



0x02. Session authentication

Back-end**Authentication**

Weight: 1

 Project will start Apr 17, 2024 6:00 AM, must end by Apr 19, 2024 6:00 AM Checker was released at Apr 17, 2024 6:00 PM

An auto review will be launched at the deadline

Background Context

In this project, you will implement a **Session Authentication**. You are not allowed to install any other module.

In the industry, you should **not** implement your own Session authentication system and use a module or framework that doing it for you (like in Python-Flask: Flask-HTTPAuth (/rltoken/_ZTQTaMKjx1S_xATshexkA)). Here, for the learning purpose, we will walk through each step of this mechanism to understand it by doing.

Resources

Read or watch:

- REST API Authentication Mechanisms - Only the session auth part (/rltoken/oofk0VhuS0ZFZTNTVrQeaQ)
- HTTP Cookie (/rltoken/peLV8xuJ4PDJMOVfQk-d2g)
- Flask (/rltoken/AI1tFR5XriGfR8Tz7YTYQA)
- Flask Cookie (/rltoken/QYfi5oW6OHUmHDzwKV1Qsw)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/uWXp4VcY3Dd9UzTtc9N5_A), **without the help of Google**:



General

- What authentication means
- What session authentication means
- What Cookies are
- How to send Cookies
- How to parse Cookies

Requirements

Python Scripts

- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using `python3` (version 3.7)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/env python3`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `pycodestyle` style (version 2.5)
- All your files must be executable
- The length of your files will be tested using `wc`
- All your modules should have a documentation (`python3 -c 'print(__import__("my_module").__doc__)'`)
- All your classes should have a documentation (`python3 -c 'print(__import__("my_module").MyClass.__doc__)'`)
- All your functions (inside and outside a class) should have a documentation (`python3 -c 'print(__import__("my_module").my_function.__doc__)'` and `python3 -c 'print(__import__("my_module").MyClass.my_function.__doc__)'`)
- A documentation is not a simple word, it's a real sentence explaining what's the purpose of the module, class or method (the length of it will be verified)

Tasks

0. Et moi et moi et moi!

mandatory

Copy all your work of the **0x06. Basic authentication** project in this new folder.

In this version, you implemented a **Basic authentication** for giving you access to all User endpoints:

- `GET /api/v1/users`
- `POST /api/v1/users`
- `GET /api/v1/users/<user_id>`
- `PUT /api/v1/users/<user_id>`
- `DELETE /api/v1/users/<user_id>`



Now, you will add a new endpoint: `GET /users/me` to retrieve the authenticated User object.

- Copy folders `models` and `api` from the previous project `0x06. Basic authentication`

- Please make sure all mandatory tasks of this previous project are done at 100% because this project (/) (and the rest of this track) will be based on it.
- Update `@app.before_request` in `api/v1/app.py`:
 - Assign the result of `auth.current_user(request)` to `request.current_user`
- Update method for the route `GET /api/v1/users/<user_id>` in `api/v1/views/users.py`:
 - If `<user_id>` is equal to `me` and `request.current_user` is `None`: `abort(404)`
 - If `<user_id>` is equal to `me` and `request.current_user` is not `None`: return the authenticated user in a JSON response (like a normal case of `GET /api/v1/users/<user_id>` where `<user_id>` is a valid user ID)
 - Otherwise, keep the same behavior

In the first terminal:

```
bob@dylan:~$ cat main_0.py
#!/usr/bin/env python3
""" Main 0
"""

import base64
from api.v1.auth.basic_auth import BasicAuth
from models.user import User

""" Create a user test """
user_email = "bob@hbtn.io"
user_clear_pwd = "H0lbertonSchool98!"

user = User()
user.email = user_email
user.password = user_clear_pwd
print("New user: {}".format(user.id))
user.save()

basic_clear = "{}:{}".format(user_email, user_clear_pwd)
print("Basic Base64: {}".format(base64.b64encode(basic_clear.encode('utf-8')).decode("utf-8"))))

bob@dylan:~$
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=basic_auth ./main_0.py
New user: 9375973a-68c7-46aa-b135-29f79e837495
Basic Base64: Ym9iQGhidG4uaW86SDBsYmVydG9uU2Nob29sOTgh
bob@dylan:~$
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=basic_auth python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
```

