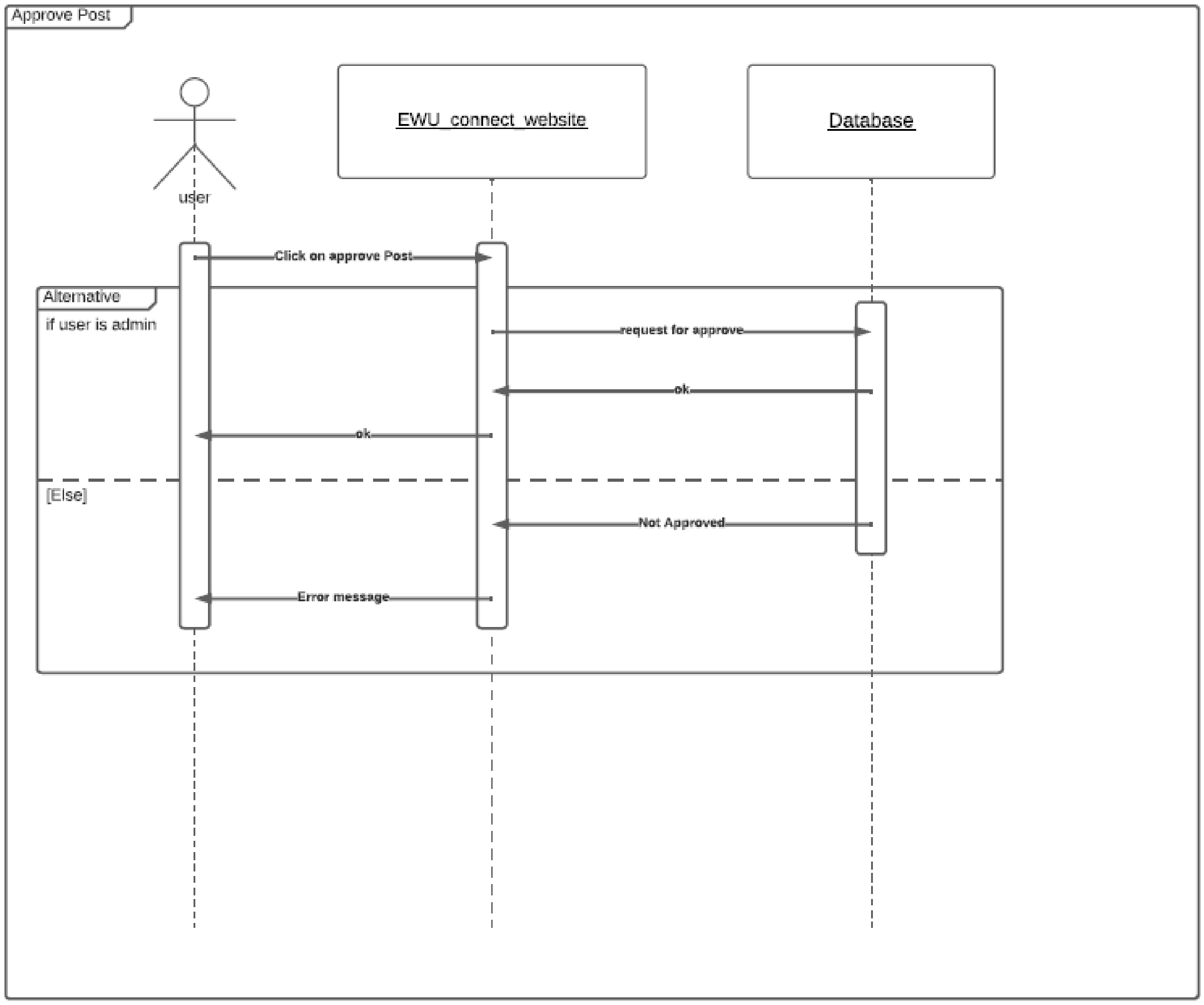
11) Create post:



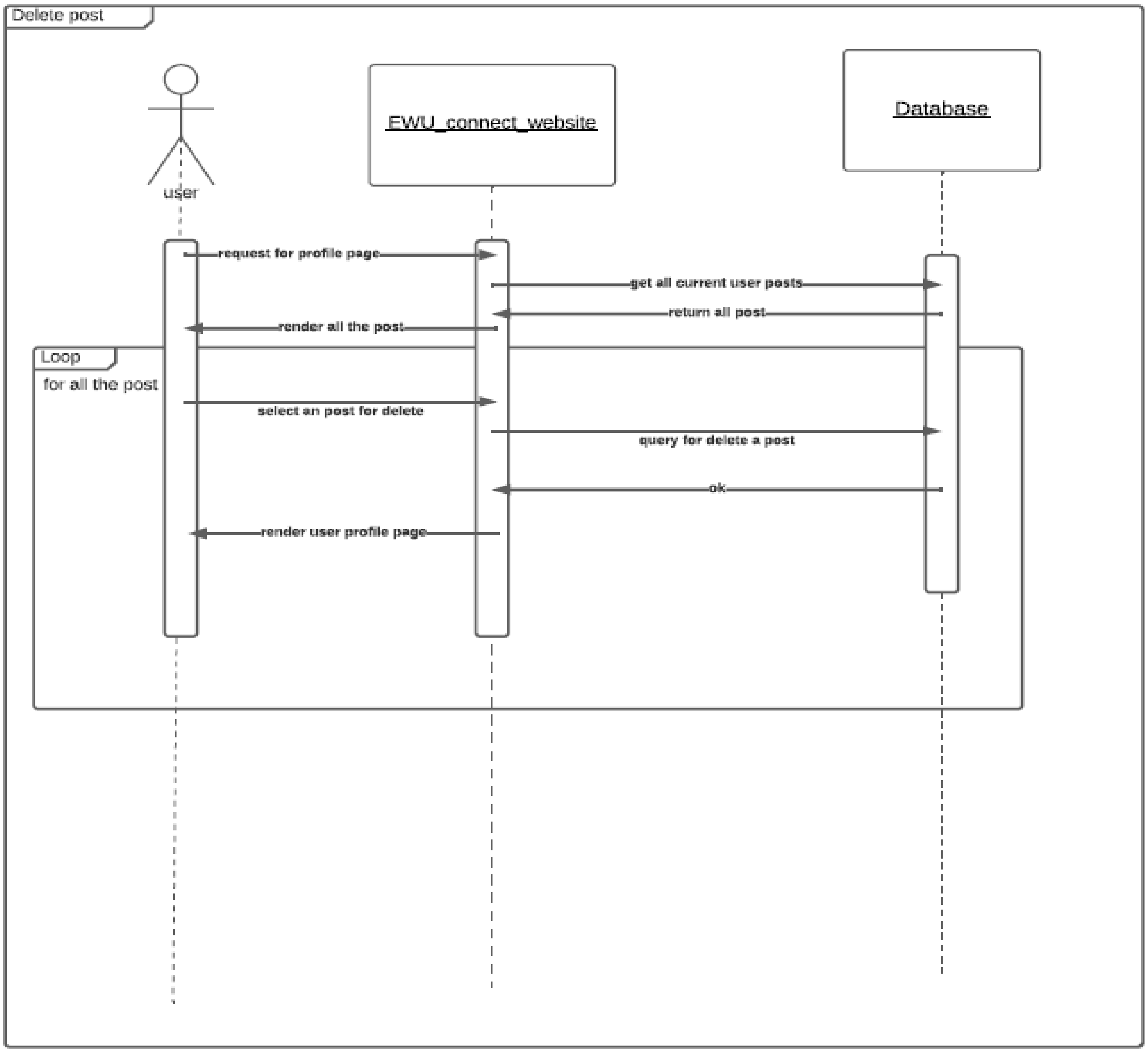
12) Modify Post:



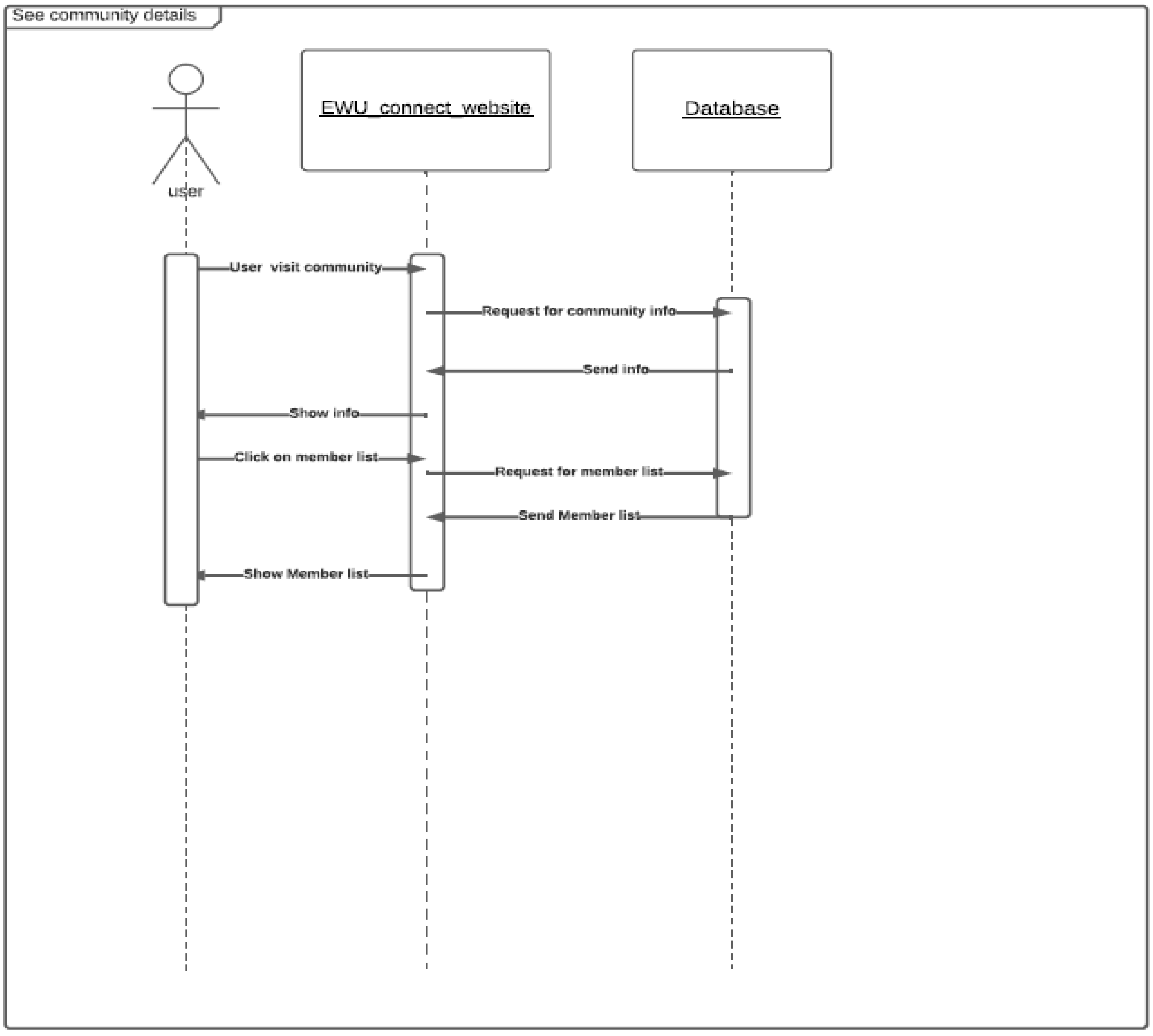
13) Approve post:



