

 Home(/)  

# 0x02. Session authentication

 Check-end My Projects(/projects/current) Authentication (/me)

Weight: 1

Projects(/projects/current)



Project over - took place from Aug 7, 2024 6:00 AM to Aug 9, 2024 6:00 AM



An auto QA review will automatically make corrections(/to\_review)



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

## In a nutshell...

- **Auto QA review:** 96.4/135 mandatory & 0.0/46 optional

- **Altogether: 71.41%**



Curriculums(/dashboards/my\_curriculums)

◦ Mandatory: 71.41%

◦ Optional: 0.0%

◦ Calculation:  $71.41\% + (71.41\% * 0.0\%) == 71.41\%$ 

Concepts(/concepts)



Conference rooms(/dashboards/video\_rooms)



## Background Context

Servers(/servers)

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



Sandboxes(/user\_containers/current)

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



**Flask-HTTPAuth**). Here, for the learning purpose, we will walk through each step of this mechanism to understand it by doing.



Video on demand(/dashboards/videos)

## Resources

### Read or watch:



- REST API Authentication Mechanisms - Only the session auth part  
Peers(/users/peers)

(/rltoken/oofk0VhuSOZFZTNTVrQeaQ)

- HTTP Cookie (/rltoken/pelV8xuJ4PDJMOVfQk-d2g)



- Flask (/rltoken/vrCaeZxNcpaQOLh-0SVhWA)

Discord(<https://discord.com/app>)

- Flask Cookie (/rltoken/QYfI5oW6OHUmHDzwKV1Qsw)

## Learning Objectives

At the end of this project, you are expected to be able to explain to anyone

(/rltoken/uWxp4Vcr3Du9UZtC9NS\_A), **without the help of Google:**

My Profile(/users/my\_profile)



# General



- What authentication means (/)
- What session authentication means



- What Cookies are
- How to use Cookies
- How to parse Cookies



## Requirements

My Planning(/planning/me)



## Python Scripts

Projects(/projects/current)

- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using python3 (version 3.7)
- All your files should have a documentation ( python3 -c 'print(\_\_import\_\_("my\_module").\_\_doc\_\_)' )
- 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)



Sandboxes(/user\_containers/current)

## Tasks



Tools(/dashboards/my\_tools)

### 0. Et moi et moi et moi!

mandatory



Video on demand(/dashboards/videos)

Score: 90.67% (Checks completed: 100.0%)

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



Peers(/users/peers)

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

- GET /api/v1/users
- POST https://discord.com/app
- GET /api/v1/users/<user\_id>
- PUT /api/v1/users/<user\_id>
- DELETE /api/v1/users/<user\_id>



Discord(/discord)



Now, you will add a new endpoint: GET /users/me to retrieve the authenticated User object.  
My Profile(/users/my\_profile)

- 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

Projects(/projects/current)

In the first terminal:

```
bob@dylan:~$ cat main_0.py
#!/usr/bin/env python3
""" Main 0
? """
    Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
import base64
from api.v1.auth.basic_auth import BasicAuth
from models.user import User
```

```
""" Curriculums(/dashboards/my_curriculums)
user_email = "bob@hbtn.io"
user_clear_pwd = "H0lbertonSchool98!"
```

```
Concepts(/concepts)
```

```
user = User()
```

```
user.email = user_email
```

```
user.password = user_clear_pwd
Conference rooms(/dashboards/video_rooms)
```

```
print("New user: {}".format(user.id))
```

```
user.save()
```

```
Servers(/servers)
```

```
basic_clear = "{}:{}".format(user_email, user_clear_pwd)
```

```
print("Basic Base64: {}".format(base64.b64encode(basic_clear.encode('utf-8')).dec
>_ode("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: Ym9iIQRhbnRlG4uaw86SDBsYmVydG9uU2Nob29sOTgh
```

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

```
Video on demand(/dashboards/videos)
```

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=basic_auth python3 -m api.v
```

```
1.app
```

```
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

```
....
```

Peers(/users/peers)

In a second terminal:


Discord(<https://discord.com/app>)

