Austin Osborn
COMP 2710-001
Lab 1 Design Portion (With Revisions)
Aeo0015@auburn.edu

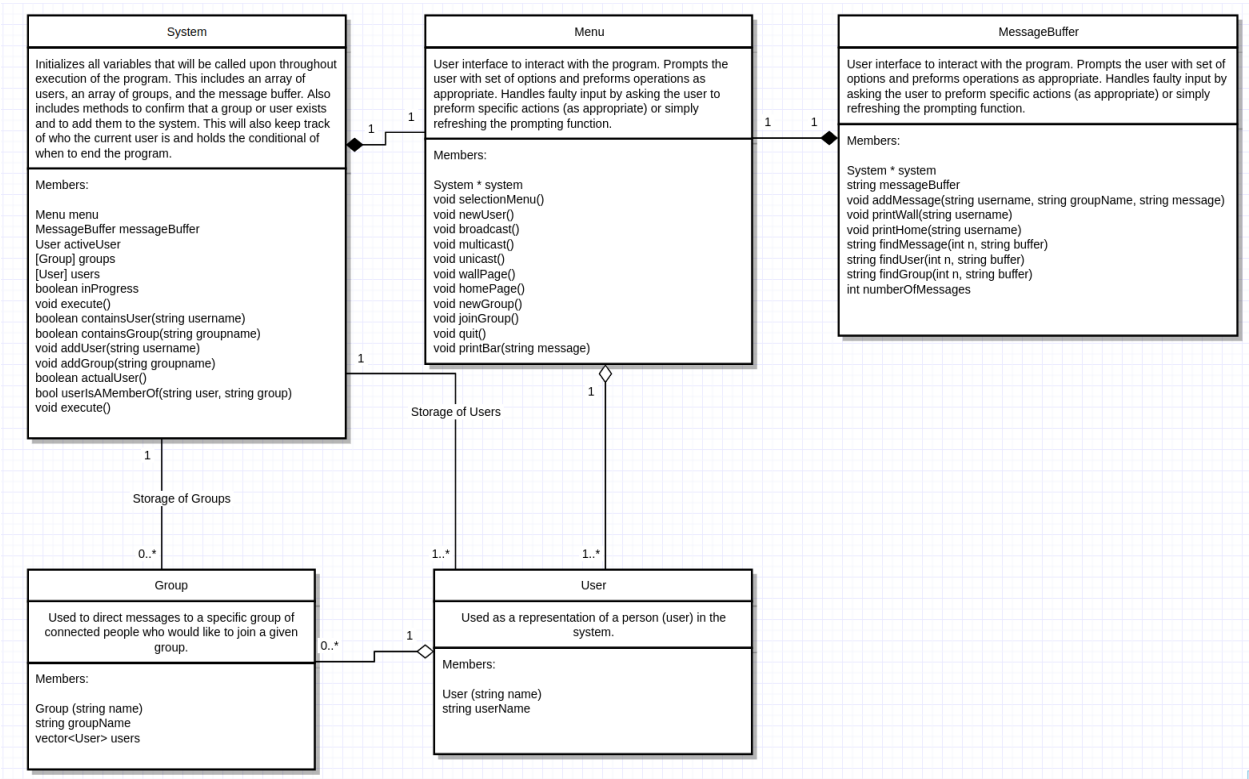
//NOTE: THIS IS THE REVISED COPY.
The only thing that changed slightly was the class diagram. Changes were minor member functions (added some convenience functions to clean up code and pointers to the System were added to Menu and MessageBuffer classes for interactivity). DFD models remained exactly the same. Use and test cases remain same.

Analysis (Use Cases)