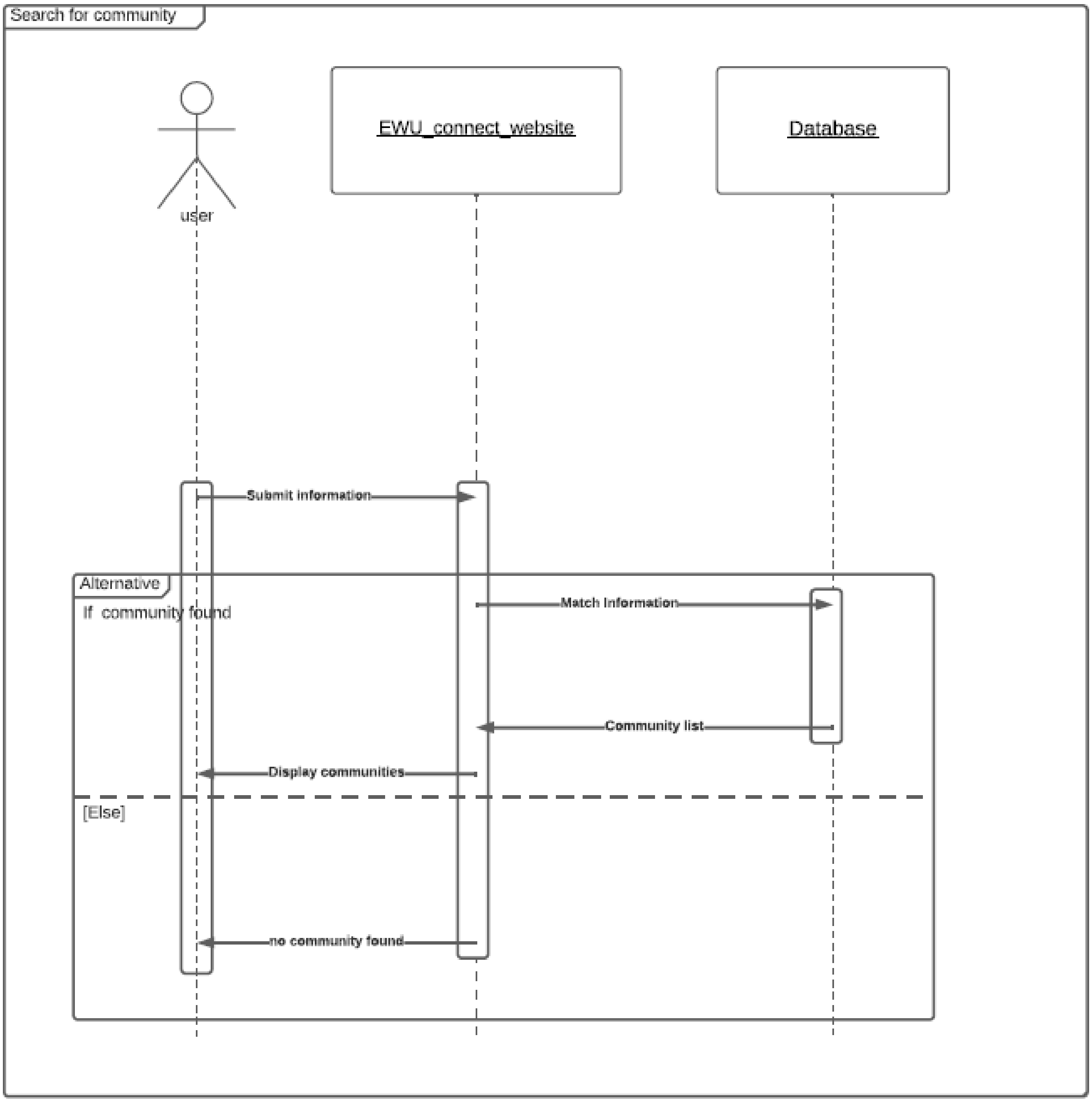
14) **Delete Post:**



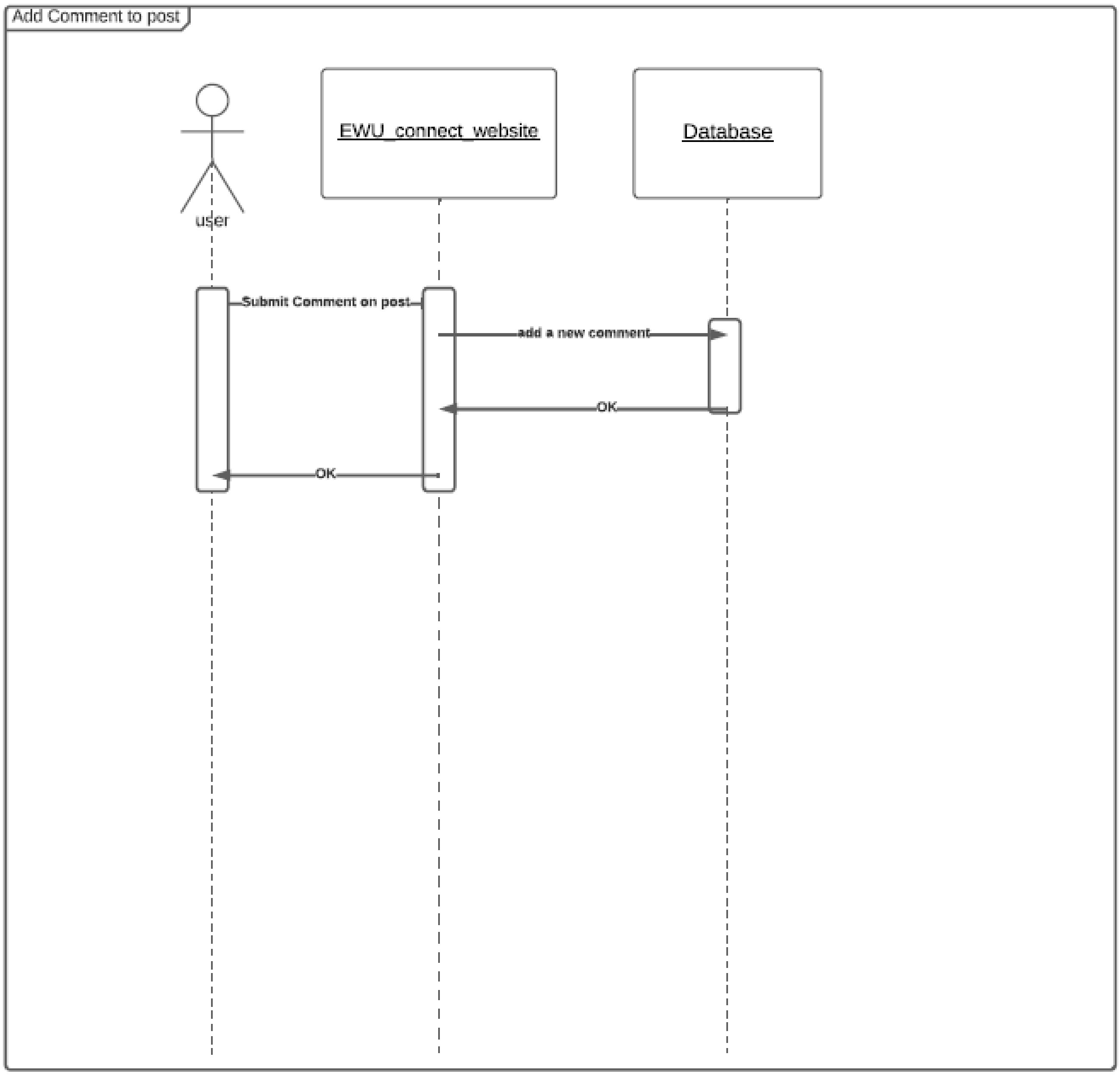
15) **<u>See community details:</u>**



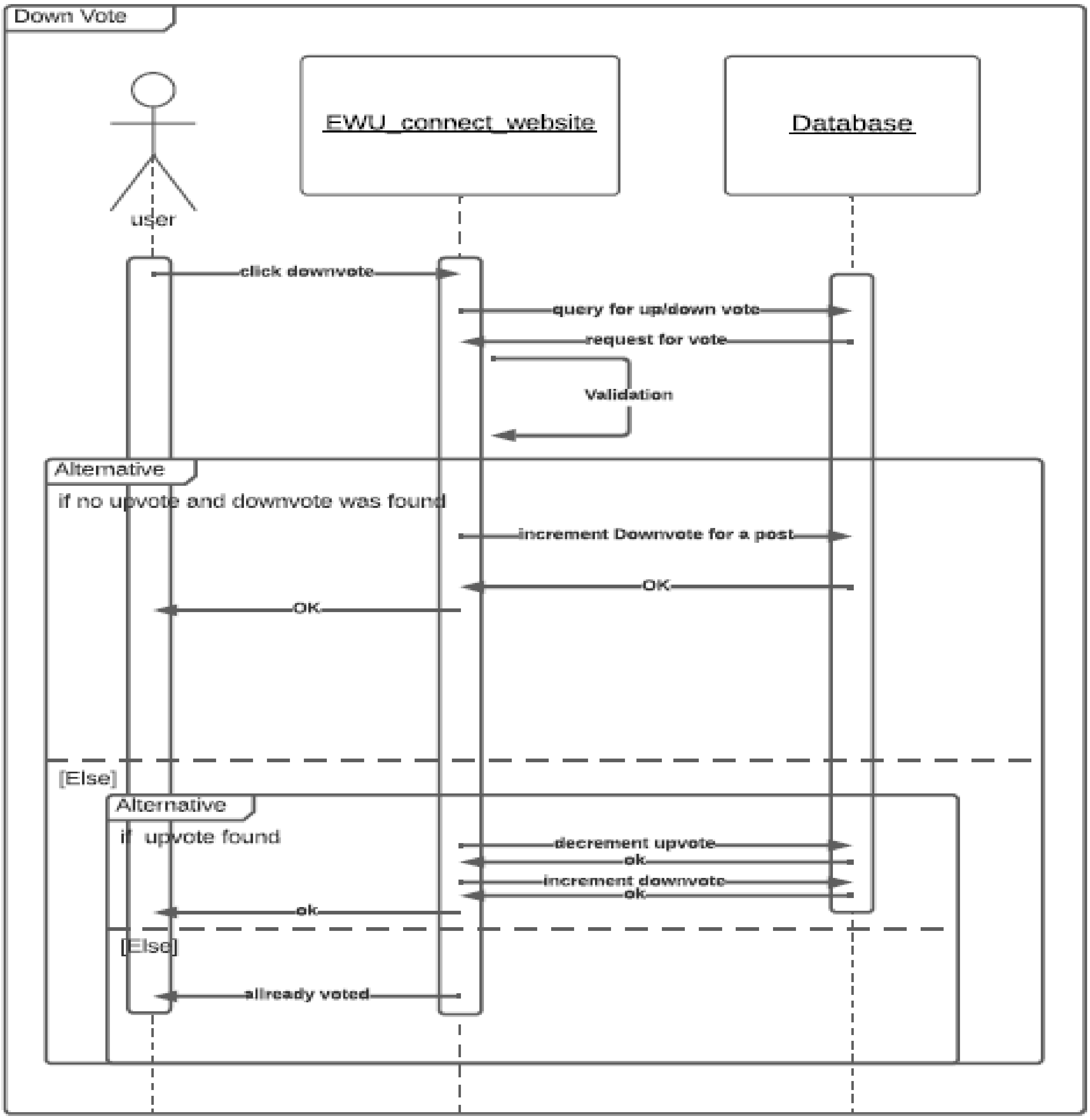
16) Search for community:



17) Add comment to post:

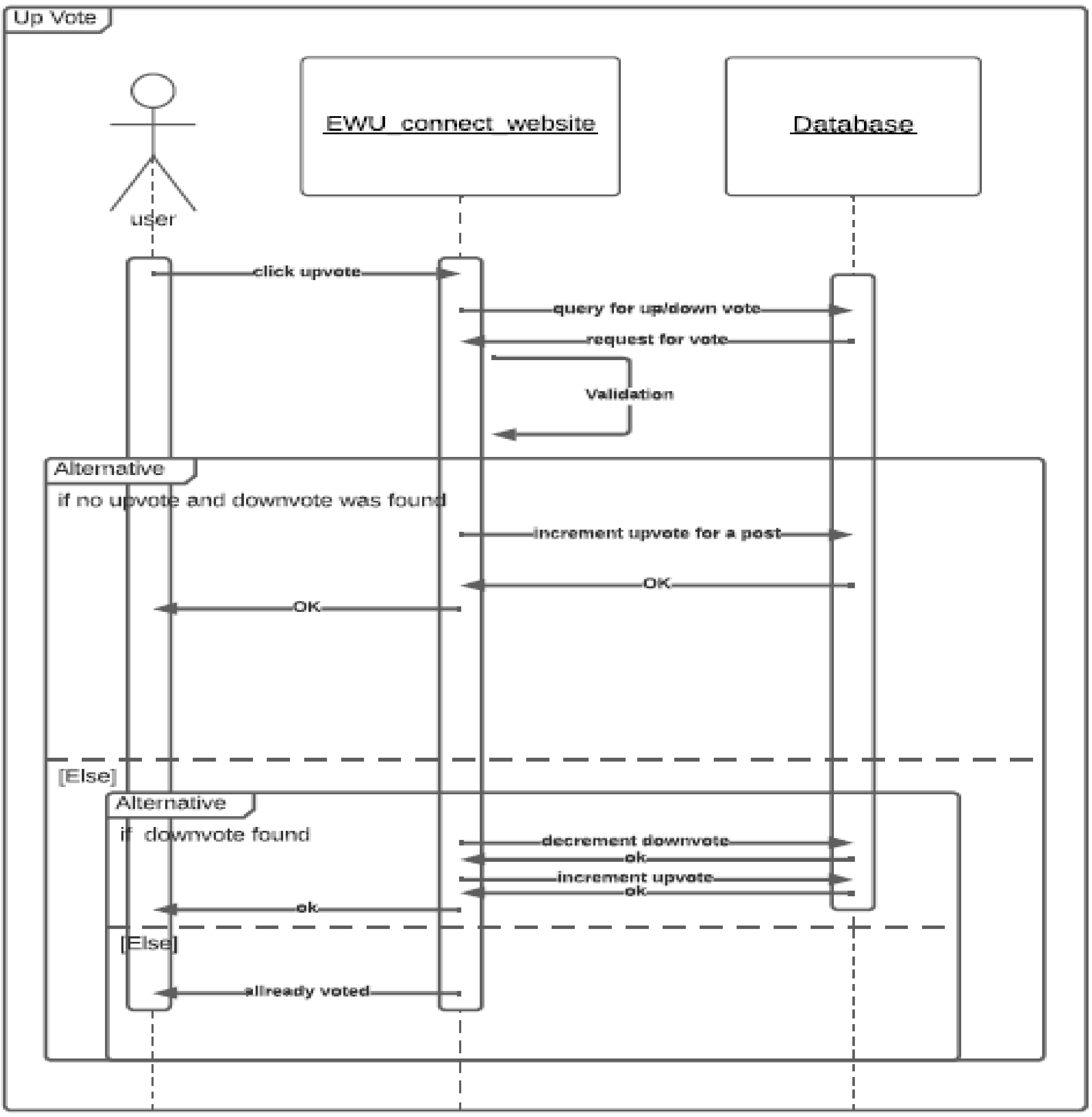


18) Down Vote:

