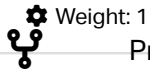



# Home(/) 0x03. User authentication service

 Check-end My  Authentication (/me)



Weight: 1

Projects(/projects/current)

 Project over - took place from Aug 12, 2024 6:00 AM to Aug 16, 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:** 70.2/89 mandatory & 4.0/4 optional

- **Altogether: 157.76%**



Curriculums(/dashboards/my\_curriculums)

◦ Mandatory: 78.88%

◦ Optional: 100.0%

◦ Calculation:  $78.88\% + (78.88\% * 100.0\%) == 157.76\%$



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>)

In the industry, you should **not** implement your own authentication system and use a module or framework that doing it for you (like in Python-Flask: Flask-User (/rltoken/9nVfotMI\_1zpEzihMzBeTA)). Here, for the learning purpose, we will walk through each step of this mechanism to understand it by doing.

My Profile(/users/my\_profile)



# Resources

Read or watch:



- Flask documentation (/rltoken/IKExyvivrW4eh0el8UV6A)
- Requests module (/rltoken/py7LuuD1u2MUwcaf8wnDzQ)
- HTTP status codes (/rltoken/cj-mc5ZHp\_KyXn1yikHC0A)



## Learning Objectives



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

Writoken/AdmZmpBujCcf5GqYFXA), **without the help of Google**:



- How to declare API routes in a Flask app
- How to get and set cookies
- How to retrieve request form data
- How to return various HTTP status codes



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

## Requirements



- Allowed editors: `vi`, `vim`, `emacs`
- All your files must be executable 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)



- You should use `SQLAlchemy 1.3.x`
- 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)

- All your functions should be type annotated
- The flask app should only interact with `Auth` and never with `DB` directly.
- Only public methods of `Auth` and `DB` should be used outside these classes



Peers(/users/peers)

## Setup



Discord(<https://discord.com/app>)  
You will need to install `bcrypt`

```
pip3 install bcrypt
```



My Profile(/users/my\_profile)

# Tasks

## 0. User model

mandatory



Home(/)

Score: 100.0% (Checks completed: 100.0%)



In this task you will create a SQLAlchemy model named `User` for a database table named `users` (by using the mapping declaration `(/rltoken/-a69l-rGqoFdXnnu6qfKdA)` of SQLAlchemy).



The model will have the following attributes:

- `id`, the integer primary key
- `email`, a non-nullable string
- `hashed_password`, a non-nullable string
- `session_id`, a nullable string
- `reset_token`, a nullable string



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



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

```
bob@dylan:~$ cat main.py
```

```
#!/usr/bin/env python3
```

```
"""
```



Main file

```
"""
```

```
from user import User
```



Concepts(/concepts)

```
print(User.__tablename__)
```



for conferences rooms(/dashboards/videos\_rooms)

```
print("{}: {}".format(column, column.type))
```



```
bob@dylan:~/servers$ python3 main.py
```

```
users
```

```
users.id: INTEGER
```



```
users.sandboxes(/user_containers/current)
```

```
users.hashed_password: VARCHAR(250)
```

```
users.session_id: VARCHAR(250)
```



```
users.reset_token: VARCHAR(250)
```

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



Video on demand(/dashboards/videos)

### Repo:

- GitHub repository: `alx-backend-user-data`
- Directory: `0x03-user_authentication_service`
- File: `user.py`



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

Check submission

Get a sandbox

View results



My Profile(/users/my\_profile)

## 1. create user

mandatory

(/)

Score: 76.67% (Checks completed: 100.0%)



Home(/)

In this task, you will complete the `DB` class provided below to implement the `add_user` method.



"""DB module  
""" My Planning(/planning/me)



```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
from sqlalchemy.orm.session import Session
```



QA Reviews I can make(/corrections/to\_review)  
from user import Base



class DB: Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

```
"""DB class  
"""
```



```
def __init__(self) -> None:  
    """Initialize a new DB instance  
    """
```



```
self._engine = create_engine("sqlite:///a.db", echo=True)  
Base.metadata.drop_all(self._engine)  
Base.metadata.create_all(self._engine)
```



```
self.__session = None  
Conference rooms(/dashboards/video_rooms)
```

@property



```
def _session(self) -> Session:  
    Servers(/servers)  
    """Memoized session object  
    """
```



```
if self.__session is None:  
    Sandboxes(/user_containers/current)  
    DBSession = sessionmaker(bind=self._engine)  
    self.__session = DBSession()
```



```
return self.__session  
Tools(/dashboards/my_tools)
```

Note that `DB._session` is a private property and hence should NEVER be used from outside the `DB`



ass. Video on demand(/dashboards/videos)

Implement the `add_user` method, which has two required string arguments: `email` and

`hashed_password`, and returns a `User` object. The method should save the user to the database. No validations are required at this stage.