In a second terminal:



```

bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status"
{
  "status": "OK"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users"
{
  "error": "Unauthorized"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users" -H "Authorization: Basic Ym9iQGhidG4uaW86SDBsYmVydG9uU2Nob29sOTgh"
[
  {
    "created_at": "2017-09-25 01:55:17",
    "email": "bob@hbbtn.io",
    "first_name": null,
    "id": "9375973a-68c7-46aa-b135-29f79e837495",
    "last_name": null,
    "updated_at": "2017-09-25 01:55:17"
  }
]
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" -H "Authorization: Basic Ym9iQGhidG4uaW86SDBsYmVydG9uU2Nob29sOTgh"
{
  "created_at": "2017-09-25 01:55:17",
  "email": "bob@hbbtn.io",
  "first_name": null,
  "id": "9375973a-68c7-46aa-b135-29f79e837495",
  "last_name": null,
  "updated_at": "2017-09-25 01:55:17"
}
bob@dylan:~$

```

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session_authentication
- File: api/v1/app.py, api/v1/views/users.py

☒ Done!

1. Empty session

☒ mandatory

Create a class `SessionAuth` that inherits from `Auth`. For the moment this class will be empty. It's the first step for creating a new authentication mechanism:

- validate if everything inherits correctly without any overloading

- validate the "switch" by using environment variables

(/)

Update `api/v1/app.py` for using `SessionAuth` instance for the variable `auth` depending of the value of the environment variable `AUTH_TYPE`, If `AUTH_TYPE` is equal to `session_auth`:

- import `SessionAuth` from `api.v1.auth.session_auth`
- create an instance of `SessionAuth` and assign it to the variable `auth`

Otherwise, keep the previous mechanism.

In the first terminal:

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
```

In a second terminal:


```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status"
{
  "status": "OK"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status/"
{
  "status": "OK"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users"
{
  "error": "Unauthorized"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users" -H "Authorization: Test"
{
  "error": "Forbidden"
}
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-user-data`
- Directory: `0x02-Session_authentication`
- File: `api/v1/auth/session_auth.py`, `api/v1/app.py`

☒ Done!

Check your code

 Get a sandbox



2. Create a session

mandatory

Update `SessionAuth` class:

- Create a class attribute `user_id_by_session_id` initialized by an empty dictionary
- (/).• Create an instance method `def create_session(self, user_id: str = None) -> str:` that creates a Session ID for a `user_id`:
 - Return `None` if `user_id` is `None`
 - Return `None` if `user_id` is not a string
 - Otherwise:
 - Generate a Session ID using `uuid` module and `uuid4()` like `id` in `Base`
 - Use this Session ID as key of the dictionary `user_id_by_session_id` - the value for this key must be `user_id`
 - Return the Session ID
 - The same `user_id` can have multiple Session ID - indeed, the `user_id` is the value in the dictionary `user_id_by_session_id`

Now you an "in-memory" Session ID storing. You will be able to retrieve an `user_id` based on a Session ID.



```

bob@dylan:~$ cat main_1.py
#!/usr/bin/env python3

""" Main 1
"""

from api.v1.auth.session_auth import SessionAuth

sa = SessionAuth()

print("{}: {}".format(type(sa.user_id_by_session_id), sa.user_id_by_session_id))

user_id = None
session = sa.create_session(user_id)
print("{} => {}: {}".format(user_id, session, sa.user_id_by_session_id))

user_id = 89
session = sa.create_session(user_id)
print("{} => {}: {}".format(user_id, session, sa.user_id_by_session_id))

user_id = "abcde"
session = sa.create_session(user_id)
print("{} => {}: {}".format(user_id, session, sa.user_id_by_session_id))

user_id = "fghij"
session = sa.create_session(user_id)
print("{} => {}: {}".format(user_id, session, sa.user_id_by_session_id))

user_id = "abcde"
session = sa.create_session(user_id)
print("{} => {}: {}".format(user_id, session, sa.user_id_by_session_id))

bob@dylan:~$
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth ./main_1.py
<class 'dict'>: {}
None => None: {}
89 => None: {}
abcde => 61997a1b-3f8a-4b0f-87f6-19d5cafee63f: {'61997a1b-3f8a-4b0f-87f6-19d5cafee63f': 'abcde'}
fghij => 69e45c25-ec89-4563-86ab-bc192dcc3b4f: {'61997a1b-3f8a-4b0f-87f6-19d5cafee63f': 'abcde', '69e45c25-ec89-4563-86ab-bc192dcc3b4f': 'fghij'}
abcde => 02079cb4-6847-48aa-924e-0514d82a43f4: {'61997a1b-3f8a-4b0f-87f6-19d5cafee63f': 'abcde', '02079cb4-6847-48aa-924e-0514d82a43f4': 'abcde', '69e45c25-ec89-4563-86ab-bc192dcc3b4f': 'fghij'}
bob@dylan:~$