My Profile(/users/my\_profile)

```

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"
[
  {
    "QA Reviews" can make 0.25 of 1.7 (view)
    "email": "bob@hbtn.io",
    "first_name": null,
    "id": "9375973a-68c7-46aa-b135-29f79e837495"
    "Evaluation quizzes" dashboard my current evaluation quizzes)
    "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"
{
  Concepts(/concepts)
    "created_at": "2017-09-25 01:55:17",
    "email": "bob@hbtn.io",
    "first_name": null,
    "id": "9375973a-68c7-46aa-b135-29f79e837495",
    "last_name": null,
    "updated_at": "2017-09-25 01:55:17"
  }
}
bob@dylan:~$
> Sandboxes(/user_containers/current)

```

-  **Repo:** Tools(/dashboards/my\_tools)
- GitHub repository: alx-backend-user-data
  - Directory: 0x02-Session\_authentication
  - Video on demand(/dashboards/videos)
  - File: api/v1/app.py, api/v1/views/users.py

 **Peers(/users/peers)**

Check submission  Get a sandbox View results

## 1. Empty session

mandatory

 Discord(<https://discord.com/app>)

Score: 100.0% (Checks completed: 100.0%)



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.

My Planning(/planning/me)

In the first terminal:



```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth python3 -m ap
Projects(/projects/current)
i.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
✓···· QA Reviews I can make(/corrections/to_review)
```

In a second terminal:



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status"
{
  "status": "OK"
}
```



Curriculums(/dashboards/my\_curriculums)

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status/"
{
  "status": "OK"
}
```



Concepts(/concepts)

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



Conference rooms(/dashboards/video\_rooms)

```
bob@dylan:~$ curl http://0.0.0.0:5000/api/v1/users
{
  "error": "Unauthorized"
}
```



Servers(/servers)

```
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users" -H "Authorization: Test"
```



Sandboxes(/user\_containers/current)

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



Tools(/dashboards/my\_tools)



Video on demand(/dashboards/videos)

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



Check solution <https://discord.com/channels/1000000000000000000/1000000000000000000> View results

## 2. Create a session

mandatory

Score: 50.0% (Checks completed: 50.0%)  
My Profile(/users/my\_profile)

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`:
  - Home (/) Return None if `user_id` is None
    - o Return None if `user_id` is not a string
    - o Otherwise:
      - My Planning (/planning/me)
        - 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`
      - Projects (/projects/current)
        - Return the Session ID
        - o The same `user_id` can have multiple Session ID - indeed, the `user_id` is the value in the dictionary `user_id_by_session_id`
  - QA Reviews (/qa-reviews/corrections-to-review)

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



Evaluation quizzes (/dashboards/my\_current\_evaluation\_quizzes)



Curriculums (/dashboards/my\_curriculums)



Concepts (/concepts)



Conference rooms (/dashboards/video\_rooms)



Servers (/servers)



Sandboxes (/user\_containers/current)



Tools (/dashboards/my\_tools)



Video on demand (/dashboards/videos)



Peers (/users/peers)



Discord (<https://discord.com/app>)



My Profile (/users/my\_profile)

```

bob@dylan:~$ cat main_1.py
#!/usr/bin/env python3
"""(?) Main 1
"""

from api.v1.auth.session_auth import SessionAuth
Home(/)

sa = SessionAuth()

print(My Planning(/planning/me).format(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 = "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:~$ SERVER_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth ./main_1.py
<class 'dict'>: {}
None => 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:~$

```

Peers(/users/peers)  
Repo:

- GitHub repository: alx-backend-user-data
- Discord(<https://discord.com/app>)
- Directory: 0x02-Session\_authentication
- File: api/v1/auth/session\_auth.py



Check submission

Mark submission

> Get a sandbox

View results

My Profile(/users/my\_profile)

### 3. User ID for Session ID

mandatory

(/)

Score: 100.0% (Checks completed: 100.0%)



Home(/)

Update SessionAuth class:

Create an instance method `def user_id_for_session_id(self, session_id: str = None) ->`



My Planning(/planning/me)

Tr: 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



QA Reviews I can make(/corrections/to\_review)

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.



