Seth Denney
COMP 2710 Lab 2
Program Design Outline

I. Use Cases

A. Case – User Views Friend Group's Messages

i. User logs in. User chooses to display his or her home page. Aubiebook displays the last three messages from the user's friend group. If more than three messages are available, Aubiebook displays an option to show all messages. The user chooses this option. All messages from the user's friend group are displayed in reverse chronological order. The user then quits Aubiebook.

B. Case – User Types a Message and Views Own Message History

i. User logs in. User chooses to post a message. After posting the message, the user chooses to display his or her wall page. Aubiebook displays the last three messages posted by the user. If more than three messages are available, Aubiebook displays an option to show all messages. The user chooses this option. All messages posted by the user are displayed in reverse chronological order. The user then quits Aubiebook.

C. Case – User Creates Profile, Switches Users, and Adds a Friend

i. User chooses to create a new user, and enters a user name. The user then switches to an existing user profile, and chooses to add the newly created user as a friend on Aubiebook. After selecting the 'Add a Friend' option, the user enters the desired user name, and the appropriate profile is added to the current user's friend list. The user then quits Aubiebook.

D. Case – User Tweets a Message and Views Tweets

i. User logs in. User chooses to tweet a message. User enters tweet, and then views tweets. Tweets from all users are displayed in reverse chronological order. Only the (up to) three latest tweets are displayed until the user opts to display more. The user then quits AubieBook.

II. Classes

A. **AubieBook** – The *Aubiebook* class holds all system-level functionality in the Aubiebook application. It contains a *main()* method that will be run as the primary driver function for Aubiebook. It uses the *User* class and the *Page* class to store message data and perform object-specific functions.

i. Methods

a) void *createUser()*
- instantiate a *User* object
- sets *currentUser* to new User object
- prints welcome banner

b) void *displayMenu()*
- display main menu options
- read in user menu selection
- call appropriate function (switch statement)

c) void *postMessage()*
- prompt for message
- read in message ('$\n' terminated)
- call *page.addMessage("message",* message*)* (sans '$')

d) void *tweet()*
- prompt for tweet
- read in tweet ('$\n' terminated)
- call *page.addMessage("tweet",* message*)* (sans '$')

e) void *displayWall()*
- call *page.displayPage(currentUser, "wall")*

f) void *displayHome()*
- *call page.displayPage(currentUser, "home")*

g) void *addFriend()*
- prompt for name of friend
- read in name
- check name against visible User files; if valid, call *currentUser.addFriend(name)*; if not valid, print error message
- call *currentUser.addFriend(name)*

h) void *quitApp()*
- exit gracefully

i) void *createFiles()*
- checks for the existence of necessary files
- if not created, create them

j) bool *findUser(*string *name)*
- checks if *name* is a valid user (if *name.txt* is a valid file)

- ii. **Data Fields**
  - a) User *currentUser*
    - pointer to currently selected *User* object
  - b) Page *page*
    - Page object that stores and prints messages

B. User – The *User* class represents a single user profile on Aubiebook. It stores the user's user name and friends list, as well as functions to interact with this data. The *User* class is used by both *Aubiebook* and *Page*.

- i. **Methods**
  - a) constructor *User(*string *name)*
    - creates/loads a user named *name*
  - b) void *addFriend(*string *friend)*
    - adds *friend* to *friendsList* if not already found in *friendsList*
  - c) boolean *isFriend(*string *friend)*
    - return true if *friend* is member of *friendsList*
  - d) string *getName()*
    - return *name*
  - e) vector<string> *getFriends()*
    - returns friendsList
- *ii.* ***Data Fields***
  - a) string *name*
    - the user name
  - b) vector<string> *friendsList*
    - vector that includes a string username for each of the user's friends

C. Page – The *Page* class contains functionality related to message posting, tweeting, and page-viewing. *Page* is used by the *Aubiebook* class.

- i. **Methods**
  - a) void display*Page(*User *user*, string *pageType)*
    - print appropriate title banner (wall or home)
    - if *pageType* is *"wall"*, print only messages where the username is *user.name*
      - display latest 3 (formatted) messages/tweets from current user in reverse chronological order
      - if more messages/tweets exist, print prompt to display remaining messages/tweets
      - if three or fewer messages/tweets exist, do not prompt
    - if *pageType* is *"home"*, print only messages where *user.isFriend(*username*)* is 'true'
      - display latest 3 (formatted) messages from current user's friends list and ANY tweets in reverse chronological order
      - if more messages/tweets exist, print prompt to display remaining messages/tweets
      - if three or fewer messages/tweets exist, do not prompt
  - b) void *addMessage(*string *username)*
    - append formatted message to front of *username.txt* (with timestamp)
  - c) void *tweet(*string *username)*
    - append formatted tweet to front of tweet*.txt* (with timestamp and *username*)
  - d) void *loadMessages()*
    - loads all of the user's messages into *messageBuffer* (dumps file contents to memory)
  - e) int *printMessages(*User& *user*, string *pageType*, int *limit*, int *prevTime)*
    - calls *getNext()* and then *printStruct()*
  - *f)* NextMessage *getNext(*User& *user*, string *pageType*, int *prevTime)*
    - check through appropriate files for next message and stores message data in NextMessage struct to return
  - g) void *printStruct(*NextMessage *next*, string *pageType)*
    - prints out data in *next* with formatting appropriate to *pageType*
- *ii.* ***Data Fields***
  - a) string *messageBuffer*
    - contains a list of buffered usernames and messages in reverse chronological order
  - b) int *messageCount*
    - counts messages displayed thus far
  - c) struct nextMessage
    - contains int *time*, string *name*, string *message*