```


Repo:

- GitHub repository: [alx-backend-user-data](#)
- Directory: `0x02-Session_authentication`
- File: `api/v1/auth/session_auth.py`



 Done!

Check your code

 Get a sandbox

3. User ID for Session ID

mandatory

Update `SessionAuth` class:

Create an instance method `def user_id_for_session_id(self, session_id: str = None) -> str:` that returns a User ID based on a Session ID:

- Return `None` if `session_id` is `None`
- Return `None` if `session_id` is not a string
- Return the value (the User ID) for the key `session_id` in the dictionary `user_id_by_session_id`.
- You must use `.get()` built-in for accessing in a dictionary a value based on key

Now you have 2 methods (`create_session` and `user_id_for_session_id`) for storing and retrieving a link between a User ID and a Session ID.




```
bob@dylan:~$ cat main_2.py
#!/usr/bin/env python3

""" Main 2
"""

from api.v1.auth.session_auth import SessionAuth

sa = SessionAuth()

user_id_1 = "abcde"
session_1 = sa.create_session(user_id_1)
print("{} => {}: {}".format(user_id_1, session_1, sa.user_id_by_session_id))

user_id_2 = "fghij"
session_2 = sa.create_session(user_id_2)
print("{} => {}: {}".format(user_id_2, session_2, sa.user_id_by_session_id))

print("---")

tmp_session_id = None
tmp_user_id = sa.user_id_for_session_id(tmp_session_id)
print("{} => {}".format(tmp_session_id, tmp_user_id))

tmp_session_id = 89
tmp_user_id = sa.user_id_for_session_id(tmp_session_id)
print("{} => {}".format(tmp_session_id, tmp_user_id))

tmp_session_id = "doesntexist"
tmp_user_id = sa.user_id_for_session_id(tmp_session_id)
print("{} => {}".format(tmp_session_id, tmp_user_id))

print("---")

tmp_session_id = session_1
tmp_user_id = sa.user_id_for_session_id(tmp_session_id)
print("{} => {}".format(tmp_session_id, tmp_user_id))

tmp_session_id = session_2
tmp_user_id = sa.user_id_for_session_id(tmp_session_id)
print("{} => {}".format(tmp_session_id, tmp_user_id))

print("---")

session_1_bis = sa.create_session(user_id_1)
print("{} => {}: {}".format(user_id_1, session_1_bis, sa.user_id_by_session_id))

tmp_user_id = sa.user_id_for_session_id(session_1_bis)
print("{} => {}".format(session_1_bis, tmp_user_id))

tmp_user_id = sa.user_id_for_session_id(session_1)
print("{} => {}".format(session_1, tmp_user_id))

bob@dylan:~$
```



```

bob@dylan:~$ API_HOST=0.0.0 API_PORT=5000 AUTH_TYPE=session_auth ./main_2.py
abcde => 8647f981-f503-4638-af23-7bb4a9e4b53f: {'8647f981-f503-4638-af23-7bb4a9e4b53f': 'abcde'}

fghij => a159ee3f-214e-4e91-9546-ca3ce873e975: {'a159ee3f-214e-4e91-9546-ca3ce873e975': 'fghij', '8647f981-f503-4638-af23-7bb4a9e4b53f': 'abcde'}

---
None => None
89 => None
doesntexist => None

---
8647f981-f503-4638-af23-7bb4a9e4b53f => abcde
a159ee3f-214e-4e91-9546-ca3ce873e975 => fghij

---
abcde => 5d2930ba-f6d6-4a23-83d2-4f0abc8b8eee: {'a159ee3f-214e-4e91-9546-ca3ce873e975': 'fghij', '8647f981-f503-4638-af23-7bb4a9e4b53f': 'abcde', '5d2930ba-f6d6-4a23-83d2-4f0abc8b8eee': 'abcde'}

5d2930ba-f6d6-4a23-83d2-4f0abc8b8eee => abcde
8647f981-f503-4638-af23-7bb4a9e4b53f => abcde
bob@dylan:~$

```

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session_authentication
- File: api/v1/auth/session_auth.py

☒ Done!