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)



Curriculums(/dashboards/my\_curriculums)



Concepts(/concepts)



Conference rooms(/dashboards/video\_rooms)



Servers(/servers)



Sandboxes(/user\_containers/current)



Tools(/dashboards/my\_tools)



Video on demand(/dashboards/videos)



Peers(/users/peers)



Discord(<https://discord.com/app>)



My Profile(/users/my\_profile)



```

bob@dylan:~$ cat main_2.py
#!/usr/bin/env python3
"""(?) Main 2
"""

from api.v1.auth.session_auth import SessionAuth
Home(/)
sa = SessionAuth()

user_id_1 = 1
session_1 = sa.create_session(user_id_1)
print("{} => {}: {}".format(user_id_1, session_1, sa.user_id_by_session_id))
Projects(/projects/current)
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("---")
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
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))
Concepts(/concepts)
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("---")
Servers(/servers)
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("Video on demand(/dashboards/videos)


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=0.0.0.0 API_PORT=5000 AUTH_TYPE=session_auth ./main_2.py
abcde => 8647f981-f503-4638-af23-7bb4a9e4b53f: {'8647f981-f503-4638-af23-7bb4a9e4b53f': 'My Profile(/users/my_profile)

```


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

---  
 None => None  
89 => None  
doesn't exist => 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:~\$

 Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)  
**Repo:**


- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session\_authentication
-  Curriculum (/dashboards/my\_curriculums)
- File: api/v1/auth/session\_auth.py



 Concepts(/concepts)

 **Session cookie** Conferences rooms(/dashboards/video\_rooms)

mandatory

 Score: 100.0% (Checks completed: 100.0%)  
Servers(/servers)

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

 Peers(/users/peers)

 Discord(https://discord.com/app)



My Profile(/users/my\_profile)

```

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()

My Planning(/planning/me)
app = Flask(__name__)

@app.route('/projects/show', methods=['GET'], strict_slashes=False)
def root_path():
    """ Root path
    QA Reviews I can make(/corrections/to_review)
    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)
....

```

Concepts(/concepts)  
in a second terminal:

```

bob@dylan:~$ curl "http://0.0.0.0:5000"
Conference rooms(/dashboards/video_rooms)
Cookie value: None
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000" --cookie "_my_session_id=Hello"
Servers(/servers)
Cookie value: Hello
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000" --cookie "_my_session_id=C is fun"
Sandboxes(/user_containers/current)
Cookie value: C is fun
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000" --cookie "_my_session_id_fake"
Tools(/dashboards/my_tools)
Cookie value: None
bob@dylan:~$

```

Video on demand(/dashboards/videos)

## Repo:

- GitHub repository: alx-backend-user-data
- Directory: src/04-session\_authentication
- File: api/v1/auth/auth.py

Discord(<https://discord.com/app>)

Check submission

Get a sandbox

View results



## 5. Before request

mandatory

My Profile(/users/my\_profile)  
Score: 91.92% (Checks completed: 100.0%)

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:

```
My Planning(/planning/me)
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)
....
Projects(/projects/current)
```

✓ In a second terminal: I can make `/corrections/to_review`

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status"
? {
  Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
  "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
Curriculums(/dashboards/my_curriculums)
{
  "error": "Not found"
}
Concepts(/concepts)
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me"
{
  Conference rooms(/dashboards/video_rooms)
  "error": "Unauthorized"
}
Servers(/servers)
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" -H "Authorization: Basic
Ym9iQGhidG4uaw86SDBSYmVydG9uU2Nob29sOTgh" # Won't work because the environment va
riable AUTH_TYPE is equal to "session_auth"
Sandboxes(/user_containers/current)
{
  "error": "Forbidden"
}
Tools(/dashboards/my_tools)
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 t
his Session ID
Video on Demand(/dashboards/videos)
{
  "error": "Forbidden"
}

bob@dylan:~$
Peers(/users/peers)
```

 **Repo:** [Discord\(https://discord.com/app\)](https://discord.com/app)

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



My Profile(/users/my\_profile)

[Check submission](#)[Get a sandbox](#)[View results](#)[\(/\)](#)

## 6. Use Session ID for identifying a User

**mandatory**[Home\(/\)](#)

Score: 10.0% (Checks completed: 10.0%)

[My Planning\(/planning/me\)](#)[Projects\(/projects/evaluation\)](#)

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

- 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.

[Evaluation quizzes\(/dashboards/my\\_current\\_evaluation\\_quizzes\)](#)

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

In the first terminal:

[Curriculums\(/dashboards/my\\_curriculums\)](#)[Concepts\(/concepts\)](#)[Conference rooms\(/dashboards/video\\_rooms\)](#)[Servers\(/servers\)](#)[Sandboxes\(/user\\_containers/current\)](#)[Tools\(/dashboards/my\\_tools\)](#)[Video on demand\(/dashboards/videos\)](#)[Peers\(/users/peers\)](#)[Discord\(https://discord.com/app\)](https://discord.com/app)[My Profile\(/users/my\\_profile\)](#)

