
Group S Development Team

TeamMe
Design Report
CSC 32200 Software Engineering Project

Version 1.0

Group Members:
Tohidul Islam
Sophie Huang
Ekramul Sawrid
Dor Ulman

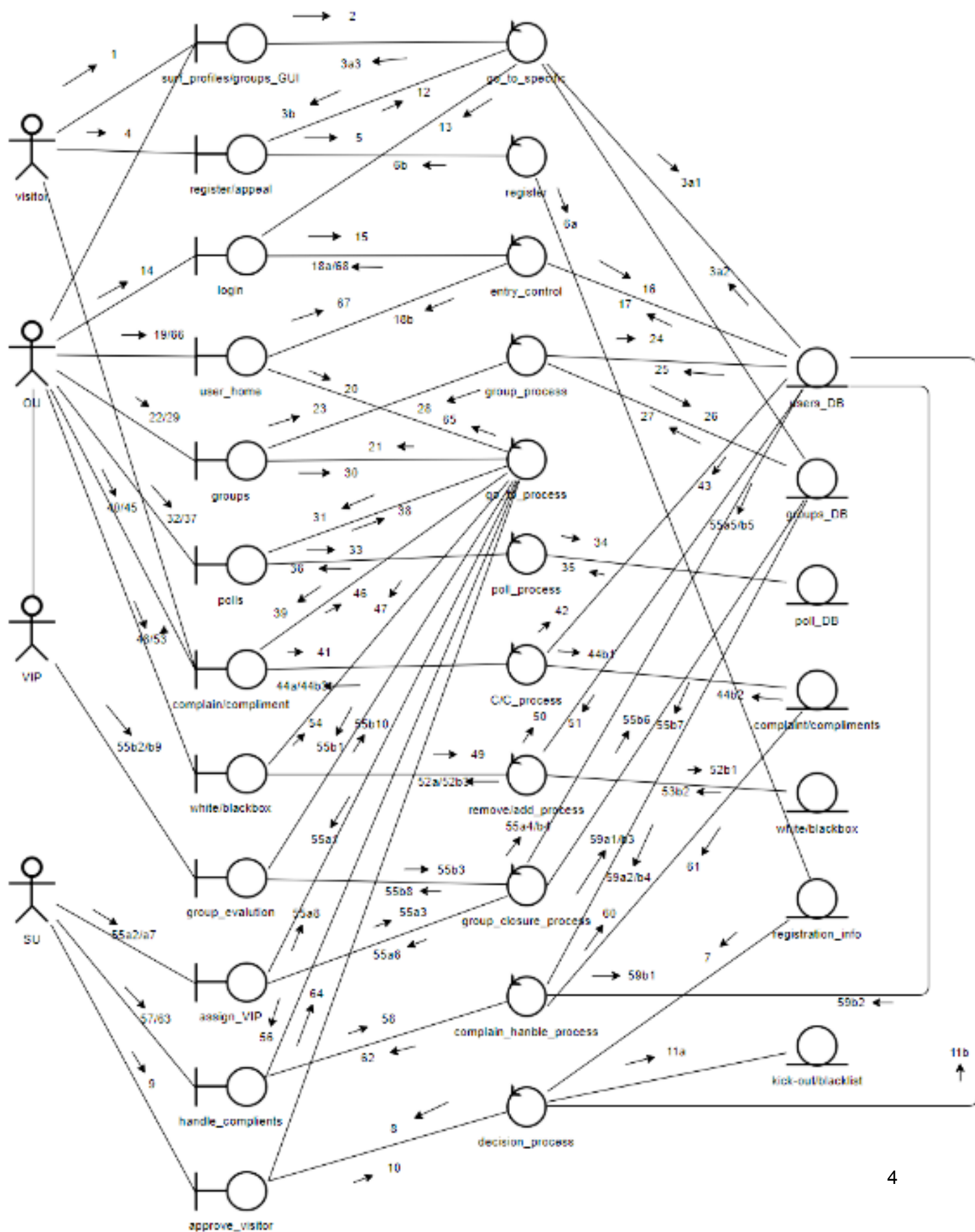
Table of Contents

1. Introduction: An overall picture of system using collaboration class diagram	3
2. All use-cases	8
a. Scenarios for each use-case	8
b. Collaboration class or petri-net diagrams for each use-case	15
3. E-R Diagram for Entire System	24
4. Detailed Design using pseudo-code	26
5. Systems Screens: Prototypes of GUI screens	39
6. Minutes of group meetings and possible concerns of teamwork.	52
7. Address of git repository	54

1. Introduction

The following (see next page) is an overall (general) picture of the system using collaboration class diagrams. For detailed collaboration or petri-net diagrams of different use-cases please refer to the next section (2. All Use-Cases).

Note: the data from the data class shown in the collaboration diagrams can be data processed form a text file (we assume toy system) and not an actual database.



Numbered Actions Description

1. Visitor surfs the application for profiles/groups.
2. Info of user/group goes to process or goes to register/login.
3. a1) User info goes to user DB.
a2) Returns user info.
a3) User profile is shown.
b) Goes to the register/appeal page.
4. Input information for registration.
5. Registration information go through process.
6. a) Valid registration information goes to registration_info_DB.
b) Invalid registration information will send back error message.
7. Registration information goes to the decision process.
8. Registration information goes to the SU.
9. SU make a decisions on the registration.
10. SU decision information goes to the decision process.
11. a) If rejected, the visitor's information goes to the kicked-out or blacklist DB.
b) If approved, the visitor's information goes the user DB.
12. The visitor sends signal to go to login page to process.
13. Signal to go login page.
14. OU inputs login information.
15. Login information goes to entry control process.
16. Login formation goes to user DB.
17. Result of login information match.
18. a) If not match, send error message.
b) Successful logn, send user information.
19. OU send signal to go to groups page.
20. Request is sent to go to process.
21. Information to display groups page.
22. OU inputs information for group activities.
23. Information for to group process.
24. Selected information goes to user DB.
25. Information or result is sent back to process.
26. Selected information goes to groups DB.
27. Information or result is sent back to process.
28. Information or result is sent back to process.
29. OU sends signal to go to polls page.
30. Request for poll page is sent to process.
31. Information to sent to polls page to open.
32. OU inputs poll activities.

33. Poll information is sent to process.
34. Poll information is sent to poll DB.
35. Information or result is sent back to process.
36. Information or result is sent back to the polls page.
37. OU sends a signal to go to the complaint/compliment page.
38. Request is sent to go to complain/compliment page to process.
39. Information sent to complain/compliment page to open.
40. OU inputs complain/compliment information.
41. Complain/compliment information is sent to complain/compliment process.
42. Send information of complained user to user DB to check.
43. Send result back to complain/compliment process.
44. a) Send result back to complain/compliment DB.
 - b1) If has group information, send information to group DB.
 - b2) Send result back to complain/compliment process.
 - b3) Send result back to complain/compliment page.
45. OU sends signal to go to white/blackbox page.
46. Send request to go to white/blackbox page to go to process.
47. Information sent to white/blackbox to open.
48. OU inputs information for white/blackbox activities.
49. Send information to remove/add process.
50. Send information to user DB to check.
51. Send information or result to process.
52. a) If no match, send error message back to white/blackbox page.
 - b1) If match, send information to white/blackbox DB.
 - b2) Send information or result back to remove/add process.
 - b3) Send information back to white/blackbox page.
53. OU send signal to go to another page
54. Request to go to another page is sent.
55. a1) For SU users only, the information for assign VIP page to open
 - a2) SU inputs VIP information to assign to a group.
 - a3) Send information to group closure process.
 - a4) Send information to users DB to check.
 - a5) Send information or result back to group closure process.
 - a6) Send information or result back to assign VIP page.
 - a7) Send signal to go to another page.
 - a8) Send request to go to another page to go to process.
 - b1) For VIP users, information is sent to group evaluation page to open
 - b2) VIP inputs information for group evaluation.
 - b3) Send information to group closure process.

- b4) Send information to user DB to update information.
- b5) Send information or result back to group closure process.
- b6) Send information to group DB to update information.
- b7) Send information or result back to group closure process.
- b8) Send information or result back to group evaluation page.
- b9) VIP sends signal to go to another page.
- b10) request to go to another page is sent to go to process.
- 56. For SU users only, information is sent to handle complaints page to open.
- 57. SU inputs a decision for a complaint.
- 58. Decision information is sent to complain handle process.
- 59. a1) If user complaint, send information to user DB to update user information.
- a2) Send information or result back to complain handle process.
- b1) If group information, send group information to group DB.
- b2) Send information or result back to complain handle process.
- b3) Send information to user DB to update user information.
- b4) Send information or result back to complain handle process.
- 60. Send result information to update complaints/compliments.
- 61. Send information or result back to complain handle process.
- 62. Send information or result back to handle complaints page
- 63. SU signal to go to another page.
- 64. Send a request to go to another page to go to process.
- 65. For an OU, VIP, or SU in this case, send information to to to user home page.
- 66. Signal to go to login page.
- 67. Send request to go to login page to entry control.
- 68. Send information to go to login page (you are logging out).

2. All Use-Cases

a. Scenarios for each Use-Case: Normal AND Exceptional Scenarios

Type of users:

- **visitor:** no need to login, just surf around
- **ordinary user (OU):** all self registered users who are approved by SU, need login
- **VIP:** OUs whose reputation scores exceed a threshold set by SU
- **super-user (SU):** one founding SU who initializes the system; one democratic SU who is voted by VIPs

Use-Case 1: Visitor Browsing/Surfing the Application

Normal scenarios:

1. Visitors open up the application home page, and it will show top 3 rated projects and user profiles. Visitors can click on the projects and user profiles to open up a new window to see more details.

Exceptional Scenarios:

We could not find any exceptional scenarios for this use-case..

Use-case 2: Login

Normal scenarios:

1. Attempt to login is by non-blacklist and kick-out users with correct login information. Login is successful.

Exceptional Scenarios:

1. Attempt to login by incorrect / non-existing information. Login failed.
2. Attempt to login by a blacklisted user. Login failed.
3. Attempt to login by kicked-out user. Login failed.

Use-Case 3: Visitors Register to be OU

Normal scenarios:

1. Visitors click the button "Register" and a new window opens up to prompt visitors to input required information. Required information includes:
 - a. Valid basic info: name, email address, interests
 - b. Valid referral info: the name and email address of the referral (OU or VIP)
 - c. Info is not for existing users.

One SU will check the legitimacy of the referral information. If SU approves, then an email with user id and password to the applying visitor and the visitor becomes an OU.

The first time the OU logs into the system, it is asked to change the initial password. And the initial reputation score of the OU is given by the reference, ranging from 0-10 if it's given by another OU, or 0-20 if given by a VIP.

Exceptional scenarios:

1. When inputting registration information:
 - a. Invalid basic info: name, email address, interests.
 - b. Invalid referral info: the name and email address of the referral (OU or VIP)
 - c. Info is of existing users.
2. Visitors try to register, but are rejected by the SU. Visitors can have one chance to appeal. If the appeal is rejected, the visitors will be blacklisted and cannot register forever.
3. If the appeal is accepted, the visitors will be given user login information and reputation scores as in the normal scenarios.

Use-Case 4: OUs form Groups

Normal scenarios:

1. OUs can send out invitations to other OUs to form a group. OUs can either accept or reject the invitations. If rejecting the invitation, OUs should provide reasons upon rejection in a form of checkboxes.

Exceptional scenarios:

1. when OUs want to accept or reject all invitations from a particular OU, they can put the OU into a white box or a black box. The white box will allow them to accept all invitations, while the black box will allow them to reject all of them with an automatic message.
2. Invited a invalid/non-existing user.
3. No response from OU. Then the user is never joined to that group.

Use-case 5: User Whitelist/blacklist Users

Normal Scenarios:

1. White/blacklist valid/existing users.

Exceptional Scenarios:

1. Attempt to white/blacklist invalid/non-existing users.
2. Attempt to white/blacklist users more than once.

Use-Case 6: Public/Private group page and Moderation/Posting After Group Formed

Normal scenarios:

1. A group web-page will be created once a group is formed by OUs. The group web-page has two sections: one for public and one for private. In the public section, information such as group projects and group members can be browsed by all visitors and users.

Exceptional scenarios:

1. If the group web-page is opened by a group member, the group web-page will have a private section where the member can edit the group page, posting updates, manage group members, and create meeting polls.
2. If taboo words are posted, the word will be changed and the consequences will be initiated.

Use-Case 7: Group Poll for a Convenient Time to Meet-Up

Normal scenarios:

1. Time slot with the most votes will be chosen after all the group members voted. Warning will be issued if a member missed two scheduled meetings. A user with a warning can appeal to the SU for keeping a straight score. Each group member can keep track of the project tasks and to which member of the group it's assigned to and which user received a warning and who appealed to the SU.

Exceptional scenarios:

1. A poll can have an equal number of votes on each time slot offered for a meet-up. In this case, a new poll should be conducted.
2. A poll must be voted by all the group members, if a poll starts and there are still missing votes, a new poll cannot be conducted until the missing votes are completed and the poll remains open.

Use-Case 8: Group Vote for Issuing Warnings, Praising and Kicking a Group Member

Normal scenarios:

1. A group member with the highest votes will be issued a warning/praise according to an anonymous poll. A group member with 3 warnings will be removed from the group and 5 points would be deducted from its score. A poll can be made to kick a user from the group, such a group member would be removed and 10 points would be deducted. OU with a negative score would be removed from the system and added to the blacklist.

Exceptional scenarios: I

1. If a poll resulted in equal votes to all group members, the poll will have no effect and a new one should be issued. A group member with a negative score after receiving 3 warnings and 5 points has been deducted, will automatically be removed from the system and added to the blacklist. Same apply if a kicked user after 10 points deduction resulted with a negative score. Only one poll can be conducted at a time.
2. If OUs doesn't vote, then the poll has no effect.

Use-case 9: Polling to Kick a Group Member Out

Normal Scenarios:

1. Make a poll to kick out a group member. Once all members vote, if the majority votes to kick out, the voted-out member will be kicked-out of the group with consequences. If not the majority vote then the poll has no effect.

Exceptional Scenarios:

1. Make a poll to kick out a invalid/non-existing group member.
2. No response from other group members. The poll has no effect.
3. The vote is a tie. The poll has no effect.

Use-Case 10: Group members voting to close a group and do exit evaluation

Normal scenarios:

1. The option with the most votes will be chosen. When a group is closing, each user receives the median value of a reputation score given to him by all the group members. Each user can decide then if to put a fellow group member into their white/black box with a given reason.

Exceptional scenarios:

1. Poll ending with a tie would be conducted again. Once a group is closed it cannot be opened again (a record is kept). Users who end up with a negative score from closing the group will be removed from the system and added to the blacklist. Also, profanity will be monitored for the user's typed reason when adding another user to white/black box.

Use-Case 11: SU Assign VIP to Evaluate Closed Group.

Normal Scenarios:

1. SU will assign a VIP user to determine if a group deserves a negative/positive score which will apply on every group member. The system will keep a list of records of groups to be showcased.