4. Session cookie

Update `api/v1/auth/auth.py` by adding the method `def session_cookie(self, request=None):` that returns a cookie value from a request:

- Return `None` if `request` is `None`
- Return the value of the cookie named `_my_session_id` from `request` - the name of the cookie must be defined by the environment variable `SESSION_NAME`
- You must use `.get()` built-in for accessing the cookie in the request cookies dictionary
- You must use the environment variable `SESSION_NAME` to define the name of the cookie used for the Session ID

In the first terminal:



```

bob@dylan:~$ cat main_3.py
#!/usr/bin/env python3

""" Cookie server
"""

from flask import Flask, request
from api.v1.auth.auth import Auth

auth = Auth()

app = Flask(__name__)

@app.route('/', methods=['GET'], strict_slashes=False)
def root_path():
    """ Root path
    """
    return "Cookie value: {}\n".format(auth.session_cookie(request))

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="5000")

bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth SESSION_NAME=_my_session_id ./main_3.py
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....

```

In a second terminal:

```

bob@dylan:~$ curl "http://0.0.0.0:5000"
Cookie value: None
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000" --cookie "_my_session_id=Hello"
Cookie value: Hello
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000" --cookie "_my_session_id=C is fun"
Cookie value: C is fun
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000" --cookie "_my_session_id_fake"
Cookie value: None
bob@dylan:~$

```

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session_authentication
- File: api/v1/auth/auth.py



☒ Done!

[Check your code](#)

[>_ Get a sandbox](#)

5. Before request

mandatory

Update the `@app.before_request` method in `api/v1/app.py` :

- Add the URL path `/api/v1/auth_session/login/` in the list of excluded paths of the method `require_auth` - this route doesn't exist yet but it should be accessible outside authentication
- If `auth.authorization_header(request)` and `auth.session_cookie(request)` return `None` ,
`abort(401)`

In the first terminal:

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth SESSION_NAME=_my_session_id python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
```

In a second terminal:

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status"
{
  "status": "OK"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" # not found but not "blocked" by an authentication system
{
  "error": "Not found"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me"
{
  "error": "Unauthorized"
}
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" -H "Authorization: Basic Ym9iQGhidG4uaw86SDBsYmVydG9uU2Nob29sOTgh" # Won't work because the environment variable AUTH_TYPE is equal to "session_auth"
{
  "error": "Forbidden"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=5535d4d7-3d77-4d06-8281-495dc3acfe76" # Won't work because no user is linked to this Session ID
{
  "error": "Forbidden"
}
bob@dylan:~$
```




Repo:

- GitHub repository: `alx-backend-user-data`

- Directory: 0x02-Session_authentication
- (/).• File: api/v1/app.py

☒ Done!

Check your code

 Get a sandbox

6. Use Session ID for identifying a User

mandatory

Update `SessionAuth` class:

Create an instance method `def current_user(self, request=None):` (overload) that returns a `User` instance based on a cookie value:

- You must use `self.session_cookie(...)` and `self.user_id_for_session_id(...)` to return the `User` ID based on the cookie `_my_session_id`
- By using this `User` ID, you will be able to retrieve a `User` instance from the database - you can use `User.get(...)` for retrieving a `User` from the database.

Now, you will be able to get a `User` based on his session ID.

