Homework 7 - Using Decentralised Storage

Recap

Decentralised storage providers allow participants to store data online without fear of location-based restrictions.

Data stored in this manner is uniquely identified based on the contents to avoid clashing. This also acts as a retrieval mechanism to ensure you're getting the promised information.

Data stored in free decentralised storage should be treated as volatile as it may not be stored for extended periods of time, especially if it's not being accessed.

Immutable storage is available from third-party providers or by 'pinning' data.

Interact with a decentralised storage provider

1. Use one of the storage providers to store data online.
    1. https://ipfs.io (https://ipfs.io)
    2. https://nft.storage/ (https://nft.storage/)
    3. https://www.storj.io/ (https://www.storj.io/) (Go library)
    4. Any other…
2. Allow others to retrieve data from location.
3. Create a front-end application to programmatically interact with storage provider.
    - Store data and receive location.
    - Retrieve data from any given location.
    - Front-end can be written in any chosen framework however **code samples use IPFS, NodeJS and ExpressJS**.
4. How would you try to delete data from IPFS?

Code snippets

Store string to IPFS

```
// Dependency
const IPFS = require('ipfs');

(async () => {
    // Initialise IPFS node
    const node = await IPFS.create();

    // Set some data to a variable
    const data = 'Hello, <YOUR NAME HERE>';

    // Submit data to the network
    const cid = await node.add(data);

    // Log CID to console
    console.log(cid.path);
})();
```

Retrieve string from IPFS

```
// Dependencies
const IPFS = require('ipfs');
const all = require('it-all');

(async () => {
    // Initialise IPFS node
    const node = await IPFS.create();

    // Store CID in a variable
    const cid = 'QmPChd2hVbrJ6bfo3WBcTW4iZnpHm8TEzWkLHmLpXhF68A';

    // Retrieve data from CID
    const data = Buffer.concat(await all(node.cat(cid)));

    // Print data to console
    console.log(data.toString());
})();
```

> Gateway for verifying CID using above example: https://ipfs.io/ipfs/QmPChd2hVbrJ6bfo3WBcTW4iZnpHm8TEzWkLHmLpXhF68A (https://ipfs.io/ipfs/QmPChd2hVbrJ6bfo3WBcTW4iZnpHm8TEzWkLHmLpXhF68A)

Resources

https://github.com/ipfs/js-ipfs/blob/master/docs/core-api/FILES.md (https://github.com/ipfs/js-ipfs/blob/master/docs/core-api/FILES.md)

https://expressjs.com/en/starter/generator.html (https://expressjs.com/en/starter/generator.html)

Helpful hints below...

Store file to IPFS

Client-side:

```
const reader = new FileReader();
reader.onloadend = function () {
    const buf = buffer.Buffer.from(reader.result);
    const route = 'addFile';
    const req = { data: buf };
    ...
    ...
}
const file = document.getElementById("file");
reader.readAsArrayBuffer(file.files[0]);
```

For IPFS add:

```
let buf = Buffer.from(obj);
```

To show retrieved data on screen

```
function toBase64(arr) {
    arr = new Uint8Array(arr)
    return btoa(
        arr.reduce((data, byte) => data + String.fromCharCode(byte), '')
        );
    }
$('#ipfs-image').attr('src', `${toBase64(response[0]
```

Model answer CID

QmVGLCTbLkkbNtyymoGhEmYtfZpGWjjJeScAm8GcCCTEeo