Exceptional Scenarios:

1. SU assigns invalid/non-existing VIP.

2. No VIPs to assign.

Use-Case 12: OU Kick-Out if negative score

Normal Scenarios:

1. OU is kicked-out of the system when its reputation score is negative.

Exceptional Scenarios:

We could not find an exceptional scenario for this use-case.

Use-Case 13: OU Promoted to VIP and VIP Demoted to OU

Normal scenarios:

1. OU with a score higher than 30 will be promoted to VIP. VIP with a score lower than 25 will be demoted to OU.

Exceptional scenarios:

Could not find any exceptional scenarios for this use-case.

Use-Case 14: Visitors and OU Complaints to SU about a Group or Other OU

Normal scenarios:

1. SU receives the complaint and decides if: to shut down the involved group or OU. Also, SU decides if to deduct score to every user in a group individually, and finally if to kick users from the system.

Exceptional scenarios:

1. Limiting the amount of reports that a user can send to SU(especially a visitor).
2. If SU decides to deduct points, the system will check if the score resulted negative, if so, the OU will be kicked from the system automatically and will be added to the blacklist.

Use-Case 15: OU Final Login for Final Processing after being Kicked

Normal scenarios:

1. Final limited login session for kicked users before removal from the system.

Exceptional scenarios:

1. Limited access and session time before removal, with no time-out limit for a session a kicked user can stay logged in for a long period of time.

Use-Case 16: how taboo words are taken care of in the system

Normal scenarios:

1. The system itself already has a list of taboo words and if any words are used by OUs, they will be automatically converted to “***”. And everytime it happens, the OU suffers a decrease of reputation score by 1.

Exceptional scenarios:

1. If the OU keeps using the same word again after the first time, the reputation score will be decreased by 5 for each use.

Use-Case 17: OU can complement other users by approaching SU

Normal scenarios:

1. SU can increase score after a certain OU has received 3 compliments.

Exceptional scenarios:

1. User signs up with multiple accounts and compliments himself through them. In that case, if the accounts used to compliment the same OU do not have any group or project activities, the compliments would be invalidated.

Use-Case 18: VIP Vote for Democrate SU

Normal scenarios:

1. Equally distributed votes for an SU among the VIPs, resulting with no one chosen and another vote should take place.

Exceptional scenarios:

1. Equally distributed votes for an SU or no votes for any SU among the VIPs, resulting with no one chosen and another vote should take place.

Creative Feature 1:

Normal Scenarios:

1. Member inserts valid commits number for a project. The commit number is then updated.

Exceptional Scenarios:

1. Member inserts invalid commits number for a project. The commit number is unchanged.

Creative Feature 2:

Normal Scenarios:

1. Displays a hierarchical tree that describes the roles of different group members.

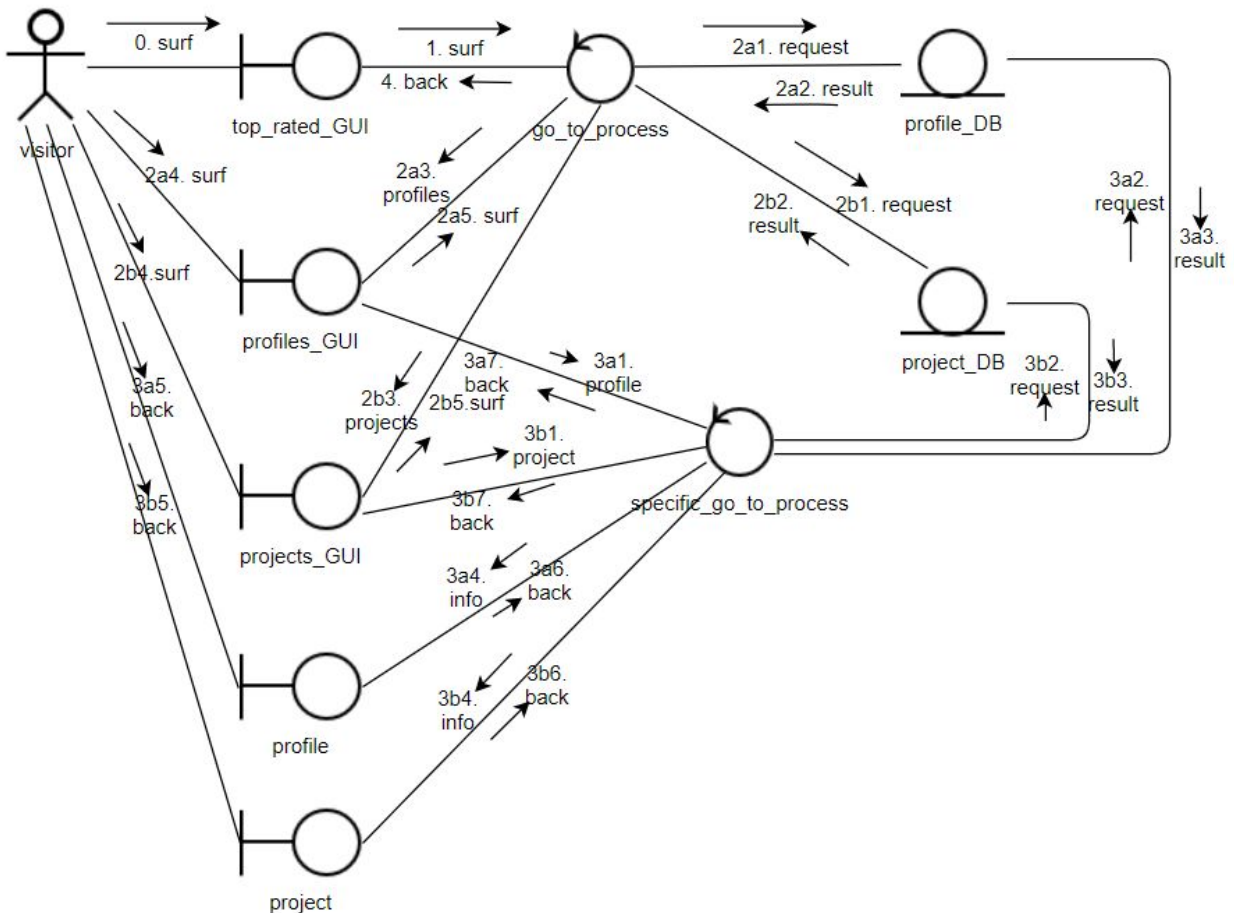
Exceptional Scenarios:

We could not find any exceptional scenarios for this use-case..

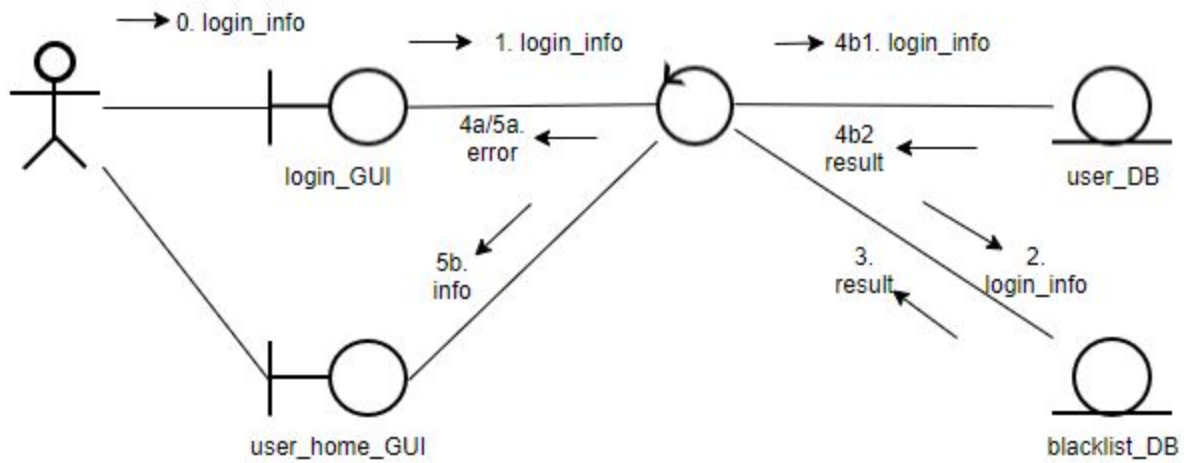
b. Collaboration or Sequence Class Diagram for each Use-Case

Choose 3 or more use cases: draw the Petri-nets instead of class diagrams. The number of the use-case corresponds to the subsection above.

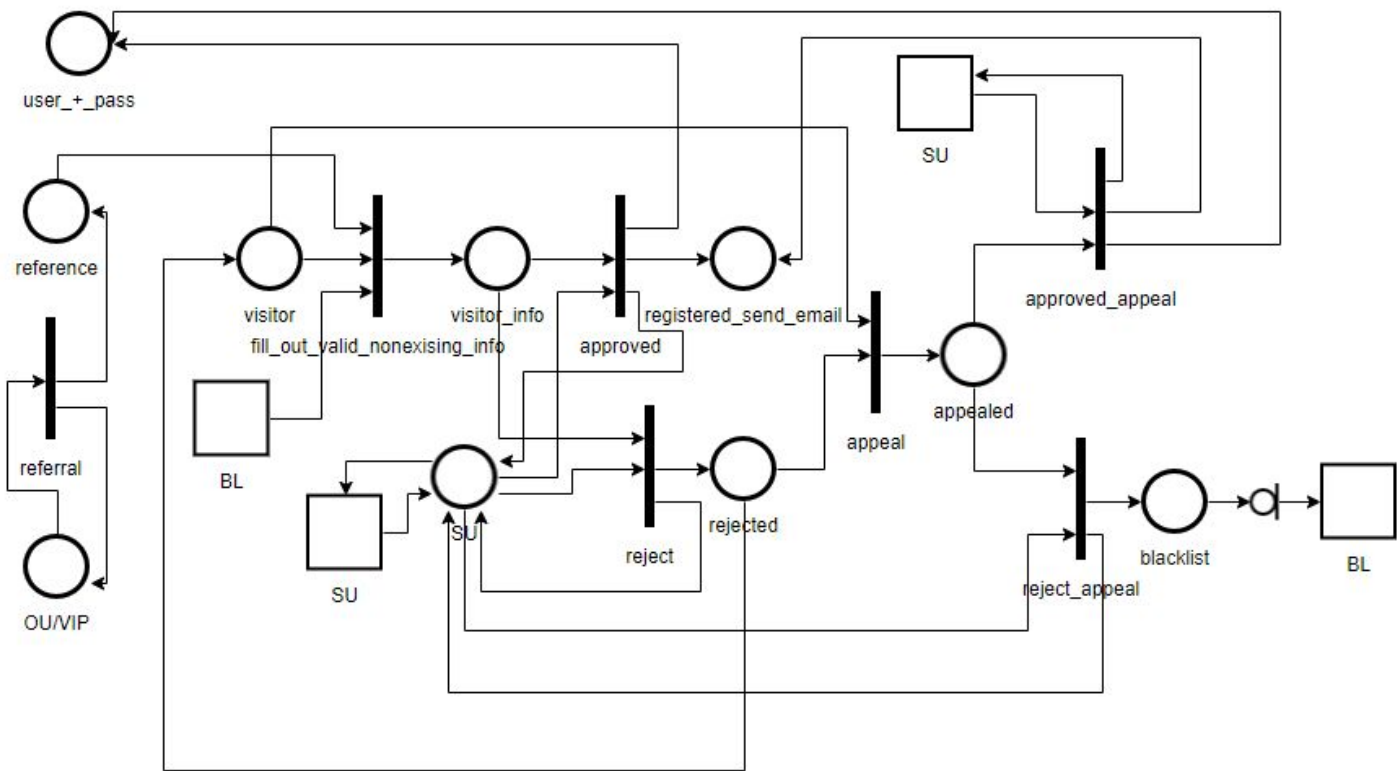
1. Visitor browsing/surfing the application



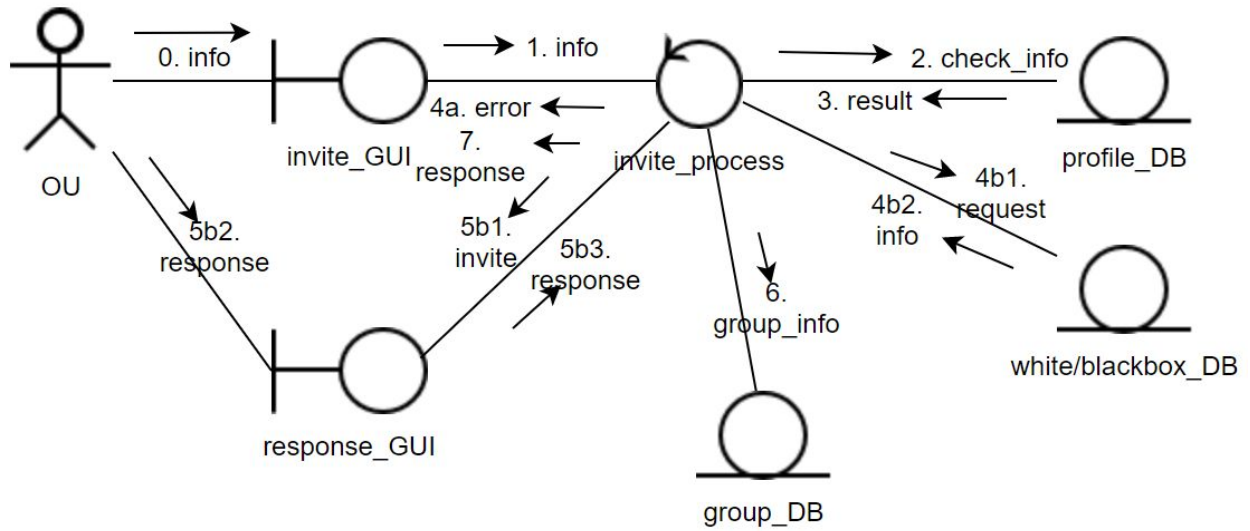
2. Login (See use-case #15 for kicked-out user login)



