Assignment 9

Contents

In this assignment, you will design a class-wide interactive Twitter-style news feed web app that displays a list of recent tweets and allows users to post new tweets to the feed, update or delete their tweets, retweet or reply to tweets, and like or unlike tweets.

You will not be interacting directly with the real twitter.com. Instead, we have created a RESTful back-end API server specifically for this course. Your web app should use AJAX + JSON to send requests to this special server. There are 7 RESTful endpoints (an endpoint is a URL coupled with an HTTP verb) that we expose for your app to use. These 7 endpoints are sufficient for implementing the functionality to display, manage, and like tweets. Each one of these endpoints are explained in detail below.

> **Careful! When you post new tweets, they will be visible to the entire class.** This also means that your app's news feed will show all of the tweets recently posted by your classmates.

1. Tweet resource

The main RESTful resource that your app must manipulate for a09 is a Tweet. Tweet objects have the following properties:

name	type	example	description
id	number	498	The unique identification number assigned to the Tweet by the server database
type	string	"retweet"	Specifies whether the Tweet is a regular tweet ("tweet"), a reply to another Tweet ("reply"), or a retweet of another Tweet ("retweet") Must be one of the following values: ["tweet", "retweet", "reply"]
body	string	"This is what the user said"	The body text of the Tweet
author	string	"Aaron S."	The first name and last initial of the

			user who posted the Tweet, as a string
parentId	number	263	If this Tweet is a reply or a retweet, this is the id of the referenced Tweet
parent	Tweet object		If this Tweet is a reply or a retweet and the referenced Tweet is available, then this is the referenced Tweet's data
isMine	boolean	false	True if the Tweet was created by the current, logged in user
isLiked	boolean	true	True if the Tweet is liked by the current, logged in user
retweetCount	number	0	The number of retweets for this Tweet
replyCount	number	1	The number of replies for this Tweet
likeCount	number	1	The number of likes for this Tweet
someLikes	array of strings	["Erin S."]	A non-exhaustive list of names of users who have liked this tweet
replies	array of Tweet objects		If available, this is an array of reply Tweets that were posted in response to this Tweet
createdAt	timestamp		The date and time at which the Tweet was posted
updatedAt	timestamp		The date and time at which the Tweet was last updated

2. API Documentation

This section describes the 7 RESTful endpoints (aka URLs) that your app must use (via AJAX + JSON) to interact with the COMP 426 Twitter server.

2.1. Index

```
Purpose:
        Retrieves a list of the most recent Tweets as a JSON array
Endpoint:
        GET  https://comp426fa19.cs.unc.edu/a09/tweets
Request Params:
        skip (integer) - Optional. If omitted, defaults to 0. Used for pagination.
                         Specifies the number of Tweets to skip before seleting
                         Tweets to retrieve.
        limit (integer) - Optional. If omitted, defaults to 50. Used for pagination.
                          Specifies the number of Tweets to retrieve. Must be in
                          the range [1, 75].
        sort (json) - Optional. Specifies the order of the Tweets to retrieve. Must
                      be a Tweet field name followed by "ASC" or "DESC" to denote
                      ascending or descending, respectively. If omitted, defaults
                      to [{createdAt: 'DESC'}] which selects newest Tweets first.
        where (json) - Optional. Specifies a filter to apply to the Tweets before
                       retrievial. If omitted, defaults to
                       {type: ['tweet', 'retweet']}, which selects only Tweets with
                       type "tweet" and "retweet" (not "reply" Tweets).
Response:
        Responds with an array in JSON format containing the selected Tweets.
```

Example Axios Request:

```
const result = await axios({
  method: 'get',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets',
  withCredentials: true,
});
```

> **Note:** It's important that you add `withCredentials: true` as a configuration to all of your axios requests. This tells axios that it's okay to send identifying cookies with the http request, and is required so that the server knows who you are when you interact with the a09 Twitter API.

