Group S Development Team

TeamMe

Memo

CSC 32200 Software Engineering Project

Version 1.0

Group Members:
Tohidul Islam
Sophie Huang
Ekramul Sawrid
Dor Ulman

Active Teaming System Requirements

Note: for all the main source code, see system.py. We have labeled each function with descriptive names and with comments. We have actively tested all completed code. To seem the working features, see the readme on how to use the application.

System Features:

- For a random visitor, the system provide a GUI showcasing the top 3 rated projects and top rating OU profiles and SU profiles to showcase the power of the system. A visitor can surf around to find more OUs/VIPs and projects

  Code Completed/Partially Implemented in GUIs: System shows top 3 rated projects/groups and top 3 rated users. Listing more OUs/VIPs not implemented on GUIs.

- Give visitor an option to register to be OU: the visitor has to fill in basic personal information such as name, email address, interest, credential and reference who are already an OU or VIP of the system. One SU will check these info to either approve or reject. If approved, the SU will send an email with account id and password, when the new OU first login, s/he is required to change the password. If rejected, the applicant has one chance to appeal and the SU will make a final decision to reverse the rejection: if still reject, then this visitor will be put in blacklist forever. The approved OU will receive an initial reputation score by the reference: an OU can give a score 0-10; a VIP can give score 0-20.

  Mostly Completed/Partially Implemented in GUIs: Everything works in the working system except not implemented: checking if a email is really real and sending email to the visitor's email after approved, and change password after first login, and for not implemented in GUIs: although we have function, appealing not implemented in GUIs but the visitor will be put into blacklist if appealed and rejected again.

- OUs can form groups by inviting other OU(s) for a certain purpose: the other OUs can accept or reject the invite. If reject, the OU should respond by the reason. An OU can put some OUs to his/her white-box: accept all invites or black-box: reject all invites with automatic message. For instance, the group could be some students taking 32200 as a study group.

  Code Completed/Not Implemented in GUIs: functions for invite, accept, reject, white/blackbox is written but not implemented in the GUIs.

- Once a group is formed, a group web-page should be made available that is accessible to all group members: some information is public to be browsed by visitors and other OUs, some could be set as private to the group members only such as evaluations and warnings. All group members can moderate and post to the group page. This page will be used for posting updates and scheduling meet-ups.

Mostly Completed/Not implemented on GUIs: Group page is made available after a group is made, group members can change group information. Although not implemented on GUIs, we have functions for group warnings. Posting is not implemented.

- Any group member can ask for a meet-up polling to find common time for all members to meet. Once all members responded, the time slot with the most votes will be chosen. If a member has missed scheduled meeting twice, s/he will receive a warning. The voted out member can appeal to the SU to possibly change the reputation scores. Each group member should have a track record for the number of assigned tasks that have been done, which is the foundation for the group warnings and the appeals of the affected group member(s).

Code Completed/Not implemented on GUIs/Not Implemented: We have functions for meeting poll, voting, and track off missed warnings, tracking commits but not implemented on GUIs. Group member appeal to SU not implemented.

- The group members can vote to issue a warning or a praise to a group member, the vote must be unanimous. A member receiving 3 warnings will be automatically removed from the group and get a 5 point reputation score deduction. The group can also vote to kick out a member directly, the member will be removed from the group and receives 10 point reputation score deduction. An OU with negative reputation score will be removed from the system and put into black-list automatically.

Code Completed/Not Implemented on GUIs: We have functions for warning polling and consequences, kick-out polling and consequences, and not implemented in GUIs. Blacklisting is completed in the working system for other features.

- The group members can vote to close the group, and conduct an exit evaluation to other members. Each member will receive the median reputation score given by all other members. And every member can decide if s/he is willing to put the other member

to her/his white-box or black-box afterwards and why. After group closure, the SU will assign a VIP to evaluate the group and determine a reputation score for the entire group to be added/deducted for all members involved. The system will keep a ranking list of finished groups to be showcased.

Mostly Completed/Not Implemented on GUIs: We have functions for everything except for members giving other members score after group closes. Not implemented on GUIs.

- An OU whose reputation score is higher than 30 will be promoted to VIP; and a VIP whose score is lower than 25 will be demoted. All VIPs can vote one VIP as the democratic SU.

Fully completed: Each time a score changes, system checks and makes changes if necessary. Created a function for this specific purpose, which other functions uses.

- Visitors and OUs can complain to SU about a group or other OUs, the SU will decide if the complaint merit action. The SU can decide to shut down the group or OU and punish all involved by a certain score deduction or even kick them out from the entire system.

Code Completed/Partially Implemented on GUIs: SU can decide to accept or reject complaint after a user complaints. We have functions for everything but some are not implemented.

- OUs who are kicked out will have the final chance to login and do some final processing and will be unable to login ever after.

Code Completed/Not implemented on GUIs: we have function but not implemented in GUIs. Currently, kicked out members can not log in.

- The entire system keep a list of taboo word list, any message by any OU with these taboo word will be converted to *** automatically and the OU's reputation score will be decreased by 1: if s/he uses the same word again later, his/her reputation score will be decreased by 5.

Code Completed/Implemented to Certain Features: For complaints only. Promoted/demoted to OU/VIP or blacklisted after each score change and if applicable.

- OUs can send compliment about other OUs to SU, and SU will increase the reputation score of the complemented OU, any OU receiving 3 compliments, regardless of the reputation score.

Fully Completed: SU can not reject OU for each 3 compliments.

Other system requirements:

- A consistent system GUI is required: don't keep popping up new windows to cause a mess

Fully Complete: Although, we do have popping windows, it does not cause a mess and take away from the user experience of using the system.

- Besides the foregoing items, each team can have a creative feature for this system, which is worth 10% of the project. A feature deemed extremely creative will receive an up to 10% bonus by discretion.

Not implemented: Due to time and problems with GUIs connection, we were able to implement the special features.

- No need to make this system web based or mobile based, the latter two can be viewed as a creative feature if your team choose to do so.

We made desktop application.

- For details not listed in the foregoing items your team is free to use your own judgment to proceed in your system design and development.

We made our own judgement while in our system design and development.

The main cause we weren't able to fully complete our project:
Although all group members are relatively familiar with Python, the programming language, we are not very familiar with working on a full blown desktop application with Python. After researching for different libraries, we decided on PyQt5, a relatively easier library. However, it's proven that we have difficulties adjusting to the PyQt5 GUI frameworks. The culprit was that the group were stuck for multiple times/periods on

GUIs fixes and there were no PyQt5 experts in our group. Although we finished almost all of our backends with OOP principles and reusable functions, we have some difficulties connecting those functions with the GUI.