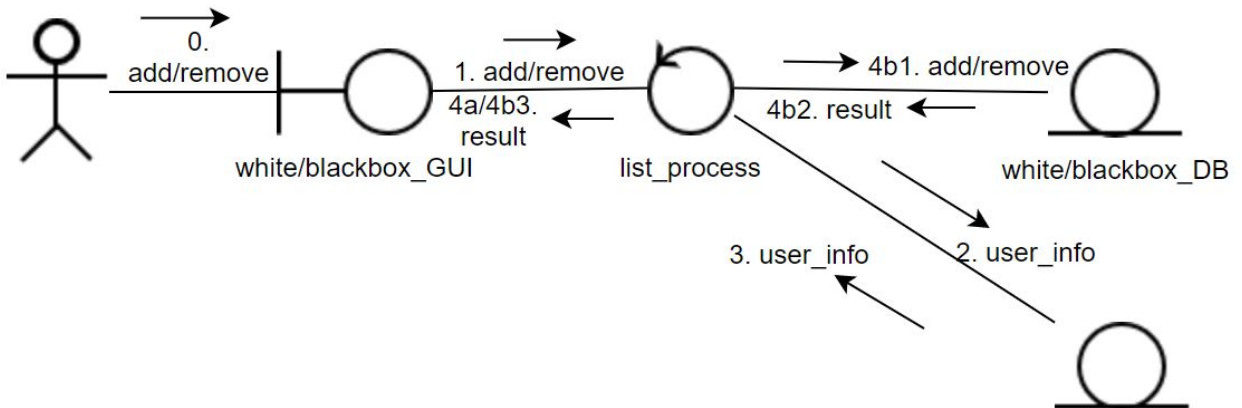
3. (Petri-Net) Visitor registration



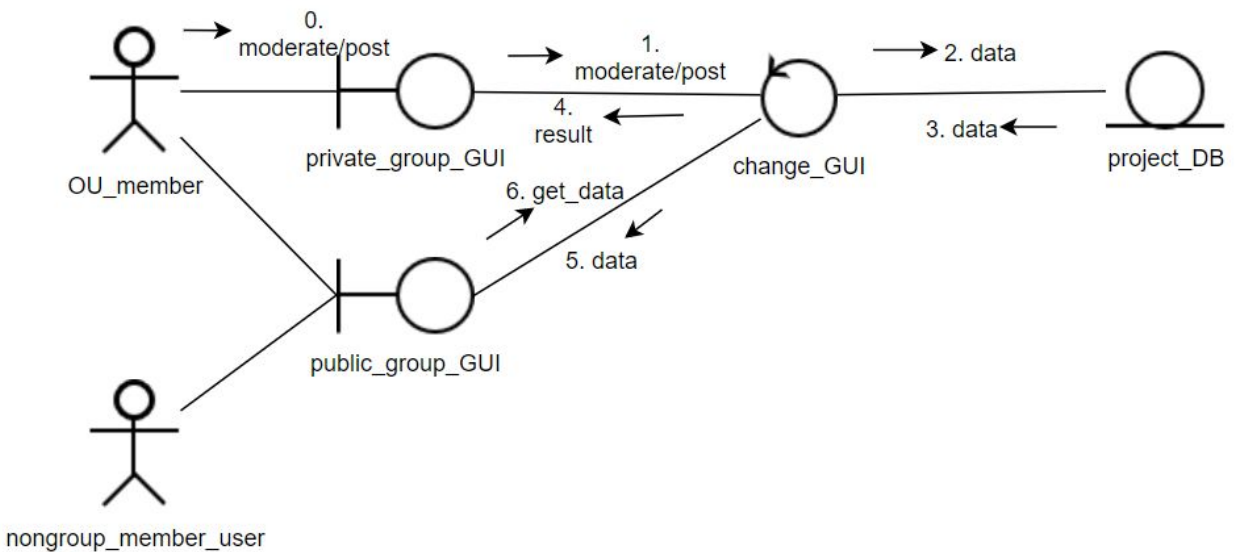
4. From Groups/Inviting



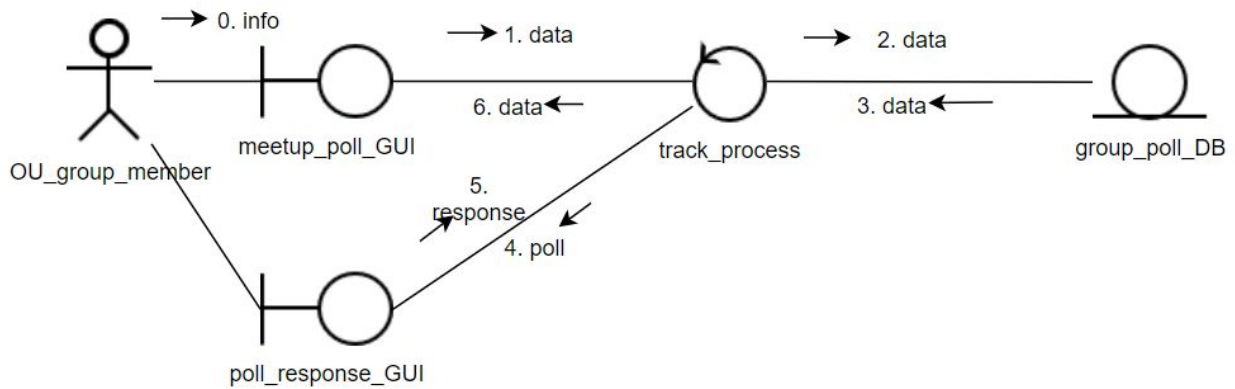
5. User whitelist or blacklist users.



6. Public/Private group page and Moderation/Posting

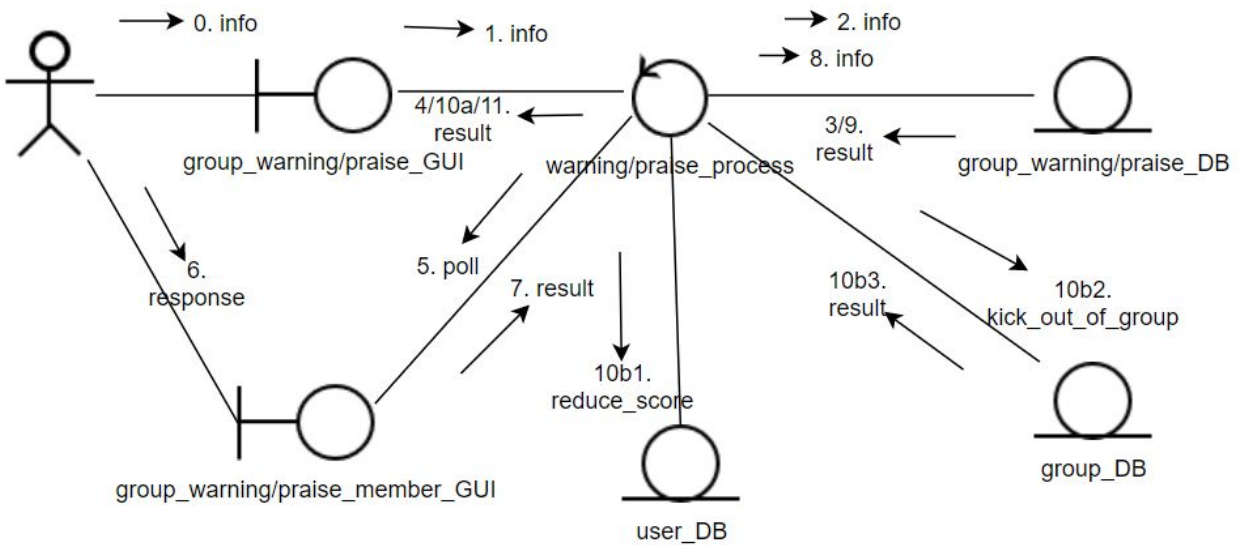


7. Polling

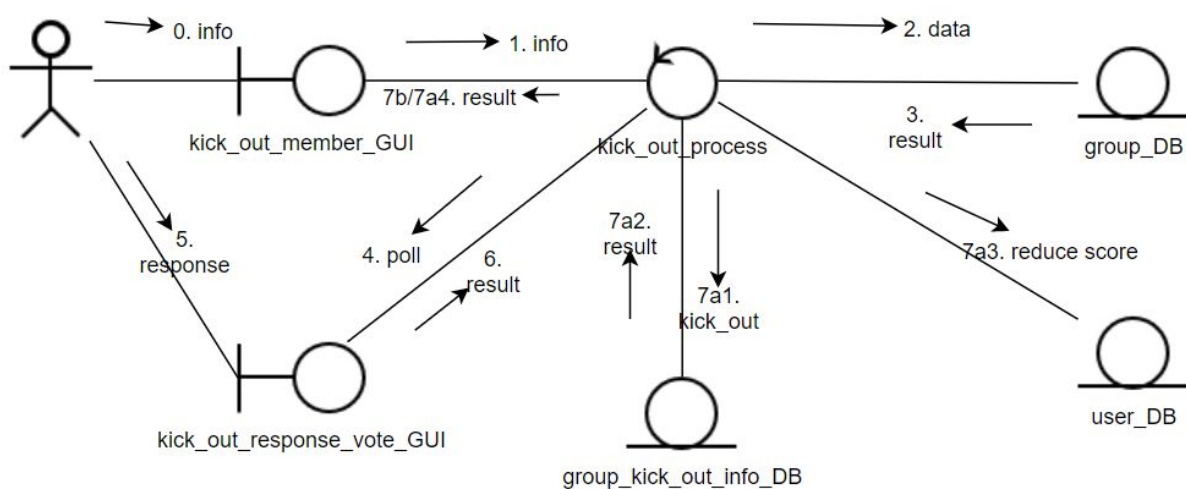


8.

