API Document

Your `app.js` web service will provide different data based upon the route. The possible endpoints are described below:

Endpoint 1: Get all yip data or yip data matching a given search term

Request Format: `/yipper/yips`

Query Parameters: `search` (optional)

Request Type (both requests): `GET`

Returned Data Format: JSON

Description 1: If the `search` parameter is not included in the request, your service should get the `id`, `name`, `yip`, `hashtag`, `likes` and `date` from the `yips` table and outputs JSON containing the information in the order of `date`s in descending order (Pass the `date` into the `DATETIME()` function in your ordering statement).

Example Request 1: `/yipper/yips`

Example Output 1: (abbreviated)

```
{
  "yips":[
    {
      "id": 25,
      "name": "Mister Fluffers",
      "yip": "It is sooooo fluffy I am gonna die",
      "hashtag": "fluff",
      "likes": 6,
      "date": "2020-07-07 03:48:28"
    },
    {
      "id": 24,
      "name": "Sir Barks a Lot",
      "yip": "Imagine if my name was sir barks a lot and I was meowing all day haha",
      "hashtag": "clown",
      "likes": 6,
      "date": "2020-07-06 00:55:08"
    },
    ...
  ]
}
```

Description 2:

If the `search` parameter is included in the request, your service should respond with all the `id`s of the `yip`s matching the term passed in the `search` query parameter (ordered by the `id`s). A "match" would be any `yip` that has the `search` term in *any* position meaning that the term "if" should match any `yip` containing the words "if", "iframe" or "sniff" (as an example, not exhaustive, more matches are possible). Your search should *not* look in `hashtag`s.

Example Request 2: /yipper/yips?search=if

Example Output 2:

```
{
  "yips" : [
    {
      "id": 8
    },
    {
      "id": 24
    }
  ]
}
```

Endpoint 2: Get yip data for a designated user

Request Format: `/yipper/user/:user`

Query Parameters: none.

Request Type: `GET`

Returned Data Format: JSON

Description: Your service should get the `name`, `yip`, `hashtag` and `date` for all the yips for a designated `user` ordered by the `date` in descending order (Pass the `date` into the `DATETIME()` function in your ordering statement). The `user` should be taken exactly as passed in the request.

Example Request: /yipper/user/Chewbarka

Example Output:

```
[
  {
    "name": "Chewbarka",
    "yip": "chewy or soft cookies. I chew them all",
    "hashtag": "largebrain",
    "date": "2020-07-09 22:26:38",
  },
  {
    "name": "Chewbarka",
    "yip": "Every snack you make every meal you bake every bite you take... I will be watc
hing you.",
    "hashtag": "foodie",
    "date": "2019-06-28 23:22:21"
  }
]
```

Endpoint 3: Update the likes for a designated yip

Request Format: `/yipper/likes` **Body Parameters:** `id` **Request Type:** `POST` **Returned Data Format:** plain text **Description:** Your service should update the `likes` for a yip (the yip your service is updating is determined by the `id` passed through the body) by incrementing the current value by 1 and responding with the new value. **Example Request:** `/yipper/likes` **Example Output:**

```
8
```

Endpoint 4: Add a new yip

Request Format: `/yipper/new`

Body Parameters: `name` and `full`

Request Type: `POST`

Returned Data Format: JSON

Description: Your service should add the new Yip information to the database and send back and output the JSON with the `id`, `name`, `yip`, `hashtag`, `likes` and `date`. The `id` should correspond with the auto-incremented `id` generated from inserting into the

database. In your Node app, the newly generated `id` can be retrieved by using the `lastID` property on the result of the query. The `name` of the user added to the database should be grabbed from the `name` body parameter. The `likes` should be set to 0, and the `yip` and `hashtag` information can be obtained from the `full` body parameter. The `date` should be the current date (the `yips` table schema will default to the current datetime upon a new inserted row).

Example Request: `/yipper/new`

Example Output:

```
{
  "id": 528,
  "name": "Chewbarka",
  "yip": "love to yip allllll day long",
  "hashtag": "coolkids",
  "likes": 0,
  "date": "2020-09-09 18:16:18"
```

Yipper Database Overview

You will be required to store all the data required for this site in the database. The file representing the database is provided to you and contains one table named `yips`. The table contains 6 columns, described below:

- `id`: the primary key, auto-increments
- `name`: represents the name of the user
- `yip`: represents the text of the "yip" without the hashtag *and* stripped of any extraneous whitespace.
- `hashtag`: represents the text of the hashtag (omitting the `#` in the entry)
- `likes`: represents the number of likes the yip has received
- `date`: represents the day and time at which the yip was made. Will default to the current datetime.

You are free and encouraged to experiment with your site/add entries/like Yips as a means of testing out your Yipper website functionality. The `yipper.db` file *can* be turned in with additional entries beyond what was provided to you.