Peers(/users/peers)



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



My Profile(/users/my\_profile)

```
bob@dylan:~$ cat main.py
#!/usr/bin/env python3
"""
Main file
"""
Home(/)
from db import DB
from user import User
My Planning(/planning/me)
my_db = DB()

user_1 = User(username="test@test.com", "SuperHashedPwd")
print(user_1.id)

user_2 = User(username="test1@test.com", "SuperHashedPwd1")
print(user_2.id)

bob@dylan:~$ python3 main.py
1
2
bob@dylan:~$
```

Curriculums(/dashboards/my\_curriculums)

#### Repo:

- Concepts(/concepts)
- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File: Conference rooms(/dashboards/video\_rooms)

Check submission Servers(/servers) Set a sandbox View results

## 2. Find user

mandatory

Sandboxes(/user\_containers/current)

Score: 72.0% (Checks completed: 100.0%)

Tools(/dashboards/my\_tools)

In this task you will implement the `DB.find_user_by` method. This method takes in arbitrary keyword arguments and returns the first row found in the `users` table as filtered by the method's input arguments. No validation of input arguments required at this point.

Make sure that SQLAlchemy's `NoResultFound` and `InvalidRequestError` are raised when no results are found, or when wrong query arguments are passed, respectively.

Warning: Peers(/users/peers)

- `NoResultFound` has been moved from `sqlalchemy.orm.exc` to `sqlalchemy.exc` between the version 1.3.x and 1.4.x of SQLAlchemy - please make sure you are importing it from `sqlalchemy.exc`



My Profile(/users/my\_profile)



In this task, you will implement the `DB.update_user` method that takes as argument a required

`user_id` integer and arbitrary keyword arguments, and returns `None`.

The method will use `find_user_by` to locate the user to update, then will update the user's attributes as passed in the method's arguments then commit changes to the database.

If an argument that does not correspond to a user attribute is passed, raise a `ValueError`.

```
bob@dylan:~$ cat main.py
#!/usr/bin/env python3
"""
Main file
Projects(/projects/current)

from db import DB
from user import User
QA Reviews I can make(/corrections/to_review)

from sqlalchemy.exc import InvalidRequestError
from sqlalchemy.orm.exc import NoResultFound
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

my_db = DB()

email = 'test@test.com'
Curriculum(/dashboards/my_curriculums)
hashed_password = "hashedPwd"

user = my_db.add_user(email, hashed_password)
Concepts(/concepts)
print(user.id)

try:
    Conference rooms(/dashboards/video_rooms)
    my_db.update_user(user.id, hashed_password='NewPwd')
    print("Password updated")
except ValueError:
    Servers(/servers)
    print("Error")

bob@dylan:~$ python3 main.py
Sandboxes(/user_containers/current)
Password updated
bob@dylan:~$
Tools(/dashboards/my_tools)
```

**Repo:** Video on demand(/dashboards/videos)

- GitHub repository: `alx-backend-user-data`
- Directory: `0x03-user_authentication_service`

**File:** `db.py`  
Peers(/users/peers)

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

[Check submission](#)

[Get a sandbox](#)

[View results](#)

## 4. Hash password

mandatory

Score: 76.67% (Checks completed: 100.0%)

My Profile(/users/my\_profile)



In this task you will define a `_hash_password` method that takes in a `password` string arguments and returns bytes.