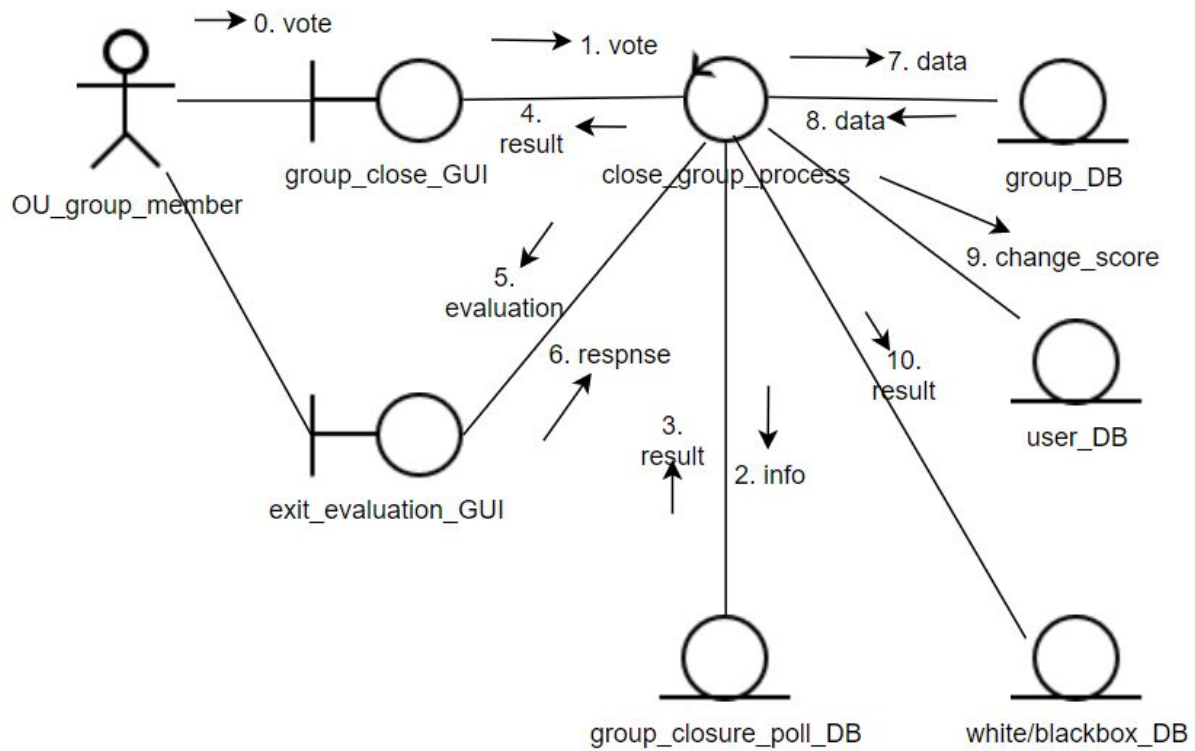
Group member issue warning/praise to OU



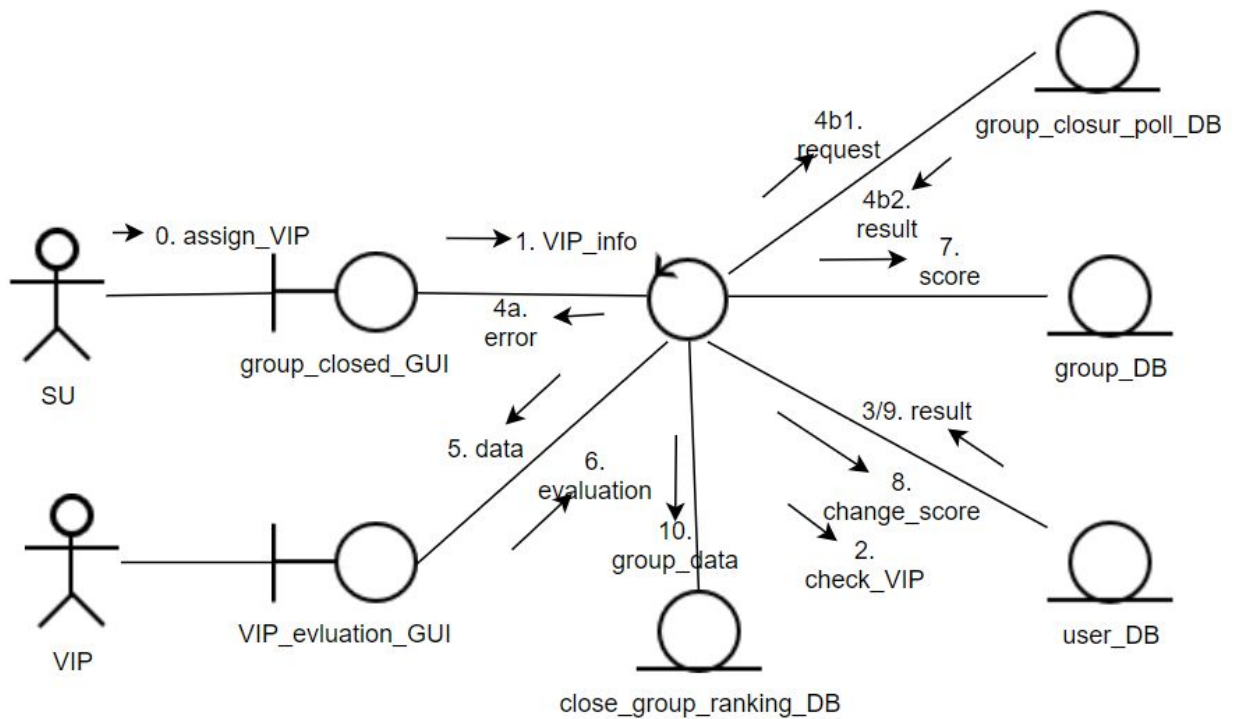
9. Vote to kick out group member



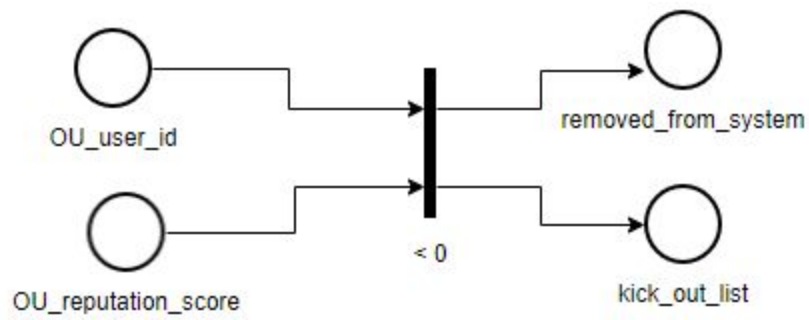
10. Group Closure and Exit evaluation



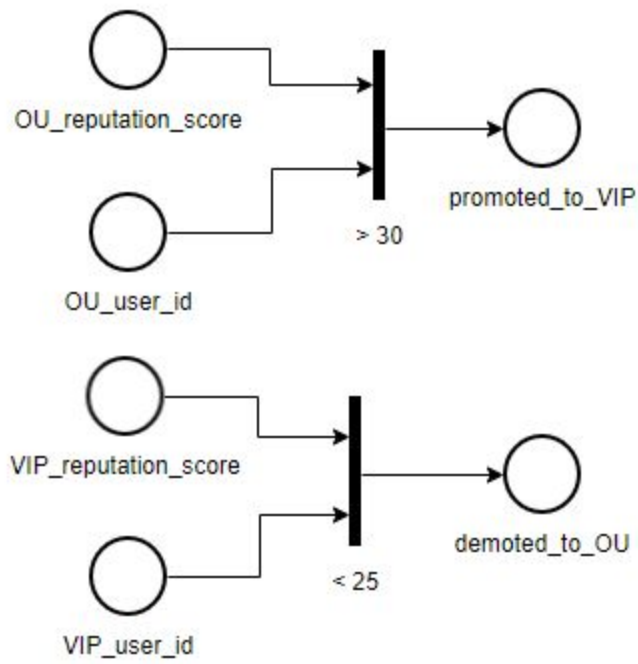
11. SU assign VIP to do evaluation



12. Kick Out of system if OU reputation score is negative

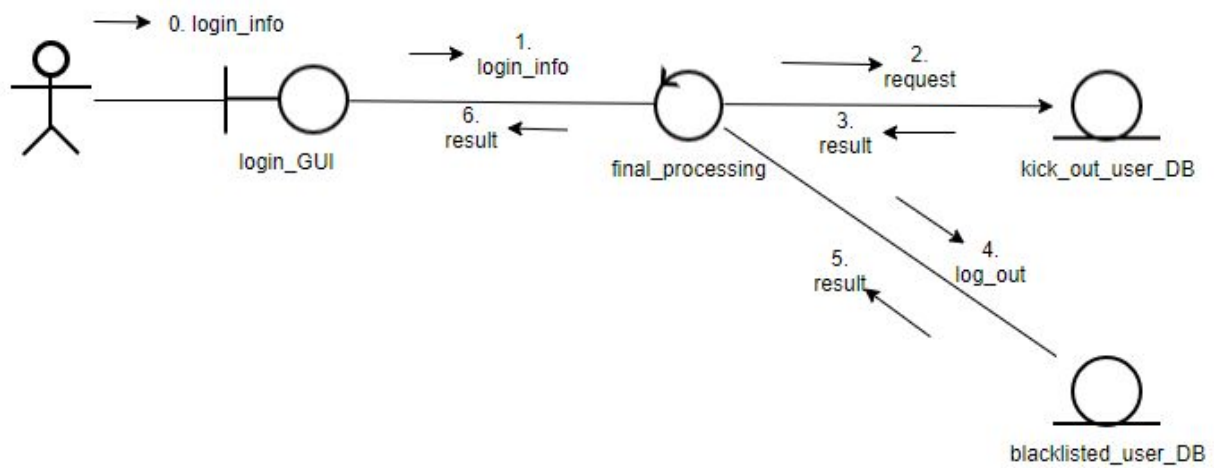


13. (Petri-Net) Promote OU. Demote VIP.

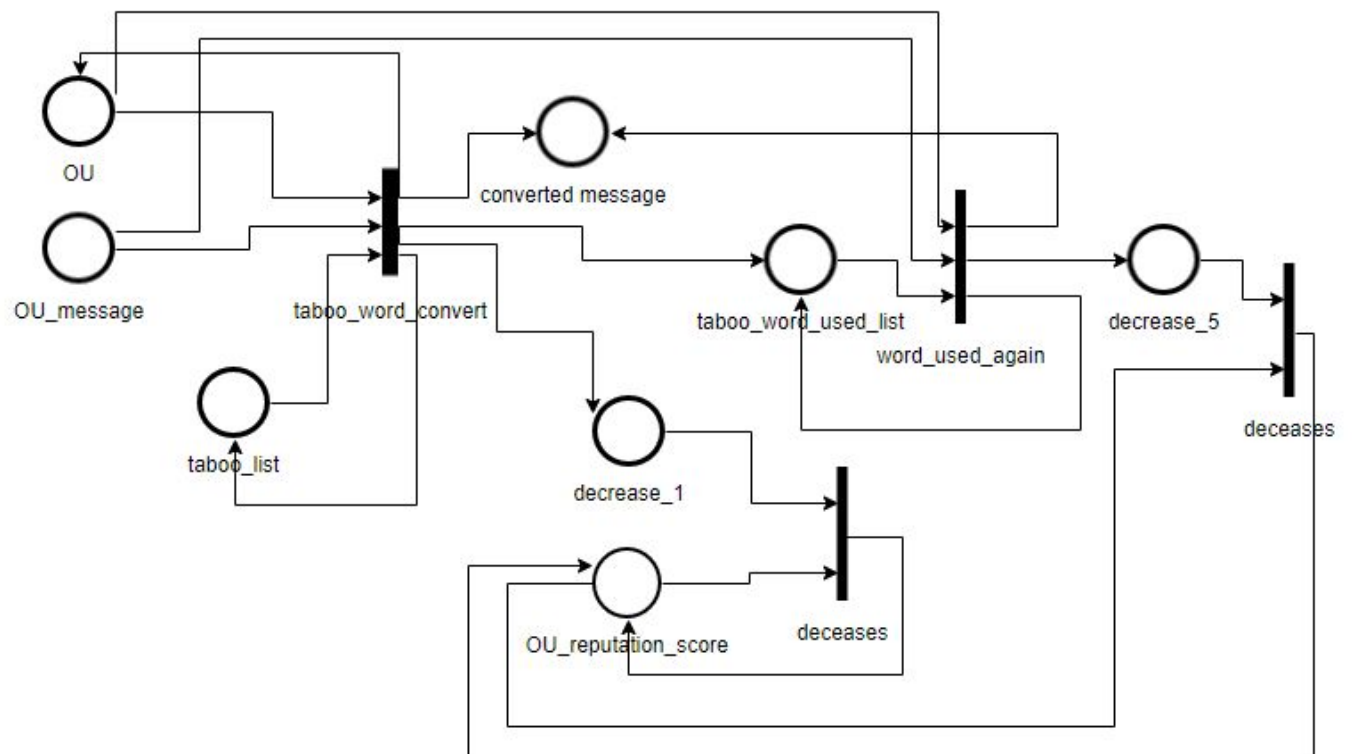


14. OU/visitor Complain about OU or group.

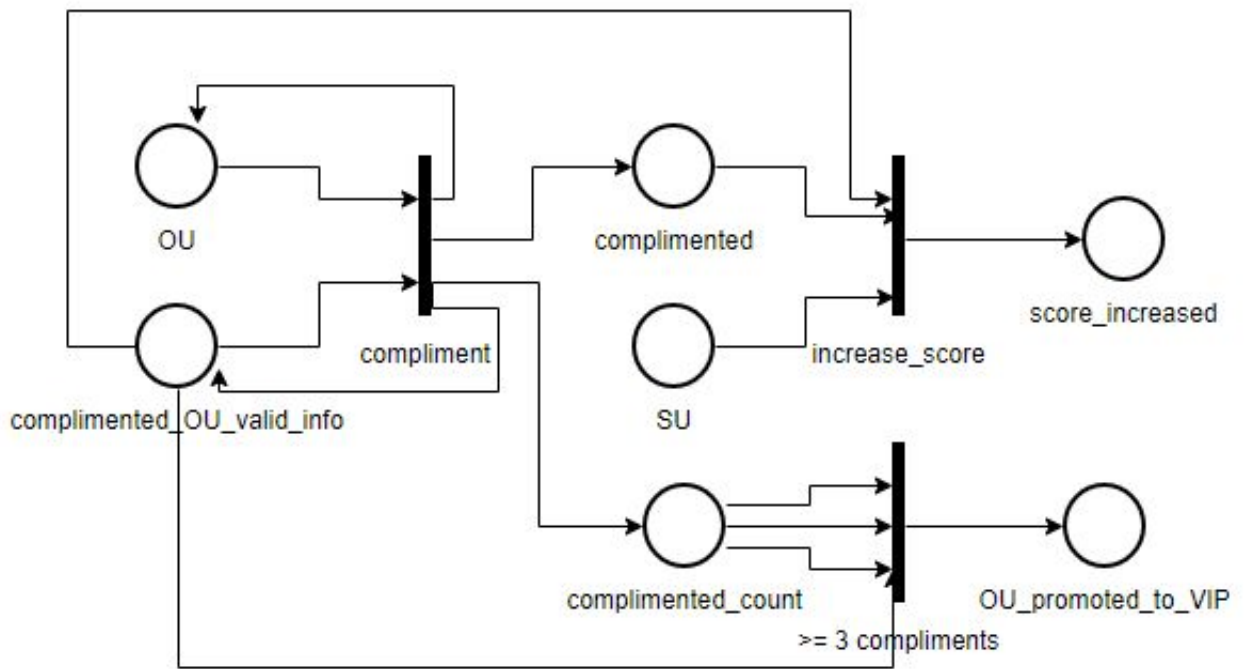




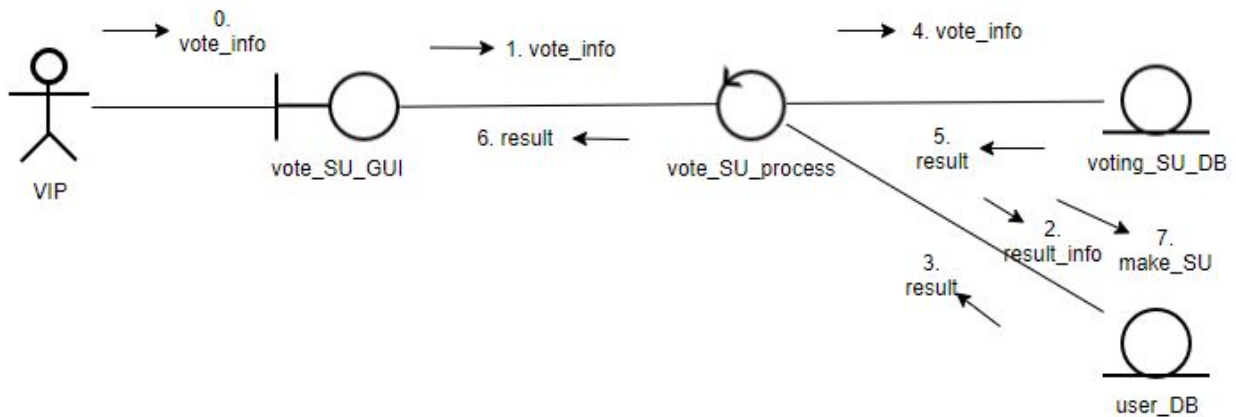
16. (Petri-Net) Taboo-Word



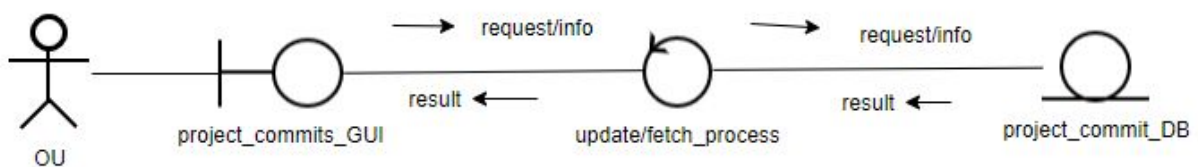
17. (Petri-Net) OU compliments.



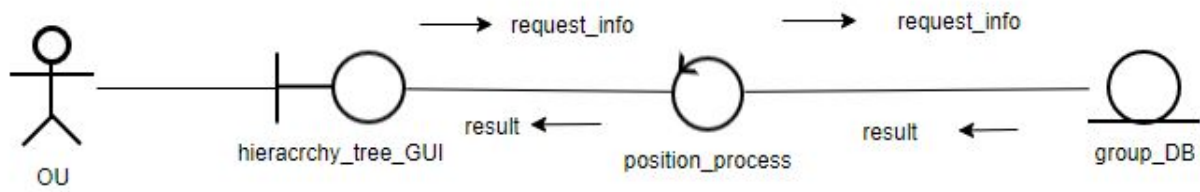
18. VIP votes for a SU.



19. Creative Feature 1:



20. Creative Feature 2: Group/Project Hierarchy Tree



3. E-R diagram for the entire system

4. Detailed design

<u>Index</u>	<u>SYSTEM METHODS</u>
1	chatFilter method for taboo words ran on every new user input, compares text to stored taboo words, if match is found text is converted to *** and updatePoints method is called. Calls updateCommit method at the beginning
2	updateCommit method, accepts groupId as parameter. All chat counts as contribution to the group; adds current date to group's list of dates if not already there, and increments the number of commits for that date by one.
3	updatePoints method taking a user and an amount to be deducted or added as parameter. Will have if statements at the end that check the amount of points and call the required system methods if conditions are met (e.g. if points < 0 call blacklist method) , calls updateRanking method
4	Blacklist method ran at end of every method that updates points, accepts userId as parameter, moves the user to the "blacklist" database, updateRanking method then called, user is returned to login screen
5	Promotion method that is ran at the end of every method that updates points, will update user's flag to a vip type
6	Demotion method that is ran at the end of every method that updates points, will update users flag to ou type
7	updateRanking method, accepts a userId as parameter and locates its position in the ranking database, compares userId's score to scores around it and shifts accordingly
8	updateRankingGrp method, accepts a groupId as parameter and locates its position in the ranking database, compares groupId's score to scores around it and shifts accordingly
9	login method that takes users inputs from login textboxes and compares them to stored values to determine if invalid credentials or successful login (this will return an int flag representing what type of user they are), This method will check if the user was kicked or blacklisted, if they were kicked then they will be allowed 1 last login before their account is moved to blacklist

10	register method, displays the registration GUI prompting visitor to fill it out, prompts superuser to accept or reject registration
11	appeal method, takes an email as a string for parameter, finds the form that corresponds to the email and re-adds it to the superuser's pending registrations list with the added appeal message
12	displayRankings method, displays the top 3 user profiles and top 3 groups, called on program startup or by the browser method
13	seeMoreUsers method, accepts an int parameter, displays the entire list of user profiles for surfing in an order based on the value of the parameter
14	seeMoreGroups method, accepts an int parameter, displays the entire list of groups for surfing in an order based on the value of the parameter
15	onClickProfile method, accepts a userId as parameter, obtains and displays the contents of the clicked user
16	onClickGroup method, accepts a groupId as parameter, obtains and displays the contents of the clicked group if it is public
17	homepage method, accepts a userId as parameter, called on home tab click or on login, obtains and loads data for the homepage
18	Browser method, called on browser tab click, displays browser GUI and calls displayRankings method
19	inbox method, accepts a userId as parameter, called on inbox tab click, obtains and loads data for the current user's inbox
20	groupManagement method, accepts a userId as parameter, called on group management tab click, obtains and loads the user's affiliated groups and their information
21	adminSU method, accepts a userId as parameter, called on admin(SU) tab click, if user is of type superuser then the data is obtained and loaded, otherwise no data will be presented
22	adminVIP method, accepts a userId as parameter, called on admin(VIP) tab click, if user is of type VIP then the data is obtained and loaded, otherwise no data will be presented
23	projectManagement method, accepts a userId as parameter, called on creative features tab click, displays chosen group's member hierarchy and contributions

