**Programming Project – Simple Facebook**

**CPE 111 Programming with Data Structures**

**Presented to**

Dr. Sally Goldin

**RUN EVERY VERB Team**

Chawakorn Boonrin (Bright)

ID 60070503415 Section C

Nathaphop Sundarabhogin (Kla)

ID 60070503420 Section C

Natthawat Tungruethaipak (Tong)

ID 60070503426 Section C

Setthawut Leelawatthanapanit (Saab)

ID 60070503466 Section D

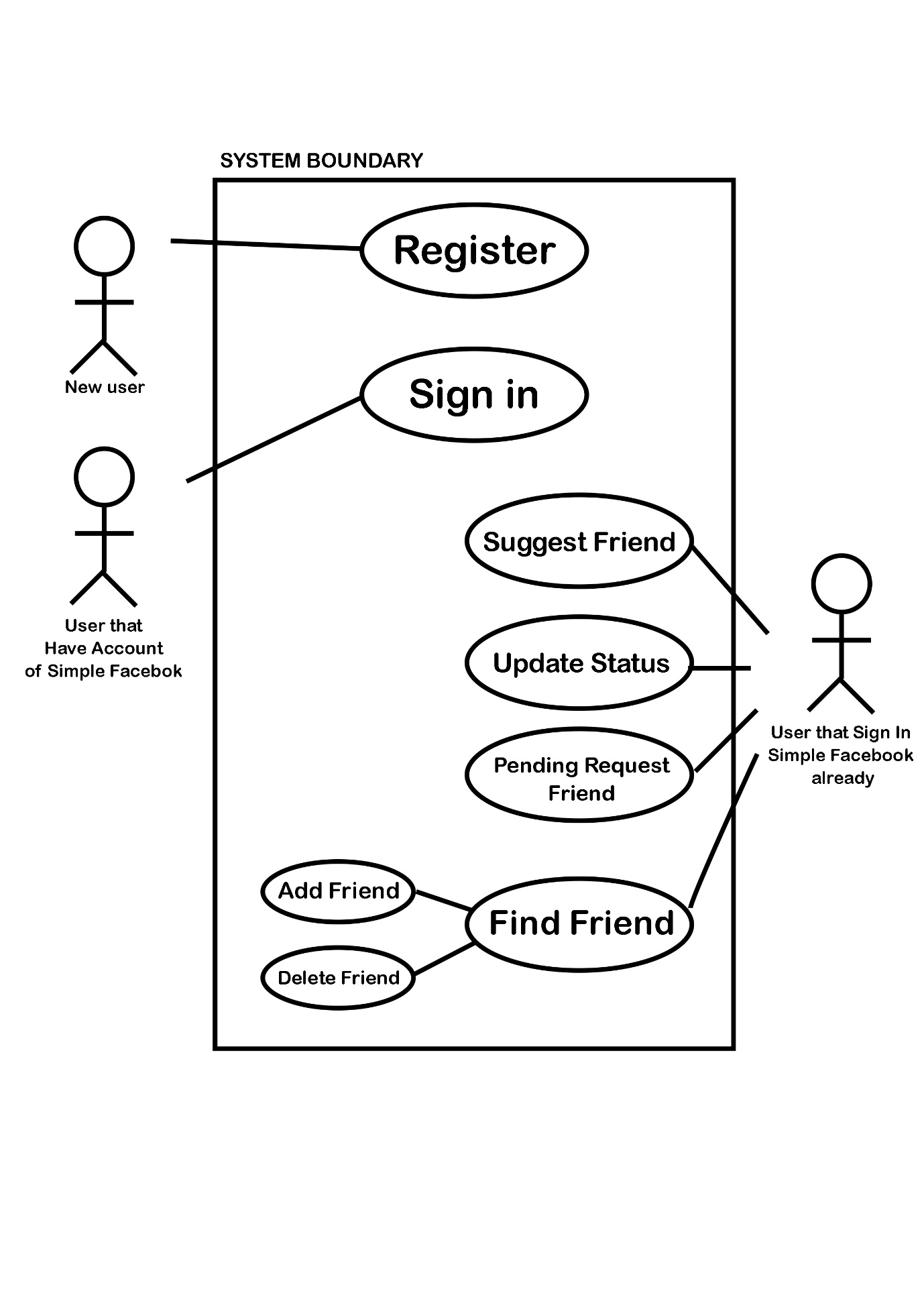
**SUMMARY STATEMENT OF PROBLEM**

This system will act as a social network that named Facebook. But it’s just simple Facebook. After user sign in, user can see status, comment on that status, find friends, add friend and delete friend. If user don’t have account, user can register a new account to login. All these works will manage, read and write data that we have in data structure.

Our program should have a data structure that represents the profile of the person, the status of each person and comment on each status. It should manage structure to know who a friend is or not a friend to show only friend’s status. And should manage to unfriend too (But if we not have data structure, it should create a new one.).

Once our program has loaded information about the profile, status, comment, and friend relations (Reading data from a file). The program asks user to register or sign in first. Then the program will show suggested friends and print friend’s status to user. After that program will loop ask user what to do in program 1) update status 2) comment 3) find friends 4) see next status. If user finds a friend, the program will ask user to add friends (If they are not a friend) or delete friends (If they are a friend). Our program must use the stored data to match the username that user signs in and check who is user’s friend and print status and comment to user. If user wants to comment on current status, our program must keep comment to data structure.

**USE CASE DIAGRAM**



**FUNCTIONAL REQUIREMENTS**

- Read a file from the database that has all user data, friends list, pending friend list and keep in a hash table about 1000 arrays ( If there are the same position it will keep in the tree. ).

- User data have E-mail, password, name, birthday, phone number, gender, status, friends and pending request.

- In status will include text, owner status, time, and all comments.

- In comment will include text, time, owner.

