COMP3322 Modern Technologies on World Wide Web

Assignment 1: A Simple Social Network Application (11%)

[Learning Outcomes 2, 3]

Due Date: 23:55, Wednesday March 7, 2018

Total mark is 100.

Overview

In this assignment, you are going to develop a simple **social network site** using technologies learned in class for creating dynamic web pages, i.e., PHP, JavaScript and AJAX. The social network application implements a few simplified functionalities, including displaying posts from friends, replying to posts from friends, and staring your friends into a "starred friends" list.

Task 1 (10 marks) Create basic index.html and prepare database tables

Create the **index.html** which renders the layout shown in the following figure, i.e., the page has three divisions: heading, starred friend list and posts. You can design any heading, showing the name of your social network site. Detailed requirements on implementing the friend list and post divisions will be given in Task 2, 3, and 4.

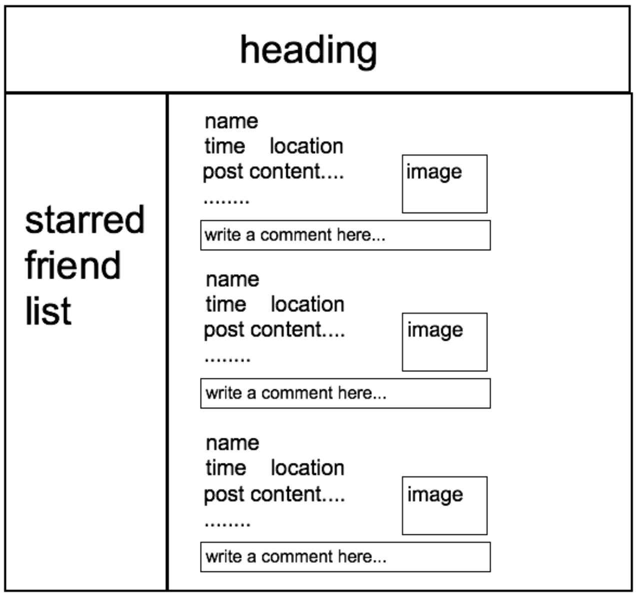

Fig. 1

Prepare the database on the server side:

1. Create a database table **posts** in your database in the server **sophia.cs.hku.hk**.

Each post entry in the table must contain the following attributes:

- **postID** – The unique ID of the entry. Assume that the postID is an integer starting from 1.
- **friendID** – The unique ID of the friend who produced the post. Assume that the friendID is an integer starting from 1.
- **time** – The date when the post was produced (less than or equal to 20 characters).
- **location** – The location where the post was produced (less than or equal to 20 characters).
- **content** – The content of the post.
- **image** – The path (can be absolute or relative path) to the image file of the post (e.g., a photo related to the post content). Let us assume that each post has one and only one associated image. The path should be less than or equal to 255 characters.

For your reference, the following SQL creates the posts table.

```
CREATE TABLE posts (
    postID int(11) NOT NULL,
    friendID int(11) NOT NULL,
    time varchar(20) NOT NULL,
    location varchar(20) NOT NULL,
    content longtext NOT NULL,
    image varchar(255) NOT NULL,
    PRIMARY KEY    (postID)
)
```

The following SQL inserts a post entry into the posts table.

```
INSERT INTO posts( postID , friendID, time, location, content, image)
VALUES (
1,
1,
'February 12 2018',
'Calgary, Alberta, Canada',
'I am currently traveling in Calgary.',
'images/1.jpg'
);
```

Please prepare at least 3 pages of posts in the database table. That is to say, if you set the initial number of the posts to display in the posts division to be 3, then you should prepare at least 9 posts in the database.

2. Create a database table **comments**. Each comment entry in the table must contain the following attributes:

- **commentID** – The unique ID of the entry. Assume that the commentID is an integer starting from 1.
- **postID** – The unique integer ID of the post which the comment was posted on.
- **time** – The date when the comment was produced (less than or equal to 20 characters).
- **commContent** – The content of the comment.

3. Create a database table **friends**. Each friend entry in the table must contain the following attributes:

- **friendID** – The unique ID of the entry. Assume that the friendID is an integer starting from 1.
- **name** – The name of the friend (less than or equal to 20 characters).
- **starred** – whether the friend has been starred or not (1 character: 'Y' for yes and 'N' for no).

Please learn from the SQL statements given in 1 to insert entries into the friends table. Please prepare a sufficient number of friend records in the table, based on the number of posts you have prepared in the posts table. The default value for "starred" attribute should be set to 'N'.

Task 2 (30 marks) display the posts

Listing posts from friends is a basic functionality in a social network system. The feature we are implementing is to display a fixed number of posts in the initial view of the page (you can specify the initial number of posts for display), and then when you scroll down the scrolling bar, more posts are loaded on the bottom of the posts division (you can specify the number of posts to load each time, which can be smaller or equal to the initial number for display). Please note that scrolling down the page will load the posts **without reloading the entire web page.**

1. JS events and event handler.

1.1 When the **index.html** is first loaded, load the initial number of the posts in the posts division. (Hint: register a JS event **onload** in the **<body>** tag)

1.2 Whenever the scrolling bar is scrolled (in this simplified implementation, we do not distinguish scrolling up or down), more posts will be loaded in the posts division. (Hint: use the JS event **onscroll**; see http://www.w3schools.com/jsref/dom_obj_event.asp and google for examples on using this event).

