Users microservice

Each user has a username, an email address, a hashed password, and a list of users that the user is following.

The following API operations should be exposed:

- `createUser(username, email, password)`

  Registers a new user account.

Endpoint: http://127.0.0.1:5000/register?info=data

Method: POST

Status Codes: HTTP 200 on success, HTTP 4xx for Client error, HTTP 5xx for server error

Input JSON: `{'data': {username: "foo", email: "bar", password: "pass"}}`

Output JSON:`{"message": "{username} was added successfully"}`

- `authenticateUser(username, password)`

  Returns `true` if the supplied password matches the hashed password stored for that username in the database.

Endpoint: http://127.0.0.1:5000/login?user=username,password=

Method: POST

Status Codes: HTTP 200 on success, HTTP 4xx Client error, HTTP 5xx Server error, 401 Incorrect

Input JSON: `{'data': {username: "foo", password: "pass"}}`

Output JSON:

`{"message": "{username} was authenticated successfully"}`

`{"message": "{username} password incorrect"}`

- `addFollower(username, usernameToFollow)`

  Start following a new user.

Endpoint: http://127.0.0.1:5000/follow? username=, usernameToFollow=

Method: PUT

Status Codes: HTTP 200 on success, HTTP 4xx Client error, HTTP 5xx Server error

input JSON: `{'data': {username: "foo", usernameToFollow: "pass"}}`

Output JSON: {**Output JSON:**

`{"message": "{username} is now following {usernameToFollow}"}`

- `removeFollower(username, usernameToRemove)`

  Stop following a user.

Endpoint:http://127.0.0.1:5000//unfollow?username=, usernameToRemove=

Method: DELETE

Status Codes: HTTP 200 on success, HTTP 4xx Client error, HTTP 5xx Server error

input JSON: `{'data': {username: "foo", usernameToRemove: "pass"}}`

Output JSON: {**Output JSON:**

`{"message": "{username} is now unfollowing {usernameToFollow}"}`

Timelines microservice

Each post to a timeline should have the author's username, the text of the post, and a timestamp showing when the post was created.

Timelines should be returned in reverse chronological order. Limit the maximum number of posts retrieved for any timeline to 25.

The following API operations should be exposed:

- `getUserTimeline(username)`

  Returns recent tweets from a user.

Endpoint: http://127.0.0.1:5000/getUserTimeline?username=

Method: GET

Status Codes: HTTP 201 on success, HTTP 4xx for Client error, HTTP 5xx for server error

Input JSON: `{'data': {username: "foo"}}`

Output JSON:

```
{"tweet": message
"day_of": date
"fk_users": username
}
```

- `getPublicTimeline()`

  Returns recent tweets from all users.

Endpoint: http://127.0.0.1:5000/getPublicTimeline

Method: GET

Status Codes: HTTP 200 on success, HTTP 4xx for Client error, HTTP 5xx for server error

Output JSON:

```
{"tweet": message
"day_of": date
"fk_users": username
}
```

- `getHomeTimeline(username)`

  Returns recent tweets from all users that this user follows.

Endpoint: http://127.0.0.1:5000/getHomeTimeline?username=

Method: GET

Status Codes: HTTP 200 on success, HTTP 4xx for Client error, HTTP 5xx for server error

Input JSON: `{'data': {username: "foo"}}`

Output JSON:

```
{"tweet": message
"day_of": date
"fk_users": username
}
```

- `postTweet(username, text)`

  Post a new tweet.

Endpoint: http://127.0.0.1:5000/postTweet?username=, tweet=

Method: POST

Status Codes: HTTP 200 on success, HTTP 4xx for Client error, HTTP 5xx for server error

Input JSON: `{'data': {username: "foo", tweet: "bar"}}`

Output JSON:

`{"message": "{username}: {tweet}, posted"}`