<u>index</u>	<u>SUPERUSER - ONLY METHODS</u>
24	Kick method, accepts a user id as parameter, moves the user to the “kicked” database, the user is then returned to the login screen and a message saying they were kicked is displayed
25	Shutdown group method, takes a group id as parameter and removes that group from the list of available groups
26	approveRegist method, accepts an email address as a string as parameter, a random account id and password will be generated and emailed to the new user. Calls the OU/VIP initialScore method, calls updateRanking method
27	rejectRegist method, accepts an email address as a string as parameter, emails visitor of rejection and allows them to appeal if first time, otherwise the visitor is blacklisted
28	assignVIP method, available after a group closure, takes a vip userId as parameter, calls the groupScore method. Takes returned value and adds/subtracts from all members involved, sets group’s score as returned value and calls the updateRankingGrp method

<u>index</u>	<u>ORDINARY USER/VIP – ONLY METHODS</u>
29	groupScore method, accepts a vip userId as parameter, prompts the user to enter a score for the given group, returns an int value
30	InitialScore method accepts a user id as parameter, prompts that user to enter an initial reputation score for the visitor that they referred, max amount based on flag type
31	complement method, takes inputs from text fields and prompts the superuser to read what was written and provide an option to increase points, only accepts userid and a string as parameter, if user id is complimented 3 times the promotion method is called
32	Complaint method, accepts a user id or group id as parameter, prompts the superuser to read what was written and punish the users involved
33	Invite method, accepts a user id as parameter, sends a prompt to that user asking if they want to join the group, allows them to accept or reject, the text

	from the reject text box prompt is returned if rejected, if the requesting user is already blackboxed then the automated message is returned
34	addToWB method, accepts a user id as parameter and adds them to calling user's whitebox
35	addToBB method, accepts a user id as parameter and adds them to calling user's blackbox
36	setAutoMsg method, sets the message for black-box'ed user's invite rejections
37	voteSU method, accepts a vip userId as parameter, prompts that user to vote for another vip that they want as a superuser, once all vip's have voted a superuser is chosen

<u><i>index</i></u>	<u><i>GROUP RELATED METHODS</i></u>
38	createGroup method, makes a group web page available, creates a new group id and adds it to list of available groups
39	visibility method, accepts groupid and an int as parameter, sets visibility variable of groupid accordingly
40	VoteClose method, prompts all group members to vote if they want to close the group, on unanimous vote the exit evaluation method is called
41	exitEvaluation method, brings up a screen displaying all other group members and prompts the user to enter a score for each member, then assigns each group member a score based on the average of the scores they received, users have the option to white or black box members during evaluation
42	closeGroup method, removes the group from each members list of available groups and prompts the superuser to assign a vip for group evaluation
43	voteKick method, accepts a userId as parameter, prompts all other group members to vote if they wish to kick userId, if the vote is unanimous then the removeFromGroup method is called
44	removeFromGroup method, accepts a userId and a point amount as a parameter, removes the current group from the user's list of groups and calls updatePoints to deduct the amount given

45	meetupPoll method, prompts group members to select a time they prefer for a meetup, after all votes are accounted for the time with the most votes is returned
46	praiseVote method, accepts a userID as parameter, prompts members to vote to praise userID, on unanimous vote number of userID's praises is incremented and a praise message is displayed
47	warningVote method, accepts a userID as parameter, prompts members to vote to warn the given userID, on unanimous vote the user's number of warning will be incremented, if three warnings then removeFromGroup is called, else a warning message is displayed

System controlled methods

- 1) `def chatFilter(groupID, userID, userInp) :`
`updateCommit(groupID)`
`Arr = [list of taboo words]`
`For i in Arr :`
`If (userInp == i) :`
`userInp = "***"`
`display userInp to chatbox`
`if(word used three times or more)`
`updatePoints(userID, -5)`
`else`
`updatePoints(userID, -1)`
`display userInp to chatbox`
- 2) `def updateCommit(groupID) :`
`//group has a dictionary with key (date) : value (# of commits)`
`//getCommit is a getter for the dictionary variable`
`date = obtain current date`
`if((groupID.getCommits()).get(date, -1) == -1) :`
`(groupID.getCommits()).add(date, 1)`
`else :`
`val = (groupID.getCommits()).get(date)`
`(groupID.getCommits()).add(date, val + 1)`
- 3) `def updatePoints(userID, amount) :`
`userID.addToScore(amount)`
`if(userID.getScore() < 0) :`
`blacklist(userID)`
`elif(userID.getScore() > 30 and userID.getFlag() == ou) :`

- ```

 promotion(userID)
 elif(userID.getScore() < 25 and userID.getFlag() == vip) :
 demotion(userID)
 else :
 updateRanking(userID)

```
- 4) 

```

def blacklist(userID) :
 userID.setScore(-1)
 updateRanking(userID)
 move user to blacklist database
 return user to login screen

```
  - 5) 

```

def promotion(userID) :
 userID.setFlag(vip)
 updateRanking(userID)

```
  - 6) 

```

def demotion(userID) :
 userID.setFlag(ou)
 updateRanking(userID)

```
  - 7) 

```

def updateRanking(userID) :
 search for userID in list of sorted users by rank
 if(userID.getScore() == -1) :
 remove from list of rankings
 elif(userID.getScore() > score one position before it) :
 while (userID score > score before it) :
 swap positions
 elif(userID.getScore() < score one position after it) :
 while (userID score < score after it) :
 swap positions

```
  - 8) 

```

def updateRankingGrp(groupID) :
 groupScore = groupID.getScore()
 for i in list of groups :
 if(groupScore > list[i].getScore()) :
 insert groupID in front
 Break;

```
  - 9) 

```

def login() :
 displays login GUI
 userInp = text from "username"
 passInp = text from "password"
 for i in blacklist list :
 if account information matches a userID:

```

```

 display "account is blacklisted" message
 return -1;
 for i in kicked list :
 if account information matches a userID:
 display "last login" message
 move userID to blacklist list
 homepage(userID)
 return -1;
 for i in users list :
 if account information matches a userID:
 if(first login) :
 display "set a password" message
 userID.setPassword(user input)
 homepage(userID)
 return 0;

10) def register() :
 display registration GUI
 if(register clicked again) :
 if(some fields are empty) :
 display "incomplete form"
 else :
 add form to superuser pending registrations list

11) def appeal(email) :
 displays appeal GUI
 for(i in list of rejected forms) :
 if(form contains email) :
 add form and appeal message to superuser pending registrations list

12) def displayRankings() :
 for i in range(3) :
 add list of ranked users at i to users textfield
 add list of ranked groups at i to groups textfield

13) def seeMoreUsers(order) :
 display profiles GUI
 if(order == 1) :
 for i in list of ranked users :
 add list of ranked users at i to textfield
 elif(order == -1) :
 x = size(list of ranked users) - 1
 for i in range(x, -1, -1) :

```



add list of ranked users at i to textfield

- ```
14) def seeMoreGroups(order) :  
    display groups GUI  
    if(order == 1) :  
        for i in list of ranked groups :  
            add list of ranked groups at i to textfield if public  
    elif(order == -1) :  
        x = size(list of ranked groups) - 1  
        for i in range(x, -1, -1) :  
            add list of ranked groups at i to textfield if public  
    elif(order == 0) :  
        for i in list of all groups :  
            add list of all groups at i to textfield if public  
  
15) def onClickProfile(userID) :  
    display profile GUI  
    for i in userID.getGroups() :  
        add userID.getGroups() at i to groups textfield  
    for i in userID.getLanguages() :  
        add userID.getLanguages() at i to program_languages_used textfield  
    display userID.getEmail() in email textfield  
    display userID.getInterests() in interests textfield  
    display userID.getScore() in reputation textfield  
  
16) def onClickGroup(groupID) :  
    if(groupID.getVisibility() == 1) :  
        display "group is private" message  
    else :  
        display group page GUI  
        display groupID.getMembers() in group members textfield  
        display groupID.getProjects() in projects textfield  
        display groupID.getLanguages() in program_languages_used textfield  
  
17) def homepage(userID) :  
    display homepage GUI  
    for i in userID.getGroups() :  
        add userID.getGroups() at i to groups textfield  
    for i in userID.getLanguages() :  
        add userID.getLanguages() at i to program languages used textfield  
    display userID.getEmail() in email textfield  
    display userID.getInterests() in interests textfield  
    display userID.getScore() in reputation textfield
```

- 18) `def browser() :`
`display browser GUI`
`displayRankings()`
- 19) `def inbox(userID) :`
`display user.getInvitations() in invitations textfield`
`display user.getPolls() in other_messages textfield`
- 20) `def groupManagement(userID) :`
`display group management GUI`
`group1 = userID.getGroups()[0]`
`display group1.getMembers() in group_members textfield`
- 21) `def adminSU(userID) :`
`if(userID.getFlag() == superuser) :`
`display admin SU GUI`
`display list of pending registrations in account_approval textfield`
`display list of complaints in group_member_complaints textfield`
`display list of compliments and complaints in compliments_&_complaints`
`textfield`
`display list of vip assignment prompts in assign_vips_to_group textfield`
- 22) `def adminVIP(userID) :`
`if(userID.getFlag() == vip) :`
`display admin VIP GUI`
`display list of pending group evaluations in group_evaluations textfield`
- 23) `def projectManagement(userID) :`
`display creative features GUI`
`groups = userID.getGroups()`
`selectedGroup = groups[value from widget]`
`display selectedGroup's member list(already in join date order) in hierarchy tree pane`
`display selectedGroup's commits with seaborn/matplotlib graphing in tracker pane`

Superuser controlled methods

- 24) `def kick(userID) :`
`move user to kicked database`
`return user to the login page`
`display kicked message`
- 25) `def shutdownGroup(groupID) :`
`remove groupID from group list`

removes groupID from all member's list of affiliated groups

- 26) def approveRegist(email) :
 generate random userID
 generate random password
 add userID to list of users
 email userID and password to the new user
 userID.setScore(initialScore(input from referral textbox))
 updateRanking(userID)
- 27) def rejectRegist(email) :
 if(first time) :
 add form information to list of rejected forms
 email visitor of rejection and chance to appeal
 else :
 email visitor of final rejection decision
 blacklist(email)
- 28) def assignVIP(userID, groupID) :
 while(userID.getFlag() == ou) :
 prompt superuser to enter another userID
 val = groupScore(userID)
 for i in list of members:
 updateScore(i, val)
 groupID.setScore(val)
 updateRankingGrp(groupID)

OU/VIP controlled methods

- 29) def groupScore(userID, groupID) :
 val = prompts userID to enter a score for the group
 return val
- 30) def initialScore(userID) :
 if(userID.getFlag() == ou) :
 initScor = prompt the user to give their friend a score from 0 to 10
 return initScor
 else :
 initScor = prompt the user to give their friend a score from 0 to 20
 return initScor
- 31) def complement(userID, text) :
 if(number of compliments for this user < 3) :
 prompts superuser to read text

```

        prompts superuser to increase score
    else
        promotion(userID)

32) def complaint(id, text) :
    prompts superuser to read text
    prompts superuser to shutdown group, deduct points, or kick

33) def invite(userID) :
    if current user not in userID's blackbox or whitebox :
        prompts userID to accept or reject the invite
        asks userID if they want to whitebox or blackbox
        if(whitebox) :
            addToWB(self.userID)
        else :
            addToBB(self.userID)
        if rejected :
            displays text from textbox
        else:
            userID added to groupID
            userID gains access to groupID features
    elif(current user in userID's whitebox) :
        userID added to groupID
        userID gains access to groupID features
    else :
        display userID.getAutoMsg()

34) def addToWB(userID) :
    adds userID to the current user's whitebox

35) def addToBB() :
    adds userID to the current user's blackbox

36) def setAutoMsg(text) :
    autoMsg = text

37) def voteSU(userID) :
    if(self.flag == vip) :
        while(userID.getFlag() == ou) :
            prompt the user to enter another userID
        increment vote count for userID
        if(sum of all votes == # of vips) :
            user with highest vote becomes superuser

```

Group related methods

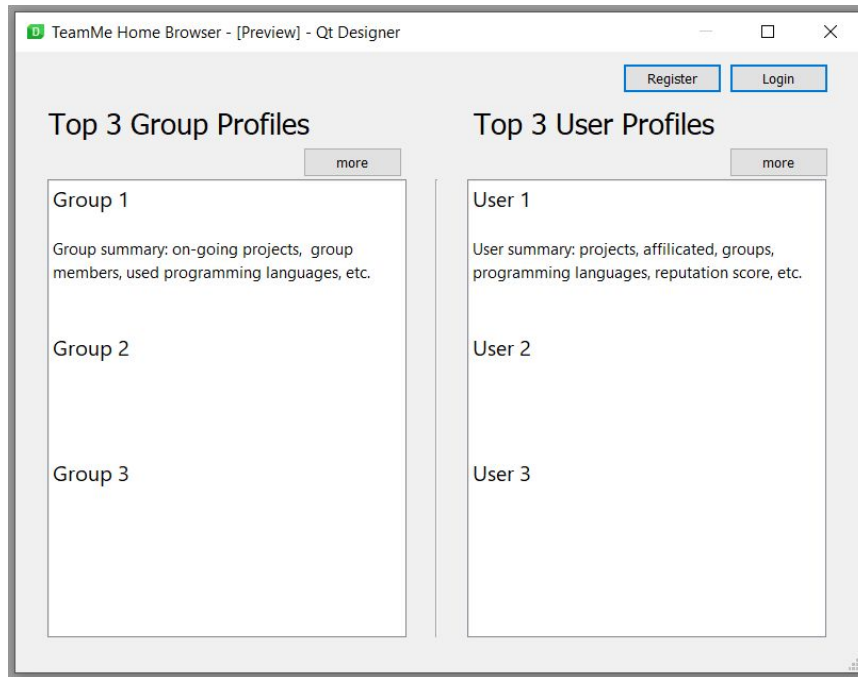
- 38) def createGroup() :
 creates a new groupID
 adds groupID to list of available groups
 adds group to creator's grouplist
 makes group page available to creator
 returns groupID
- 39) def visibility(groupID, val) :
 if(val != 0 or val != 1)
 invalid parameter message
 groupID.setVisibility(val)
- 40) def voteClose() :
 display a prompt to all group members, let yes clicked = 1, no clicked = 0
 adds votes to self.groupID's group_closure_and_evaluation_DB
 if(number of yes' equal to group size)
 clear group_closure_and_evaluation_DB
 exitEvaluation()
- 41) def exitEvaluation() :
 displays exit evaluation UI for each member
 average of inputs stored in group_closure_and_evaluation_DB
 position of input based on pos of members in member list
 pos = 0
 if(all members done with evaluation) :
 for i in member list:
 updatePoints(i, group_closure_and_evaluation_DB[pos])
 pos = pos + 1
 closeGroup(groupID)
- 42) def closeGroup(groupID) :
 removes groupID from each member's list of groups
 visibility(groupID, 0)
 prompts the superusers to enter a vip userID for group evaluation
- 43) def voteKick(userID) :
 prompts group members to vote to kick userID
 counter = 0;
 increment on yes click
 if(number of yes' equal to group size - 1) :
 removeFromGroup(userID, -10)

