COMP 2710

Software Construction

Lab 1

Robinson Davis

September 19, 2014

1 Analysis

This program will create a basic message board for at least four different users across various groups, and will support different ways to communicate - broadcast, multicast, and unicast messages - along with a home page and wall page. All messages displayed will always be presented in reverse chronological order. Only one user is supported at a time, but this program does provide support for switching users. The message board will rely on a menu to provide this functionality, which will be described in greater detail. The menu is as follows:   
  
 1) Create a new user

* Prompts for a username of a new user to be created. This username must not already exist. New user will be logged in automatically upon successful creation.

2) Broadcast a message

* Prompts the current user for a broadcast message. This message will be shared with all users on the message board. Multi-line input will be received as necessary until $$ is entered. The two symbols $$ will not appear in any messages.

3) Multicast a message

* Prompts the user for a recipient group. If the group exists, the current user will then be prompted for the multicast message to be shared only with other members of the group on the message board. Multi-line input will be received as necessary until $$ is entered. The two symbols $$ will not appear in any messages.

4) Unicast a message

* Prompts the user for a recipient username. If that user exists, the current user will then be prompted for a unicast message. This message will only be shared with the desired recipient user. Multi-line input will be received as necessary until $$ is entered. The two symbols $$ will not appear in any messages.

5) Display Wall page

* Provides the current user with a wall page which will display all messages shared by the current user. Two messages will appear, and the user will me prompted for more. If the user decides to show more messages, all remaining messages will be shown.

6) Display Home page

* Provides the current user with a home page which will display all shared messages by other users on the message board viewable by the current user. Two messages will appear, and the user will me prompted for more. If the user decides to show more messages, all remaining messages will be shown.

7) Create a new group

* Prompts for the new name of the group to be created. This group name must not already exist.

8) Join a new group

* Prompts user for the name of the group they wish to join and adds them to it. The user must not already be a member of this group.

9) Switch to a different user

* Prompts the current user for the username of the user wishing to log in and then logs them in. This user must exist.

10) Quit Auburn Messaging System

* Quits the Auburn Messaging System.