- Connect user that are friends together and linked pending request into user data.

- If there isn’t any database file, create new file.

- The system must allow the user to register, sign in or exit program.

- If the user signs up, get input and validate information correct or not. If finished register, add to the hash table.

- If user signs in, get E-mail and Password from the user. Then send E-mail to the hash function and search in the hash table. If found user data, let user login. If not, print not found a message.

- (1) When user login program will show suggest friend by see friend of friend relation. If the user does not have any friend, the program will random in a hash table to suggest a friend.

- Then print how many friends pending request are there.

- After display suggests friend. Go through friend list, print one friend’s status and all comment.

- Get command from user, there are command as below

* + ‘/c <text>’ Comment current status
  + ‘/s <text>’ Update new status to Simple Facebook
  + ‘/home’ Go to first status (It work like a button refresh)
  + ‘/help’ Show all command and display some

instruction

* + ‘/next’ Go to next status
  + ‘/pending’ Go to see who are request to be your friend in sample facebook
  + ‘/profile’ See all user’s information.
  + ‘/logout’ Log out from facebook
  + ‘/find <e-mail>’ Find user in simple facebook

After command ‘/pending’ user can use only this command

* + ‘/acceptf’ Accept to be friend
  + ‘/denyf’ Deny to be friend

Or ‘/home’

After search friend, user can use only this command

* + ‘/addfriend’ Add friend that person to see his news
  + ‘/delfriend’ Delete that friend

Or ‘/home’

- For ‘/home’, It work like a refresh button in Real Facebook. It will random a new status to show to user and start at (1) again.

- For ‘/help’, will show to user that first-time login ( and suggest this command when user input wrong command ).

- For ‘/next’, It will random friends to print status.

- For ‘/pending’, It will show amount of friend request and show only one user and some data ( name, gender, age ). Then asked to accept or deny friends. But you can’t ignore to see next friend pending ( You have to decide to be friend or not that user. Otherwise, ignore to decide and go back to home . ).

- When log out it will go back to ask register or sign in again.