```

bob@dylan:~$ cat main_4.py
#!/usr/bin/env python3
"""(?) Main 4
"""

from flask import Flask, request
from app.views.auth.session_auth import SessionAuth
from models.user import User

""" Create a Flask app
    Home(/)
    My Planings(/planning/me)
    user_email = "bobsession@hbbtn.io"
    user_clear_pwd = "fake pwd"
    Projects(/projects/current)
    user = User()
    user.email = user_email
    user.password_clear = user_clear_pwd
    user.save()
    QA Reviews I can make/corrections/to_review)
    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
        Conference rooms(/dashboards/video_rooms)
        request_user = sa.current_user(request)
        if request_user is None:
            Servers(/servers)
            return "User not found\n"
        return "User found: {}\n".format(request_user.id)

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

bob@dylan:~$ python3 main_4.py
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: 9d1648aa-da79-4692-8236-5f9d7f9e9485 has a Session ID: 9d1648aa-da7
9-4692-8236-5f9d7f9e9485
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....

```

Peers(/users/peers)  
In a second terminal:

Discord(<https://discord.com/app>)



My Profile(/users/my\_profile)

```

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 My Planning (planning@me)
bob@dylan:~$

```


 Projects(/projects/current)

#### Repo:

- ✓ • [QA Reviews I can make/corrections to review](#)
- [GitHub repository: atx-backend-user-data](#)
- [Directory: 0x02-Session\\_authentication](#)
- ? • [File: api/v1/auth/session\\_auth.py](#)
- [Evaluation quizzes\(dashboards/my\\_current\\_evaluation\\_quizzes\)](#)

Check submission

Mark submission

 Get a sandbox

View results

## 7. New view for Session Authentication

mandatory

 Score: 61.58% (checks completed: 94.74%)

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

Conference rooms(dashboards/video\_rooms)

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

Servers(/servers)

- `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 `dashboards/video_rooms`)
  - 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

My Profile(/users/my\_profile)

- 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:



Home(/)

```
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
```



My Planning(/planning/me)

\* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

....



Projects(/projects/current)

In a second terminal:



QA Reviews I can make(/corrections/to\_review)



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)



Curriculums(/dashboards/my\_curriculums)



Concepts(/concepts)



Conference rooms(/dashboards/video\_rooms)



Servers(/servers)



Sandboxes(/user\_containers/current)



Tools(/dashboards/my\_tools)



Video on demand(/dashboards/videos)



Peers(/users/peers)



Discord(<https://discord.com/app>)



My Profile(/users/my\_profile)



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>  
Home (/)  
bob@dylan:~\$  
bob@dylan:~\$ curl "http://0.0.0.0:5000/api/v1/auth\_session/login" -XPOST  
{  
 "error": "email missing"  
}  
My Planning (/planning/me)  
bob@dylan:~\$  
Projects (/projects/current)  
bob@dylan:~\$ curl "http://0.0.0.0:5000/api/v1/auth\_session/login" -XPOST -d "email=guillaume@hbtn.io"  
{  
 "error": "password and make corrections/to\_review"  
}  
QA Reviews (/reviews/qa)  
bob@dylan:~\$  
Evaluation quizzes (/dashboards/my\_evaluation/quizzes)  
bob@dylan:~\$ curl "http://0.0.0.0:5000/api/v1/auth\_session/login" -XPOST -d "email=guillaume@hbtn.io" -d "password=test"  
{  
 "error": "no user found for this email"  
}  
Curriculums (/dashboards/my\_curriculums)  
bob@dylan:~\$  
bob@dylan:~\$ curl "http://0.0.0.0:5000/api/v1/auth\_session/login" -XPOST -d "email=bobsession@hbtn.io" -d "password=test"  
{  
 Concepts (/concepts)  
 "error": "wrong password"  
 }  
 }  
Conference rooms (/dashboards/video\_rooms)  
bob@dylan:~\$  
bob@dylan:~\$ curl "http://0.0.0.0:5000/api/v1/auth\_session/login" -XPOST -d "email=bobsession@hbtn.io" -d "password=fake pwd"  
{  
 Servers (/servers)  
 "created\_at": "2017-10-16 04:23:04",  
 "email": "bobsession@hbtn.io",  
 "file\_sandboxes (/user\_containers/current)"  
 "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",  
 "last\_name": null,  
 "upload (/dashboards/my\_tools)"  
 "updated\_at": "2017-10-16 04:23:04"  
 }  
}  
bob@dylan:~\$  
Video on demand (/dashboards/vod)  
bob@dylan:~\$ curl "http://0.0.0.0:5000/api/v1/auth\_session/login" -XPOST -d "email=bobsession@hbtn.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  
Peers (/users/peers)  
Discord (https://discord.com/app)

