Assigned: 06 September

Homework #2 – Node.js

EE 547: Fall 2022

Due: Tuesday, 16 September at 23:59. Late penalty: 10% per 24-hours before 18 September at 23:59. Submission instructions will follow separately on canvas.

1. **Simple Node.js webserver** Use the Node.js `http` package to create a simple webserver. Use only standard (*i.e.,* built-in) packages. Listen on port 8088. Your server must respond to `GET` requests for the following four paths. Your server should respond to all other `GET` requests with http status code 404.

   - `GET /ping`

     Respond with status code 204 and empty body.

   - `GET /anagram?p=[string]`

     Read string p from the query. Compute the number of distinct anagrams for p as in Homework 1. Respond with status code 400 and empty body if the string is invalid or empty. Respond with http status code 200 and write the following object literal to the response body as a JSON string (`JSON.stringify()`):

     ```
     {
       "p":     "[p string]",
       "total": "[number of distinct anagrams]"     # NOTE: string type
     }
     ```

   - `GET /secret`

     Respond with http status code 200 if the file /tmp/secret.key exists and write the content of the file /tmp/secret.key to the response body. If the file does not exist respond with http status code 404.

   - `GET /status`

     Respond with http status code 200. Write the following dictionary to the response body as a JSON string:

     ```
     {
       "time": "[UTC TIME in ISO-8601]",     # YYYY-MM-DDTHH:mm:ssZ
       "req":  [number of requests received since server start],
       "err":  [number of 404 errors, c.f., /secret and invalid path]
     }
     ```

2. **Object oriented API proxy**

A. Setup

Create a Twitter developer account.

1. Create a Twitter account https://twitter.com/i/flow/signup. You may use a Twitter account that you already own or create a new one. You can skip providing personal information if you choose.

2. Enable developer access. Sign-up at https://developer.twitter.com/en. This step requires a verified email and phone number. You can set these in your user Settings and Privacy, https://twitter.com/settings/account.

3. Create a new App through the developer Dashboard https://developer.twitter.com/en/portal/dashboard. Name the app `EE547_HW`. Save the generated keys in a secure location. You only need the Bearer Token for this assignment.

B. Setup

Implement class to perform Twitter API REST requests using the axios library. Use the following interface.

```
const axios = require('axios')
const { EntityNotFoundError } = require('./error');

// https://developer.twitter.com/en/docs/api-reference-index
class TwitterApi {
  constructor(bearerToken) {
    // complete me
  }

  getTweet(tweetId, callback) {
    // /twitter-api/tweets/lookup/api-reference/get-tweets-id
  }

  getTimeline(userId, callback) {
    // /twitter-api/tweets/timelines/api-reference/get-users-id-tweets
  }

  recentSearch(query, callback) {
    // /twitter-api/tweets/search/api-reference/get-tweets-search-recent
  }

  retweetBy(tweetId, callback) {
    // /twitter-api/tweets/retweets/api-reference/get-tweets-id-retweeted_by
  }
```

```
  getUser(userId, callback) {
    // /twitter-api/users/lookup/api-reference/get-users-id
  }

  getUserByUsername(userName, callback) {
    // /twitter-api/users/lookup/api-reference/get-users-by-username-username
  }

  getTimelineByUsername(userName, callback) {
    // complete me
  }
}

exports.TwitterApi = TwitterApi;
```

C. Object Syntax

Use object literals and arrays of object literals for all (non-empty) data responses. Use the following format for entity objects.

Tweet

```
{
  body:           string
  createdAt:      string  ISO-8601
  publicMetrics   TweetPublicMetrics
  tweetId:        string  LongInt
  userId:         string  LongInt
}
```

TweetPublicMetrics

```
{
  retweetCount:   int
  replyCount:     int
  likeCount       int
}
```

User

```
{
  createdAt:      string  ISO-8601
  description:    string
  location:       string
  name:           string
  publicMetrics   TweetPublicMetrics
  userId:         string  LongInt
  userName:       string
  verified:       boolean
}
```

`UserPublicMetrics`

```
{
  followersCount: int
  followingCount: int
  tweetCount      int
}
```

D. EntityNotFoundError

Raise an EntityNotFoundError for any request that yields a "Not Found Error" or returns with status 404. The following code describes the imported EntityNotFoundError error object. Your submission should require this from the `./error` file. Do not implement the class directly.

```
class EntityNotFoundError extends Error { };
```