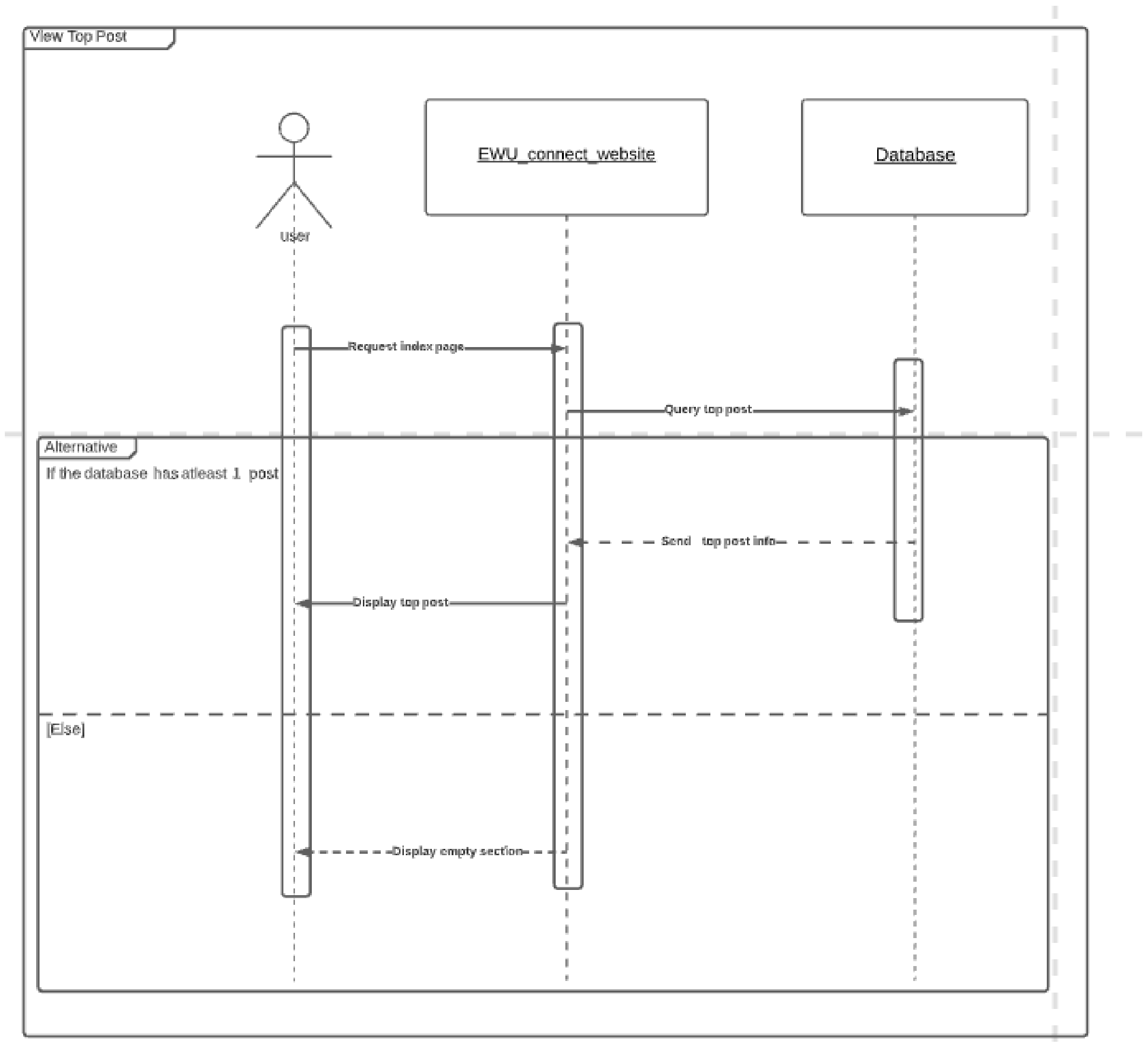


Down Vote
user
EWU_connect_website
Database
click downvote
query for up/down vote
request for vote
Validation
Alternative
if no upvote and downvote was found
increment Downvote for a post
OK
OK
[Else]
Alternative
if upvote found
decrement upvote
ok
increment downvote
ok
ok
[Else]
allready voted

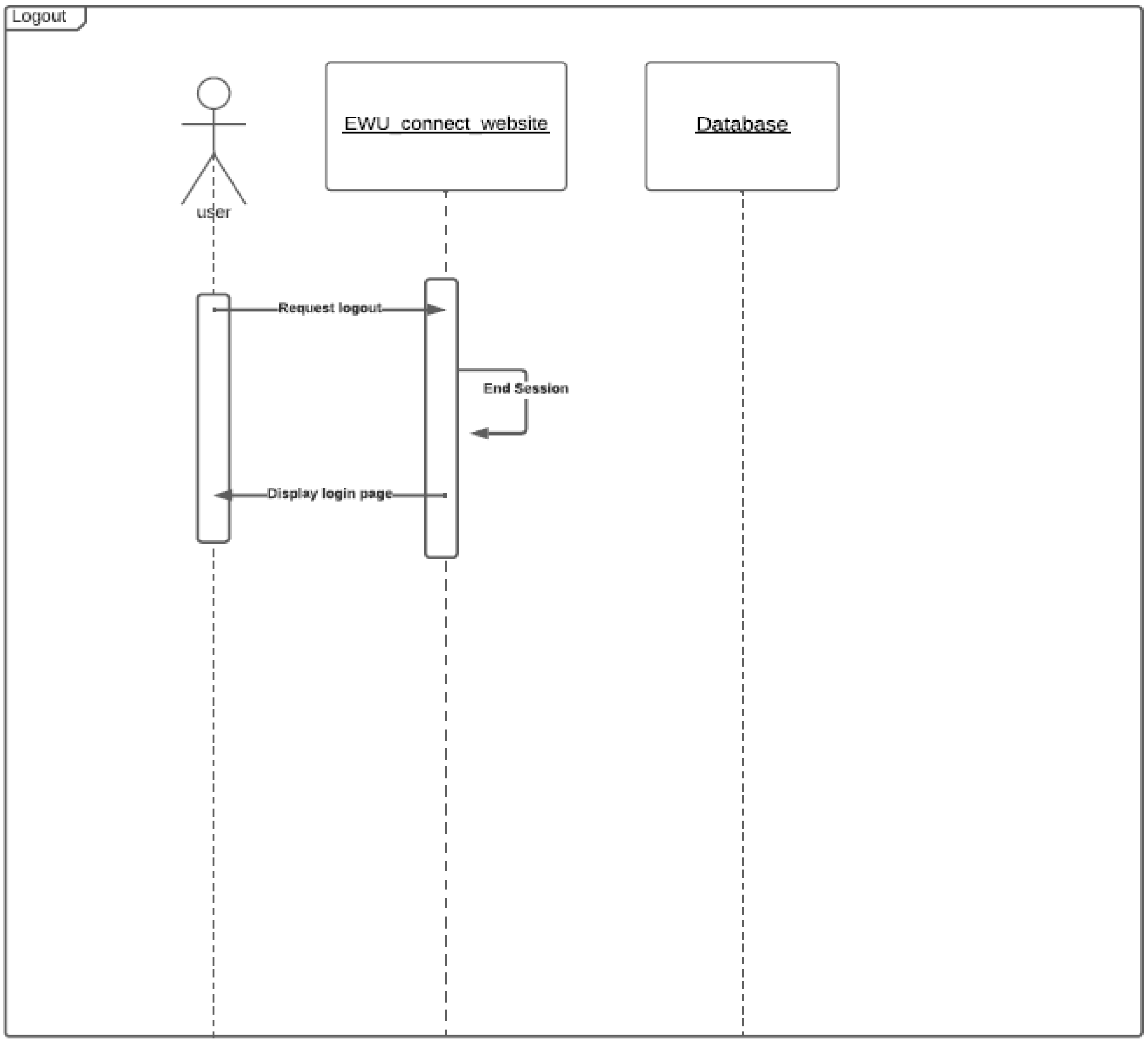
19) **Up Vote:**


Up Vote
user
EWU_connect_website
Database
click upvote
query for up/down vote
request for vote
Validation
Alternative
if no upvote and downvote was found
increment upvote for a post
OK
OK
[Else]
Alternative
if downvote found
decrement downvote
ok
increment upvote
ok
ok
[Else]
allready voted

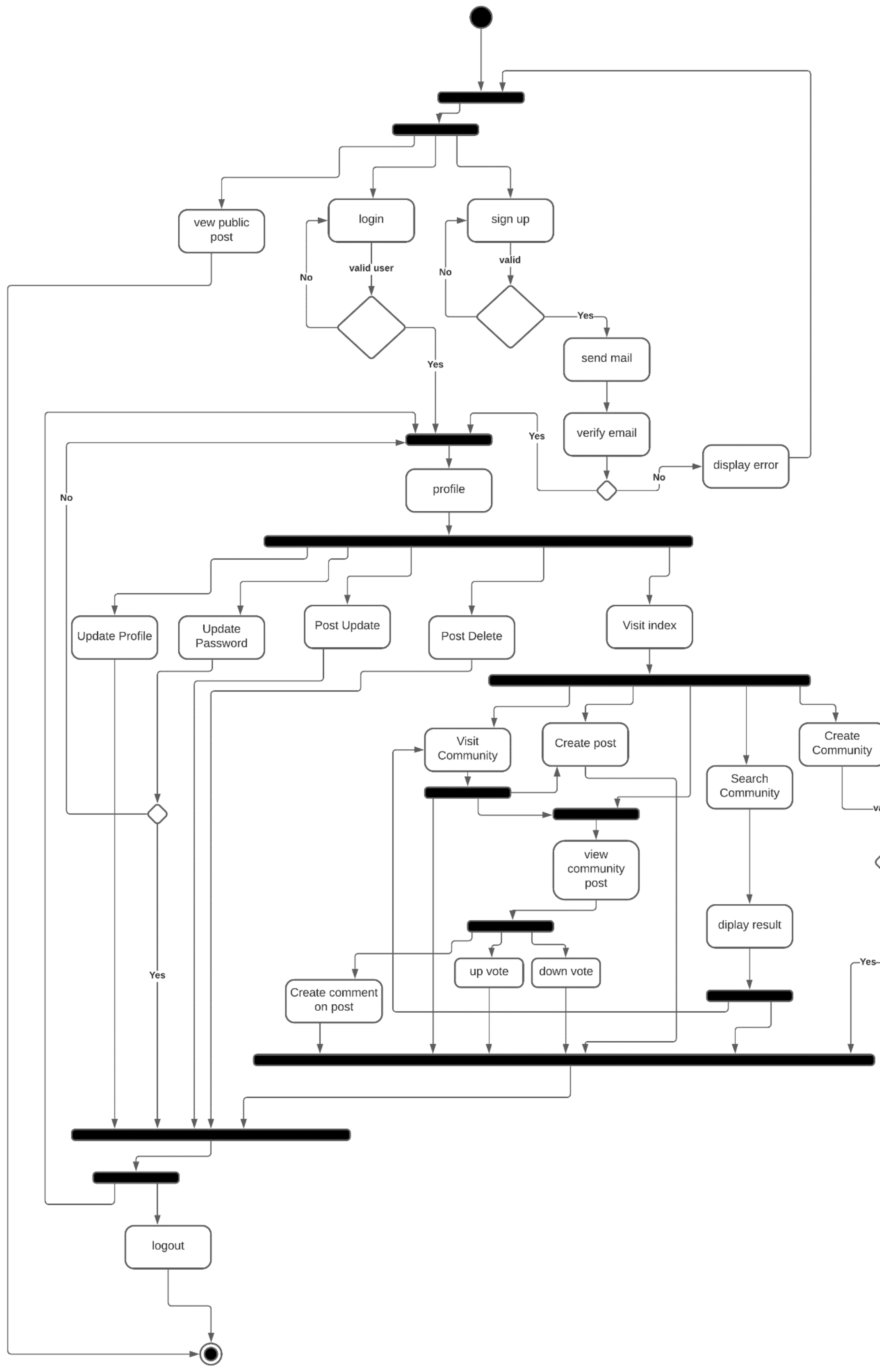
20) View top post:



21) **Logout:**


Logout
user
EWU_connect_website
Database
Request logout
End Session
Display login page

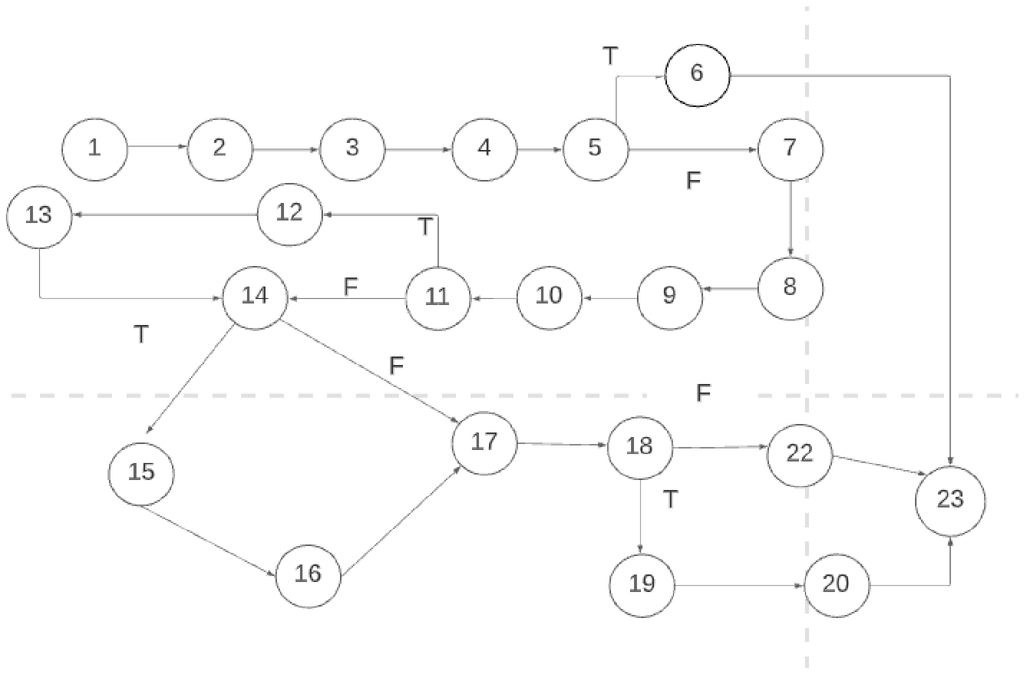
EWU Connect Activity Diagram:



White Box Testing 1:

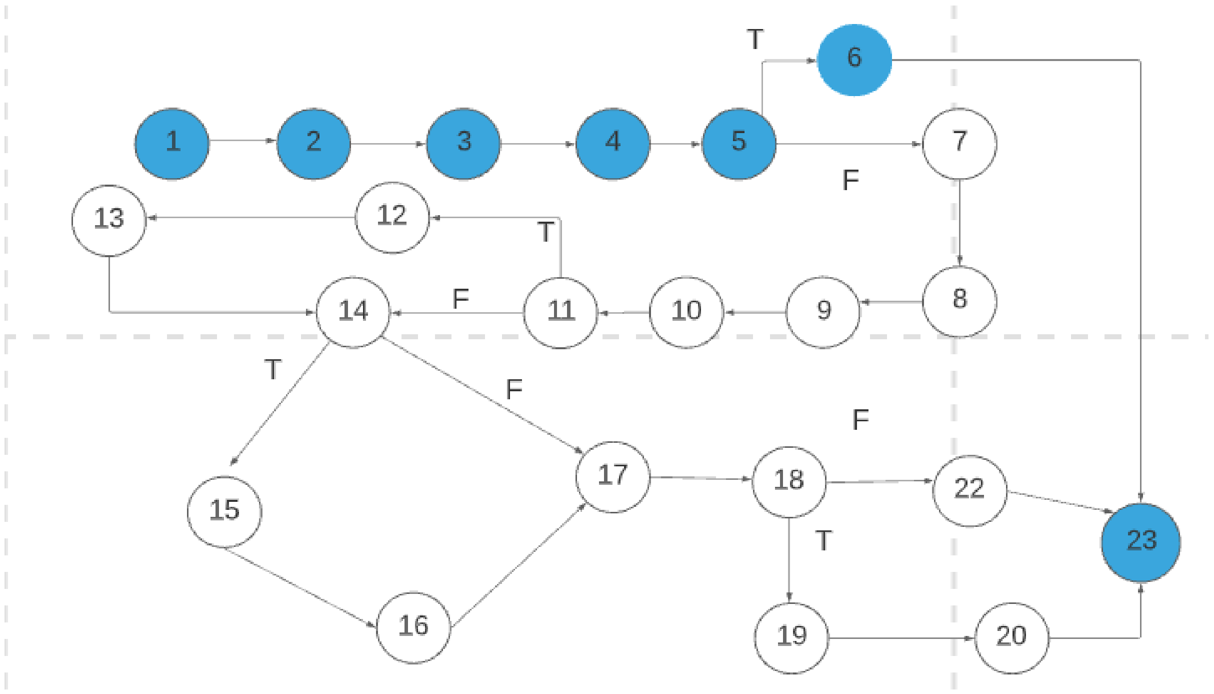
```
1. begin
2.   community_name:=input()
3.   community_tag_name=generate_tag()
4.   result=query(select tag_name from community where tag_name=community_tag_name)
5.   if(length(result)!=0)
6.      redirect ()
7.   about=input()
8.   public=input()
9.   cover_photo_url=defualt
10.  avater_photo_url=defualt
11.  if(  Files['cover_photo'].is_exist)
12.     cover_photo_url:=Files['cover_photo'].url
13.     upload(cover_photo_url)
14.  if(  Files['avater_photo'].is_exist)
15.     avater_photo_url:=Files['avater_photo'].url
16.     upload(avater_photo_url)
17.  res=query()
18.  if(res.status==true)
19.     query()//make current user as mod
20.     redirect()
21.  else
22.     return 'error'
23.  end
```

“Community Create “Control flow Graph:

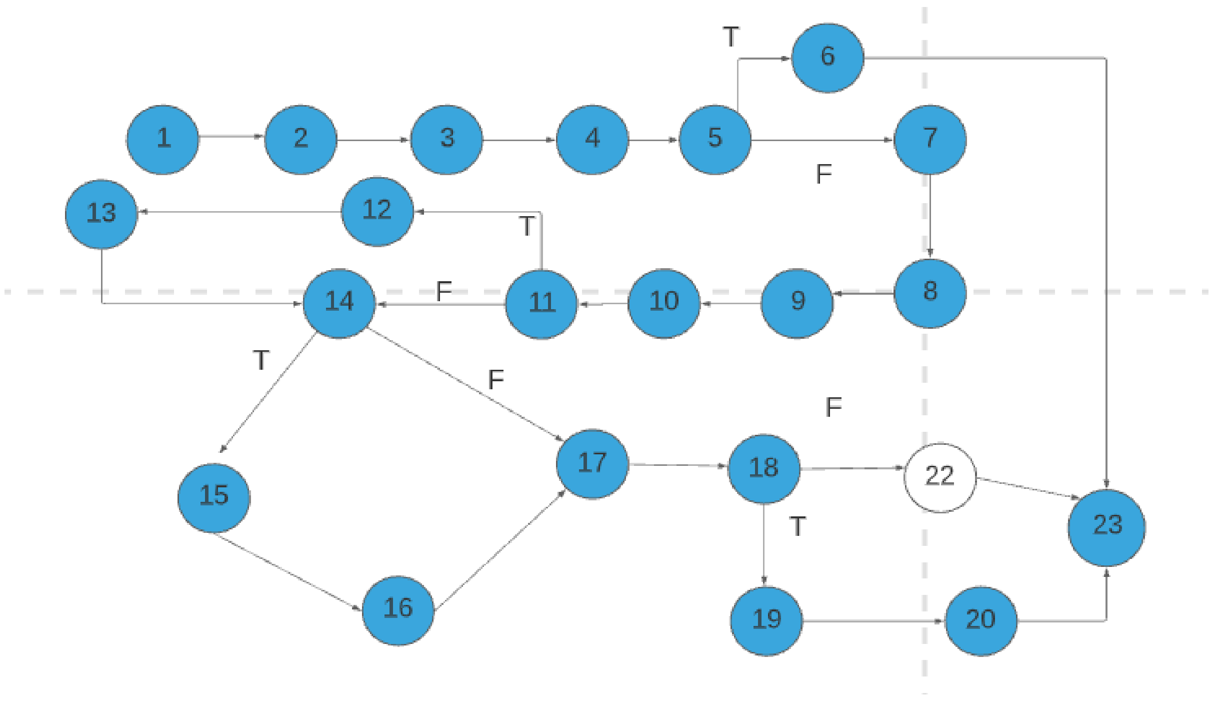


Statement Coverage:

test case 1:
community_name = 'cse18' [input]
tag_name='cse18' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}



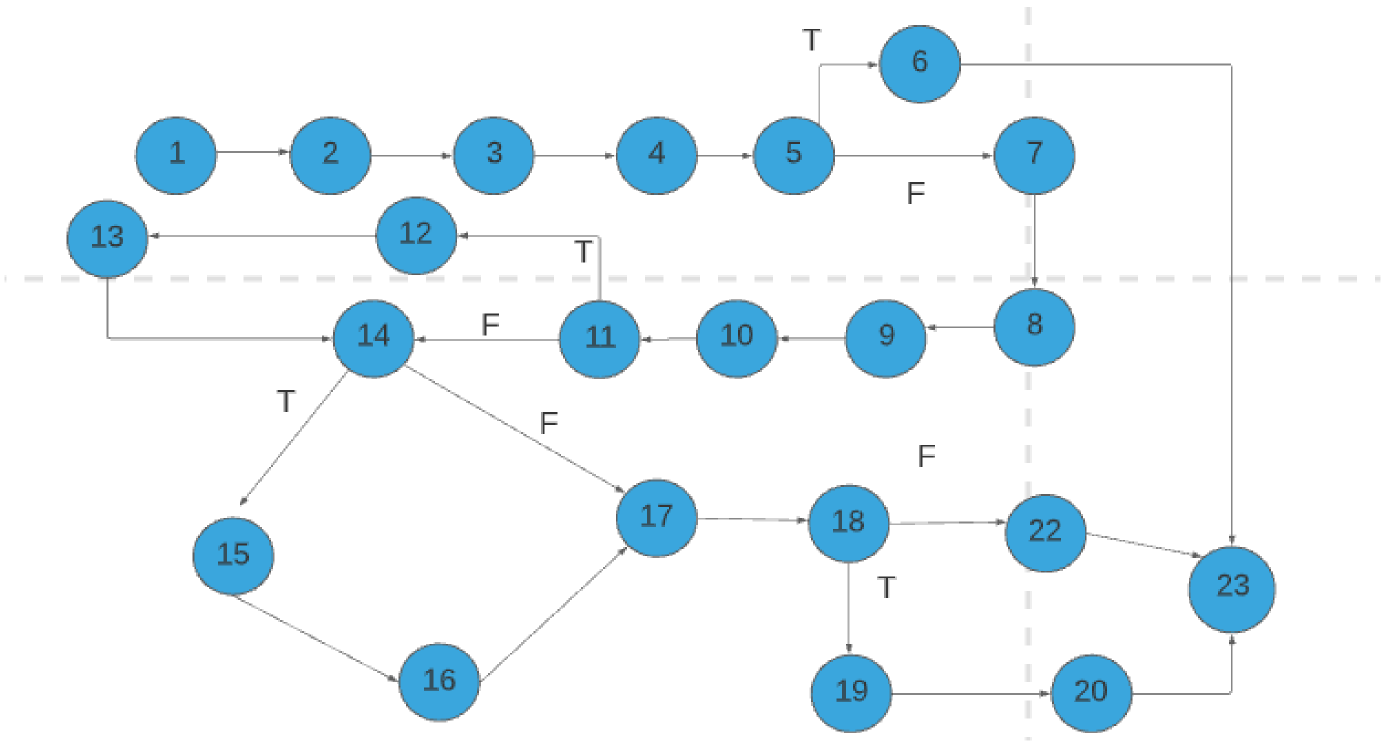
test case 1:
community_name = 'cse18' [input]
tag_name='cse18' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}
test case 2:
community_name = 'cse18' [input]
tag_name='cse2018' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}



test cas 1:
community_name = 'cse18' [input]
tag_name='cse18' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}

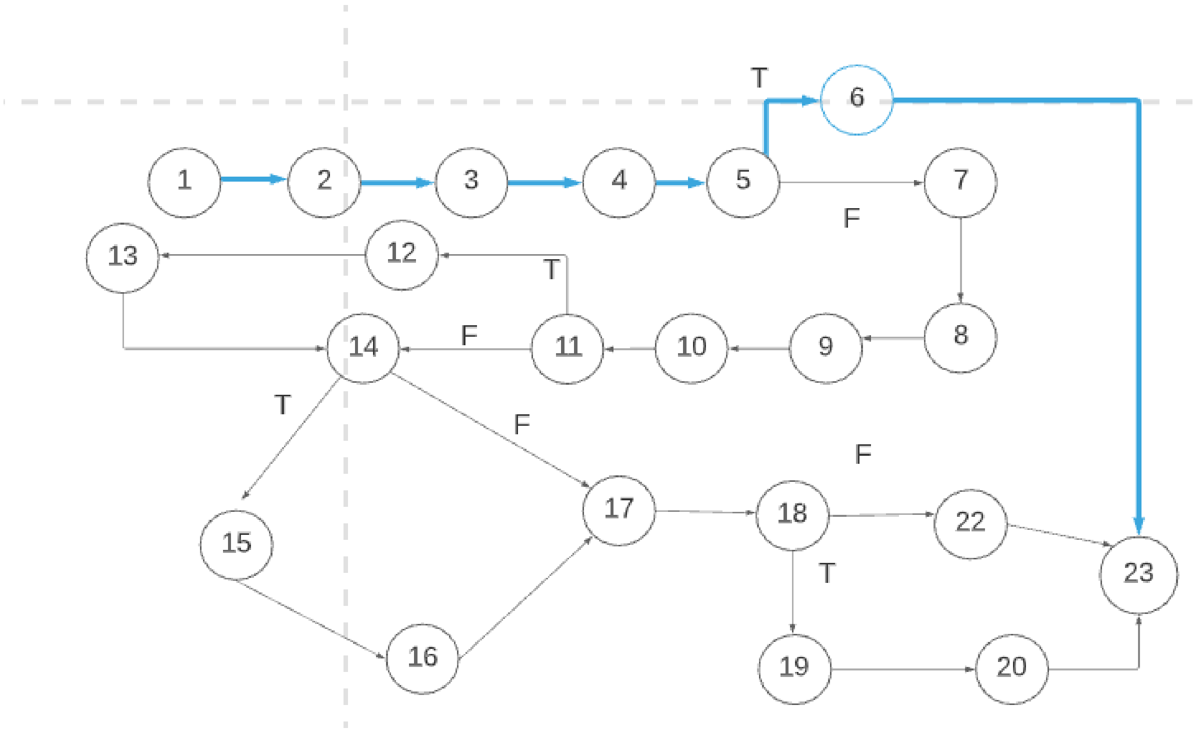
test case 2:
community_name = 'cse18' [input]
tag_name='cse2018' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}

test case 3:
community_name = 'cse18' [input]
tag_name='cse2018' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:false,data:{}}