```

< 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
<
{
  "My Planning(/planning/me)
  "created_at": "2017-10-16 04:23:04",
  "email": "bobsession@hbtn.io",
  "first_name": null,
  "id": "cf3dde1-ff24-49e4-a40b-2540333fe992",
  "last_name": null,
  "updated_at": "2017-10-16 04:23:04"
}
QA Reviews I can make(/corrections/to_review)
* Closing connection 0
bob@dylan:~$
bob@dylan:~$ curl -i 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@hbtn.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) and servers (servers)

**>Repo:** Sandboxes(/user\_containers/current)

- GitHub repository: alx-backend-user-data
- Directory: 0x02-Session authentication
- File: api/v1/views/session\_auth.py, api/v1/views/\_\_init\_\_.py

Video on demand(/dashboards/videos)

Check submission Mark submission Get a sandbox View results

## 8. Logout

mandatory

Peers(/users/peers)

Score: 39.72% (Checks completed: 61.11%)

Discord(<https://discord.com/app>)

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.request.cookies.get('session\_id', None) to get the session ID from the cookie



- 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`:



- `SlashPlanning(/planning/me)`

- 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.



`Projects(/projects/current)`

- If `destroy_session` returns False, `abort(404)`

- Otherwise, return an empty JSON dictionary with the status code 200



`QA Reviews I can make(/corrections/to_review)`

In the first terminal:



```
bob@dy:~/Evaluation$ curl -H "Host: localhost:5000" -H "Cookie: 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)
....
```



`Curriculum(/dashboards/my_curriculum)`

In a second terminal:



`Concepts(/concepts)`



`Conference rooms(/dashboards/video_rooms)`



`Servers(/servers)`



`Sandboxes(/user_containers/current)`



`Tools(/dashboards/my_tools)`



`Video on demand(/dashboards/videos)`



`Peers(/users/peers)`



`Discord(https://discord.com/app)`



`My Profile(/users/my_profile)`

```


bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/auth_session/login" -XPOST -d "email=bobsession@hbtn.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/7.34.0
> Accept: */*
> Content-Length: 42
> Content-Type: application/x-www-form-urlencoded
* upload completely sent off: 42 out of 42 bytes
* HTTP request sent, awaiting response...
< HTTP/1.0 200 OK
< Content-Type: application/json
< Set-Cookie: quizses17a0b79-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@hbtn.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@hbtn.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?

 **My Planning(/planning/me)**  
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.

 **Projects(/projects/current)**

**Repo:**


- ✓ **GitHub Repository** [make corrections to a view](#)
- Directory: 0x02-Session\_authentication
  - File: api/v1/auth/session\_auth.py, api/v1/views/session\_auth.py
- ? **Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)**

Check submission

Mark submission


> Get a sandbox

View results

 **Expirations** [Curriculums\(/dashboards/my\\_curriculums\)](#)

#advanced





 Score: 0.0% (0 concepts / 0 concepts)

 Actually you have 2 authentication systems:

- Conference rooms(/dashboards/video\_rooms)
- Basic authentication
- Session authentication

 **Servers(/servers)**  
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:
  -  **Tools(/dashboards/my\_tools)**
    - 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
-  **Video on demand(/dashboards/videos)**
  - 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
  -  **Peers(/users/peers)**
    - 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"):
  -  **Discord(<https://discord.com/app>)**
    - 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/mwc3EnlWLNJ2rvzvZT8eA)`
- Otherwise, return `user_id` from the session dictionary



Home(/)



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:

`Projects(/projects/current)`