The returned bytes is a salted hash of the input password, hashed with `bcrypt.hashpw`.

```
bob@dylan:~$ cat main.py
#!/usr/bin/env python3
"""
Main file
My Planning(/planning/me)
"""
from auth import _hash_password

Projects(/projects/current)
print(_hash_password("Hello Holberton"))

bob@dylan:~$ python3 main.py
QA Reviews I can make(/corrections/to_review)
b'$2b$12$eUDdeuBtrD41c8dXvzh95ehsWYCCA14VH1JbESzgbgZT.eMMzi.G2'
bob@dylan:~$
```

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

#### Repo:

- GitHub repository: `alx-backend-user-data`
- Directory: `Curriculums(/dashboards/my_curriculums)`
- Directory: `0x03-user_authentication_service`
- File: `auth.py`

Concepts(/concepts)

Check submission

> Get a sandbox

View results

Conference rooms(/dashboards/video\_rooms)

### 5. Register user

mandatory

Servers(/servers)  
Score: 82.5% (Checks completed: 100.0%)

> In this task, you will implement the `Auth.register_user` in the `Auth` class provided below:

```
from db import DB
Tools(/dashboards/my_tools)

class Auth:
    """Auth class to interact with the authentication database.
    Video on demand(/dashboards/videos)
    """

    def __init__(self):
        self._db = DB()
        Peers(/users/peers)
```

Note that `Auth._db` is a private property and should NEVER be used from outside the class.

Discord(<https://discord.com/app>)  
`Auth.register_user` should take mandatory `email` and `password` string arguments and return a `User` object.

If a user already exist with the passed email, raise a `ValueError` with the message `User <user's email> already exists`.

My Profile(/users/my\_profile)



If not, hash the password with `_hash_password`, save the user to the database using `self._db` and return the User object.

```
(/)  
bob@dylan: $ cat main.py  
#!/usr/bin/env python3  
""" Home(/)  
Main file  
"""  
from MyPlanning(Planning/me)  
  
email = 'me@me.com'  
password = 'password' (password/Pwd)  
  
auth = Auth()  
QA Reviews I can make(/corrections/to_review)  
try:  
    user = auth.register_user(email, password)  
    print("successfully created a new user!")  
    Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)  
except ValueError as err:  
    print("could not create a new user: {}".format(err))  
  
try:  
    Curriculums(/dashboards/my_curriculums)  
    user = auth.register_user(email, password)  
    print("successfully created a new user!")  
except ValueError as err:  
    Concepts(/concepts)  
    print("could not create a new user: {}".format(err))  
  
bob@dylan:~$ python3 main.py  
successfully created a new user!  
could not create a new user: User me@me.com already exists  
bob@dylan:~$  
Servers(/servers)
```

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

- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- Tools(/dashboards/my\_tools)
- File: auth.py

Video on demand(/dashboards/videos)  
Check submission > Get a sandbox View results

## 6. Basic Flask app

mandatory

Peers(/users/peers)

Score: 79.0% (Checks completed: 100.0%)

Discord(<https://discord.com/app>)  
In this task, you will set up a basic Flask app.

Create a Flask app that has a single GET route ( "/" ) and use `flask.jsonify` to return a JSON payload of the form:



```
{"message": "Bienvenue"}  
My Profile(/users/my_profile)
```




Add the following code at the end of the module:

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

 Home(/)

### Repo:

-  [MyBlindBox \(Planning\)](#)
  - Directory: 0x03-user\_authentication\_service
-  [Projects \(/projects/current\)](#)


 Check submission  Get a sandbox  View results  
QA Reviews I can make (/corrections/to\_review)

## 7. Register user


mandatory

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


Score: 82.5% (Checks completed: 100.0%)

 In this task, you will implement the end-point to register a user. Define a `users` function that implements the `POST /users` route.

Import the `Auth` object and instantiate it at the root of the module as such:

 Concepts (/concepts)


```
from auth import Auth
```

 Conference rooms (/dashboards/video\_rooms)


```
AUTH = Auth()
```

 Servers (/servers)


The end-point should expect two form data fields: "email" and "password". If the user does not exist, the end-point should register it and respond with the following JSON payload:

 Sandboxes (/user\_containers/current)  

```
{ "email": "<registered email>", "message": "user created" }
```

 If the user is already registered, catch the exception and return a JSON payload of the form

```
{ "message": "email already registered" }
```

 Video on demand (/dashboards/videos)  
and return a 400 status code

Remember that you should only use `AUTH` in this app. `DB` is a lower abstraction that is proxied by `Auth`.