- 44) def removeFromGroup(userID, amount) :
 add userID to group_kick_out_poll_DB
 remove group from userID's list of groups
 updatePoints(userID, amount)
- 45) def meetUpPoll() :
 prompts group members to select a time for a meeting
 votes stored in group_poll_DB
 if(number of votes equal to group size) :
 return time with most votes
- 46) def praiseVote(userID) :
 prompts group members to vote to praise userID
 counter = 0;
 increment on yes click
 if(number of yes' equal to group size - 1) :
 increment value for user in group_praise_DB
 display a praise message to userID
- 47) def warningVote(userID) :
 prompts group members to vote to warn userID
 counter = 0;
 increment on yes click
 if(number of yes' equal to group size - 1) :
 increment value for user in group_warning_DB
 if(value in group_warning_DB == 3) :
 removeFromGroup(userID, -5)
 else :
 display a warning message to userID

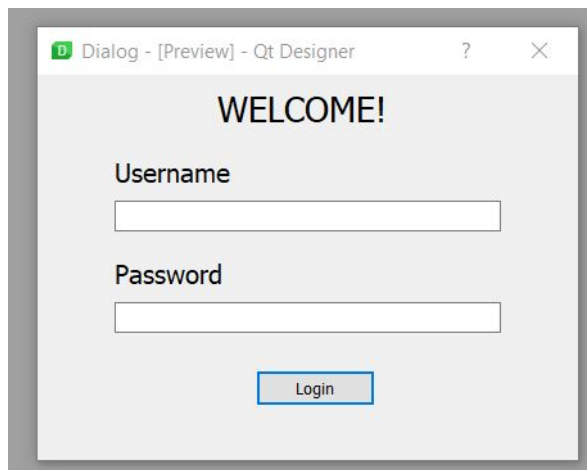
5. System Screens

Demonstrate major GUI screens of the system and a prototype of one functionality of your own choice.

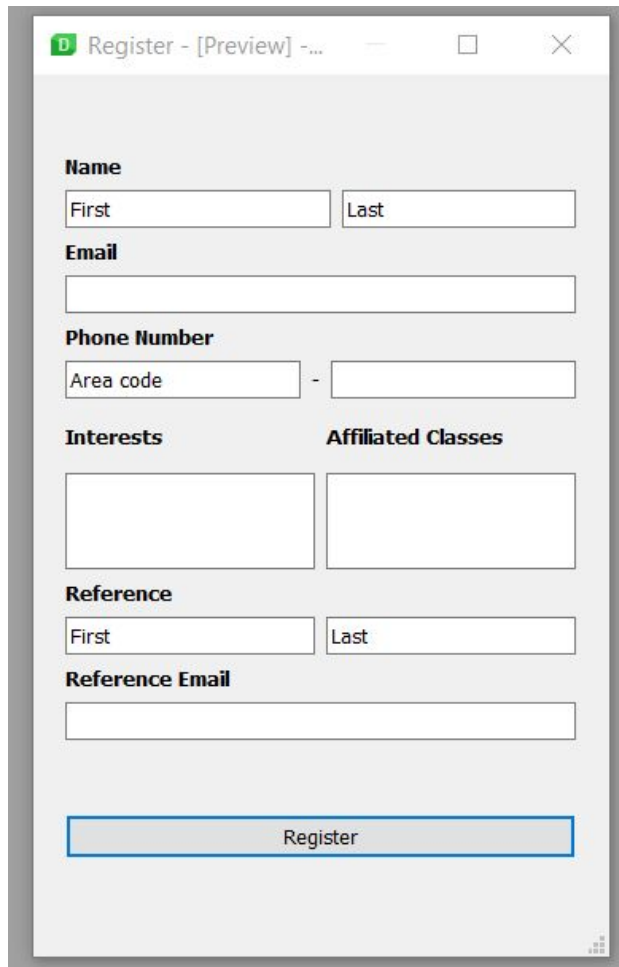
top_rated_GUI: a browsing screen for visitors to see top rated group and user profiles.



login_GUI: a dialog that allows visitors to login



register_GUI: a dialog that allows visitors to register with required information



The image shows a 'Register - [Preview]' dialog box with a light gray background. It contains several input fields and a button. At the top, there's a title bar with a green icon, the text 'Register - [Preview] -...', and standard window controls. The main area has the following sections: 'Name' with 'First' and 'Last' text boxes; 'Email' with a single text box; 'Phone Number' with 'Area code' and a text box separated by a hyphen; 'Interests' and 'Affiliated Classes' each with a large text box; 'Reference' with 'First' and 'Last' text boxes; and 'Reference Email' with a single text box. At the bottom is a 'Register' button. A small version number '1.0.0.0' is visible in the bottom right corner.

Name

First Last

Email

Phone Number

Area code -

Interests **Affiliated Classes**

Reference

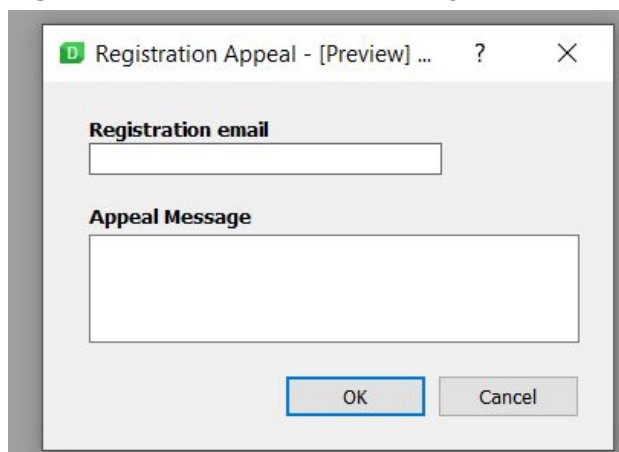
First Last

Reference Email

Register

1.0.0.0

registration_appeal_GUI: a dialog that allows visitors to send appeal of registration to the SU.



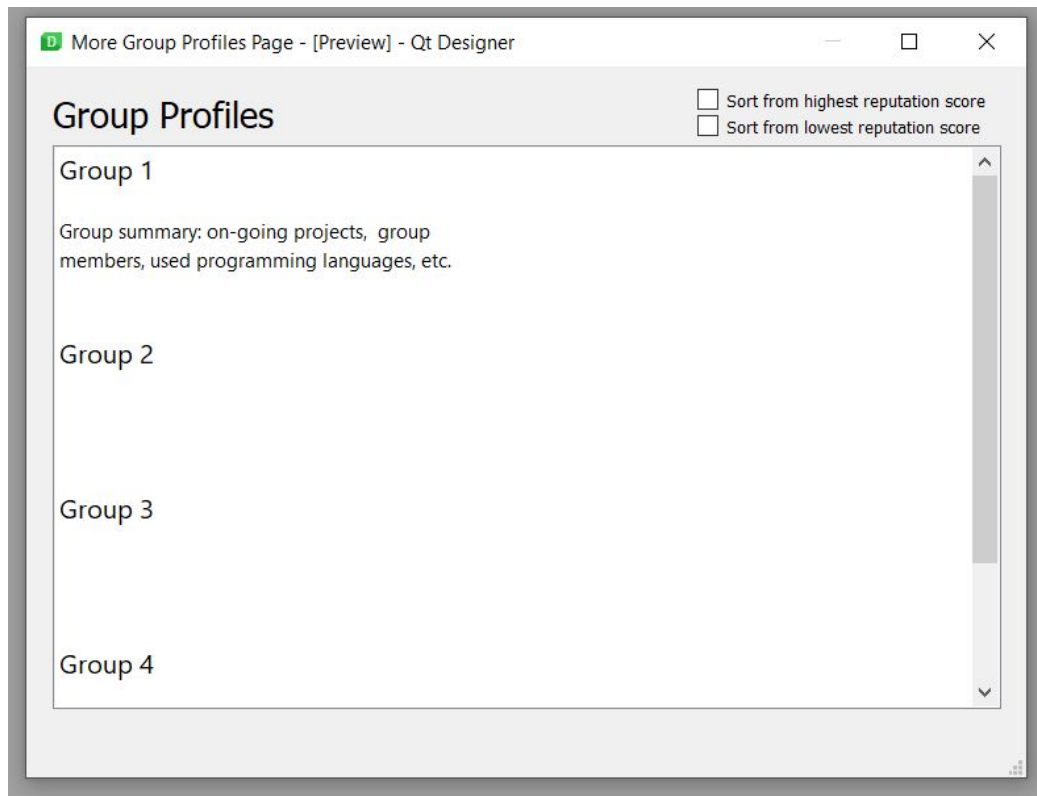
The image shows a 'Registration Appeal - [Preview]' dialog box with a light gray background. It contains two input fields and two buttons. The title bar has a green icon, the text 'Registration Appeal - [Preview] ...', a question mark icon, and a close button. The main area has the following sections: 'Registration email' with a text box; and 'Appeal Message' with a larger text box. At the bottom are 'OK' and 'Cancel' buttons.

Registration email

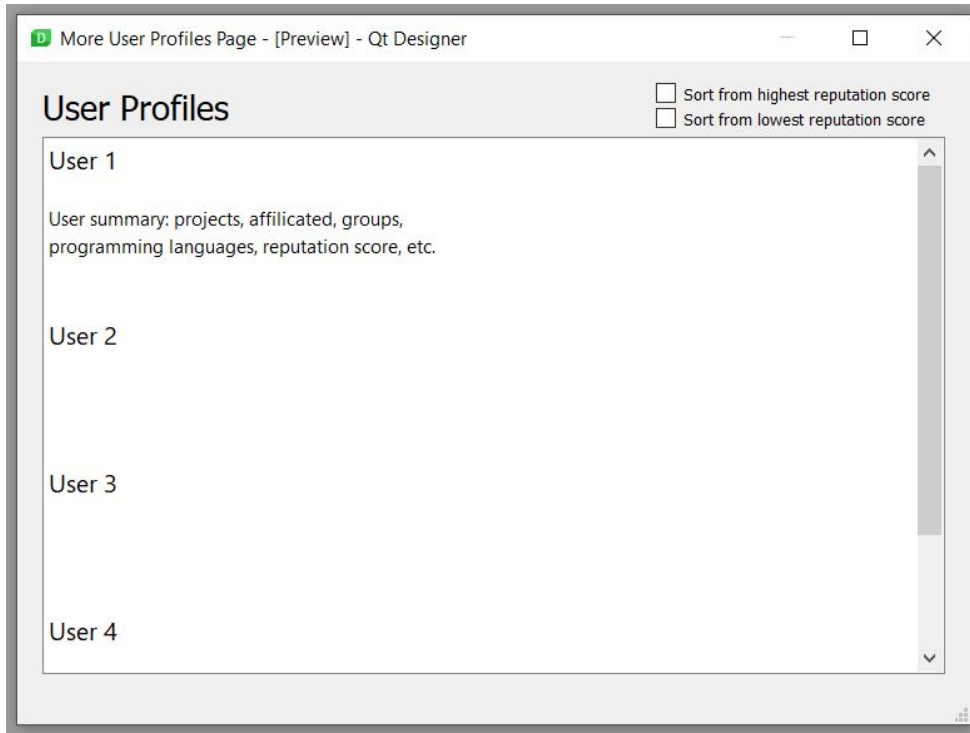
Appeal Message

OK Cancel

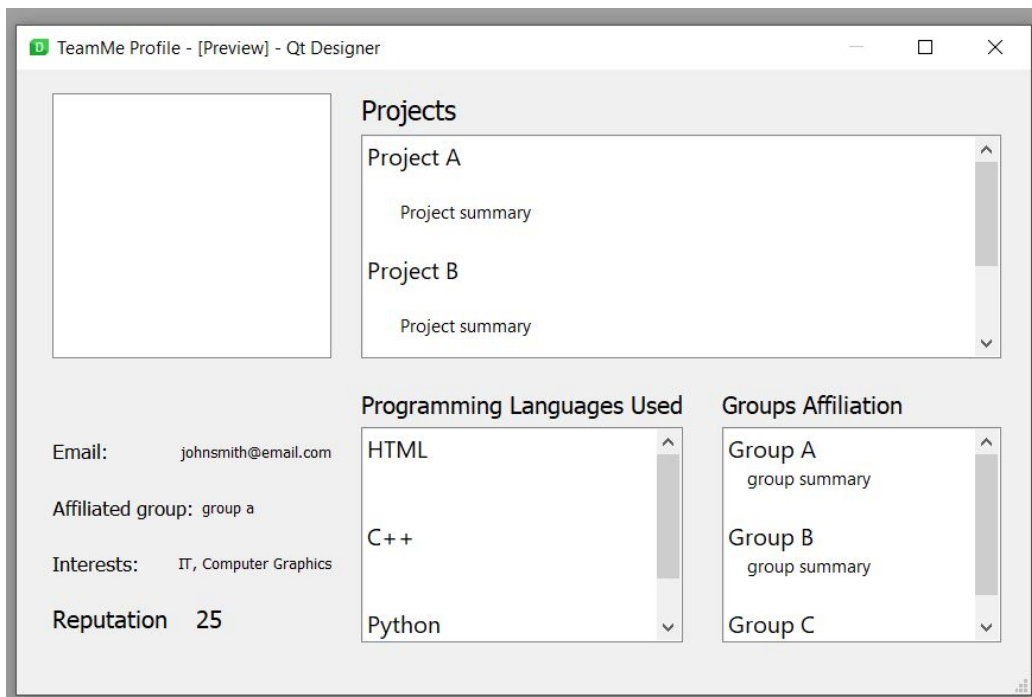
groups_GUI: when visitors click on “see more” button under top rated group profiles, they will be directed to this GUI.