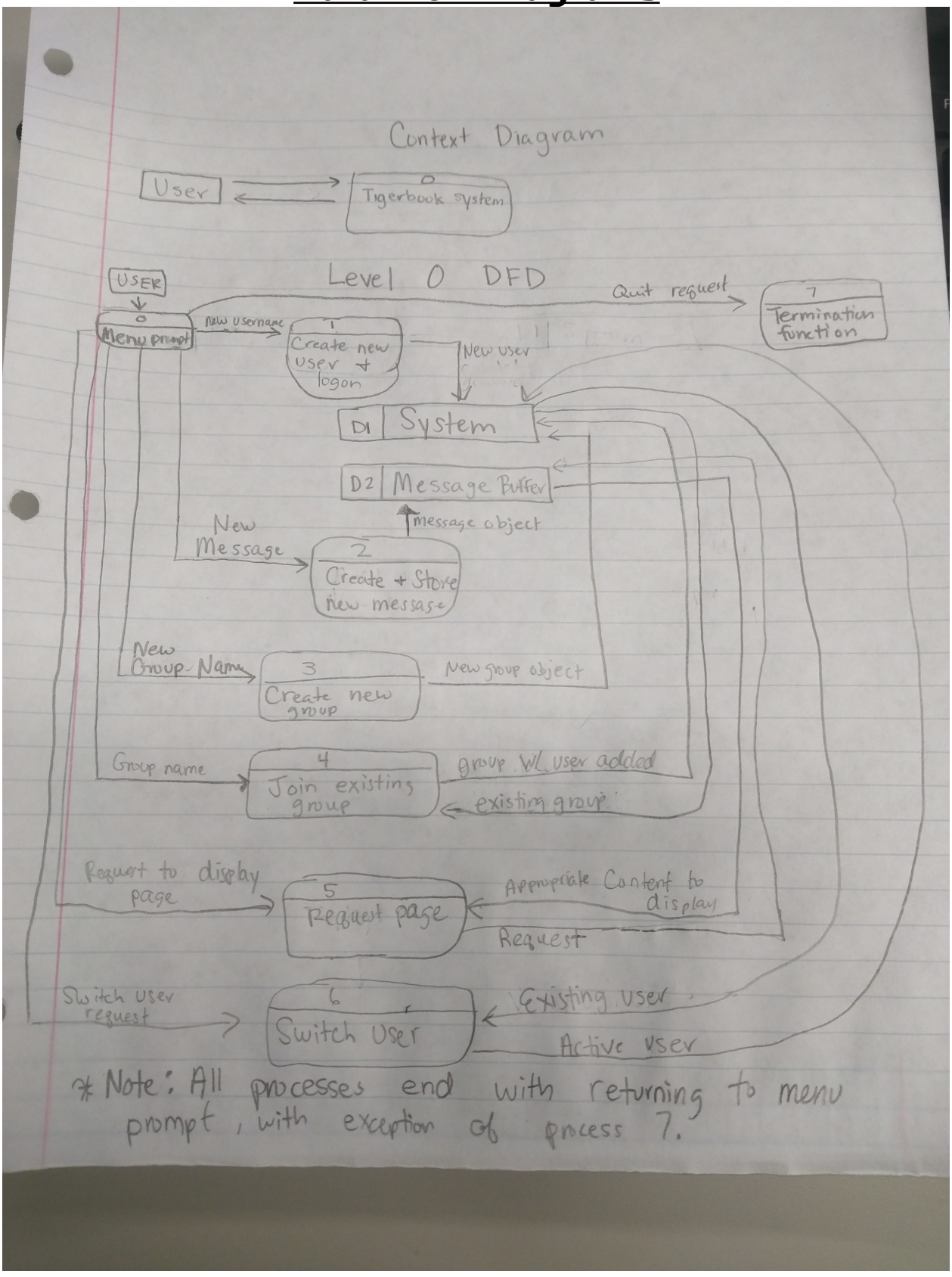
- **Initial execution:** The user is greeted with a greeting message and presented with a menu of options (selection prompt). The first thing they should do is create a new user, as all other options in the menu are contingent upon at least one user being present.
- **Create a new user:** The user selects this option from a list of options. They then enter a name. This user is now logged in as the active user.
- **Send a message:** The user selects one of three options (broadcast, multicast, or unicast) depending on who they want to send the message to. They are then prompted to enter the recipient user or group (as appropriate). If the recipient is valid, then they can enter a message and end their message by going to a new line and typing "^!". This message is then added to the universally-shared message buffer.
- **Display a wall page:** The user wants to select this option to see a list of all messages they have sent. If they haven't sent any, they will be told their wall is empty. Otherwise, the system will show the user up to 2 messages they have sent, and if there are more asks the user if they want to display all.
- **Display a home page:** The user wants to select this option to see a list of all messages directed at them. This includes all broadcast messages, any multicast message for which they are a member of the group, and any unicast message directed at them. The system will show up to 2 messages, and if there are more will ask the user if they want to display all.
- **Create a group:** The user selects this option to create a new group that users can join. The user is prompted to enter a name for this new group. If it doesn't already exist, it's created, otherwise the user is alerted that their input is invalid and directed back to the selection prompt.

