COMP3322B Modern Technologies on World Wide Web

Assignment 1 (22%)

[Learning Outcome 2]
Due by: 23:59, Friday April 1 2022

Overview

In this assignment, you are going to develop a web-based **news feed application** using NodeJS/Express.js, JavaScript/jQuery, AJAX, HTML and CSS. The application implements a few simplified functionalities, including displaying news lists, switching between different news list pages, searching news headlines, displaying a news entry, displaying comments posted to a piece of news, and posting comments on a piece of news after login. The web-based news feed application is to be implemented by the following code in an Express app:
app.js
./public/newsfeed.html
./public/javascripts/script.js
./public/stylesheets/style.css

and accessed at http://127.0.0.1:8081/newsfeed.html .

Preparations

1. Following steps in **setup_nodejs_runtime_and_examples_1.pdf**, install the Node.js environment and the Express framework, and create an Express project named **NewsFeed**.

2. Following steps in **AJAX_JSON_MongoDB_setup_and_examples.pdf**, install MongoDB, run MongoDB server, and create a database "assignment1" in the database server.

3. Insert a few user documents into a **userList** collection in the database in the format as follows:

```
db.userList.insert({'name':'Amy', 'password':'123456', 'icon':'images/amy.jpg'})
```

Each user document in the **userList** collection contains the following key-value pairs:

- **_id**: The unique ID of the document, which you do not need to include when inserting a document, and MongoDB server will add automatically into each inserted document. You can find out such a _id using db.userList.find().
- **name**: The registered name of the user.
- **password**: The registered password of the user.
- **icon**: The path (which can be absolute or relative path) to the file of the icon used to represent that user (e.g., a photo of the user). For example, you can put the icon images under "public/images" in your Express app directory, and specify the correct path to each

image when inserting each user document.

4. Insert a number of news feed documents into a **newsList** collection in the database in the format as follows:

db.newsList.insert({'headline':'Jason Day signs with Bridgestone Golf', 'content':' Twelve-time PGA TOUR winner Jason Day has signed with Bridgestone Golf to use its golf ball.', 'time':new Date(), 'comments':[{ 'userID': ObjectId("507f1f77bcf86cd799439011"), 'time':ISODate("2022-03-02T22:31:55Z"), 'comment': 'Fantistic!'}]})

Each news feed document in the **newsList** collection contains the following key-value pairs:

- **_id**: The unique ID of the document, which you do not need to include when inserting a document, and MongoDB server will add automatically into each inserted document. You can find out such a _id using db.newsList.find().
- **headline**: The headline of the news.
- **time**: The time when the news was posted, which should be a Date object (https://www.w3schools.com/js/js_dates.asp)
- **content**: The textual content of the news feed.
- **comments**: An array of comments on the news feed.

Each comment entry in the **comments** array is an object, containing the following key-value pairs:

- **userID**: _id of the user posting the comment. You can find it out using db.userList.find().
- **time**: The time when the comment was posted, which should be a Date object.
- **comment**: The textual content of the comment.

Insert at least 15 news feed documents into the **newsList** collection for testing your program. You can insert some comment entries when inserting news documents for testing purpose, or you can insert comments later through the web page you have developed.

Task 1 (10 marks) Create basic newsfeed.html and app.js

1. *Client side*

Create **newsfeed.html** which renders a layout as sketched in Fig. 1, which contains 3 divisions: (i) , containing a search input textbox, a button showing "Search news headlines", and a login or logout link; (ii) , to display the news entries; (iii) , to contain page indices for navigating to other news list pages. Create the search input textbox and the search button in the "header" division. (The "Log in" or "Log out" link in the "header" division, the news entries in the "news" division, and the page indices in the "pageindex" division are to be dynamically generated in later tasks.)

Fig. 1 The news feed page

(Red dashed boxes mark the three divisions only and should not appear in the page display)

2. *Server side*

In **app.js**, add necessary code for loading the MongoDB database you have created, creating an instance of the database, and passing the database instance for usage of all middlewares.

Also load any other modules and add other middlewares which you may need to implement this application.

Add the middleware for serving static file in the "./public" directory.

We will let the server run on the port 8081 and launch the Express app using command "node app.js".

