COMPUTER SCIENCES AND SOFTWARE ENGINEERING

AUBURN UNVERSITY

COMP 2710

**Software Construction**

Fall 2014

**Lab 2**

**Distribution Messaging System**

**Due: October 13, 2014**

**Analysis**

Use Cases:

1. **Broadcast a message:** The program will prompt the user for the message which is then stored into the message buffer. To end a message, the user will enter a new line with a string “$$”followed by the enter key.
2. **Multicast a message:** The program will prompt for a group name first, then it prompts for a message to be sent to this group. To end a message, the user will enter a new line with a string “$$”followed by the enter key.
3. **Unicast a message:** The program will prompt for a user name first, then it prompts for a message to be sent to the user. To end a message, the user will enter a new line with a string “$$”followed by the enter key.
4. **Display wall page:** The program will first display a title indicating that it is displaying the current user’s wall page, e.g. “Joe’s Wall Page”. It then displays the *two* latest messages in the current user’s wall page, in reverse chronological order. It only displays the user’s messages and the group (for multicast) or recipient (for unicast) or “All” (for broadcast) in reverse chronological order. After displaying the two latest messages, it will then prompt the user if they want more messages. If the response is “no”, then it will stop displaying messages, but if the response is “yes”, it will display all the remaining messages from that user. If there are two or fewer messages then the program will not prompt for more messages. The messages are all displayed in reverse chronological order, i.e. the most recently posted messages will be displayed before the earlier messages.
5. **Display home page:**  The program first displays the current user home page, e.g. “Joe’s Wall Page”. And then it displays only the two latest messages (either from broadcast, multicast or unicast messages), whichever are the two latest messages. After displaying the latest two messages, it will then prompt the user if they want more. If the response is “no”, then it will stop displaying messages, but if the response is “yes”. It will display all the remaining (multicast, broadcast or unicast) messages for that user. If there are two or fewer messages then the program will not prompt for more messages. The messages are all displayed in reverse chronological order, i.e. the most recently posted messages will be displayed before the earlier messages.
6. **Create a group:**  The program will prompt for the name of the group. It checks if the group’s name is already an existing group. If so, it will display an error message and prompts for another group name.
7. **Join a group:** The program will prompt for the name of the group. It checks if the group’s name is already an existing group. If not, it will display an error message and prompts for another group name. Next, the program will add the current user name to the group.
8. **Quit the Auburn Messaging System:** The program will post a thank you message and then exit.

**Design**

* **Classes**
* **System Menu –** This class will be the main. It handles user decisions for system use and calls to the other classes.
  + **Functions**
    - runProgram – presents the user with the menu. Is the gateway function to all other classes’ functions and variables. If case is to quit, system shuts down.
    - Sleep – allows the system to pause momentarily
  + **Variables**
  + **Error-Handling**
    - mainError – handles any and all error output to user throughout all system functions
* **User –** This class handles the user-related objects and information; User Manager
  + **Functions**
    - isUser
  + **Variables**
    - userList
    - currentUser
  + **Error**
    - userError – handles any and all user function errors
* **Group –** This class handles the user-related objects and information; Group Manager
  + **Functions**
    - createGroup
    - deleteGroup
    - joinGroup
    - isGroup
    - isInGroup
  + **Variables**
    - groupList – list of groups
    - new\_Group
    - doubleBuffer – List of users with their group assignment next to their names.
  + **Error**
    - groupError – handles any and all group function errors
* **Message –** This class handles all message-related objects and functionality; Message Manager
  + **Functions**
    - broadcastMessage
    - mulitcastMessage
    - unicastMessage
  + **Variables**
    - messageBuffer
    - currentTimeStamp = message.size() + 1
  + **Error**
    - messageError – handles any and all message function errors
* **Pages** – This class handles the wall and home pages; Page Manager
  + **Functions**
    - displayWallpage
    - displayHomepage
    - checkMessages

**Class Diagram**



**Data Flow Diagram**

**Testing**

\*\***Note:** Usernames and Groupnames should only be [a-Z] and not include spaces, punctuation or numbers. This is valid for input, creation and searching within the program. Anything not following this rule will be invalid and result in an error. This note is stated here to prevent redundancy within the below testing scenarios.