- **Join a group:** The user selects this option to add themselves to an already existing group. They are prompted to enter the name of the group they would like to enter. If this group exists, they are added to the group. Otherwise, they are told the group doesn’t exist and asked if they’d like to create it. If they choose this option, the group is created and the user is added to it.
- **Switch users:** The user would like to change from one User to another User. They are prompted to enter the name of the User they would like to switch to. If it exists, the active User is changed. Otherwise, the user is asked if they would like to create this User and one is created if they choose “yes”. Otherwise, they are greeted once again with the selection prompt.
- **Quit:** The user would like to close the program. They choose this option and the program will terminate.

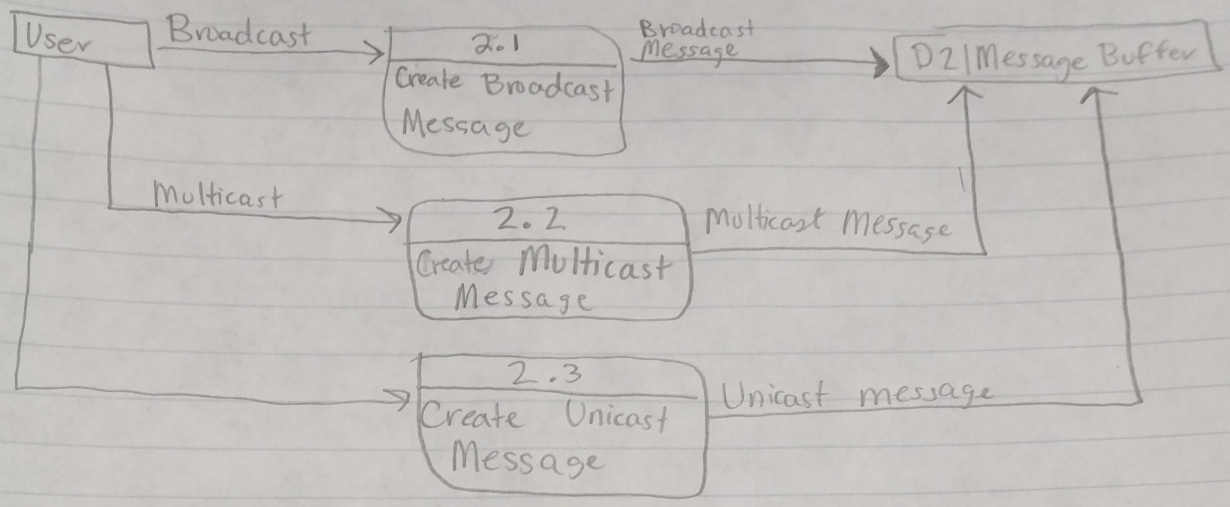
Class Diagram



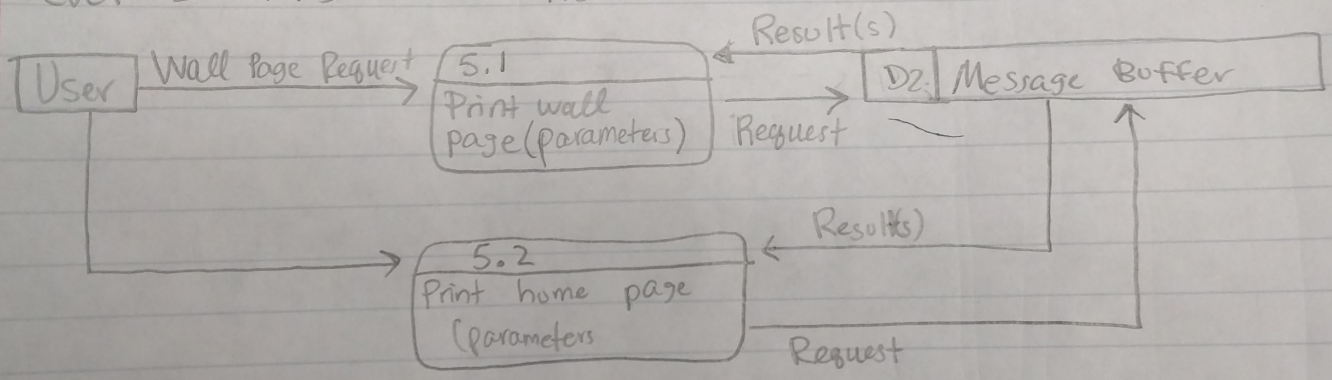
Data Flow Diagrams



Level 1 DFD for Process 2



Level 1 DFD for Process 5



Testing (Test Cases)

