User Groups Design Document

OVERVIEW:

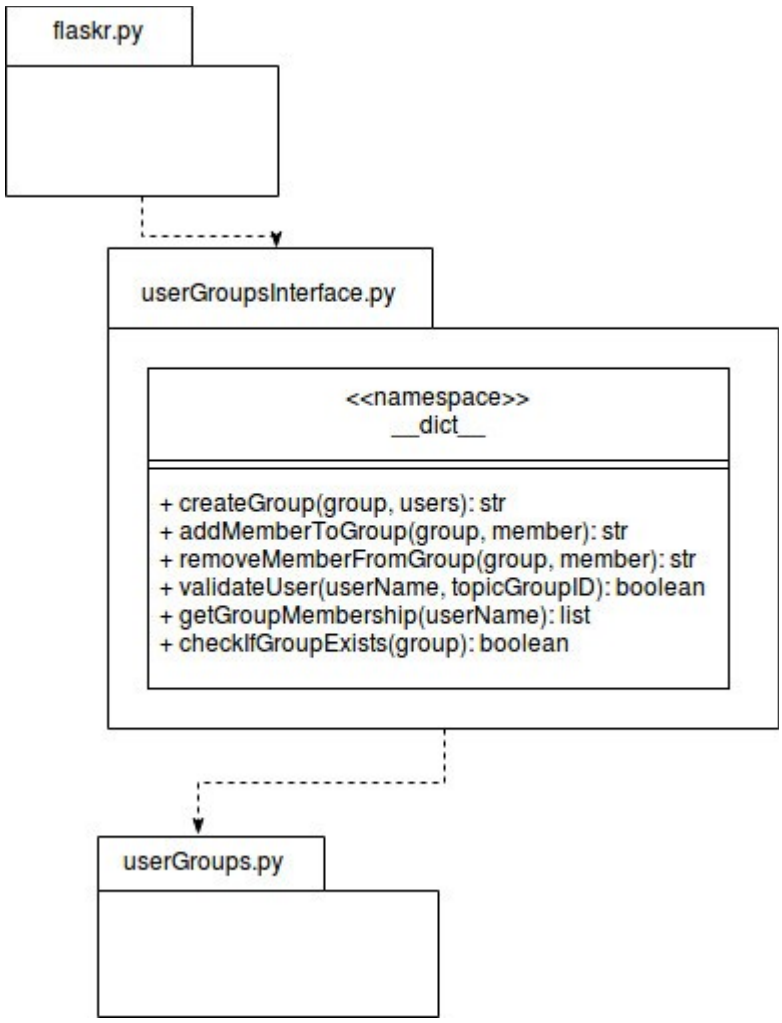
The user groups module is responsible for the creation, editing, and handling of user groups and requests related to the user groups. Requests for information on user groups primarily comes from the topics section as it's meant to be a way of restricting a topic to members of a particular group.

The user groups code was designed and implemented in a compartmentalized and modularized way. This was done so that it could both operate on it's own with minimal changes to the main flaskr.py file and so that it could be changed or replaced with a completely different system while having little to no impact on the way the main flaskr.py code uses it. Persistence has been integrated into the main schema.db initialized by flaskr.py

The code includes two files: userGroupsInterface.py and userGroups.py.

userGroupsInterface.py is the interface code between flaskr.py and userGroups.py. flaskr.py will import userGroupsInterface.py and call its methods to action anything relating to the user groups functional requirement implemented at this time. userGroupsInterface.py will then take the information it was passed and alter it to be usable by the userGroups.py code. Once the information is in a usable form the interface will call the appropriate method in userGroups.py and pass it the required information. userGroups.py will then process the request and return the information specified in the method that was called. userGroupsInterface.py will then take that response, alter it as needed, and return the appropriate information (as specified in the method) to flaskr.py. If any changes are made to the way the user groups functional requirement is implemented in the future, flaskr.py can maintain the same method calls to userGroupsInterface.py as long as userGroupsInterface.py is modified to send the appropriate information to the new implementation. The system should still work with minimal to no effect on the flaskr.py code as long as userGroupsInterface.py then changes the response to one that is usable by flaskr.py.

userGroups.py is the code that actually implements everything related to the user groups. It is imported by userGroupsInterface.py and its methods are then called by userGroupsInterface.py. Any changes to the implementation of the user groups functional requirement will be done here.



DESCRIPTION/USE:

Placement: place userGroups.py and userGroupsInterface.py into the first flaskr file (where the setup.py file is located). The user_groups.html page needs to be added to the templates folder as well.

To use: import userGroupsInterface.py. Call methods from userGroupsInterface.py with the data specified in the doc string and you will get data returned in the form described by the doc string. Some functions return string messages which can be used with flash to give feedback to the user on what happened during the operation (i.e. adding a user to a group).

DESIGN WEAKNESSES:

The implementation could be considered poor and adhoc. The interface was designed to work specifically with flaskr.py and uses some of flaskr.py's quirks. When entering data in certain web forms it is passed as a list object rather than a string. The interface automatically assumes it's getting a list even though it only wants a string. Currently the interface turns the data into a string in a crude way (i.e. without checking if it is actually a list object). More type checking in the interface would be beneficial as any object type can be passed in python which may cause errors.

Implementing the code would be much better if objects were used. Currently the code is methods only and it deals with lists and strings primarily. If objects were implemented it would help with type issues and make it easier to pass information.

Persistence is in the form of strings stored in tables with sqlite. Using a more secure or even just a more object friendly system would be beneficial, especially if objects were used in the implementation.

Security was not a requirement during the initial prototype phase but it should be considered again. Specifically, the database should be secured against direct viewing and information gathering by repeated quiring of the database and individuals in or not in groups.

Better understanding of client expectations would lead to better implementation and design, especially when considering future integrations into the system.

The database is hard coded into userGroups.py. If the file were to change name the functionality would break. It would be better if the code received either the database path or a connection to the database.