 Terminal 1: Peers (/users/peers)

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



My Profile (/users/my\_profile)

```
bob@dylan:~$ python3 app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
* Home(/)
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
My Planning(/planning/me)
```

Terminal 2:  
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 -XPOST localhost:5000/users -d 'email=bob@me.com' -d 'password=
mySuperPwd' -v
Note: Unnecessary use of -X or --request, POST is already inferred.
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 5000 (#0)
> POST /users HTTP/1.1
> Host: localhost:5000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Length: 40
> Content-Type: application/x-www-form-urlencoded
* upload completely sent off: 40 out of 40 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: application/json
< Content-Length: 53
< Server: Werkzeug/1.0.1 Python/3.7.3
< Date: Wed, 19 Aug 2020 00:03:18 GMT
<
{"email":"bob@me.com","message":"user created"}
Curriculums(/dashboards/my_curriculums)
bob@dylan:~$
bob@dylan:~$ curl -XPOST localhost:5000/users -d 'email=bob@me.com' -d 'password=
mySuperPwd' -v
Note: Unnecessary use of -X or --request, POST is already inferred.
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 5000 (#0)
> POST /users HTTP/1.1
> Host: localhost:5000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Type: application/x-www-form-urlencoded
* upload completely sent off: 40 out of 40 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 400 BAD REQUEST
< Content-Type: application/json
< Content-Length: 39
< Server: Werkzeug/1.0.1 Python/3.7.3
< Date: Wed, 19 Aug 2020 00:03:33 GMT
<
{"message": "email already registered"}
bob@dylan:~$
```

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

## Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File: my\_profile (/users/my\_profile)



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

## 8. Credentials validation

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

Score: 49.17% (Checks completed: 66.67%)

[My Planning\(/planning/me\)](#)

this task, you will implement the `Auth.valid_login` method. It should expect email and password required arguments and return a boolean.

[Try locating the user by email. If it exists, check the password with `bcrypt.checkpw`. If it matches return `True`. In any other case, return `False`.](#)

Project's project's current

[QA Reviews can make \(/corrections/to\\_review\)](#)

```
#!/usr/bin/env python3
"""
```

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

```
"""
```

```
from auth import Auth
```



```
email = 'bob@bob.com'
password = 'MyPwDorBob'
auth = Auth()
```

[Concepts\(/concepts\)](#)

```
auth.register_user(email, password)
```



```
print(auth.valid_login(email, password))
Conference rooms(/dashboards/video_rooms)
print(auth.valid_login(email, "WrongPwd"))
```

[Servers\(/servers\)](#)

```
print(auth.valid_login("unknown@email", password))
```



```
bob@dylan:~$ python3 main.py
True Sandboxes(/user_containers/current)
```

```
False
```

```
False
```



```
bob@dylan:~/dashboards/my_tools)
```

[Video on demand\(/dashboards/videos\)](#)**Repo:**

- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File peers(/users/peers)

[Discord \(/https://discord.com/app\)](#)[Check submission](#)[Get a sandbox](#)[View results](#)

## 9. Generate UUIDs

[My Profile\(/users/my\\_profile\)](#)**mandatory**

Score: 16.61% (Checks completed: 100.0%)



In this task you will implement a `_generate_uuid` function in the `auth` module. The function should return a string representation of a new UUID. Use the `uuid` module.



Note that the method is private to the `auth` module and should **NOT** be used outside of it.



Repo: My Planning(/planning/me)

- GitHub repository: `alx-backend-user-data`
- Directory: `0x03-user_authentication_service`
- File: `auth.py`



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

Check submission

> Get a sandbox

View results



10. Get session ID (/dashboards/my\_current\_evaluation\_quizzes)

mandatory

Score: 82.5% (Checks completed: 100.0%)



Curriculums(/dashboards/my\_curriculums)

In this task, you will implement the `Auth.create_session` method. It takes an `email` string argument and returns the session ID as a string.



Concepts(/concepts)

The method should find the user corresponding to the email, generate a new UUID and store it in the database as the user's `session_id`, then return the session ID.



Remember that only public methods of `server.py` can be used.



bob@dylan:~\$ cat main.py



#!/usr/bin/env python3

"""

Main file



""" Sandboxes(/user\_containers/current)

from auth import Auth



email Tools(/dashboards/my\_tools)

password = 'MyPwdOfBob'

auth = Auth()



Video on demand(/dashboards/videos)

auth.register\_user(email, password)

print(auth.create\_session(email))



print(auth.create\_session("unknown@email.com"))

Peers(/users/peers)

bob@dylan:~\$ python3 main.py

5a006849-343e-4a48-ba4e-bbd523fcca58



None Discord(https://discord.com/app)

bob@dylan:~\$



Repo:

- My Profile(/users/my\_profile)
- GitHub repository: `alx-backend-user-data`

- Directory: 0x03-user\_authentication\_service



- File: auth.py

(/)

Check submission

>\_ Get a sandbox

View results



Home(/)

## 11. Log in

mandatory



My Planning(/planning/me)

Score: 82.5% (*Checks completed: 100.0%*)



Projects(/projects/current)

In this task, you will implement a `login` function to respond to the `POST /sessions` route.

The request is expected to contain form data with `"email"` and a `"password"` fields.



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

If the login information is incorrect, use `flask.abort` to respond with a 401 HTTP status.



Otherwise, create a new session for the user, store it the session ID as a cookie with key `"session_id"` on the response and return a JSON payload of the form