In the first terminal:



```

bob@dylan:~$ cat main_4.py
#!/usr/bin/env python3

""" Main 4
"""

from flask import Flask, request
from api.v1.auth.session_auth import SessionAuth
from models.user import User

""" Create a user test """
user_email = "bobsession@hbbtn.io"
user_clear_pwd = "fake pwd"

user = User()
user.email = user_email
user.password = user_clear_pwd
user.save()

""" Create a session ID """
sa = SessionAuth()
session_id = sa.create_session(user.id)
print("User with ID: {} has a Session ID: {}".format(user.id, session_id))

""" Create a Flask app """
app = Flask(__name__)

@app.route('/', methods=['GET'], strict_slashes=False)
def root_path():
    """ Root path
    """
    request_user = sa.current_user(request)
    if request_user is None:
        return "No user found\n"
    return "User found: {}\n".format(request_user.id)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port="5000")

bob@dylan:~$
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth SESSION_NAME=_my_session_
id ./main_4.py
User with ID: cf3dde1-ff24-49e4-a40b-2540333fe992 has a Session ID: 9d1648aa-da79-4692-8236
-5f9d7f9e9485
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....

```

In a second terminal:



```

bob@dylan:~$ curl "http://0.0.0.0:5000/"
No user found
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/" --cookie "_my_session_id=Holberton"
No user found
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/" --cookie "_my_session_id=9d1648aa-da79-4692-8236-5f9d7f9e9485"
User found: cf3dde1-ff24-49e4-a40b-2540333fe992
bob@dylan:~$

```

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session_authentication
- File: api/v1/auth/session_auth.py

☒ Done!

7. New view for Session Authentication

mandatory

Create a new Flask view that handles all routes for the Session authentication.

In the file `api/v1/views/session_auth.py`, create a route `POST /auth_session/login` (= `POST /api/v1/auth_session/login`):

- Slash tolerant (`/auth_session/login == /auth_session/login/`)
- You must use `request.form.get()` to retrieve email and password parameters
- If email is missing or empty, return the JSON `{ "error": "email missing" }` with the status code 400
- If password is missing or empty, return the JSON `{ "error": "password missing" }` with the status code 400
- Retrieve the User instance based on the email - you must use the class method `search` of User (same as the one used for the BasicAuth)
 - If no User found, return the JSON `{ "error": "no user found for this email" }` with the status code 404
 - If the password is not the one of the User found, return the JSON `{ "error": "wrong password" }` with the status code 401 - you must use `is_valid_password` from the User instance
 - Otherwise, create a Session ID for the User ID:
 - You must use `from api.v1.app import auth` - **WARNING: please import it only where you need it** - not on top of the file (can generate circular import - and break first tasks of this project)
 - You must use `auth.create_session(...)` for creating a Session ID
 - Return the dictionary representation of the User - you must use `to_json()` method from User
 - You must set the cookie to the response - you must use the value of the environment variable `SESSION_NAME` as cookie name - tip (`/rltoken/3WDIzYbVvdJJAf70ljWK6g`)



In the file `api/v1/views/__init__.py` , you must add this new view at the end of the file.

(/)

In the first terminal:

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth SESSION_NAME=_my_session_
id python3 -m api.v1.app
  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
```

In a second terminal:




```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XGET
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST
{
  "error": "email missing"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=guillaume@hbbtn.io"
{
  "error": "password missing"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=guillaume@hbbtn.io" -d "password=test"
{
  "error": "no user found for this email"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=bobsession@hbbtn.io" -d "password=test"
{
  "error": "wrong password"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=bobsession@hbbtn.io" -d "password=fake pwd"
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=bobsession@hbbtn.io" -d "password=fake pwd" -vvv
Note: Unnecessary use of -X or --request, POST is already inferred.
* Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> POST /api/v1/auth_session/login HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Length: 42
> Content-Type: application/x-www-form-urlencoded
```



```

>
(4)upload completely sent off: 42 out of 42 bytes
* HTTP 1.0, assume close after body