1. **System Welcome:**
   1. Check if the menu is printed correctly.
2. **Broadcast a message:** 
   1. Check if the program prompts the user for the message.
      1. Check if the program allows multiple lines for the message
   2. Check if the message is then stored into the message buffer.
   3. Check if a new line with “$$” entered, followed by the “enter” key ends the message.
   4. The final string “$$”must not be stored in the message buffer, neither should it be displayed in the wall or home pages.
3. **Multicast a message:** T
   1. Check if the program prompts the user for a group name
      1. Checks if it is valid
      2. Then either errors if invalid or continues if valid
   2. Check if the program prompts the user for the message.
      1. Check if the program allows multiple lines for the message
   3. Check if the message is then stored into the message buffer.
   4. Check if a new line with “$$” entered, followed by the “enter” key ends the message.
   5. The final string “$$”must not be stored in the message buffer, neither should it be displayed in the wall or home pages.
4. **Unicast a message:**
   1. Check if the program prompts the user for a user name
      1. Checks if it is valid
      2. Then either errors if invalid or continues if valid
   2. Check if the program prompts the user for the message.
      1. Check if the program allows multiple lines for the message
   3. Check if the message is then stored into the message buffer.
   4. Check if a new line with “$$” entered, followed by the “enter” key ends the message.
   5. The final string “$$”must not be stored in the message buffer, neither should it be displayed in the wall or home pages.
5. **Display wall page:** 
   1. Check if program first displays a title indicating that it is displaying the current user’s wall page.
   2. Check if it displays the two latest messages in the current user’s wall page
      1. Must be in reverse chronological order. (The most recently posted messages will be displayed before the earlier messages.)
      2. Messages from different users must be separate by a blank line.
   3. Check if the program only displays the user’s messages and the group (for multicast) or recipient (for unicast) or “All” (for broadcast)
      1. Must be in reverse chronological order.
   4. Check if the program, after displaying the two latest messages, it then prompts the user if they want more messages.
      1. Check if the response is “no”, then it will stop displaying messages
      2. Check if the response is “yes”, it will display all the remaining messages from that user.
   5. Check if there are two or fewer messages then the program will not prompt for more messages.
6. **Display home page:** 
   1. Check if the program first displays the current user home page, e.g. “Joe’s Wall Page”.
   2. Check if it displays the two latest messages in the current user’s wall page
      1. Must be in reverse chronological order. (The most recently posted messages will be displayed before the earlier messages.)
      2. When displaying the message from each user, it must display the sender name followed by a string “>>” and a ‘\n’ and then followed by the message.
      3. The group name or the recipient name or “All” must not be displayed.
      4. Messages from different users must be separate by a blank line.
   3. Check if the program, after displaying the two latest messages, it then prompts the user if they want more messages.
      1. Check if the response is “no”, then it will stop displaying messages
      2. Check if the response is “yes”, it will display all the remaining messages from that user.
   4. Check if there are two or fewer messages then the program will not prompt for more messages.
7. **Create a group:** 
   1. Check if the program prompts for a name to be entered for a new group, correctly.
   2. Before adding the group, it must check if it has already been created.
      1. Check if so, then it displays an error message and prompts for another group name.
   3. Check if it creates the group.
8. **Join a group:**
   1. Check if the program properly prompts for a group name.
   2. Check if the program checks if the group has already been created.
      1. Check if not, then it displays an error message and prompts for another group name.
   3. Check if the program adds the current user name to the group.
9. **Quit the Auburn Messaging System:** 
   1. Check if the program posts a thank you message and then exits.