Example Response:

```
200 OK
```


```
[{
    "id": 11,
    "type": "retweet",
    "body": "The force is strong in my family.",
    "parent": {
      "id": 4,
      "type": "tweet",
      "body": "May the force be with you.",
      "author": "Obi-Wan K.",
      "isMine": false,
      "isLiked": false,
      "retweetCount": 1,
      "replyCount": 2,
      "likeCount": 3,
      "someLikes": ["Yoda", "Rey", "Mace W."],
      "createdAt": 1565457409819,
      "updatedAt": 1565457409819
    },
    "parentId": 4,
    "author": "Luke S.",
    "isMine": false,
    "isLiked": false,
    "retweetCount": 0,
    "replyCount": 0,
    "likeCount": 0,
    "someLikes": [],
    "createdAt": 1565457495114,
    "updatedAt": 1565457495114
  }, {
    "id": 7,
    "type": "tweet",
    "body": "Now this is podracing",
    "author": "Anakin S.",
    "isMine": false,
    "isLiked": false,
    "retweetCount": 1,
    "replyCount": 2,
    "likeCount": 1,
    "someLikes": ["Mace W."],
    "createdAt": 1565457409819,
    "updatedAt": 1565457409819
}]
```


2.2. Create

```
Purpose:
        Creates a new tweet in the database
Endpoint:
        POST  https://comp426fa19.cs.unc.edu/a09/tweets
Request Params:
        body (string) - Required. The body text of the Tweet. Can be up to 280
                        characters in length.
        type (string) - Optional. Defaults to "tweet". Specifies whether the Tweet
                        is a regular Tweet, a reply to another Tweet, or a retweet
                        of another Tweet. Must be one of the following values:
                        ["tweet", "retweet", "reply"].
        parent (number) - Optional, but required if type is set to "retweet" or
                          "reply". Specifies the id of the associated Tweet.
Response:
        Upon successful creation, this route responds with the created
        Tweet's field data in JSON format.
```

Example Axios Request

```
const result = await axios({
  method: 'post',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets',
  withCredentials: true,
  data: {
    body: "Great shot, kid, that was one in a million."
  },
});
```

Example Response

```
201 Created
```


```
{
  "id": 13,
  "type": "tweet",
  "body": "Great shot, kid, that was one in a million.",
  "author": "Han S.",
  "isMine": true,
  "isLiked": false,
  "retweetCount": 0,
  "replyCount": 0,
  "likeCount": 0,
  "someLikes": [],
  "createdAt": 1565533143707,
  "updatedAt": 1565533143707
}
```


2.3. Read

```
Purpose:
        Gets details about a specific Tweet
Endpoint:
        GET  https://comp426fa19.cs.unc.edu/a09/tweets/:id
        Note: Replace ":id" in the above route with the id of the Tweet to retrieve
Request Params:
        This route does not accept request params, but the ID of the Tweet
        that should be retrieved must be specified in the URL.
Response:
```

```
This route responds with details about the Tweet's field data in
JSON format.
```

Tip: This endpoint provides the most complete information about any given Tweet by filling out the `replies` and `parent` fields. If your app has basic information about a Tweet received from the Index endpoint, but it needs more complete information about that Tweet, use the Read endpoint to fill in the `replies` and/or `parent` fields.

Example Axios Request

```
const result = await axios({
  method: 'get',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets/12',
  withCredentials: true,
});
```

Example Response


```
{
  "id": 12,
  "type": "retweet",
  "body": "This is the most recent tweet. It demonstrates a retweet of a reply.",
  "parent": {
    "id": 10,
    "type": "reply",
    "body": "This tweet is a reply to a reply, demonstrating a reply chain.",
    "parent": {
      "id": 9,
      "type": "reply",
      "body": "This tweet represents a reply to an original tweet.",
      "parent": {
        "id": 7,
        "type": "tweet",
        "body": "This was the original tweet.",
        "author": "Darth V.",
        "isMine": false,
        "isLiked": false,
        "retweetCount": 1,
        "replyCount": 2,
        "likeCount": 0,
        "someLikes": [],
        "createdAt": 1565457409819,
        "updatedAt": 1565457409819
      },
      "parentId": 7,
      "author": "Emperor P.",
      "isMine": false,
      "isLiked": false,
      "retweetCount": 0,
      "replyCount": 1,
      "likeCount": 0,
      "someLikes": [],
      "createdAt": 1565457451243,
      "updatedAt": 1565457451243
    },
    "parentId": 9,
    "author": "Kylo R.",
    "isMine": false,
    "isLiked": false,
    "retweetCount": 1,
    "replyCount": 0,
    "likeCount": 0,
```

```
    "someLikes": [],
    "createdAt": 1565457473429,
    "updatedAt": 1565457473429
  },
  "parentId": 10,
  "author": "Snoke",
  "isMine": false,
  "isLiked": false,
  "retweetCount": 0,
  "replyCount": 0,
  "likeCount": 0,
  "someLikes": [],
  "createdAt": 1565457503068,
  "updatedAt": 1565457503068
}
```

2.4. Update

```
Purpose:
        Updates a specific tweet
Endpoint:
        PUT  https://comp426fa19.cs.unc.edu/a09/tweets/:id
        Note: Replace ":id" in the above route with the id of the Tweet to update
Request Params:
        body (string) - Required. The new body text of the Tweet. Can be up to 280
                        characters in length.
Response:
        Upon successful update, this route responds with the updated
        Tweet's new data in JSON format.
```

> **Note:** You can only update Tweets that you created. The server will respond with `403 Forbidden` if you try to update someone else's Tweet.