- When find friends, user have to find by E-mail only. If found, print that user profile and then program will check that both users are friend or not. If not, programd will allow user to use only command ‘/addfriend’. But if yes, program will allow user to use only command ‘/delfriend’. Otherwise, user have to use ‘/home’.

**OPTIONAL**

- Let user to like all status and print amount of a person who like that status.

- Let user to change name, change password or delete user after using command ‘/setting’.

- Display all user status that user have been post.

- After search friend’s, user can see all that friend’s status.

- Can ignore to see next pending friend.

- Print amount of person who comments that status.

- Make hash table from constant number to dependent on amount of data.

**SIMPLE FACEBOOK USE CASES**

**Uses case name:** Register

**Actor:** New user

**Goal:** Let new user to create own account to login Simple Facebook

**Main success narrative:**

1. Ask E-mail, password, name, age, birthday, phone, gender.
2. Send E-mail to hash function and get position of hash table.
3. If there are same position, this data will keep in tree.

**Alternative narrative 1:**

1. If user input the same E-mail that have in data structure.
2. Print error message and ask again.

**Alternative narrative 2:**

1. Password must contain at least one character and one number, some allowed punctuation.
2. But if not, print error and ask again.

**Uses case name:** Sign in

**Actor:** User that have account of simple facebook

**Goal:** Let user to input E-mail and password to access to Simple Facebook

**Main success narrative:**

1. Ask E-mail and password. (\*Note: Our ID is E-mail\*)
2. send E-mail to hash function and search in hash table.
3. If found user data, let user login.

**Alternative narrative 1:**

1. If doesn’t found data of user by E-mail.
2. Print not found message and ask again or suggest for register.

**Alternative narrative 2:**

1. If found data of user by E-mail. But, the password doesn’t correctly.
2. Print an error message and ask again.

**Uses case name:** Suggest friend

**Actor:** User that sign in Simple Facebook already

**Goal:** Suggest new friend to user

**Main success narrative:**

1. Check another user that may relative to user that login.
2. Print suggest friend to user.

**Alternative narrative 1:**

1. If there are friends of friend.
2. Suggest friend of friend to user.

**Alternative narrative 2:**

1. If user don’t have any friend or any friend of friend.
2. Random suggest friend to user.

**Alternative narrative 3:**

1. If random but can’t found any user.
2. Print that don’t have any friends to suggest.

**Uses case name:** Update Status

**Actor:** User that sign in Simple Facebook already

**Goal:** Let user to update status on Simple Facebook

**Main success narrative:**

1. Get text status that user want to update.
2. Keep text that user input into data structure.
3. When user’s friend signs in to our Simple Facebook, Our program should manage to display that status to user’s friend.

**Uses case name:** Comment

**Actor:** User that sign in Simple Facebook already.

**Goal:** Let user to comment status that current display.

**Main success narrative:**

1. Get text comment that user want to comment.
2. Keep text into data structure of status that now display on the screen.
3. Can display both status and comment correctly.

**Uses case name:** Find friends

**Actor:** User that sign in Simple Facebook already.

**Goal:** Let user to find another user by E-mail.

**Main success narrative:**

1. Get E-mail of another user that want to find.
2. Send to hash function and get address of data back.
3. Display that user’s profile.
4. Then ask user to add friend or delete friend.

**Alternative narrative 1:**

1. If not found user.
2. Print error message.

**Alternative narrative 2:**

1. If both user are friend.
2. Allowed user to use command only delete friends or home.

**Alternative narrative 3:**

1. If both user are not friend.
2. Allowed user to use command only add friends or home.

**Uses case name:** Add friends

**Actor:** User that sign in Simple Facebook already and find another user.

**Goal:** Let user to add another user to be friend by send pending request.

**Main success narrative:**

1. Get command add friend from user.
2. Check that both user are not friends.
3. Keep pending request to user that we want to be friend.