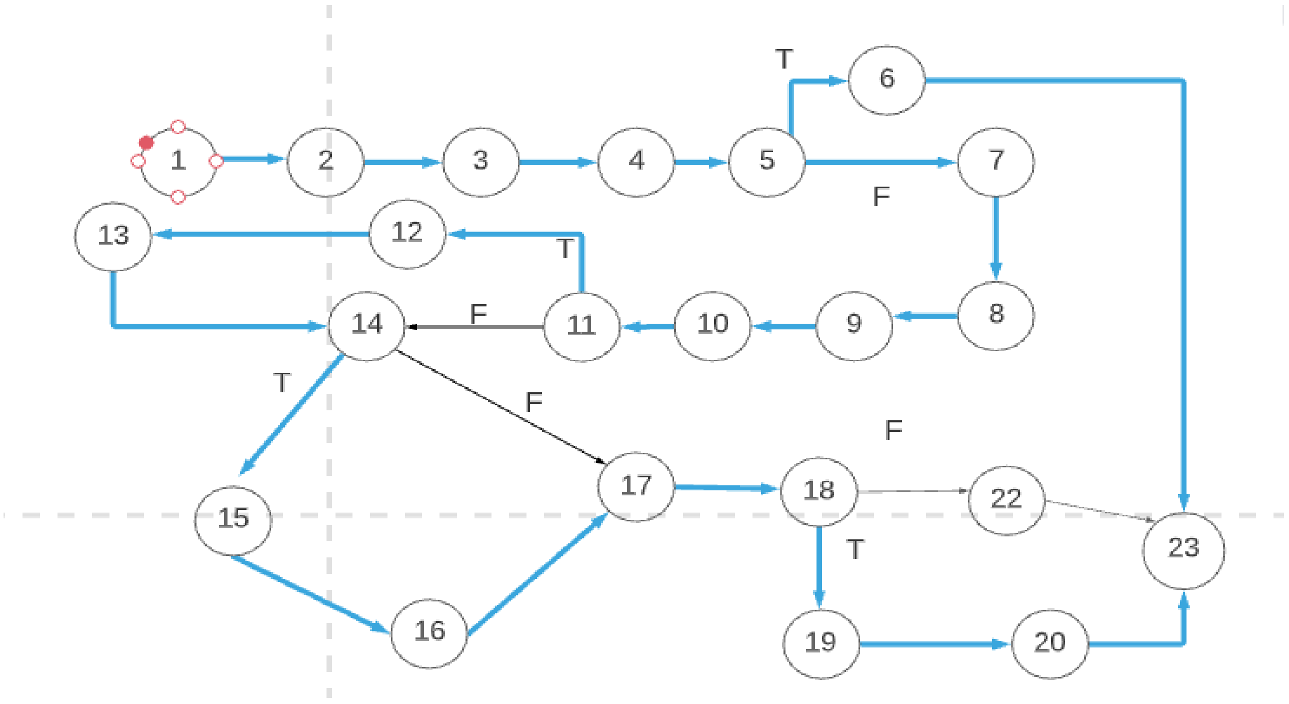
Edge Coverage:

test case 1:
community_name = 'cse18' [input]
tag_name='cse18' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}

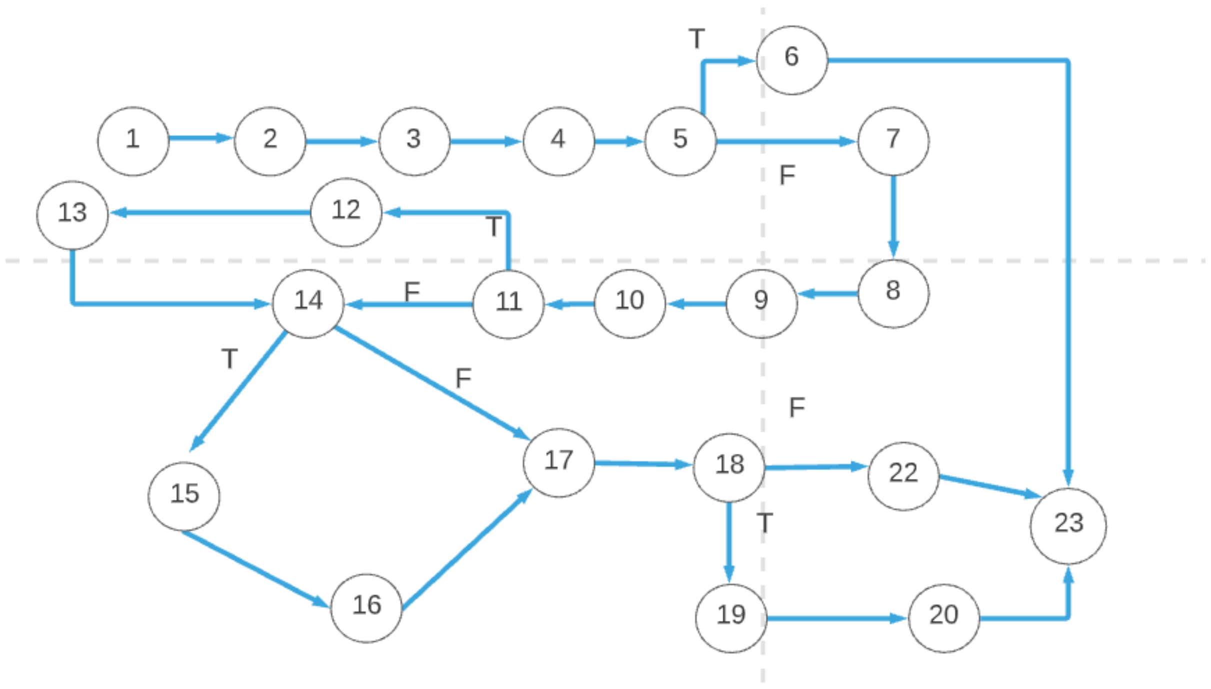


test case 1:
community_name = 'cse18' [input]
tag_name='cse18' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}

test case 2:
community_name = 'cse18' [input]
tag_name='cse2018' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}



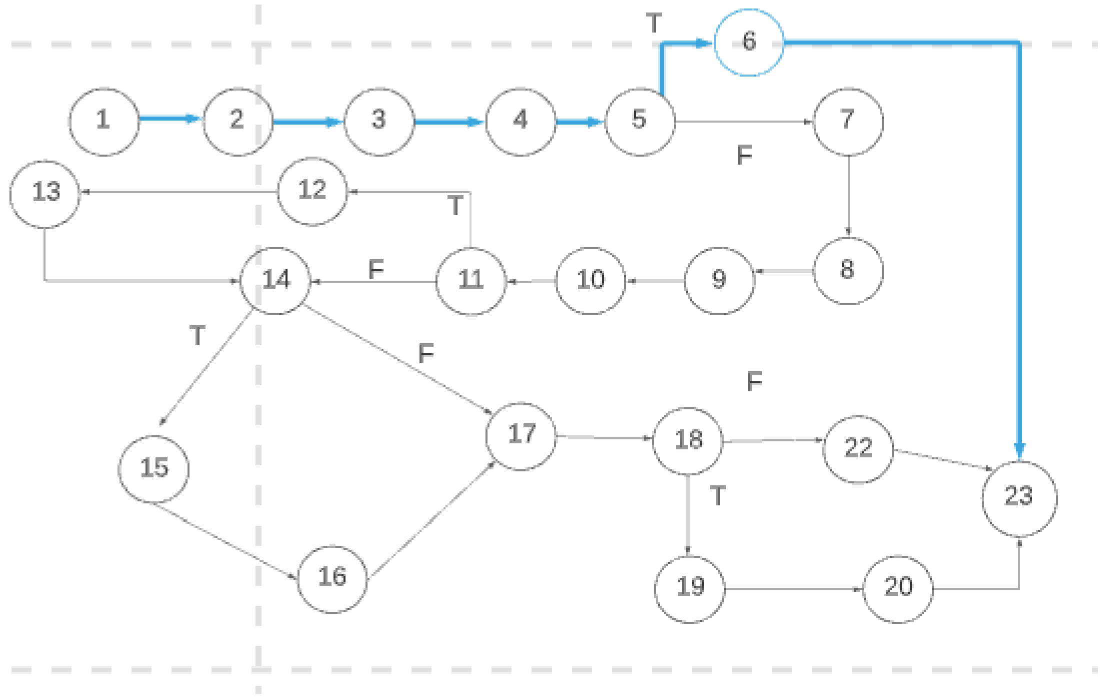
test case 1:
 community_name = 'cse18' [input]
tag_name='cse18' [database]
cover_photo='cover.jpg' ;(file exist)
avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}
test case 2:
 community_name = 'cse18' [input]
tag_name='cse2018' [database]
cover_photo='cover.jpg' ;(file exist)
 avater_photo ='image.jpg';(file exist)
res={status:true,data:{}}
test case 3:
community_name = 'cse18' [input]
tag_name='cse2018' [database]
cover_photo=null ;(file does not exist)
avater_photo =null ;(file does not exist)
res={status:false,data:{}}



Condition Coverage:

test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

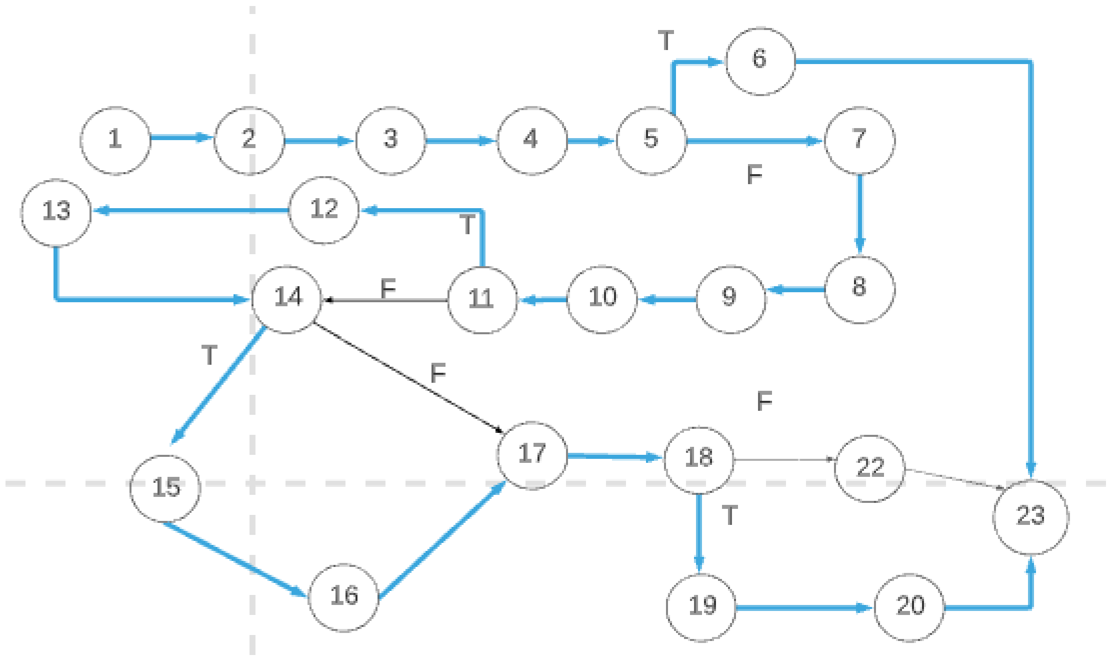


test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

test case 2:

if the first condition false, second condition true, third condition true, fourth condition true and test cases: community_name = 'cse18' [input], tag_name='cse18' [database], cover_photo='cover.jpg' (file exist) ,avater_photo ='image.jpg' (file exist) ,res={status:true,data:{}}



test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

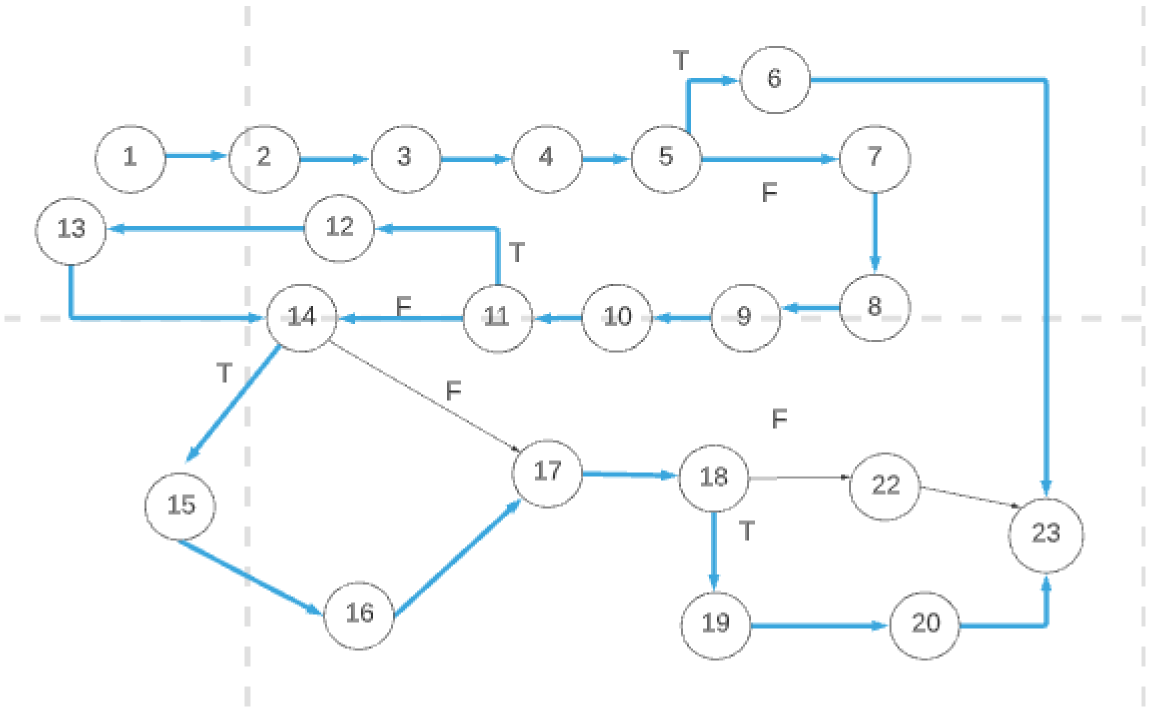
test case 2:

if the first condition false, second condition true, third condition true, fourth condition true and test cases: community_name = 'cse18' [input], tag_name='cse18' [database], cover_photo='cover.jpg' (file exist) ,avater_photo ='image.jpg' (file exist) ,res={status:true,data:{}}

test case 3:

if the first condition false, second condition false,third condition true,Fourth condition true,

test cases are community_name = 'cse18' [input], tag_name='cse18' [database] ,cover_photo='null (file does exist), avater_photo ='image.jpg';(file exist), res={status:true,data:{}}



test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

test case 2:

if the first condition false, second condition true, third condition true, fourth condition true and test cases: community_name = 'cse18' [input], tag_name='cse18' [database], cover_photo='cover.jpg' (file exist) ,avater_photo ='image.jpg' (file exist) ,res={status:true,data:{}}