Example Axios Request

```
const result = await axios({
  method: 'put',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets/13',
  withCredentials: true,
  data: {
    body: "Great shot, kid! That was one in a million!"
  },
});
```

> **Note:** The only field you can update is `body`.

Example Response

```
200 OK
```

```
{
  "id": 13,
  "type": "tweet",
  "body": "Great shot, kid! That was one in a million!",
  "author": "Han S.",
  "isMine": true,
  "isLiked": false,
```

```
  "retweetCount": 0,
  "replyCount": 0,
  "likeCount": 0,
  "someLikes": [],
  "createdAt": 1565533143707,
  "updatedAt": 1565533143707
}
```

2.5. Destroy

```
Purpose:
        Permanently deletes a specific Tweet
Endpoint:
        DELETE  https://comp426fa19.cs.unc.edu/a09/tweets/:id
        Note: Replace ":id" in the above route with the id of the Tweet to destroy
Request Params:
        This route does not accept request params, but the id of the Tweet
        that should be deleted must be specified in the URL.
Response:
        Upon successful deletion, this route responds with an empty response body.
```

> **Note:** You can only destroy Tweets that you created. The server will respond with `403 Forbidden` if you try to destroy someone else's Tweet.

Example Axios Request

```
const result = await axios({
  method: 'delete',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets/13',
  withCredentials: true,
});
```

Example Response

```
204 No Content
```

2.6. Like

```
Purpose:
        Likes a specific Tweet
Endpoint:
        PUT  https://comp426fa19.cs.unc.edu/a09/tweets/:id/like
        Note: Replace ":id" in the above route with the id of the Tweet to like
Request Params:
        This route does not accept request params, but the ID of the Tweet
        that should be liked must be specified in the URL.
Response:
        Upon successful unlike, this route responds with an empty response body.
```

Example Axios Request

```
const result = await axios({
  method: 'put',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets/11/like',
```

```
  withCredentials: true,
});
```

Example Response

```
204 No Content
```

2.7. Unlike

```
Purpose:
        Unlikes a specific Tweet
Endpoint:
        PUT  https://comp426fa19.cs.unc.edu/a09/tweets/:id/unlike
        Note: Replace ":id" in the above route with the id of the Tweet to unlike
Request Params:
        This route does not accept request params, but the ID of the Tweet
        that should be unliked must be specified in the URL.
Response:
        Upon successful unlike, this route responds with an empty response body.
```

Example Axios Request

```
const result = await axios({
  method: 'put',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets/13/unlike',
  withCredentials: true,
});
```

Example Response

```
204 No Content
```

3. Authentication

To use the COMP 426 Twitter API, your browser must first be logged in to the COMP 426 server. That's how the system knows which user is creating/updating/liking/deleting Tweets. Luckily, a login script is included in the a09 directory. **You must use this login script in the same browser window before switching to your app.** Here's how to use it:

1. Start Browsersync in the `/a09` project directory
2. In your browser, navigate to localhost:3000/login
3. Enter your COMP 426 username and password
4. Press the login button

> If your app isn't working because you keep getting `401 Unauthorized` HTTP status code responses, it's likely that you are not logged in.

Note: It's important that you add `withCredentials: true` as a configuration to all of your axios requests. This tells axios that it's okay to send identifying cookies with the http request, and is required so that the server knows who you are when you interact with the a09 Twitter API.

Note: It's okay for your app to assume that the user had already logged in before they navigated to your index.html page.

4. Retweets and Replies

When a Tweet is first created via the Create endpoint (see Section 2.2), it might be marked on the server as a "retweet" or a "reply." To accomplish this, your app must provide the server with the id of an existing Tweet to which the new Tweet will refer. This process uses the special `type` and `parent` fields on the Tweet resource as explained below.

