COMPUTER SCIENCES AND SOFTWARE ENGINEERING
AUBURN UNVERSITY

COMP 2710

Software Construction

Fall 2014

Lab 1
Auburn Messaging System

Due: September 26, 2014

Analysis

Use Cases:

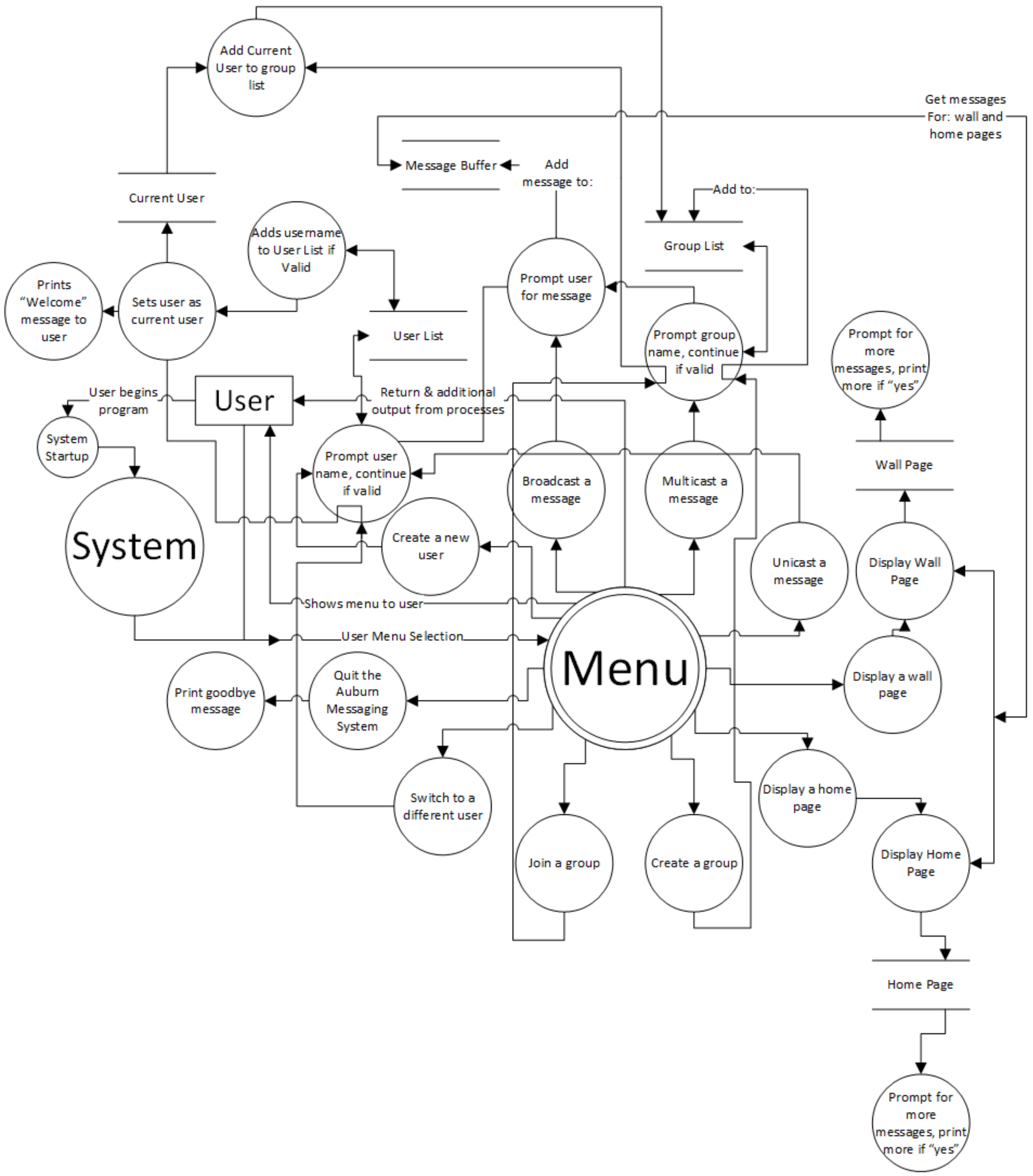
1. **Create a new user:** The program prompts for a name to be entered as a new user. Before adding the name it checks if it is already has been created. When a user is created, that user becomes the current user.
2. **Broadcast a message:** The program will prompt the user for the message which is then stored into the message buffer. To end a message, the user will enter a new line with a string "$$"followed by the enter key.
3. **Multicast a message:** The program will prompt for a group name first, then it prompts for a message to be sent to this group. To end a message, the user will enter a new line with a string "$$"followed by the enter key.
4. **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.
5. **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.
6. **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.