```
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
```



\* Run `QA-Review` to make corrections (to preview)

....



In a second terminal, `Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)`



`Curriculums(/dashboards/my_curriculums)`



`Concepts(/concepts)`



`Conference rooms(/dashboards/video_rooms)`



`Servers(/servers)`



`Sandboxes(/user_containers/current)`



`Tools(/dashboards/my_tools)`



`Video on demand(/dashboards/videos)`



`Peers(/users/peers)`



`Discord(https://discord.com/app)`



`My Profile(/users/my_profile)`

```


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/7.55.1
> Accept: */*
> Content-Length: 42
> Content-Type: application/x-www-form-urlencoded
* upload completely sent off: 42 out of 42 bytes
* HTTP request sent, awaiting response...
< HTTP/1.0 200 OK
< Content-Type: application/json
? < Set-Cookie: quizses=15d963-8dd2-46f0-9e43-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
< Curriculumums(/dashboards/my_curriculumums)
{
  "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:~$ 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=
eea5d963-8dd2-46f0-9e43-fd05029ae63f"
{
  "error": "Forbidden"
}
My Planning(/planning/me)
bob@dylan:~$

```

 Projects(/projects/current)

**Repo:**

- ✓ • GitHub repository: aly-backend-user-data
- Directory: 0x02-Session\_authentication
- File: api/v1/auth/session\_exp\_auth.py, api/v1/app.py

? Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

## 0. Sessions in database


#advanced

 Score: 0.0% (Checks completed: 0.0%)

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 example, it will be in a file like user ).

Create a new model UserSession in models/user\_session.py that inherits from Base :

 • Implement the user\_session method (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:

My Profile(/users/my\_profile)



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



Home(/)

In a second terminal:



My Planning(/planning/me)



Projects(/projects/current)



QA Reviews I can make(/corrections/to\_review)



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)



Curriculums(/dashboards/my\_curriculums)



Concepts(/concepts)



Conference rooms(/dashboards/video\_rooms)



Servers(/servers)



Sandboxes(/user\_containers/current)



Tools(/dashboards/my\_tools)



Video on demand(/dashboards/videos)



Peers(/users/peers)



Discord(<https://discord.com/app>)



My Profile(/users/my\_profile)

```

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/7.55.1 (Ubuntu)
> Accept: */*
> Content-Length: 42
> Content-Type: application/x-www-form-urlencoded
* upload completely sent off: 42 out of 42 bytes
* HTTP request sent, awaiting response...
< HTTP/1.0 200 OK
< Content-Type: application/json
? < Set-Cookie: quizses4ashbcbdrmyfad-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
< Curriculumums(/dashboards/my_curriculumums)
{
  "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"
> 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:~$ 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 60
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users/me" --cookie "_my_session_id=
bacadfad-3c3b-4830-b1b2-3d77dfb9ad13"
{
  "error": "Forbidden"
}
My Planning(/planning/me)
bob@dylan:~$
```

Projects(/projects/current)

**Repo:**

✓

- GitHub repository: alx-backend-user-data
- QA Reviews I can make(/corrections/to\_review)
- Directory: 0x02-Session\_authentication
- File: api/v1/auth/session\_db\_auth.py, api/v1/app.py, models/user\_session.py

?

Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

Check submission

Mark submission

> Get a sandbox

View results

Curriculums(/dashboards/my\_curriculums)

Concepts(/concepts)

Copyright © 2024 ALX, All rights reserved.

Conference rooms(/dashboards/video\_rooms)

Servers(/servers)

Sandboxes(/user\_containers/current)

Tools(/dashboards/my\_tools)

Video on demand(/dashboards/videos)

Peers(/users/peers)

Discord(https://discord.com/app)



My Profile(/users/my\_profile)