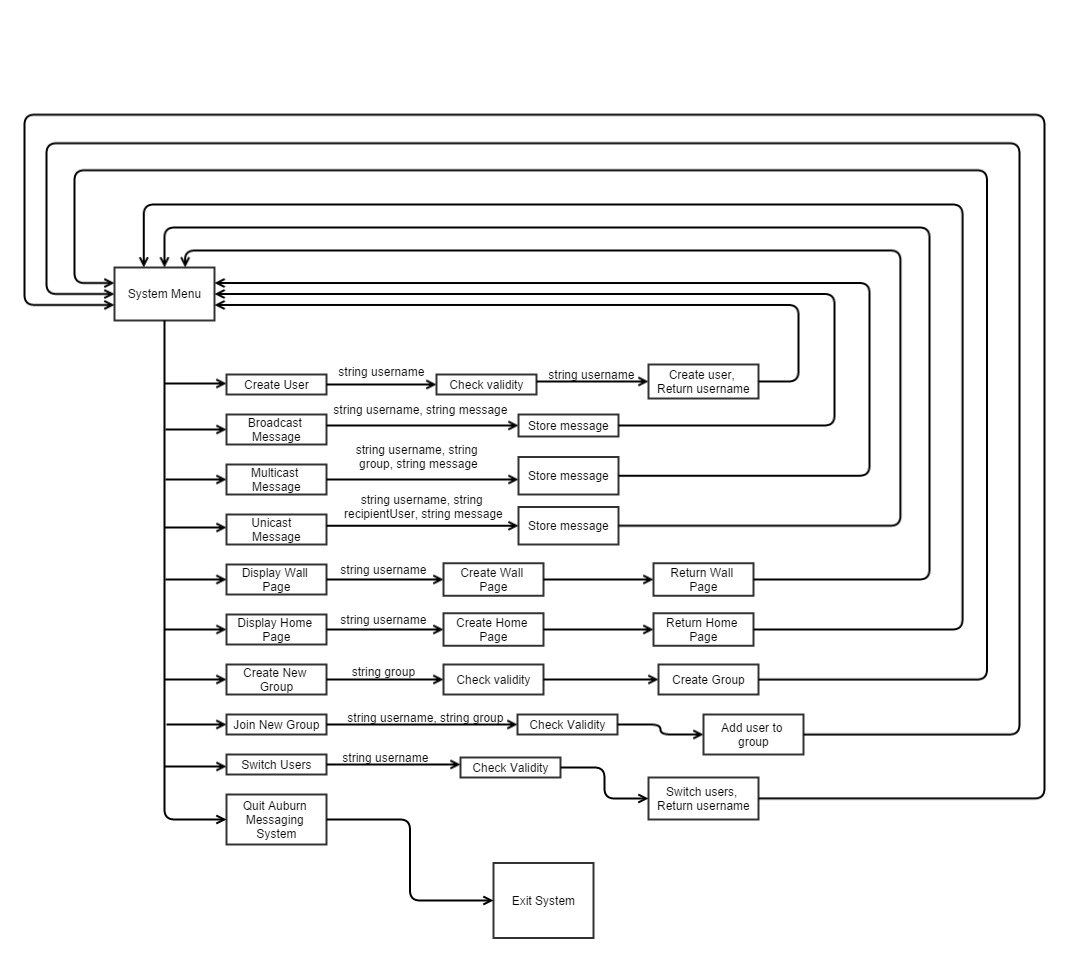
2 Design

There will be numerous classes to help separate the functionality of this program. They are as follows:

1. **Menu class** – Used to print menu and receive choice from user.
   1. Variables
      1. None known.
   2. Functions:
2. **User class** – A class used to manager user’s data.
   1. Variables:
      1. map<string, vector<string> > usersWithGroups -> A map which uses strings as keys for vectors of strings representing usernames.
   2. Functions
      1. void newUser(string username)
         1. Check if parameter username already exists. Display error if so and reprompt; if not, display success message and create user.
         2. To create the new user, add parameter username to usersWithGroups map with key “All” to represent the default group, which every user is a member of.
      2. boolean existingUser(string username)
         1. Check if parameter username exists under ANY key in the user hashmap.
      3. void newGroup(string group)
         1. Check if parameter group already exists. Display error if so; if not, display success message and create group.
         2. To create a group, create a new key in the hashmap from parameter group.
      4. boolean existingGroup(string group)
         1. Check if parameter group exists as a key in the user hashmap.
      5. boolean joinGroup(string username, string group)
         1. Checks if parameter group is a valid group and that username being added is not already a member.
         2. Adds user to usersWithGroups map using parameter group as key and parameter username as value.
      6. void displayHome(string username)
         1. Copy message\_buffer (so as to not edit message\_buffer in any way) and retrieve necessary information (all shared messages viewable by current user).
         2. Format and output two messages first, and prompt user for rest.
      7. void displayWall(string username)
         1. Copy message\_buffer (so as to not edit message\_buffer in any way) and retrieve necessary information (all shared messages viewable by current user).
         2. Format and output two messages first, and prompt user for rest.

1. **System and Menu class** – Used to run the program, and create necessary objects for execution.
   1. Variables:
      1. string currentUser -> used to keep track of current user
   2. Functions:
      1. int main()
         1. Used to run program.
         2. Will handle user’s choice using the getChoice() function, and will call the corresponding function according to the menu.
         3. The calls to the corresponding functions will be handled with a switch loop.
         4. Supports switching users. The username being switched to will be checked for previous existence. If not, display error stating user does not exist.
         5. Supports ending the program. Most code in main() will be in a while loop. When user chooses to exit program, loop will be terminated.
      2. void displayMenu()
         1. Displays the menu for the user.
      3. int getChoice()
         1. Prompts user for choice.
         2. Checks if choice is valid.
         3. Display error if choice is invalid and reprompt for correct choice.
         4. Return choice if valid.
2. **MessageBuffer class –** Used to maintain the message buffer.
   1. Variables
      1. string message\_buffer -> the message buffer used to store all messages by users.
   2. Functions
      1. void add(string username, string group, string message)
         1. Adds a user’s message to the message buffer.
         2. Group will be “all” for broadcast messages, a specific group for multicast messages, and another valid username for unicast messages.