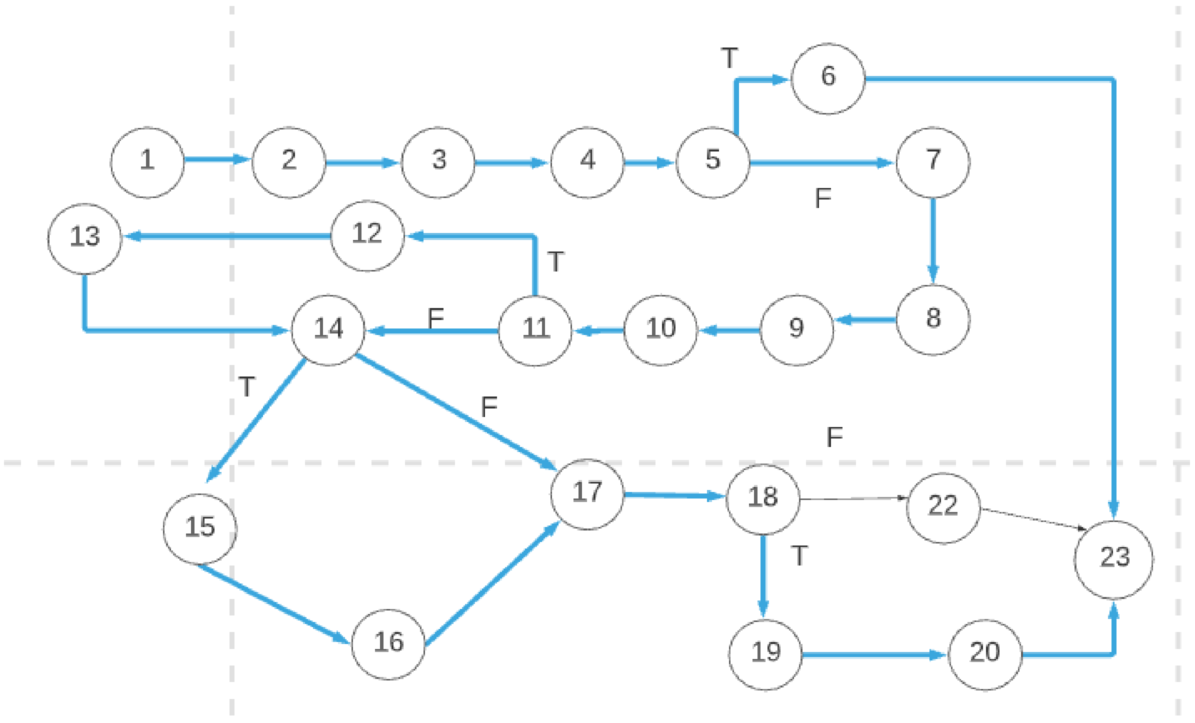
test case 3:

if the first condition false, second condition false,third condition true,Fourth condition true,

test cases are community_name = 'cse18' [input], tag_name='cse18' [database] ,cover_photo='null (file does exist), avater_photo ='image.jpg';(file exist), res={status:true,data:{}}

test case 4: if the first condition false, second condition false, third condition false, Fourth condition true,

test cases are community_name = 'cse18' [input] tag_name='cse18' [database] cover_photo='null (file does exist) avater_photo ='null (file does exist) res={status:true,data:{}}


T
F
T
F
T
F
F
T
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
22
23

test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

test case 2:

if the first condition false, second condition true, third condition true, fourth condition true and test cases: community_name = 'cse18' [input], tag_name='cse18' [database], cover_photo='cover.jpg' (file exist) ,avater_photo ='image.jpg' (file exist) ,res={status:true,data:{}}

test case 3:

if the first condition false, second condition false,third condition true,Fourth condition true,

test cases are community_name = 'cse18' [input], tag_name='cse18' [database] ,cover_photo='null (file does exist), avater_photo ='image.jpg';(file exist), res={status:true,data:{}}

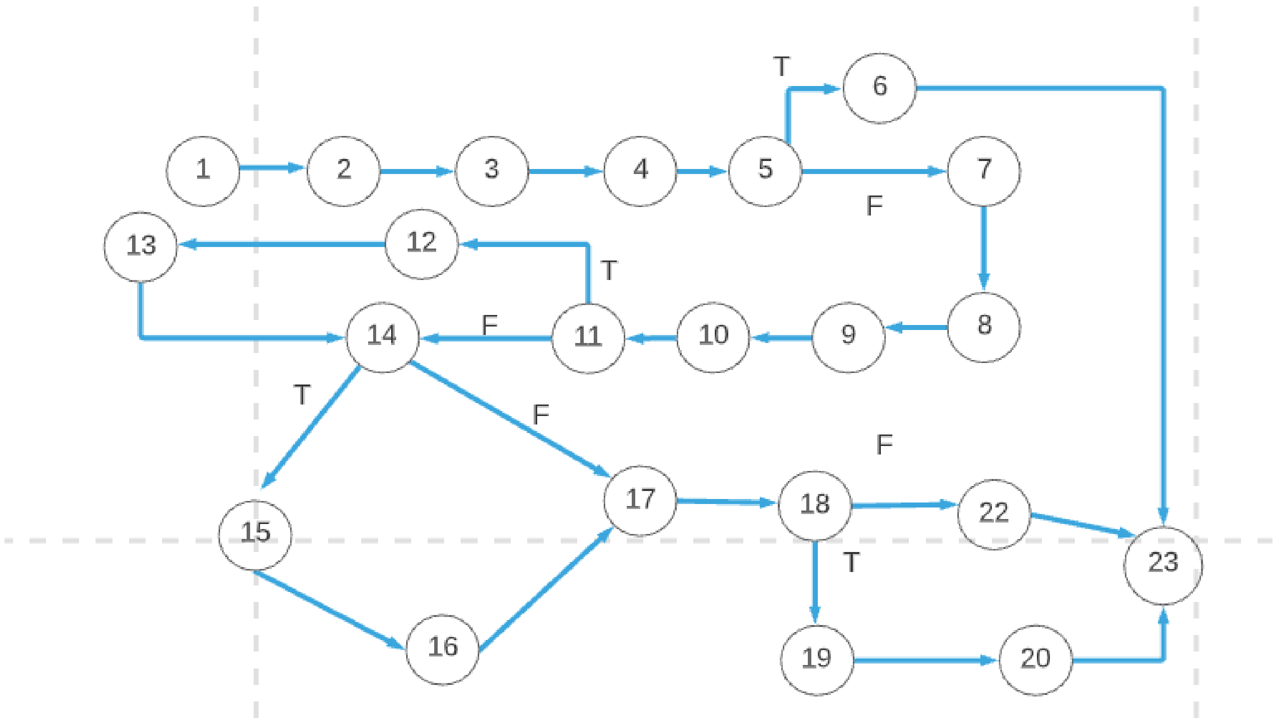
test case 4: if the first condition false, second condition false, third condition false, Fourth condition true,

test cases are community_name = 'cse18' [input] tag_name='cse18' [database] cover_photo='null (file does exist) avater_photo ='null (file does exist) res={status:true,data:{}}

test case 5:

if the first condition false, second condition false ,third condition false, fourth condition false,

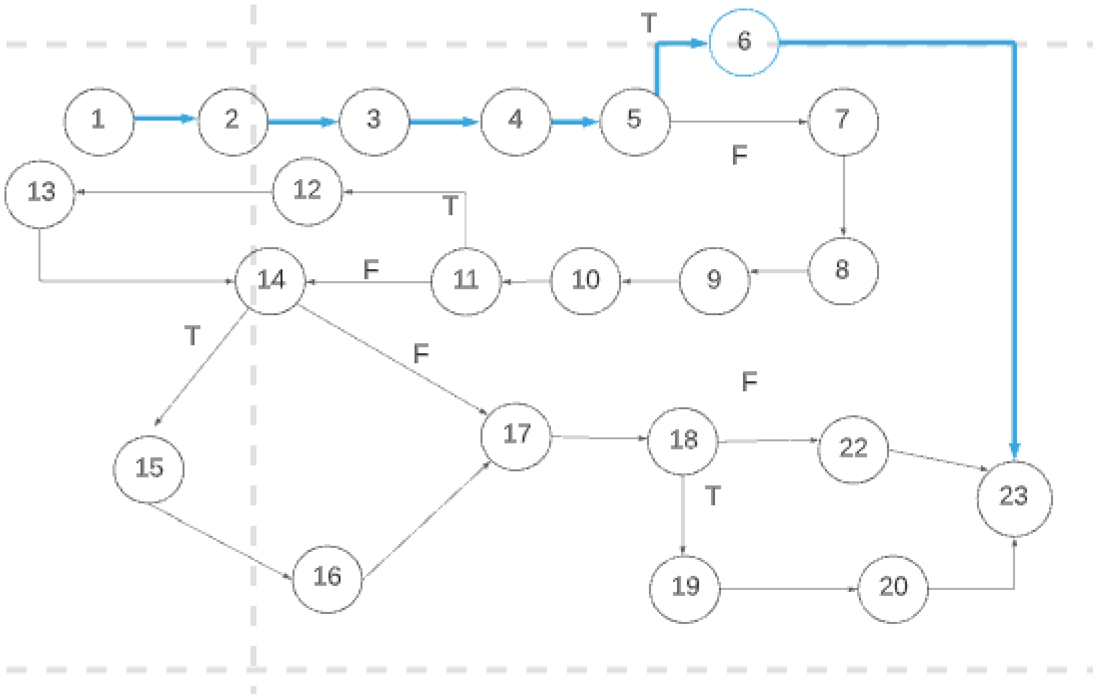
test cases are community_name = 'cse18' [input] ,tag_name='cse18' [database] ,cover_photo='null (file does exist) ,avater_photo ='null (file does exist) ,res={status:false,data:{}}



<u>Path Coverage:</u>

test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

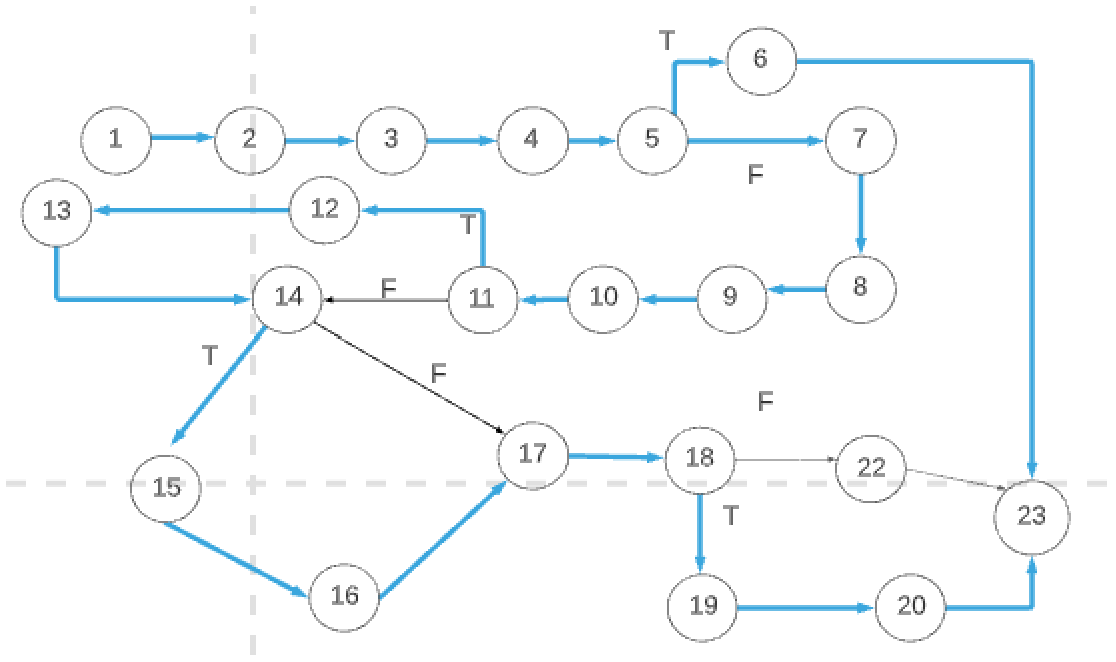


test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

test case 2:

if the first condition false, second condition true, third condition true, fourth condition true and test cases: community_name = 'cse18' [input], tag_name='cse18' [database], cover_photo='cover.jpg' (file exist) ,avater_photo ='image.jpg' (file exist) ,res={status:true,data:{}}