Task 2 (25 marks) Display the news list

1. *Client side*

The "Log in" or "Log out" link in the "header" division, the news entries in the "news" division, and the page indices in the "pageindex" division are dynamically generated in a JavaScript function loadNewsList(pageindex) (Note: put all your client-side scripts in **script.js**). The function should be invoked (i) when newsfeed.html is loaded for the first time, (ii) when the "Search news headlines" button is clicked, and (iii) when a page index is clicked. In case (i) and case (ii), loadNewsList(1) is called; in case (iii), loadNewsList(pageindex) is invoked, where pageindex is the page index that the user clicks.

In loadNewsList(pageindex), the search string that the user enters in the search input textbox is retrieved. If the search input box is empty, the search string should be "". An AJAX HTTP GET request is sent to the server side for "/retrievenewslist", carrying pageindex and the search string. It will obtain the following information from the server: (a) the total number of news entries in the newsList collection in the database, whose headlines contain the search string; (b) _id, headline, time and simplified content of 5 news entries (or less if the page is the last page) among the news entries whose headlines contain the search string, according to the pageindex; (c) the login status of the user.

With the responded data, it creates the HTML content on the page as follows:

(a) Display "Log in" or "Log out" in the "header" division. If the login status is 0, "Log in" is displayed; if the login status is 1, "Log out" is displayed. "Log in" should be a hyperlink linking to the URL "/login?newsID=0". "Log out" should not link to any URL, but an event handler function logout() should be registered with its "onclick" event (how to implement logout() will be described in Task 4).

(b) Create or replace the news entries in the "news" division with the received news entries. The news entries are ordered in reverse chronological order, i.e., the latest news is displayed at the top. Each news headline is a hyperlink, linking to /displayNewsEntry?newsID=xx, where xx should be the _id value of the corresponding news entry. The time can be displayed in the format of Date toLocaleString() (see https://www.w3schools.com/jsref/jsref_tolocalestring.asp).

(c) Produce the page indices in the "pageindex" division. The page indices are created according to the overall number of news entries in the newsList collection in the database whose headlines contain the search string — the number received from the server. We display 5 news entries per page. For example, if the overall number is 17, we need 4 pages — 5 entries on pages "1", "2", "3", and 2 entries on page "4". The index of the page current being displayed should be displayed in a highlight color, different from the color of the other page indices. When a page index i is clicked, loadNewsList(i) should be invoked.

You should use proper HTML elements and styling rules at your choice to implement the page display, following the layout sketch given in Fig. 1.

2. *Server side*

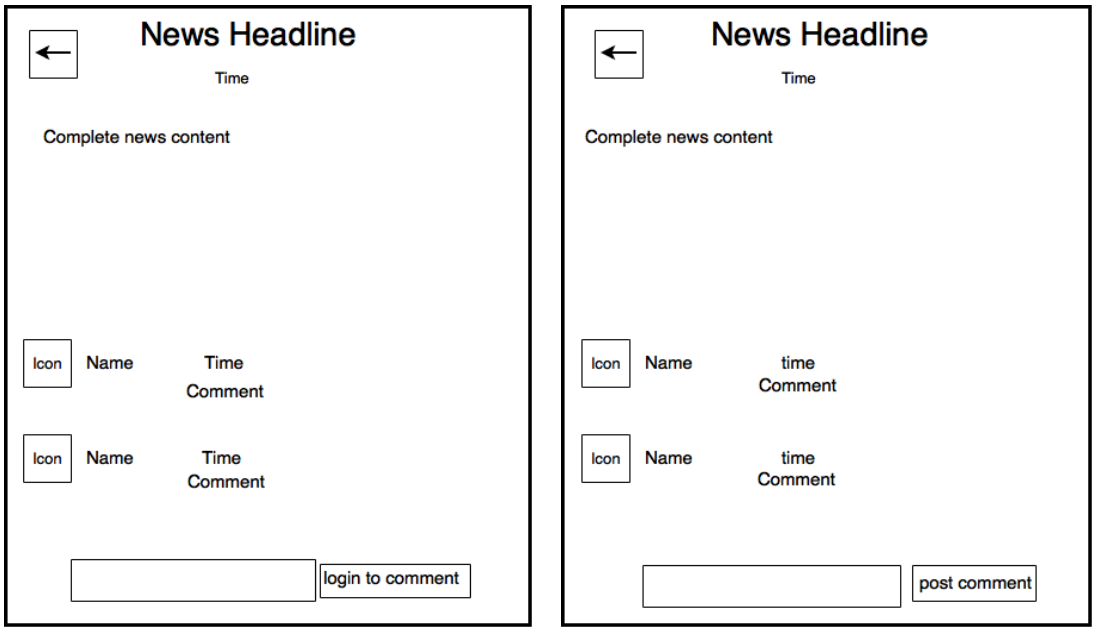
In **app.js**, create the following middleware to handle requests from the client side:

• **HTTP GET requests for http://127.0.0.1:8081/retrievenewslist**. In the middleware, find the news entries in the newsList collection in the database whose headlines contain the search string sent in the request (as substring). Identify the news entries to be returned to the client according to the pageindex and the total number of entries obtained above. In addition, check whether a "userID" cookie has been set for the user (which is set after the

user has successfully logged in, to be implemented in Task 4): if so, the login status is 1; otherwise, the login status is 0. Then return all the data (refer to the second paragraph in Task 2 "1. Client side" above on the data to be returned) to the user using a JSON string: (1) you should design the concrete format of the JSON string and parse it accordingly in loadNewsList(pageindex); (2) only the first 10 words of the content of a news entry should be sent to the user as the "simplified content", instead of the complete content.

Task 3 (30 marks) Display a news entry

After a user clicks a news headline on the news list display page shown in Fig. 1, a page as sketched in Fig. 2 will be displayed: if the user has not logged in before, he sees Fig. 2(a); if the user has successfully logged in, he sees Fig. 2 (b).



(a) (b)

Fig. 2 The news entry display page.

1. *Server side*

In **app.js**, create the following middlewares to handle requests from the client:

• **HTTP GET requests for http://127.0.0.1:8081/displayNewsEntry**. In the middleware:

(1) Retrieve the news entry from the newsList collection in the database according to the "newsID" carried in the request, including all comments posted on this news. For each comment, further retrieve the name and the icon of the user who posted the comment from the userList collection according to the "userID" of the comment document. (**Note**: to match

an ID with the _id field in a document in the database, you may need to cast the ID string to an ObjectId using monk.id(ID) first).

(2) Generate appropriate HTML contents and styling rules for the client side to display the news entry and comments according to the layout sketch in Fig. 2.

- The comments should be displayed in reverse chronological order with the newest comments at the top. The comment post time can be displayed in the format of Date toLocaleString().
- A button displaying an arrow should appear in the top-left corner of the page, which links back to newsfeed.html.
- A comment input textbox and a button besides it should appear at the bottom of the page. If the current user has not logged in (by checking if the "userID" cookie has been set), the comment input textbox should be disabled (**Hint**: use the disabled attribute https://www.w3schools.com/tags/att_input_disabled.asp), and "login to comment" should be displayed on the button; otherwise, the comment input textbox should be enabled, and "post comment" should be displayed on the button. The "login to comment" button should link to the URL "/login?newsID=xx", where xx should be the ID value of the news entry displayed on the current page. The "post comment" button does not link to a URL; instead, an event handler function postComment() is registered with its "onclick" event.

The middleware should send all the generated HTML contents back to the client. You can implement postComment() in **script.js** and link to script.js in the HTML contents sent to the client. You should also link to the CSS file (for styling the HTML contents) in the HTML contents sent to the client.

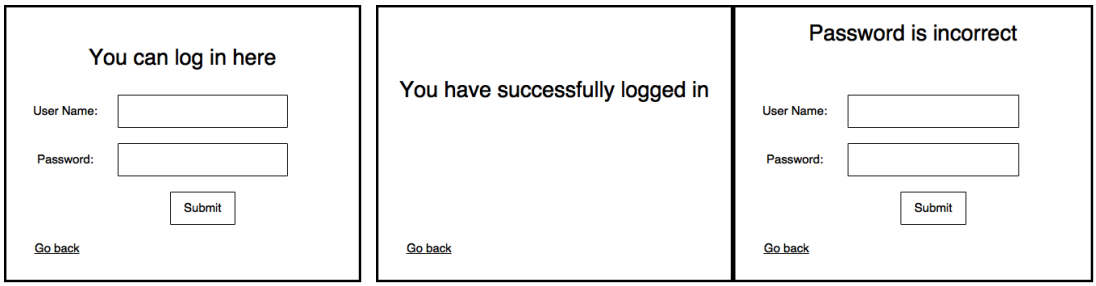
• **HTTP POST requests for http://127.0.0.1:8081/handlePostComment**. In the middleware, insert the comment posted by the user into the comments array of the respective news entry in the newsList collection in the database. Then retrieve all the latest comments on the news entry from the newsList, that have not been displayed on the user's news entry display page. For each comment, further retrieve the name and the icon of the user who posted the comment from the userList. The retrieved data should be returned to the user in a JSON string. You should design the format of the JSON string and parse it accordingly in the client-side code (aka in postComment() whose implementation is presented below).