2.1 Data Flow Diagram



3 Tests

1. Effective testing for this message board can most easily be obtained through case testing. That is, testing each usage case and ensuring that proper output is achieved. Again, this will be examined per menu item.
   1. Create a new user
      1. Create initial user, and ensure that initial user is successfully created and logged in.
      2. Try to create a new user using the same username, and ensure that it is unsuccessful.
      3. Try once more to create a new user using a different username, and ensure that the new user is successfully created and logged in.
   2. Broadcast a message
      1. Enter message, ensure that “$$” stops input.
      2. Check message buffer and ensure that the message has been successfully read in the format of:  
          “|<” + currentUser + “::” + “All>|” + message  
         where the “$$” is excluded in the message.
      3. Broadcast a second message and ensure that the next message is added to the message buffer in the same format and in reverse chronological order.
   3. Multicast a message
      1. Enter message (after entering the group), ensure that “$$” stops input.
      2. Check message buffer and ensure that the message has been successfully read in the format of:  
          “|<” + currentUser + “::” + group + “>|” + message  
         where group is the chosen recipient group for the multicast message and the “$$” is excluded in the message.
      3. Multicast a second message to a different group, and ensure the proper group has been saved and that the message has been stored in the message buffer in reverse chronological order.
   4. Unicast a message
      1. Enter a message (after entering the desired recipient user) ensure that “$$” stops input.
      2. Check message buffer and ensure that the message has been successfully read in the format of:  
          “|<” + currentUser + “::” + recipientUser + “>|” + message  
         where recipientUser is the chosen recipient username for the unicast message and the “$$” is excluded in the message.
      3. Unicast a second message to a different user, and ensure the proper user has been saved and that the message has been stored in the message buffer in reverse chronological order.
   5. Display Wall Page
      1. Ensure that only messages shared, not received, by the current user are displayed.
      2. Two messages should appear prompting the user for more.
      3. If no is chosen, ensure that the menu is redisplayed.
      4. If yes is chosen, ensure that all remaining messages are displayed, followed by the menu.
      5. Also ensure that if there are two are less messages viewable by the current user that the option to display more messages does not display, and instead redirect user directly back to the menu.
   6. Display Home Page
      1. Ensure that only valid messages received, not shared, by the current user are displayed.
         1. This includes all broadcast messages from all users, all multicast messages from all users in groups in which the current user is a member of, and unicast messages where the current user is the recipient user.
      2. Two messages should appear prompting the user for more.
      3. If no is chosen, ensure that the menu is redisplayed.
      4. If yes is chosen, ensure that all remaining messages are displayed, followed by the menu.
      5. Also ensure that if there are two are less messages viewable by the current user that the option to display more messages does not display, and instead redirect user directly back to the menu.
   7. Create a new group.
      1. Create a new group, and ensure the group has been successfully created (and that the current user is not automatically added to group)
      2. Try to create a new group using the same name as the previous group, and ensure that it is unsuccessful.
      3. Try once more to create a new group using a different group name, and ensure the second group has been created successfully.
   8. Join a new group
      1. Enter a group name that the current user is not a member of and ensure they have been successfully added.
      2. Enter a group name that the current user is currently a member of and ensure the action is unsuccessful.
   9. Switch to a different user
      1. Enter an existing username and ensure new user is set as current user.
      2. Enter a non-existing username and ensure the action fails.
   10. Quit Auburn Messaging System
       1. Ensure the program exits successfully.