4.1. Replying to a tweet

To reply to a Tweet, your app must first know the id of the original Tweet. Then, make a Create request and specify the optional `type` and `parent` properties. For example, suppose the original Tweet has an id of 11. The request to create a reply might be formulated like this:

Example Axios Request

```
const result = await axios({
  method: 'post',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets',
  withCredentials: true,
  data: {
    "type": "reply",
    "parent": 11,
    "body": "Great shot, kid! That was one in a million!"
  },
});
```

Example Response

```
201 Created
```

```
{
  "id": 13,
  "type": "reply",
  "body": "Great shot, kid! That was one in a million!",
  "parent": {
    "id": 11,
    "type": "tweet",
    "body": "The force is strong in my family.",
    "author": "Luke S.",
    "isMine": false,
    "isLiked": false,
    "retweetCount": 0,
    "replyCount": 0,
    "likeCount": 0,
    "someLikes": [],
    "createdAt": 1565457495114,
    "updatedAt": 1565457495114
  },
  "parentId": 11,
  "author": "Han S.",
  "isMine": true,
  "isLiked": false,
  "retweetCount": 0,
  "replyCount": 0,
  "likeCount": 0,
  "someLikes": [],
```

```
  "createdAt": 1565533143707,
  "updatedAt": 1565533432346
}
```

4.2. Retweeting a tweet

To retweet a Tweet, your app must first know the id of the original Tweet. Then, make a Create request and specify the optional `type` and `parent` properties. For example, suppose the original Tweet has an id of 11. The request to retweet might be formulated like this:

Example Axios Request

```
const result = await axios({
  method: 'post',
  url: 'https://comp426fa19.cs.unc.edu/a09/tweets',
  withCredentials: true,
  data: {
    "type": "retweet",
    "parent": 11,
    "body": "My father had it, I have it, and my twin sister has it."
  },
});
```

> **Note:** When you retweet an existing Tweet, you have the option of adding more text to your retweet in the `body` field. This allows users to retweet an existing tweet while appending their own content to the original.

Example Response

```
201 Created
```

```
{
  "id": 19,
  "type": "retweet",
  "body": "My father had it, I have it, and my twin sister has it.",
  "parent": {
    "id": 11,
    "type": "tweet",
    "body": "The force is strong in my family.",
    "author": "Luke S.",
    "isMine": true,
    "isLiked": false,
    "retweetCount": 0,
    "replyCount": 0,
    "likeCount": 0,
    "someLikes": [],
    "createdAt": 1565457495114,
    "updatedAt": 1565457495114
  },
  "parentId": 11,
  "author": "Luke S.",
  "isMine": true,
  "isLiked": false,
  "retweetCount": 0,
  "replyCount": 0,
  "likeCount": 0,
  "someLikes": [],
  "createdAt": 1565533143707,
```

```
  "updatedAt": 1565533432346
}
```

5. Requirements

At minimum, your app must allow users to:

- Retrieve and display a "news feed" containing the 50 most recent tweets. Use the Index route (see Section 2.1) to retrieve the news feed data from the server.
- For each tweet in the news feed, display:
    - The name of the user who posted the tweet
    - The tweet body text
    - The number of likes
    - The number of retweets
    - Whether the current user has liked the tweet
    - A button that allows the user to toggle whether they **like the tweet** (see Sections 2.6 and 2.7)
    - A button that allows the user to **reply to the tweet** (see Section 4.2)
    - A button that allows the user to **retweet the tweet** (see Section 4.1)
- For tweets that were posted by the logged-in user, include a button that allows the user to edit their tweet. When the user clicks this button, create and display an edit form with a `<textarea>` field that is prepopulated with the tweet's current body text. After the user finishes making changes, use the Update route (see Section 2.4) to save the new tweet body text to the server.
- For tweets that were posted by the logged-in user, include a button that allows the user to delete their tweet. When the user clicks this button, use the Destroy endpoint (see Section 2.5) to save the user's action to the server.
- Provide a button that allows the user to post a new tweet. When the user clicks this button, create and display an edit form with a `<textarea>` field to receive the tweet body text. After the user finishes composing the tweet, use the Create endpoint (see Section 2.2) to save the new tweet on the server.

Extra Credit Opportunities

- Ability to selectively view a tweet's replies
- Particularly pleasing user interfaces and/or designs
- Support for infinite scrolling

Note: A maximum of 10 points extra credit will be given total for these opportunities. *Extra credit applies to your overall assignment grade only*.