III. Test Cases

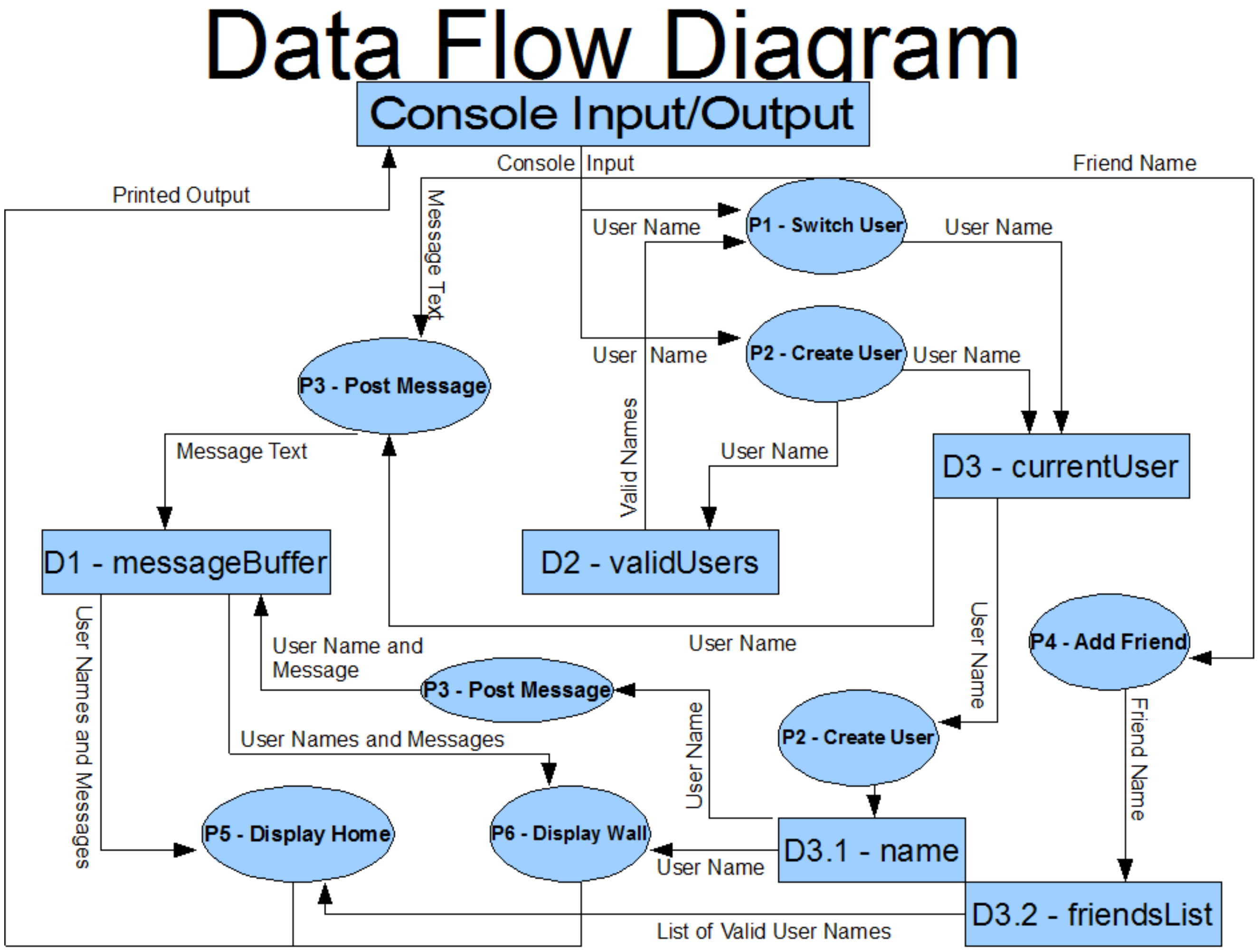
A. **Invalid Command**
   i. **Scenario –** User enters a command (menu option ID number, yes/no, user name) that is either a) incorrectly typed data (integer instead of lettered name, non-integer characters for an integer menu option, etc.) or b) correctly typed, but not within the domain of acceptable options ("100" where only "1" - "7" would be allowed, or the name of a user that has not yet created a profile)
   ii. **Result –** System should print an error message, and either a) prompt for a correct entry, or b) return to the menu view.

B. **Add Invalid Friend**
   i. **Scenario –** User attempts to add a friend who is either a) the user, b) already a friend, or c) not a valid user.
   ii. **Result –** Aubiebook should print an appropriate error message and return to the menu.

C. **Display Wall Page**
   i. **Scenario –** User chooses to display Wall Page (option w).
   ii. **Result –** Wall Page should display ONLY posts and tweets from current user in reverse chronological order. Wall Page should only display last three messages, and then prompt the user to display the rest if there exist more. Wall Page should also not display the user's name.

D. **Display Home Page**
   i. **Scenario –** User chooses to display Home Page (option h).
   ii. Result – Home Page should display all tweets as well as posts from current user and friends in reverse chronological order. Home Page should only display last three messages, and then prompt the user to display the rest if there exist more. Home Page should display each user's name.

IV. Data Flow Diagram


Data Flow Diagram
Console Input/Output
Printed Output
Console Input
Friend Name
Message Text
User Name
P1 - Switch User
User Name
User Name
P2 - Create User
User Name
P3 - Post Message
Valid Names
Message Text
User Name
D3 - currentUser
D1 - messageBuffer
D2 - validUsers
User Names and Messages
User Name and Message
User Name
User Name
P4 - Add Friend
P3 - Post Message
User Names and Messages
User Name
P2 - Create User
Friend Name
P5 - Display Home
P6 - Display Wall
D3.1 - name
User Name
D3.2 - friendsList
List of Valid User Names