API Specification And Help

2018-10-01

Revision 1.0

Receipt Scanner App

Noah Cowie

login(username:*string*, password:*string*)

Returns: either redirect (307) to login.html if unsuccessful or to app.html if successful, set cookie (receipt-user-token) on successful return.	`Method: GET (form encoded)`

. To implement this correctly, use the following resources:

`https://stackoverflow.com/questions/11556958/` note: you will use a GET method, not a POST, unlike this example.

`https://www.tutorialspoint.com/flask/flask_cookies.htm`

For now, set the cooke `receipt-user-token` to `xxxxxxxx`.

`https://www.w3schools.com/html/html_forms.asp`

Frontend: Christian. Backend: Jules

Jules: make a simple HTML document if you need something to test against. Just serve it using the following example:

`https://stackoverflow.com/questions/20646822/`

submitphoto(time:*unix timestamp*, photo:*file/photo*)

Returns: either 200 good on successful upload, 400 forbidden on bad request, or 401 unauthorized if the cookie is not set	`Method: POST (form encoded)`

Resources:

`http://flask.pocoo.org/docs/0.12/patterns/fileuploads/`

Note that we do not want to save them to disk, and so do not need most of this tutorial. Instead, just log the name of the file, to demonstrate you received it.

`https://www.tutorialspoint.com/flask/flask_cookies.htm`

Make sure the cookie `receipt-user-token` is set! for now, ignore the contents. Note that to ensure this works correctly, you can click on the 'i' in the

URL bar in FireFox, then choose 'clear cookies and browsing data'.
`https://www.w3schools.com/html/html_forms.asp`
Frontend: Unassigned. Backend: Jules

getreqs(*none*)

Returns: a comma-separated list of pairs of recommendations or 401 if bad cookie. See description.	`Method: GET`

`https://www.tutorialspoint.com/flask/flask_cookies.htm`
This must read the cookie `receipt-user-token` (ignore the contents for now) and if it is not present, send a 401.
The response body should be text and look like this:

```
more, greenbeans
less, soda
less, pizza
more, apple juice
same, peaches
same, pears
```

Each line should have one of {more, less, same} followed by a comma and a space, then the food item that is being suggested (or not) and a newline.
There should be fewer than 10 and more than 1 suggestion.
The client can interpret these messages to produce bubbly user-facing messages.
For now, just return the example list. Returning text is done by just doing `return "hello world!";` or by making a variable and building your string in it, then returning the variable. I recommend the second one.
Frontend: Unassigned. Backend: Jules

Helpful Information

A couple resources for all endpoints:

1. To print a console message from inside Flask, use `app.logger.warn("message here"`

2. To return an error code of some description, use `abort(401);` (for error 401)

3. To return a redirect, use `return redirect("/url.here.html");`

4. To run your code (it doesn't work until it's running correctly!): `http://flask.pocoo.org/docs/0.12/quickstart/`

A couple general notes:

1. You will need to test your code on a machine with Flask. All of the machines in the CS lab have it. Make sure you are comfortable with running Flask.

2. Code is only done when (1) it runs, (2) you have tested it and (3) it is pushed to either the local server or the github repository.

3. Good commit messages start with a specific, active verb, name the WHY, and are less than 50 characters.
   NEVER: `Changed some things. Edited file helloworld.html. I love long commits.`
   BETTER: `Fix bug causing random login failure`
   ALSO GOOD: `Make login page buttons less blocky`
   Further reading: `https://chris.beams.io/posts/git-commit/`

4. Commit often. One change per commit, not `finished scrum 1`.

5. Always `git pull` before starting work – make sure you're up-to-date!

6. Confused by git? `http://rogerdudler.github.io/git-guide/`

7. Google problems! If you use code from an example, make sure to cite it using a comment (Python comments start with `#`).