test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

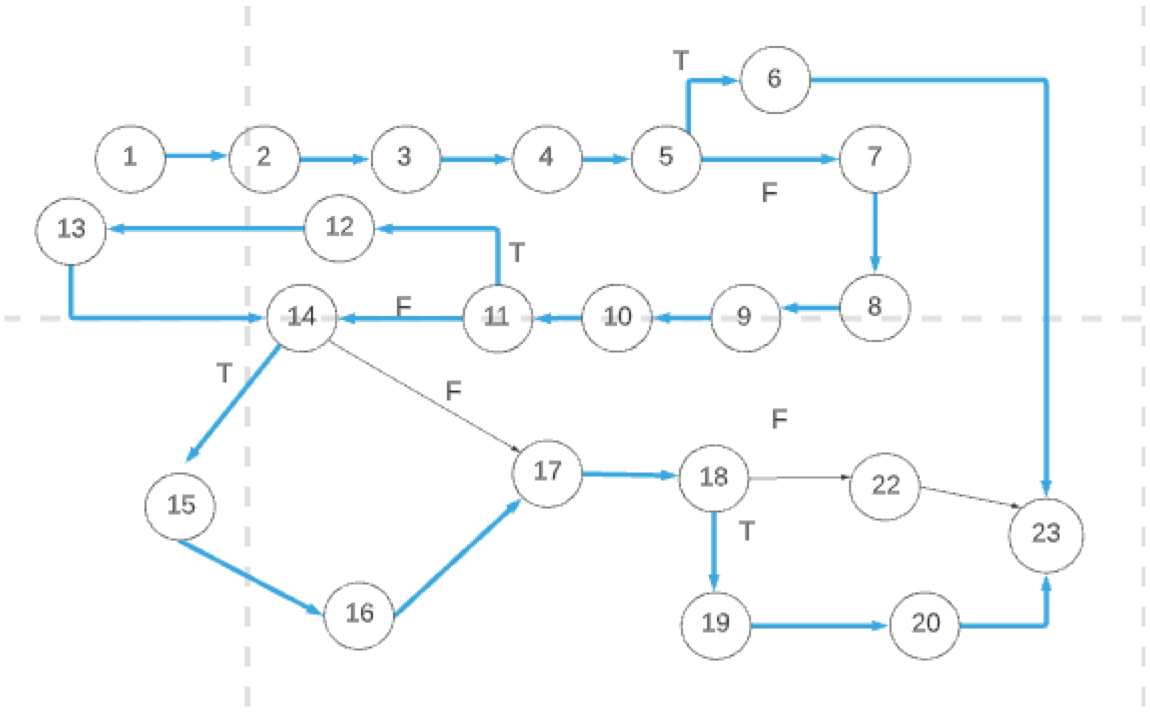
test case 2:

if the first condition false, second condition true, third condition true, fourth condition true and test cases: community_name = 'cse18' [input], tag_name='cse18' [database], cover_photo='cover.jpg' (file exist) ,avater_photo ='image.jpg' (file exist) ,res={status:true,data:{}}

test case 3:

if the first condition false, second condition false,third condition true,Fourth condition true,

test cases are community_name = 'cse18' [input], tag_name='cse18' [database] ,cover_photo='null (file does exist), avater_photo ='image.jpg';(file exist), res={status:true,data:{}}



test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

test case 2:

if the first condition false, second condition true, third condition true, fourth condition true and test cases: community_name = 'cse18' [input], tag_name='cse18' [database], cover_photo='cover.jpg' (file exist) ,avater_photo ='image.jpg' (file exist) ,res={status:true,data:{}}

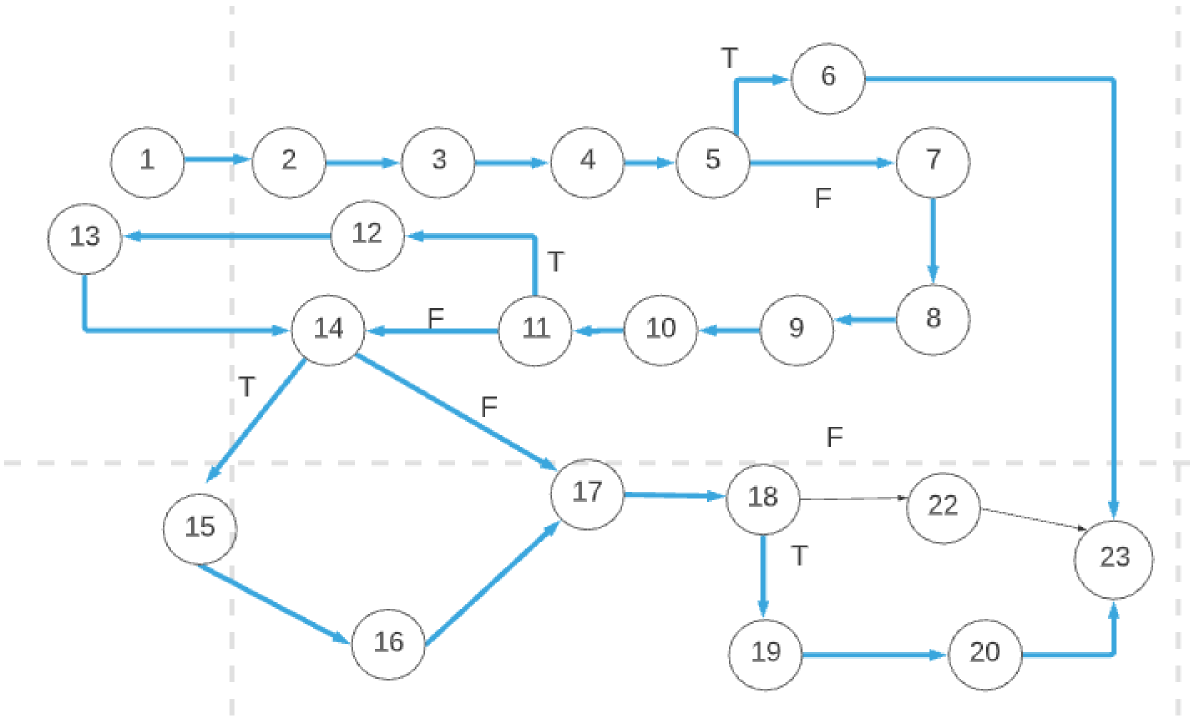
test case 3:

if the first condition false, second condition false,third condition true,Fourth condition true,

test cases are community_name = 'cse18' [input], tag_name='cse18' [database] ,cover_photo='null (file does exist), avater_photo ='image.jpg';(file exist), res={status:true,data:{}}

test case 4: if the first condition false, second condition false, third condition false, Fourth condition true,

test cases are community_name = 'cse18' [input] tag_name='cse18' [database] cover_photo='null (file does exist) avater_photo ='null (file does exist) res={status:true,data:{}}


T
F
T
F
T
F
F
T
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
22
23

test case 1: if the first condition true ,

and test cases: community_name = 'cse18' [input] tag_name='cse18' [database], cover_photo='cover.jpg' ;(file exist) ,avater_photo ='image.jpg';(file exist) ,res={status:true,data:{}}

test case 2:

if the first condition false, second condition true, third condition true, fourth condition true and test cases: community_name = 'cse18' [input], tag_name='cse18' [database], cover_photo='cover.jpg' (file exist) ,avater_photo ='image.jpg' (file exist) ,res={status:true,data:{}}

test case 3:

if the first condition false, second condition false,third condition true,Fourth condition true,

test cases are community_name = 'cse18' [input], tag_name='cse18' [database] ,cover_photo='null (file does exist), avater_photo ='image.jpg';(file exist), res={status:true,data:{}}

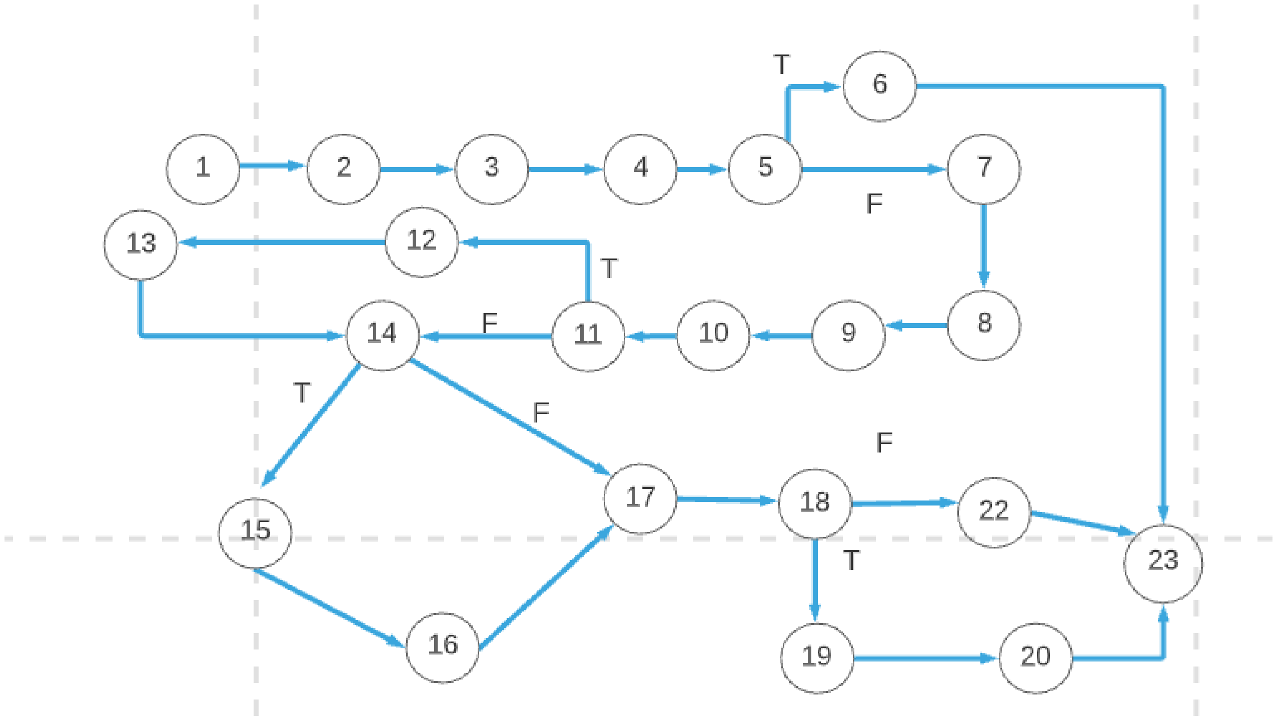
test case 4: if the first condition false, second condition false, third condition false, Fourth condition true,

test cases are community_name = 'cse18' [input] tag_name='cse18' [database] cover_photo='null (file does exist) avater_photo ='null (file does exist) res={status:true,data:{}}

test case 5:

if the first condition false, second condition false ,third condition false, fourth condition false,

test cases are community_name = 'cse18' [input] ,tag_name='cse18' [database] ,cover_photo='null (file does exist) ,avater_photo ='null (file does exist) ,res={status:false,data:{}}



Possible paths are for the following conditions:

T-X-X-X

F-T-T-T

F-T-T-F

F-T-F-T

F-T-F-F

F-F-T-T

F-F-T-F

F-F-F-T

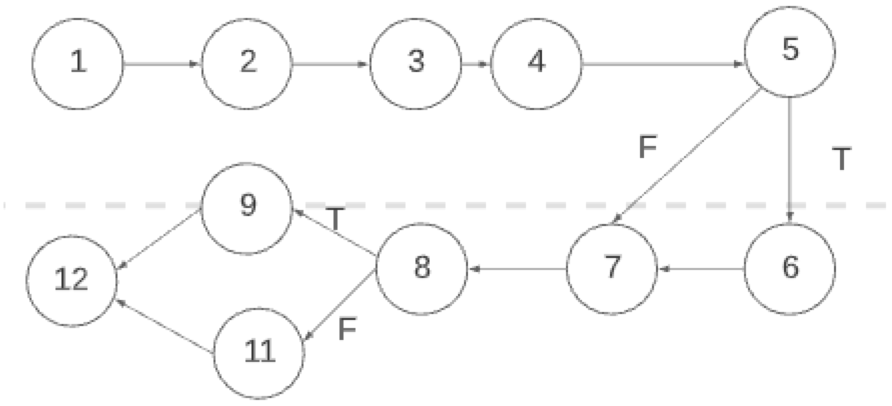
F-F-F-F

There fore there are 9 possible paths.