7. **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.
8. **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.
9. **Switch to a different user:** The program will prompt for a user name, then checks if the name is valid by checking if it has been created. If so, it switches current user to the one entered. If it is not a valid user, it displays the error message and repeatedly asks for the user name.
10. **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**
            - promptMenu – presents the user with the menu. Is the gateway function to all other classes' functions and variables.
            - quitSystem
        - **Variables**
        - **Error-Handling**
            - errorReturn – handles any and all error output to user throughout all system functions
    - **User –** This class handles the user-related objects and information
        - **Functions**
            - createNewUser
            - displayWallpage
            - displayHomepage
            - switchUser
        - **Variables**
            - userList
            - userName
            - currentUser
    - **Group –** This class handles the user-related objects and information
        - **Functions**
            - createGroup
            - joinGroup
        - **Variables**
            - groupList – a collection of collections
            - groupName
    - **Message Manager –** This class handles all message-related objects and functionality
        - **Functions**
            - broadcastMessage
            - mulitcastMessage

        - unicastMessage
    - **Variables**
        - messageNew
        - messageBuffer


Add Current User to group list
Get messages For: wall and home pages
Message Buffer
Add message to:
Add to:
Current User
Group List
Adds username to User List if Valid
Prompt user for message
Prompt group name, continue if valid
Prints "Welcome" message to user
Sets user as current user
User List
Prompt for more messages, print more if "yes"
User begins program
User
Return & additional output from processes
System Startup
Prompt user name, continue if valid
Broadcast a message
Multicast a message
Wall Page
System
Create a new user
Unicast a message
Display Wall Page
Shows menu to user
User Menu Selection
Menu
Print goodbye message
Quit the Auburn Messaging System
Display a wall page
Display a home page
Switch to a different user
Display Home Page
Join a group
Create a group
Home Page
Prompt for more messages, print more if "yes"

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:**
    a. Check if the menu is printed correctly.
2. **Create a new user:**
    a. Check if the program prompts for a name to be entered for a new user, correctly.
    b. Before adding the name it must check if it is already has been created.
    c. Check if when a user is created, that user becomes the current user.
    d. Check if the printed prompt/welcome banner is correct with the user's name included – centered on screen and surrounded by a box.
3. **Broadcast a message:**
    a. Check if the program prompts the user for the message.
        i. Check if the program allows multiple lines for the message
    b. Check if the message is then stored into the message buffer.
    c. Check if a new line with "$$" entered, followed by the "enter" key ends the message.
    d. The final string "$$"must not be stored in the message buffer, neither should it be displayed in the wall or home pages.
4. **Multicast a message:** T
    a. Check if the program prompts the user for a group name
        i. Checks if it is valid
        ii. Then either errors if invalid or continues if valid
    b. Check if the program prompts the user for the message.
        i. Check if the program allows multiple lines for the message
    c. Check if the message is then stored into the message buffer.
    d. Check if a new line with "$$" entered, followed by the "enter" key ends the message.
    e. The final string "$$"must not be stored in the message buffer, neither should it be displayed in the wall or home pages.
5. **Unicast a message:**
    a. Check if the program prompts the user for a user name
        i. Checks if it is valid
        ii. Then either errors if invalid or continues if valid
    b. Check if the program prompts the user for the message.
        i. Check if the program allows multiple lines for the message
    c. Check if the message is then stored into the message buffer.
    d. Check if a new line with "$$" entered, followed by the "enter" key ends the message.
    e. The final string "$$"must not be stored in the message buffer, neither should it be displayed in the wall or home pages.
6. **Display wall page:**
    a. Check if program first displays a title indicating that it is displaying the current user's wall page.

    b. Check if it displays the two latest messages in the current user’s wall page
        i. Must be in reverse chronological order. (The most recently posted messages will be displayed before the earlier messages.)
        ii. Messages from different users must be separate by a blank line.
    c. Check if the program only displays the user’s messages and the group (for multicast) or recipient (for unicast) or “All” (for broadcast)
        i. Must be in reverse chronological order.
    d. Check if the program, after displaying the two latest messages, it then prompts the user if they want more messages.
        i. Check if the response is “no”, then it will stop displaying messages
        ii. Check if the response is “yes”, it will display all the remaining messages from that user.
    e. Check if there are two or fewer messages then the program will not prompt for more messages.

7. **Display home page:**
    a. Check if the program first displays the current user home page, e.g. “Joe’s Wall Page”.
    b. Check if it displays the two latest messages in the current user’s wall page
        i. Must be in reverse chronological order. (The most recently posted messages will be displayed before the earlier messages.)
        ii. 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.
        iii. The group name or the recipient name or “All” must not be displayed.
        iv. Messages from different users must be separate by a blank line.
    c. Check if the program, after displaying the two latest messages, it then prompts the user if they want more messages.
        i. Check if the response is “no”, then it will stop displaying messages
        ii. Check if the response is “yes”, it will display all the remaining messages from that user.
    d. Check if there are two or fewer messages then the program will not prompt for more messages.
8. **Create a group:**
    a. Check if the program prompts for a name to be entered for a new group, correctly.
    b. Before adding the group, it must check if it has already been created.
        i. Check if so, then it displays an error message and prompts for another group name.
    c. Check if it creates the group.
9. **Join a group:**
    a. Check if the program properly prompts for a group name.
    b. Check if the program checks if the group has already been created.
        i. Check if not, then it displays an error message and prompts for another group name.
    c. Check if the program adds the current user name to the group.
10. **Switch to a different user:**
    a. Check if the program prompts for a user name

   b. Check if the program then checks if the name is valid by checking if it has been created.
      i. If so, it switches current user to the one entered. If it is not a valid user, it displays the error message and repeatedly asks for the user name.
      ii. The program must not exit or terminate when this naming error occur.
   c. Check if when the program has switched to a different user, the program prompts a welcome back banner, e.g. saying “Welcome back to Auburn Messaging System, Jane”, which should be centered on the screen and surrounded by a box.

11. **Quit the Auburn Messaging System:**
   a. Check if the program posts a thank you message and then exits.