A. System Wide

1. Program is first executed.
   a) Program should display box containing “Welcome to tigerbook social network”
   b) Menu should appear displaying user list of choices
   c) System class should have empty array of Users, empty array of Groups, and no active users
2. User inputs a string on menu prompt and presses enter
   a) If input invalid, system should ask again for valid input
   b) If input is valid, behavior is proper for whatever function is called (specific examples will be listed in part B of “Testing (Test Cases)”

B. Specific Objects/Functions

1. Broadcast message request sent
   a) Program pulls from System member, activeUser, to ensure there is a user signed in (if not, this info is conveyed to the user and asked to create a new user (since this can only happen when no initial user is created by flow of program))
   b) Prompts user to enter their message
   c) Message_Buffer should add a message in the following format: “<! activeUser_username_here:#All!>message_here” with no errors. Uses messageBuffer member addMessage(string username, string groupname, string message) to do this.
2. Mutlicast message request sent
   a) Program pulls from System member, activeUser, to ensure there is a user signed in (if not, this info is conveyed to the user and asked to create a new user (since this can only happen when no initial user is created by flow of program))
   b) Prompts user to enter receiving group
      - If group doesn’t exist, then user is told this and asked if they would like to create group
        - If “yes”, then a new group is created and added to the System class array of Group.
        - User is also asked if they would like to join newly created group.
          - If “yes”, user is added to this group.
   c) Prompts user to enter their message
   d) Message_Buffer should add a message in the following format: “<! activeUser_username_here:#group_name_here!>message_here” with no errors. Uses messageBuffer member addMessage(string username, string groupname, string message) to do this.

3. Unicast message request sent
    a) Program pulls from System member, activeUser, to ensure there is a user signed in (if not, this info is conveyed to the user and asked to create a new user (since this can only happen when no initial user is created by flow of program))
    b) Prompts user to enter receiving user
        - If user doesn't exist, the user is told this and the program should restart menu selection prompt.
    c) Prompts user to enter their message
    d) Message_Buffer should add a message in the following format: "<! activeUser_username_here:#recipient_user_here!>message_here" with no errors. Uses messageBuffer member addMessage(string username, string groupname, string message) to do this.
4. Create group request sent
    a) User is prompted to input name of their new group
    b) System member containsGroup(string groupName) is called.
        - If returns false, then new group is created
        - If returns true, then user is told group already exists and is reprompted to menu selection.
5. Join group request sent
    a) User is prompted to input name of group they wish to join
    b) System member containsGroup(string groupName) is called.
        - If returns true, then user is added to group with that name
        - If returns false, then user is alerted that the group does not exist and asked if they would like to create it.
            - If "yes", then new group with that name is created and user is asked if they would like to join this newly created group
                - If "yes", then user is added to array of User's in this newly created group
6. Request wall page sent
    a) printWall(string username) is called with System.activeUser.username used as parameter.
        - Method should output first two messages of which the user is sending (ie, <!username part)
        - Method will then prompt user asking if they would like to display all.
7. Request home page sent
    a) printHome(string username) is called with System.activeUser.username used as parameter
        - Method should output first two messages either unicasted to them, containing a group of which they are a part of (except for messages sent by themselves), and any broadcast messages
        - Method should then prompt user asking if they would like to display all
8. Switch user request sent

a) User should be prompted to enter the name of the User they would like to switch to
b) System method switchActiveUser(string username) should be called with the parameter of the name they just typed in.
    - containsUser(string username) is called to ensure this user exists
        - If not, they are told the user doesn’t existing
        - If so, the activeUser is set to the selected User object

9. Quit program request sent
    a) System member inProgress is set to false and the main while loop is broken.
    b) Program exits gracefully with exit code 0.