2. *client side*

In **script.js**, implement postComment() as follows: the comment that the user has entered in the comment input textbox is obtained. If the comment input textbox is empty, an alert box will pop up, showing "No comment has been entered", and then the function exists. Otherwise, an AJAX HTTP POST request is sent for /handlePostComment, carrying the comment that the user has entered in the comment input textbox, the newsID, the current time, and the post time of the latest comment currently displayed on the news entry page. Upon receiving the new comments (that the current page has not displayed) returned by the

server, update the comments display part of the page by inserting these new comments at the top, and empty the comment input textbox.

Task 4 (25 marks) Implement login and logout



(a) (b) (c)

Fig. 3 The login page.

1. Login

When the "Log in" link on the news feed page (Fig. 1) is clicked, or when the "login to post comment" button is clicked on a news entry display page (Fig. 2), an HTTP GET request is sent for "/login".

[*Server side*] In **app.js**, implement the following middleware to handle these requests:

• **HTTP GET requests for http://127.0.0.1:8081/login?newsID=xx**. In the middleware, produce the HTML contents for the login page as shown in Fig. 3(a), which contains: (i) a login form with a username input textbox, a password input textbox, and a submit button; (ii) a heading on top of the form which initially displays a message such as "You can login here", and (ii) a hyperlink "Go back" at the bottom of the page. The "Go back" hyperlink should link to newsfeed.html if the newsID value carried in the request is 0, or link to /displayNewsEntry?newsID=xx if the newsID value carried in the request is larger than 0 (xx should be replaced by this newsID value). An event handler function login() should be registered with the "onclick" event of the submit button (login() will be implemented in script.js and you should link to script.js in the HTML contents that this middleware produces). The middleware then sends the generated HTML contents to the client.

• **HTTP GET requests for http://127.0.0.1:8081/handleLogin?username=xx&password=xx**. In the middleware, check whether the input username and password match those of any user document in the userList collection in the database. If so, set a cookie "userID" for the user, containing the _id value of the user document, and return the string "login success"; otherwise, it returns an error message such as "Username is incorrect" or "Password is incorrect".

[*Client side*] In **script.js**, implement the function login() as follows:

Retrieve the input username and password in the respective input textboxes. If either

username or password input is empty, an alert box pops up, displaying “Please enter username and password”, and the function exits. Otherwise, send an AJAX HTTP GET request for /handleLogin, carrying the input username and password. If the string returned by the server side is “login success”, replace the heading and the login form on the page by a heading “You have successfully logged in” (Fig. 3(b)); otherwise, replace the heading on the page by the error message received (e.g., Fig. 3(c)).

2. logout

[*Client side*] When “Log out” in the news feed page (Fig. 1) is clicked, the event handler function logout() is invoked. Implement logout() in **scripts.js** as follows: send an AJAX HTTP GET request for /handleLogout. When the response message is received from the server, change the “Log out” link to the “Log in” link in the “header” division of the news feed page.

[*Server side*] In **app.js**, implement the following middleware to handle the requests:

• **HTTP GET requests for http://127.0.0.1:8081/handleLogout**. In the middleware, unset the “userID” cookie for the user, and return a string “logout success”.

Task 5 (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. You can use a template engine to produce the HTML content or hardcode HTML content generation in app.js.

3. Maintain a good programming style: avoid redundant code; the code should be easy to understand and maintain.

4. You should carefully test all the functionalities stated in this handout. When testing comment posting, you can launch two browsers and log in as two users to post comments to the same news. We will check your assignment submission in a similar fashion when marking.

Submission

You should submit the following files in the specified folder structure in a zip file:

(1) app.js

(2) /public/newsfeed.html

(3) /public/javascripts/script.js

(4) /public/stylesheets/style.css

Please submit the .zip file 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.