1.3 Implement AJAX codes in the event handler functions to communicate with the server for retrieval of posts from the database. You are suggested to follow the 6 steps you learnt in the lecture to implement the AJAX logic.

Each post (name, date, location, content, etc.) should be displayed following the layout in Fig. 1. An input textbox should be displayed underneath each post, which by default shows "write a comment here…".

2. Server-side processing logic.

Create a PHP file **handlePostDisplay.php**, which should retrieve the following information from the database: (1) a number of posts from the posts table, according to the request from the client side; (2) names of the friends publishing those posts from the friends table; (3) existing comments on those posts from the comments table. Then it sends needed information back to the client.

Task 3 (20 marks) post comments

You can enter your comment in the input textbox underneath each post. When you press "enter", the comment typed should be sent to the server and stored into the comments table in the database. After the comment has been successfully stored into the database, on the client side, you should display the comment above the input textbox, in the format of "You said: comment just entered", with the date when the comment was added into the database displayed below (please refer to Fig. 2). Then the input textbox should display "write a comment here…" again. All these should not cause reload of the entire web page.

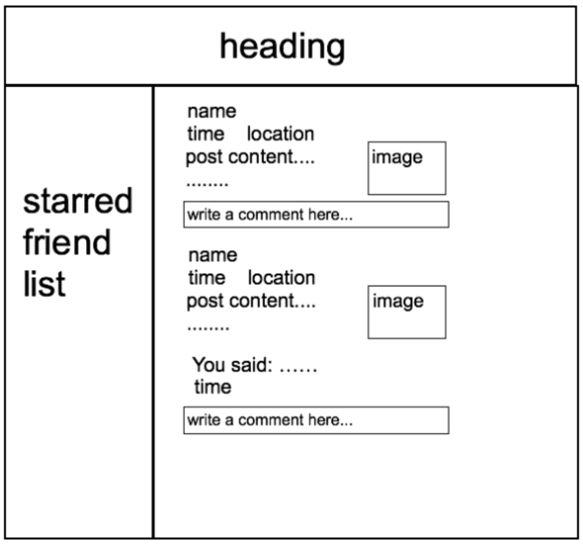

Fig. 2

Similar to Task 2, you should decide the JS event to trigger, implement the event handling AJAX code on the client side, and a PHP file **handleComments.php** on the server side to receive your comment and store it into the comments table in the database. Note that there can be multiple comment entries displayed above the input textbox, if you have posted multiple comments on the same post.

Task 4 (10 marks) display starred friend list

Upon loading of the HTML page, a list of starred friends should be displayed in the starred friend list division. Especially, names of the friends whose "starred" field is 'Y' in the friends table in the database should be retrieved and displayed in the division. You should decide how to implement the client-side code and a PHP file **handleFriendlistDisplay.php** on the server side for retrieving starred friends from the server side and displaying them on the client side.

Task 5 (20 marks) star/unstar a friend

Within the posts division on the right, please add a star icon following the name of the friend in each post. There should be two versions of the star icon: one to display when the friend is not starred (e.g., a greyed one), and one to display when the friend is starred (e.g., a highlighted one). When you click on the star icon, a request to change the starred status of this friend will be sent to the server. Upon receiving the request, the server should check the friends table in the database: if this friend is not starred yet (its "starred" attribute is 'N'), star this friend (change its "starred" attribute to 'Y'); if the friend was starred before, un-star it (change its "starred" attribute from 'Y' to 'N').

When the starred status of the friend has been successfully changed in the database, on the client side, (1) in the posts division, the star icon following the friend's name should be changed to the correct starred or un-starred version, and (2) in the starred friend list division, the list should be updated: if a friend is starred, add its name into the list; if a friend is un-starred, remove its name from the list.

You should decide the JS event to trigger, implement the event handling AJAX code on the client side, and a PHP file **handleStarringFriend.php** to implement the server-side logic. All the above functionalities should not cause reload of the entire web page, but only partial update of the page.

Task 6 (10 marks) Style the page using CSS

Please use a separate **style.css** file to include all your styling rules.

1. (**5 marks**) Use CSS styling you choose to make your page look nice with good layout
2. (**5 marks**) Implement Responsive Web Design to make your page look nice on screens of different sizes.

Notes:

1. You can use either basic JavaScript or jQuery to implement client-side scripting.
2. Maintain a good programming style: avoid redundant code; the code should be easy to understand and maintain.
3. You should carefully test all the functionalities stated in this handout.

Submission

You should submit the following files:
(1) Index.html
(2) style.css
(3) handlePostDisplay.php
(4) handleComments.php
(5) handleFriendlistDisplay.php
(6) handleStarringFriend.php
(7) exported table structure you created in your database (on phpMyAdmin, you can export your tables on the "Export" tab: choose "Quick - display only the minimal options", Format: "SQL", and click "Go".)

Please compress all the above files in a .zip file and submit it on Moodle:

(1) Login Moodle.
(2) Find “Assignments” in the left column and click “Assignment 1”.
(3) Click “Add submission”, browse your .zip file and save it. Done.
(4) You will receive an automatic confirmation email, if the submission was successful.
(5) You can “Edit submission” to your already submitted file, but ONLY before the deadline.