**Alternative narrative 1:**

1. If both users are friend already.
2. Print error message and suggest command for delete friend.

**Uses case name:** Pending Request Friend

**Actor:** User that sign in Simple Facebook already.

**Goal:** Let user to check pending request and add to friend.

**Main success narrative:**

1. See which user are send pending request.
2. Ask user to accept or deny.
3. If user accept, manage to data to keep both user to be friend.

**Alternative narrative 1:**

1. If user deny to be friend.
2. Reject that data and let user to see another friend.

**Alternative narrative 2:**

1. User can’t ignore current pending request to see next pending request.
2. User must to decide current pending request to be friend or not.
3. Or can ignore it by go back to home.

**Uses case name:** Delete friends

**Actor:** User that sign in Simple Facebook already and find another user.

**Goal:** Let user to add another user to be friend.

**Main success narrative:**

1. Get command delete friend from user.
2. Check that both user are friends.
3. Manage data to keep both data of user to be unfriend.

**Alternative narrative 1:**

1. If both users are not friend.
2. Print error message and suggest command for add friend.

There is some knowledge that related to basic binary tree function. And we’ll used these words for basic function in binary tree.

**add tree node function:** Get new node and root note of tree then add new node to binary treewhich compare with E-mail. If there are same E-mail, it should print error.

**search tree function:** Find node that want to know in binary treewhich find with E-mail. If found data, return address of user back. If not, return NULL.

**delete tree node function:** Get node that want to delete and root node of tree then find node that want to delete in tree and deleted it. If can delete, return 1, If can’t, return 0.

**Add New User Data**

Ask data from user

Send E-mail to hash function

Get position of hash function

If there isn’t any data

Set that position in hash table to new data

End if

Else if there is some data

Get root node tree of that position in hash table

Send new data and root node tree to **add tree node** **function**

End if

**Find User Data**

Set found user to NULL

Get E-mail from user

Send E-mail to hash function

Get position of hash function

If that position of hash table is NULL

Print not found message

Return found user

End if

Else

Get root node tree of that position in hash table

Send E-mail and root node tree to **search tree function**

Set found user to receive address of user after send to search

If found user equal NULL

Print not found message

Return found user

End if

Else if found user not equal NULL

Print data profile

Return found user

End if

End if

**Add Friend**

Set current user to user that login

\*After find another user\*

Set another user to user that we found

Set found user to NULL

Set friend relation to 0

Send root node of friend of current user and E-mail of another user to **search tree function**

Set found user to receive address of user after send to search

If found user is NULL

Set friend relation to 0

Else if found user is not NULL

Set friend relation to 1

Get command from user

If user want to be friend

If friend relation equal to 0

Create new linked list of pending requests

Set user in new linked list to current user

Set next node of new linked list to head node of friend pending request of found user

Set head node of friend pending request of found user to new linked list

End if

Else if friend relation equal to 1

Print error message

End if

End if

**Friend Pending Request**

Set current user to user that login

Set temporary request to NULL

Get Command from user

If user want to see friend pending request

While friend pending request not equal to NULL

Print head node of friend pending request

Ask command for user

If accept friend

Send head node of friend pending request and root node of friend to **add tree node** **function**

Send current user and root node of friend of head node of pending request to **add tree node** **function**

Set head node of pending request to next node of head node

End if

If deny friend

Set temporary request to next list of head node

Free linked list of head node

Set head node of friend pending request to temporary friend

End if

**Delete Friend**

Set current user to user that login

\*After find another user\*

Set another user to user that we found

Set found user to NULL

Set friend relation to 0

Send root node of friend of current user and E-mail of another user to **search tree function**

Set found user to receive address of user after send to search

If found user is NULL

Set friend relation to 0

Else if found user is not NULL

Set friend relation to 1

Get command from user

If user want to delete friend

If friend relation equal to 1

Send found user and root node of friend of current user to **delete tree node function.**

Send current user and root node of friend of found user to **delete tree node function.**

If value that return after send to delete equal to 0

Print error message.

End if

End if

Else if friend relation equal to 0

Print error message

End if

End if