< HTTP/1.0 200 OK
< Content-Type: application/json
< Set-Cookie: _my_session_id=df05b4e1-d117-444c-a0cc-ba0d167889c4; Path=/
< Access-Control-Allow-Origin: *
< Content-Length: 210
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Mon, 16 Oct 2017 04:57:08 GMT
<
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
* Closing connection 0
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=df05b4e1-d117-444c-a0cc-ba0d167889c4"
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
bob@dylan:~$

```

Now you have an authentication based on a Session ID stored in cookie, perfect for a website (browsers love cookies).

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session_authentication
- File: api/v1/views/session_auth.py, api/v1/views/__init__.py

☒ Done!

[Check your code](#)

[> Get a sandbox](#)

8. Logout

mandatory 

Update the class `SessionAuth` by adding a new method `def destroy_session(self, request=None):` that deletes the user session / logout:

- If the `request` is equal to `None`, return `False`

- If the request doesn't contain the Session ID cookie, return False - you must use (/) self.session_cookie(request)
- If the Session ID of the request is not linked to any User ID, return False - you must use self.user_id_for_session_id(...)
- Otherwise, delete in self.user_id_by_session_id the Session ID (as key of this dictionary) and return True

Update the file api/v1/views/session_auth.py , by adding a new route DELETE /api/v1/auth_session/logout :

- Slash tolerant
- You must use from api.v1.app import auth
- You must use auth.destroy_session(request) for deleting the Session ID contains in the request as cookie:
 - If destroy_session returns False , abort(404)
 - Otherwise, return an empty JSON dictionary with the status code 200

In the first terminal:

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth SESSION_NAME=_my_session_id python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
```

In a second terminal:



```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=bobsession@hbbtn.io" -d "password=fake pwd" -vvv
```

Note: Unnecessary use of -X or --request, POST is already inferred.

```
* Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> POST /api/v1/auth_session/login HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Length: 42
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 42 out of 42 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: application/json
< Set-Cookie: _my_session_id=e173cb79-d3fc-4e3a-9e6f-bcd345b24721; Path=/
< Access-Control-Allow-Origin: *
< Content-Length: 210
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Mon, 16 Oct 2017 04:57:08 GMT
<
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
* Closing connection 0
```

```
bob@dylan:~$
```

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=e173cb79-d3fc-4e3a-9e6f-bcd345b24721"
```

```
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
```

```
bob@dylan:~$
```

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/logout" --cookie "_my_session_id=e173cb79-d3fc-4e3a-9e6f-bcd345b24721"
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>405 Method Not Allowed</title>
<h1>Method Not Allowed</h1>
<p>The method is not allowed for the requested URL.</p>
```

```
bob@dylan:~$
```

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/logout" --cookie "_my_session_id=
```



```
e173cb79-d3fc-4e3a-9e6f-bcd345b24721" -XDELETE
```



```
bob@dylan:~$
```

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=e173cb79-d3fc-4e3a-9e6f-bcd345b24721"
```

```
{
  "error": "Forbidden"
}
```

```
bob@dylan:~$
```

Login, logout... what's else?

Now, after getting a Session ID, you can request all protected API routes by using this Session ID, no need anymore to send User email and password every time.

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session_authentication
- File: api/v1/auth/session_auth.py, api/v1/views/session_auth.py

☐ Done?

9. Expiration?

#advanced

Actually you have 2 authentication systems:

- Basic authentication
- Session authentication

Now you will add an expiration date to a Session ID.

Create a class `SessionExpAuth` that inherits from `SessionAuth` in the file `api/v1/auth/session_exp_auth.py`:

- Overload `def __init__(self):` method:
 - Assign an instance attribute `session_duration`:
 - To the environment variable `SESSION_DURATION` casts to an integer
 - If this environment variable doesn't exist or can't be parse to an integer, assign to 0
- Overload `def create_session(self, user_id=None):`
 - Create a Session ID by calling `super()` - `super()` will call the `create_session()` method of `SessionAuth`
 - Return `None` if `super()` can't create a Session ID
 - Use this Session ID as key of the dictionary `user_id_by_session_id` - the value for this key must be a dictionary (called "session dictionary"):
 - The key `user_id` must be set to the variable `user_id`
 - The key `created_at` must be set to the current datetime - you must use `datetime.now()`
 - Return the Session ID created
- Overload `def user_id_for_session_id(self, session_id=None):`



(/)

- Return None if session_id is None
- Return None if user_id_by_session_id doesn't contain any key equals to session_id
- Return the user_id key from the session dictionary if self.session_duration is equal or under 0
- Return None if session dictionary doesn't contain a key created_at
- Return None if the created_at + session_duration seconds are before the current datetime.
datetime - timedelta (/rltoken/mwc3EnIWLNJ2rvzvgZT8eA)
- Otherwise, return user_id from the session dictionary

Update api/v1/app.py to instantiate auth with SessionExpAuth if the environment variable AUTH_TYPE is equal to session_exp_auth.