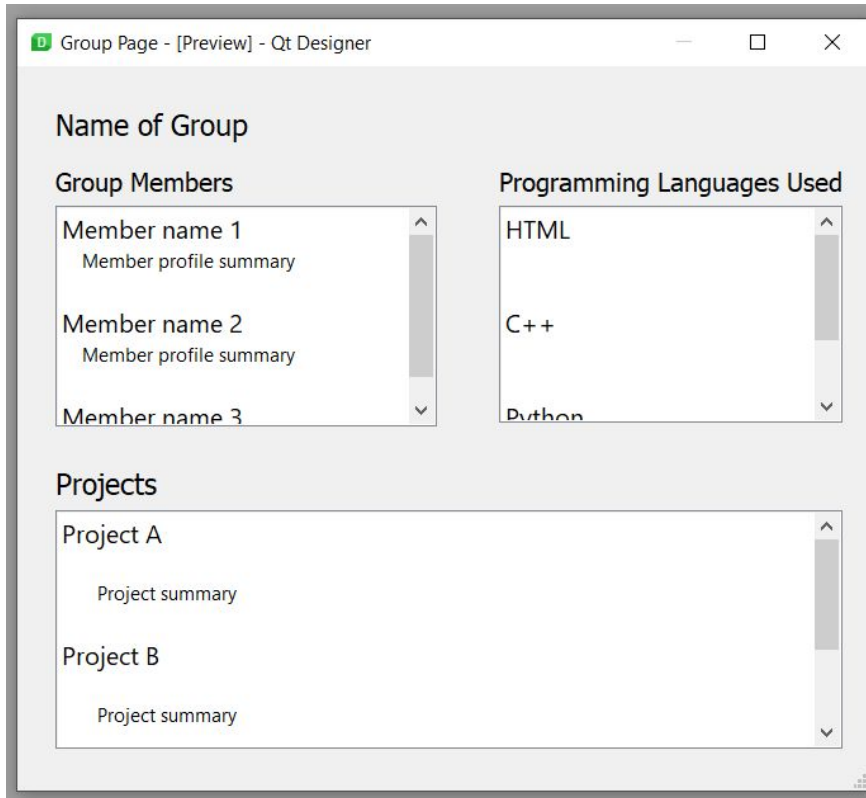
profiles_GUI: when visitors click on “see more” button under top rated user profiles, they will be directed to this GUI



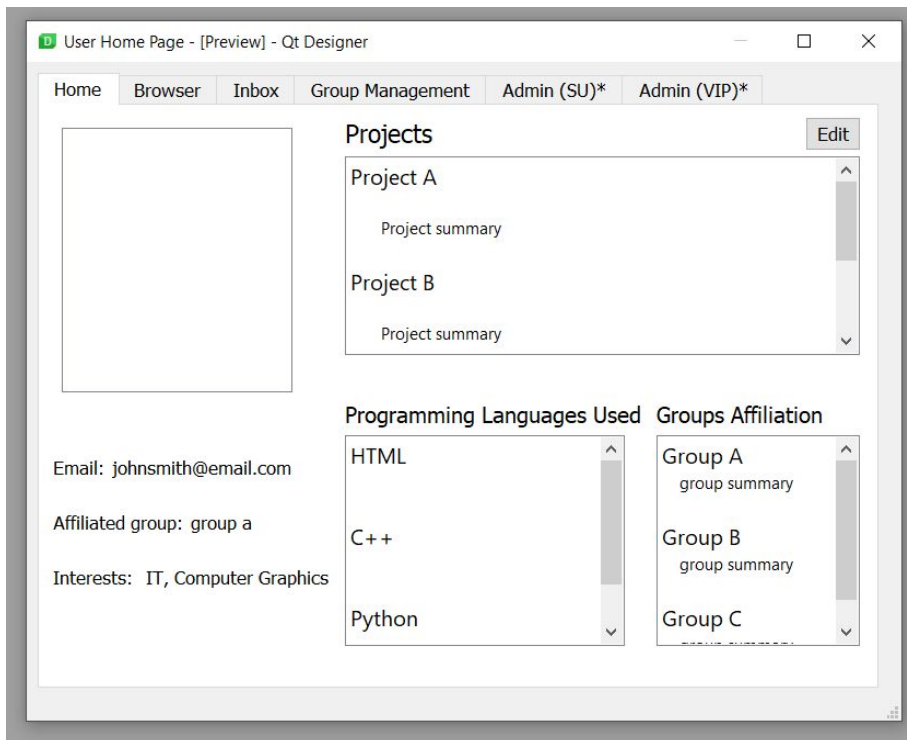
profile_GUI: when click on a summary of the user profile, a detailed profile GUI such as this will be opened



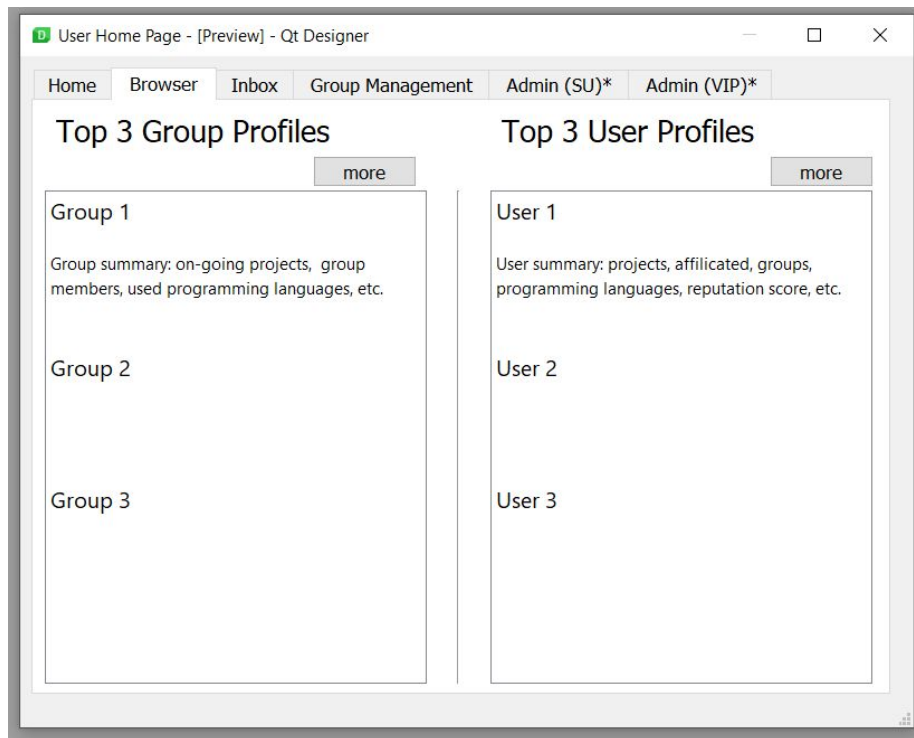
group_page_public_GUI: when click on a summary of the group profile, a detailed profile GUI such as this will be opened



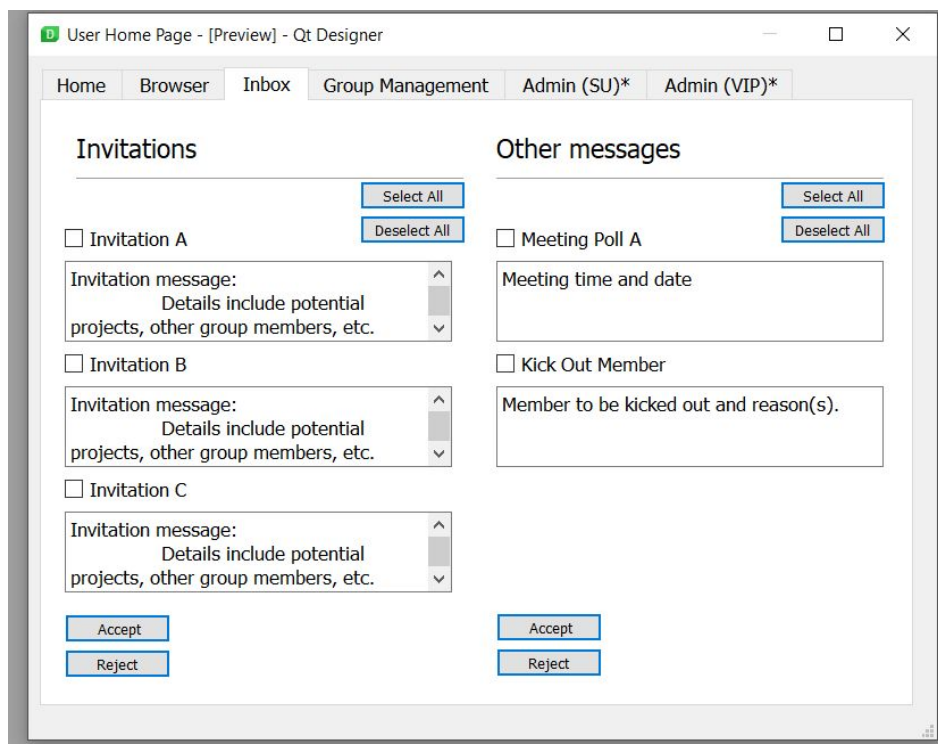
user_home_GUI: this GUI appears after visitors log in successfully, it has multiple menu tabs that allow users to navigate different features



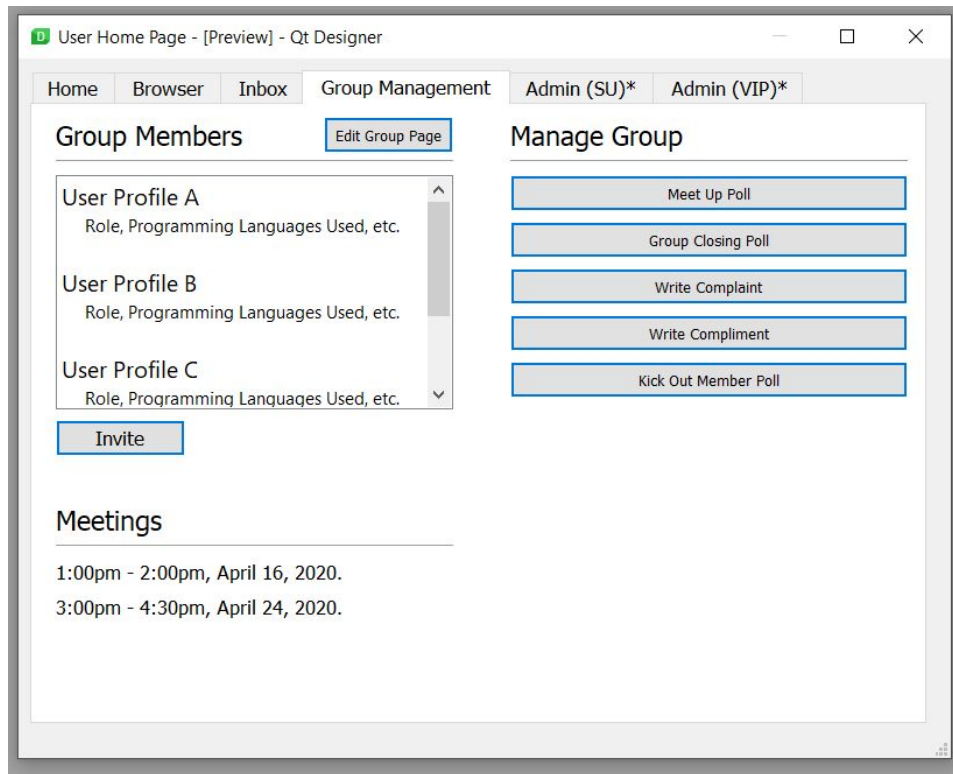
user_home_browser_GUI: this GUI allows logged in users to surf the system.



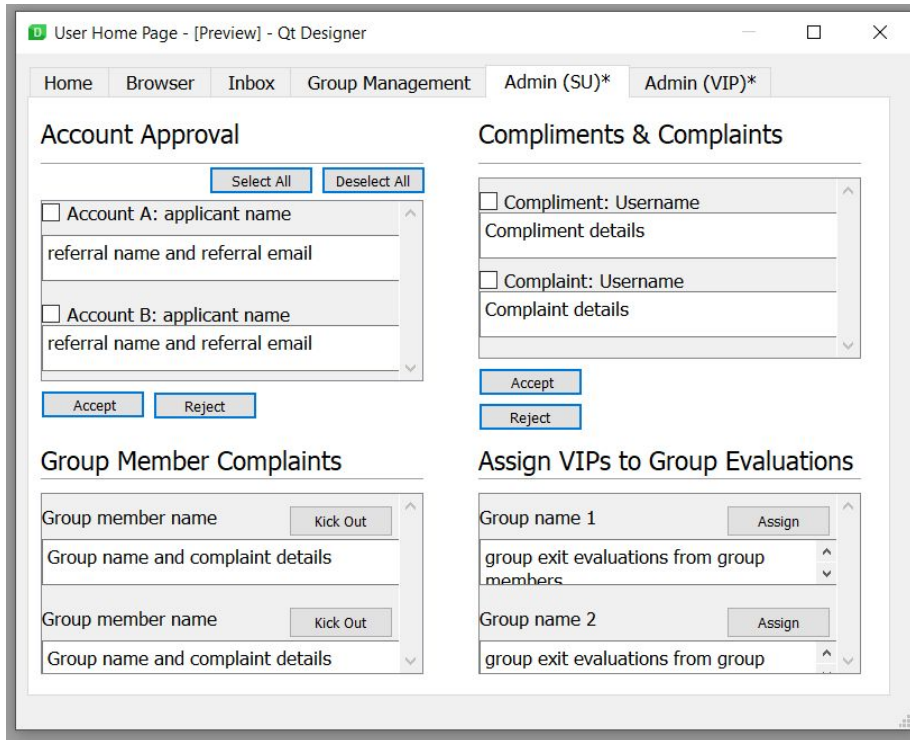
inbox_GUI: this GUI focuses on invitations and communication features.



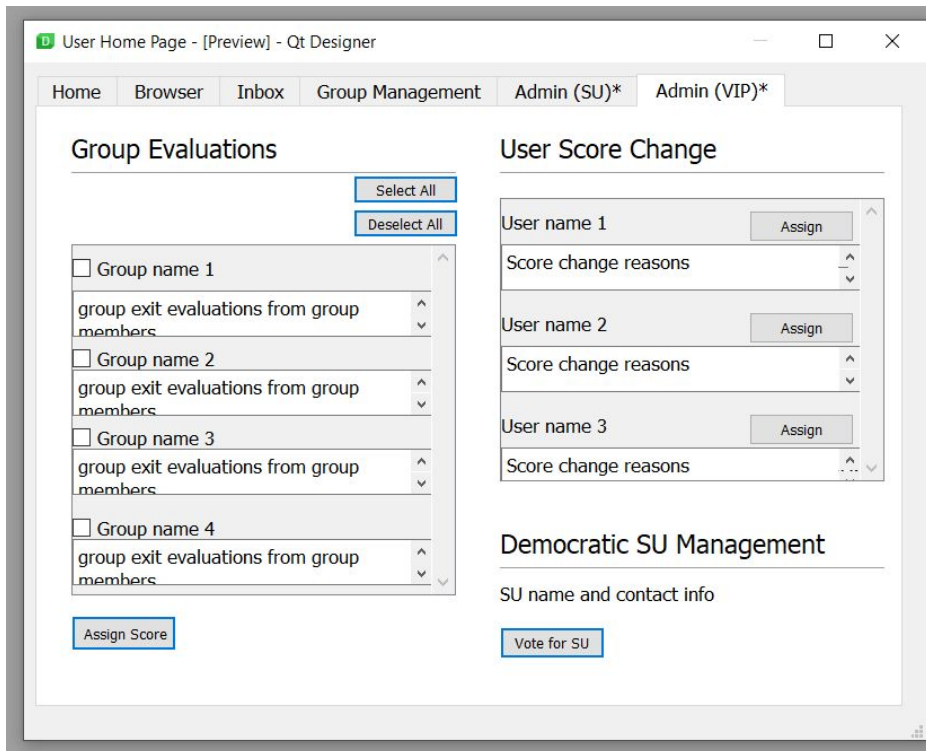
group_management_GUI: this GUI contains multiple group management features that lead to multiple different GUIs



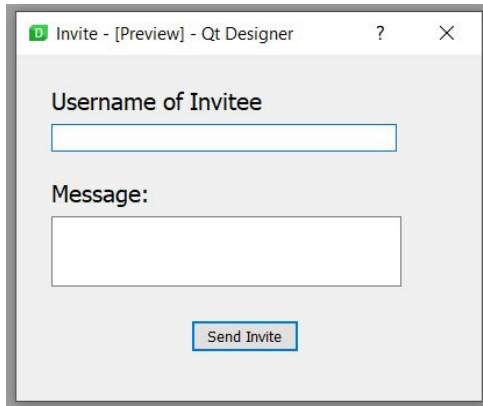
admin_SU_GUI: this GUI serves as a platform for SU to manage accounts, groups, compliments, complaints, etc.