White box testing on Token verification

```
1.    began
2.    pText=inputToken()
3.    data=split(pTExt,'.')
4.    len=length(data)
5.    if ( len < 3 )
6.        return error_message
7.    expired_time= data[len -1 ]
8.    if ( expired_time < currentTime() )
9.            return error_message
10.   else
11.   return data[0],data[1]
12.   end
```

Token Verification Control flow graph:



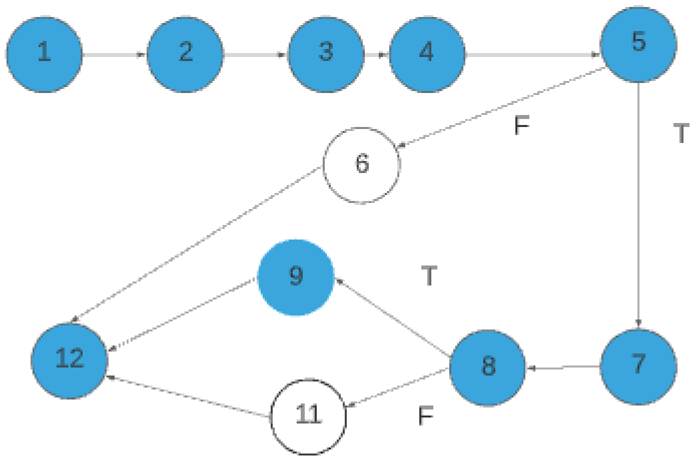
Statement coverage:

case 1:

pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"



case 1:

pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

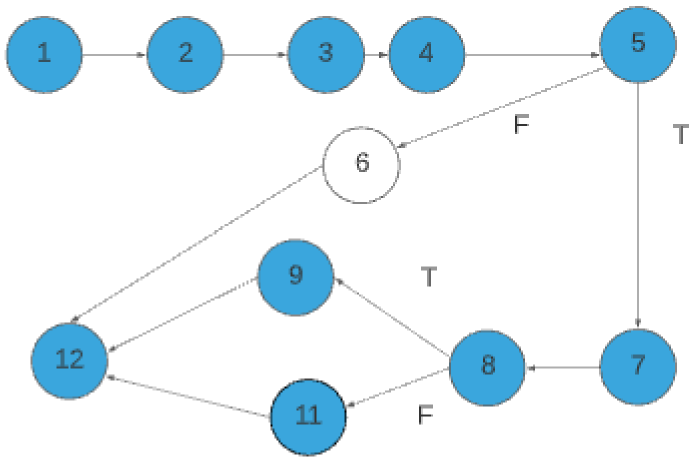
current_time="15-5-2021"

case 2:

pText='1.joy.20-5-2021'

data=[1,"joy","20-5-2021"]

current_time="15-5-2021"



,

case 1:

pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"

case 2:

pText='1.joy.20-5-2021'

data=[1,"joy","20-5-2021"]

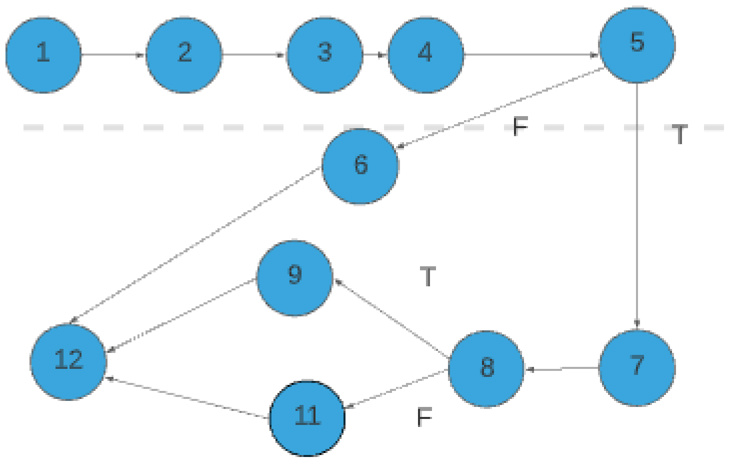
current_time="15-5-2021"

case 3:

pText='1.joy.981.20-5-2021'

data=[1,"joy",981,"20-5-2021"]

current_time="15-5-2021"



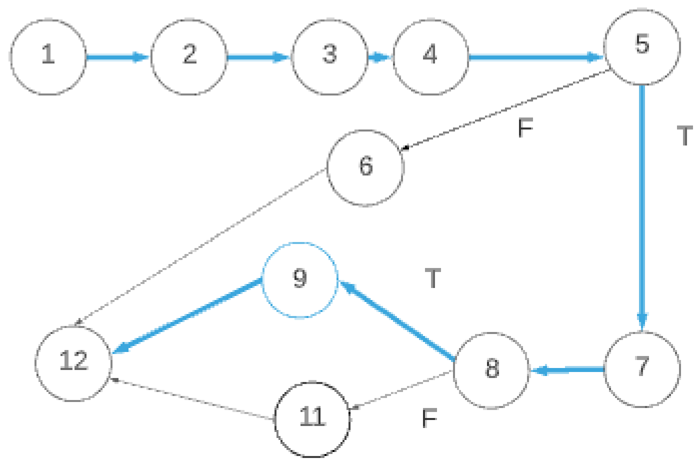
Edge coverage:

case 1:

pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"



case 1:

pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

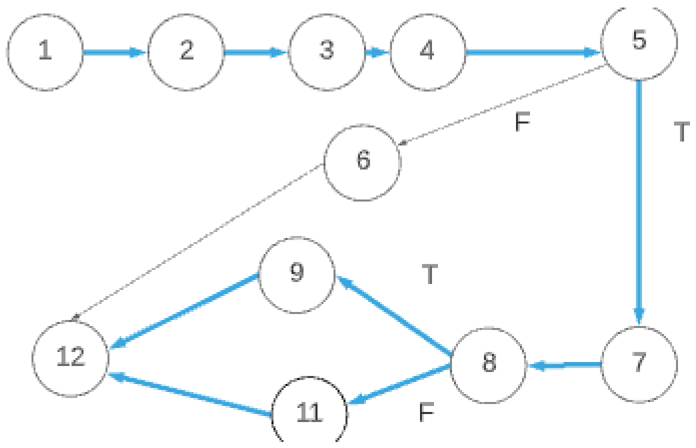
current_time="15-5-2021"

case 2:

pText='1.joy.20-5-2021'

data=[1,"joy","20-5-2021"]

current_time="15-5-2021"



case 1:

pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"

case 2:

pText='1.joy.20-5-2021'

data=[1,"joy","20-5-2021"]

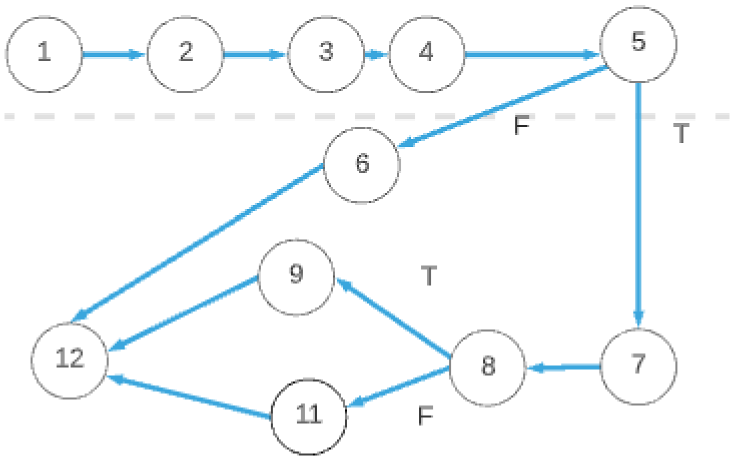
current_time="15-5-2021"

case 3:

pText='1.joy.981.20-5-2021'

data=[1,"joy",981,"20-5-2021"]

current_time="15-5-2021"



Condition Coverage:

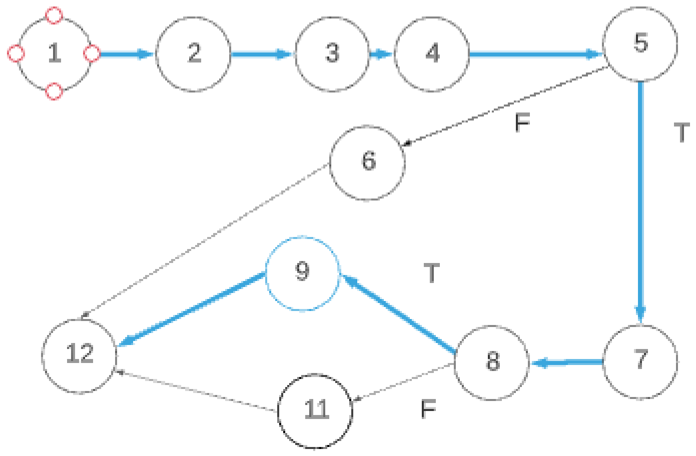
case 1:

if the first condition : true , second condition : true ,

and pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"



case 1:

if the first condition : true , second condition : true ,

and pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"

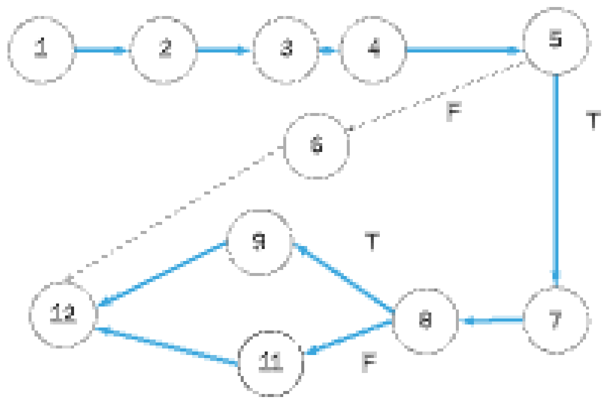
case 2:

if the first condition : true,second condition : false and

pText='1.joy.20-5-2021'

data=[1,"joy","20-5-2021"]

current_time="15-5-2021"



case 1:

if the first condition : true , second condition : true ,

and pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"

case 2:

if the first condition : true,second condition : false and

pText='1.joy.20-5-2021'

data=[1,"joy","20-5-2021"]

current_time="15-5-2021"

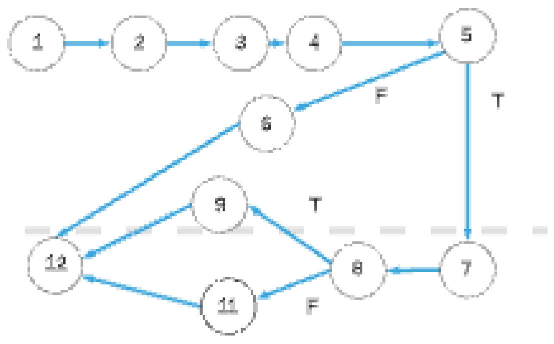
case 3:

if the first condition : false

pText='1.joy.981.20-5-2021'

data=[1,"joy",981,"20-5-2021"]

current_time="15-5-2021"



Path coverage

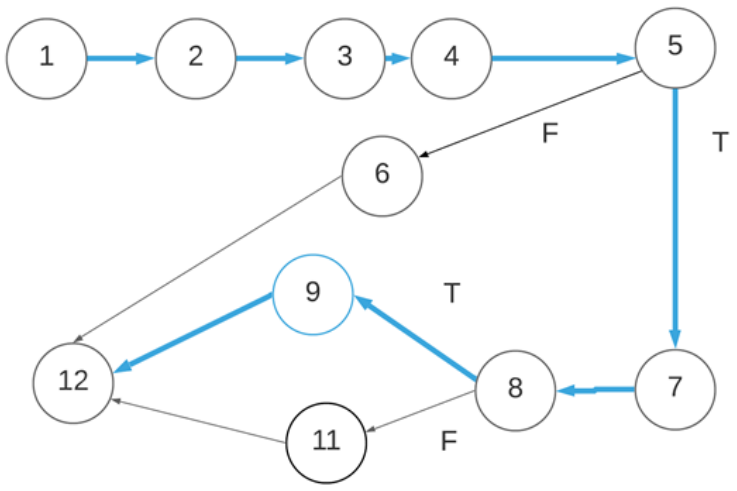
case 1:

if the first condition : true , second condition : true ,

and pTText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"



case 1:

if the first condition : true , second condition : true ,

and pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"

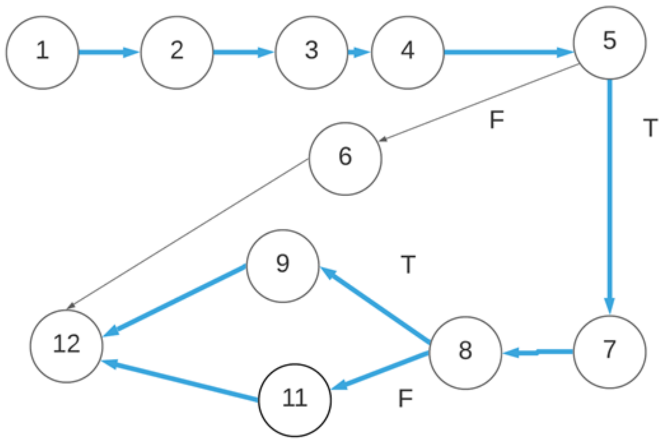
case 2:

if the first condition : true,second condition : false and

pText='1.joy.20-5-2021'

data=[1,"joy","20-5-2021"]

current_time="15-5-2021"



case 1:

if the first condition : true , second condition : true ,

and pText='1.joy.10-5-2021'

data=[1,"joy","10-5-2021"]

current_time="15-5-2021"

case 2:

if the first condition : true,second condition : false and

pText='1.joy.20-5-2021'

data=[1,"joy","20-5-2021"]

current_time="15-5-2021"

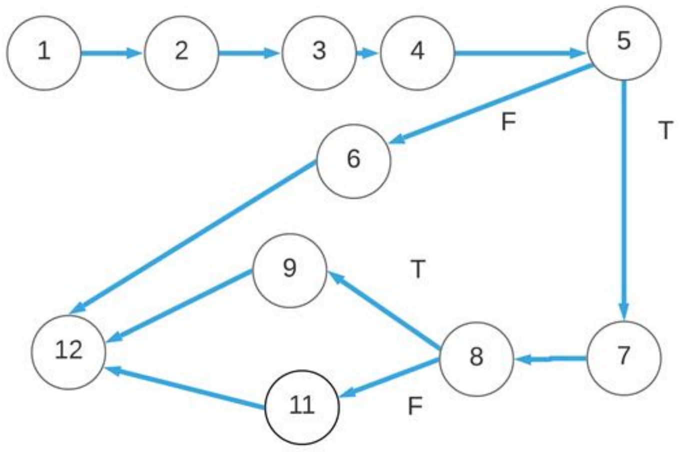
case 3:

if the first condition : false

pText='1.joy.981.20-5-2021'

data=[1,"joy",981,"20-5-2021"]

current_time="15-5-2021"



Possible Path are for following conditions:

T-T

T-F

F-X

That means, there are 3 possible path.

Conclusion:

Our goal was to create a personalized virtual platform for people connected to EWU. This was a big challenge for us. We had to learn and implement as fast as possible. We thoroughly analysed the requirements for the project and were able to implement more than 85.00% of our functional requirement in the short span of one and a half months. We have learnt a lot in our attempt to make the project work. If we get time and budget our platform will surely rival Facebook and google classroom in the academic arena of East West University

Source code:

https://github.com/tz01x/ewu_connect/

References:

https://www.w3schools.com/php/DEFAULT.asp

https://app.lucidchart.com/

https://github.com/PHPMailer/PHPMailer

https://getbootstrap.com/