In the first terminal:

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_exp_auth SESSION_NAME=_my_session_id SESSION_DURATION=60 python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
```

In a second terminal:



```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=bobsession@hbbtn.io" -d "password=fake pwd" -vvv
```

Note: Unnecessary use of -X or --request, POST is already inferred.

```
* Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> POST /api/v1/auth_session/login HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Length: 42
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 42 out of 42 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: application/json
< Set-Cookie: _my_session_id=eea5d963-8dd2-46f0-9e43-fd05029ae63f; Path=/
< Access-Control-Allow-Origin: *
< Content-Length: 210
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Mon, 16 Oct 2017 04:57:08 GMT
<
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
* Closing connection 0
```

```
bob@dylan:~$
```

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=eea5d963-8dd2-46f0-9e43-fd05029ae63f"
```

```
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
```

```
bob@dylan:~$
```

```
bob@dylan:~$ sleep 10
```

```
bob@dylan:~$
```

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=eea5d963-8dd2-46f0-9e43-fd05029ae63f"
```

```
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
```

```

    "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  (/)"last_name": null,
    "updated_at": "2017-10-16 04:23:04"
  }
bob@dylan:~$
bob@dylan:~$ sleep 51 # 10 + 51 > 60
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=eea5d963-8d
d2-46f0-9e43-fd05029ae63f"
{
  "error": "Forbidden"
}
bob@dylan:~$

```

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session_authentication
- File: api/v1/auth/session_exp_auth.py, api/v1/app.py

☐ Done?

10. Sessions in database

#advanced

Since the beginning, all Session IDs are stored in memory. It means, if your application stops, all Session IDs are lost.

For avoid that, you will create a new authentication system, based on Session ID stored in database (for us, it will be in a file, like User).

Create a new model UserSession in models/user_session.py that inherits from Base :

- Implement the def __init__(self, *args: list, **kwargs: dict): like in User but for these 2 attributes:
 - user_id:string
 - session_id:string

Create a new authentication class SessionDBAuth in api/v1/auth/session_db_auth.py that inherits from SessionExpAuth :

- Overload def create_session(self, user_id=None): that creates and stores new instance of UserSession and returns the Session ID
- Overload def user_id_for_session_id(self, session_id=None): that returns the User ID by requesting UserSession in the database based on session_id
- Overload def destroy_session(self, request=None): that destroys the UserSession based on the Session ID from the request cookie

Update api/v1/app.py to instantiate auth with SessionDBAuth if the environment variable AUTH_TYPE is equal to session_db_auth .

In the first terminal:


```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_db_auth SESSION_NAME=_my_session_id SESSION_DURATION=60 python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
```

In a second terminal:



```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=bobsession@hbbtn.io" -d "password=fake pwd" -vvv
```

Note: Unnecessary use of -X or --request, POST is already inferred.

```
* Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> POST /api/v1/auth_session/login HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Length: 42
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 42 out of 42 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: application/json
< Set-Cookie: _my_session_id=bacadfad-3c3b-4830-b1b2-3d77dfb9ad13; Path=/
< Access-Control-Allow-Origin: *
< Content-Length: 210
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Mon, 16 Oct 2017 04:57:08 GMT
<
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
* Closing connection 0
```

```
bob@dylan:~$
```

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=bacadfad-3c3b-4830-b1b2-3d77dfb9ad13"
```

```
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
```

```
bob@dylan:~$
```

```
bob@dylan:~$ sleep 10
```

```
bob@dylan:~$
```

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=bacadfad-3c3b-4830-b1b2-3d77dfb9ad13"
```

```
{
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbbtn.io",
  "first_name": null,
```

```
"id": "cf3ddee1-ff24-49e4-a40b-2540333fe992",  
(/)"last_name": null,  
"updated_at": "2017-10-16 04:23:04"
```

```
}  
bob@dylan:~$  
bob@dylan:~$ sleep 60  
bob@dylan:~$  
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=bacadfad-3c  
3b-4830-b1b2-3d77dfb9ad13"  
{  
  "error": "Forbidden"  
}  
bob@dylan:~$
```

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session_authentication
- File: api/v1/auth/session_db_auth.py, api/v1/app.py, models/user_session.py

☐ Done?

Check your code

[> Get a sandbox](#)