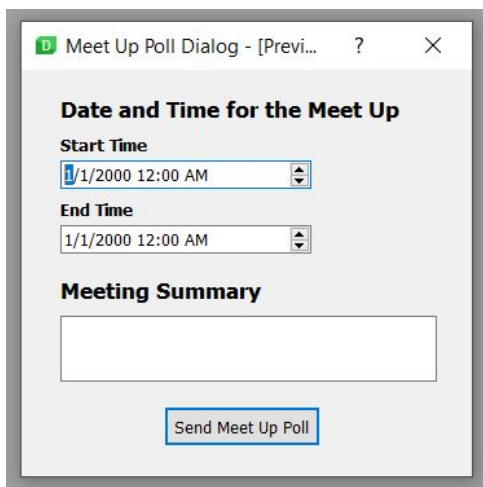
admin_VIP_GUI: this GUI serves as a platform for VIP to evaluate group exits, vote for a democratic SU and promote/demote users.



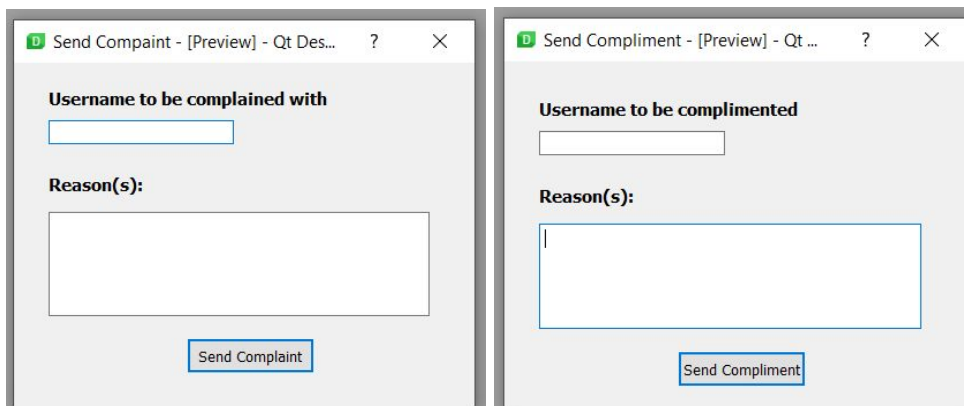
invite_GUI: a dialog that allows users to invite others to their group by the usernames.



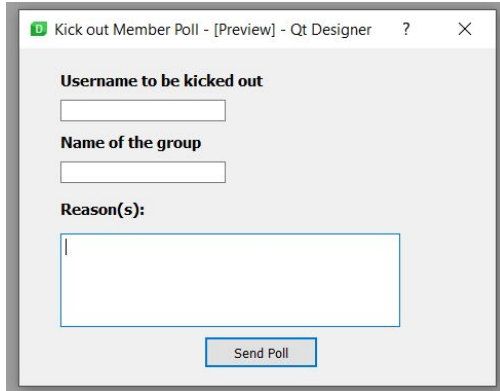
meet_up_poll_GUI: a dialog that allows group members to vote for a meet up time.



user_complain_and_compliment_GUI: 2 similar dialogs that allow users or group members to send compliments/complaints.

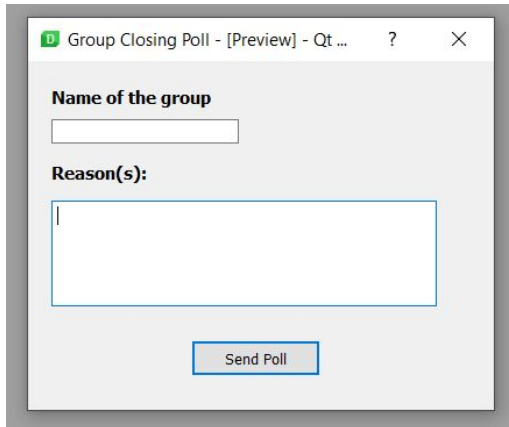


kick_out_member_poll_GUI: a dialog that allows users vote to kick out a group member with reasons specified



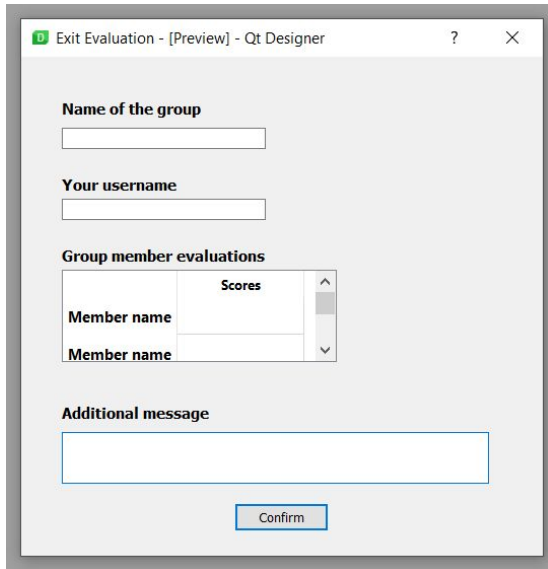
The screenshot shows a Qt Designer window titled "Kick out Member Poll - [Preview] - Qt Designer". The dialog box has a light gray background and contains the following elements: a label "Username to be kicked out" above a single-line text input field; a label "Name of the group" above another single-line text input field; a label "Reason(s):" above a larger multi-line text input field; and a "Send Poll" button at the bottom right.

group_closure_poll_GUI: a dialog that allows users to vote to close their groups with reasons specified

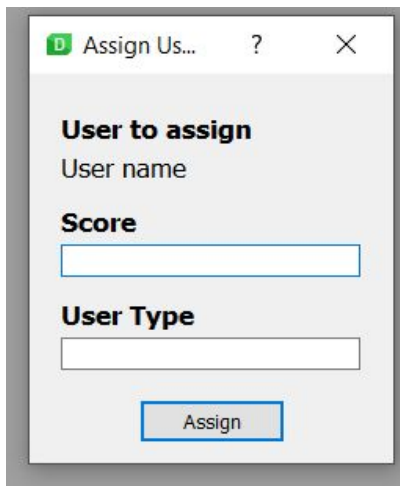


The screenshot shows a Qt Designer window titled "Group Closing Poll - [Preview] - Qt ...". The dialog box has a light gray background and contains the following elements: a label "Name of the group" above a single-line text input field; a label "Reason(s):" above a larger multi-line text input field; and a "Send Poll" button at the bottom right.

exit_evaluation_GUI: allows group members to put down scores for all other group members and send such report to VIP to do final evaluation.



promote_demote_user_GUI: a dialog that allows VIP to assign scores and promote/demote OUs.



assign_VIP_to_evalute_closed_group_GUI: a dialog that allows SU to assign a particular VIP to evaluate a group closure.

Assign VIP to Group Evaluation... ? X

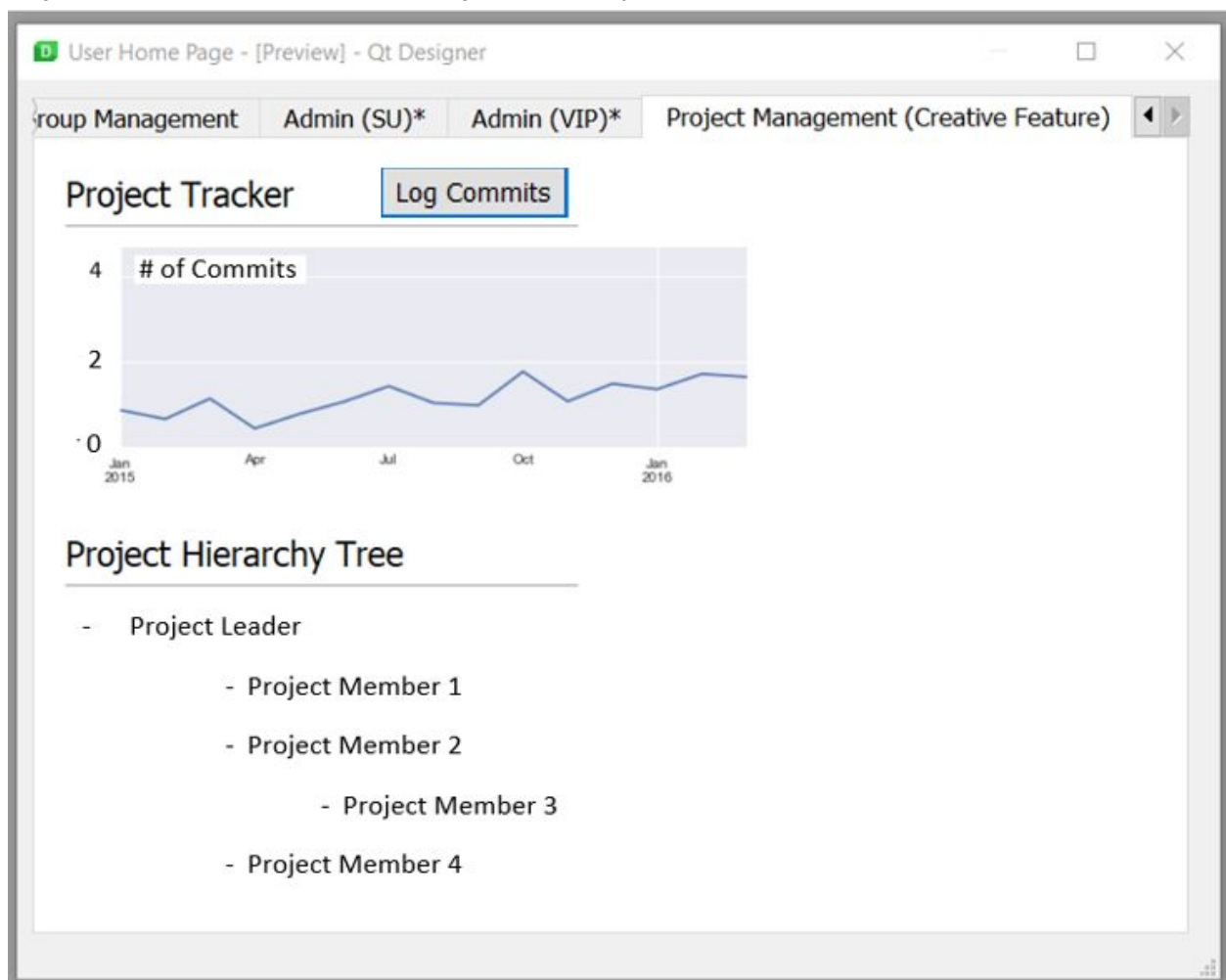
Group name

group exit evaluations from group members

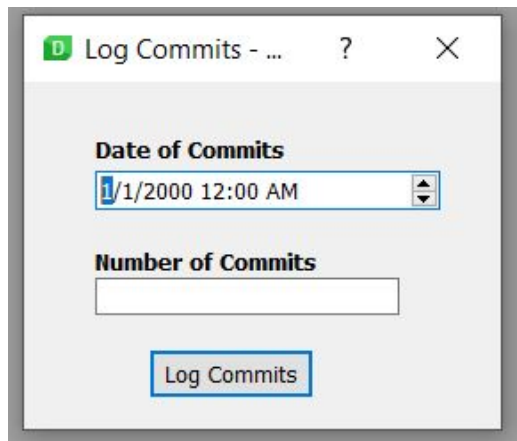
VIP to assign

Assign

creative_features_GUI: a new tab in the user home page that allows users to see the historical project commits timeline and the project hierarchy tree.



log_commits_GUI: a dialog that allows users to log date and number of commits so that they show on the project tracker.



The image shows a Windows-style dialog box titled "Log Commits - ...". It contains two input fields: "Date of Commits" with a date-time picker showing "1/1/2000 12:00 AM", and "Number of Commits" with an empty text box. A "Log Commits" button is at the bottom.

Date of Commits
1/1/2000 12:00 AM

Number of Commits

Log Commits

6. Minutes of group meetings and possible concerns of team work.

02/10/20

45 - 60 minutes group meeting. Discussed possible project ideas and added new proposed new ideas.. Talked about the specs, pros, cons, and tools needed for each project. A feasibility inspection was done on the entire group. Assigned each group member homework: to present a project you think is good, explain why, and to list all important technical skills needed.

02/17/20

45 - 60 minutes group meeting. Discussed the proposals of every group member. We inspected what every project required some challenges that might arise.

03/09/20

45 - 60 minutes group meeting. Discussed general specifications of the project and talked about what could be our creative feature for the project. We also discussed what tools and platform the application will be built with and run on. Also, we talked about how each of us were going to contribute to the project proposal.

03/23/20

2 -3 hours group meeting. Discussed final issues or concerns of the specification report and made finishing comments. Overall system discussed and agreed on temporary finalization.

04/22/20

45 - 60 minutes online group meeting. Discussed outline of application and ways of effective collaboration and synchronization of work. Also addressed any possible technical and social concerns. Timeline of development were agreed upon. Assignment of responsibilities are made.

04/24/20

2 - 2 ½ hours online group meeting. Discussed technical details such as data structures and updated the group on the progress so far on the design report. Good programming and software development practices were reviewed by the group. Synchronization of the group were talked about. Addressed any concerns the group had.

04/29/20

30 - 45 minutes of online group meeting. Discussed what every member of the group completed and what needed to be done. Discussed any issues in the design report. Updated your timeline of project completion.

There will possibly be more group meetings as the group continues into the next phase of implementing the software system.

Each group member worked and is continuing to work extremely hard. So far, the group experience is going smoothly and the collaboration has been great and productive. We strive our best to be team players and be very professional. We tried to split the work up equally, have inviting and constructive discussions, and contribute as much as each group member can to the reports and project.

7. Github Link

<https://github.com/tislam35/TeamMe>