```
{"email": "<user email>", "message": "logged in"}
```



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 -XPOST localhost:5000/users -d 'email=bob@bob.com' -d 'password=mySuperPwd'

{ "email": "bob@bob.com", "message": "user created" }

bob@dylan:~\$ curl -XPOST localhost:5000/sessions -d 'email=bob@bob.com' -d 'password=mySuperPwd' -v

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

\* Trying 127.0.0.1...

\* TCP\_NODELAY set

\* Connected to localhost (127.0.0.1) port 5000 (#0)

> POST /sessions HTTP/1.1

> Host: localhost:5000

> User-Agent: curl/7.58.0

> Accept: \*/\*

> Content-Type: application/x-www-form-urlencoded

>

\* upload completely sent off: 37 out of 37 bytes

\* HTTP 1.0, assume close after body

< HTTP/1.0 200 OK

< Content-Type: application/json

< Content-Length: 46

< Set-Cookie: session\_id=163fe508-19a2-48ed-a7c8-d9c6e56fabd1; Path=/  
Curriculums(/dashboards/my\_curriculums)

< Server: Werkzeug/1.0.1 Python/3.7.3

< Date: Wed, 19 Aug 2020 00:12:34 GMT

<

{ "email": "bob@bob.com", "message": "logged in" }

\* Closing connection 0

bob@dylan:~\$ curl -XPOST localhost:5000/sessions -d 'email=bob@bob.com' -d 'password=BlaBla' -v

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

\* Trying 127.0.0.1...

\* TCP\_NODELAY set

\* Connected to localhost (127.0.0.1) port 5000 (#0)

> POST /sessions HTTP/1.1

> Host: localhost:5000

> User-Agent: curl/7.58.0

> Accept: \*/\*

> Content-Length: 34

> Content-Type: application/x-www-form-urlencoded

>

\* upload completely sent off: 34 out of 34 bytes

\* HTTP 1.0, assume close after body

< HTTP/1.0 401 UNAUTHORIZED

< Content-Type: text/html; charset=utf-8

< Content-Length: 338

< Server: Werkzeug/1.0.1 Python/3.7.3

< Date: Wed, 19 Aug 2020 00:12:45 GMT

<

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

<title>401 Unauthorized</title>


<h1>Unauthorized</h1>

<p>The server could not verify that you are authorized to access the URL requested. You either supplied the wrong credentials (e.g. a bad password), or your brows

er doesn't understand how to supply the credentials required.</p>

\* Closing connection 0


bob@dylan:~\$  
(/)

 Repo: Home(/)

- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File: app.py

 Projects(/projects/current)

Check submission

 Get a sandbox

View results

 QA Reviews I can make(/corrections/to\_review)  
**12. Find user by session ID**

mandatory


? Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)  
Score: 79.0% (Checks completed: 100.0%)

In this task, you will implement the `Auth.get_user_from_session_id` method. It takes a single `session_id` string argument and returns the corresponding `User` or `None`.

 Curriculums(/dashboards/my\_curriculums)


If the session ID is `None` or no user is found, return `None`. Otherwise return the corresponding user.

 Remember to only use public methods of `self._db`.  
Concepts(/concepts)


 Repo: Conference rooms(/dashboards/video\_rooms)

- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File: auth.py

 Servers(/servers)

 Sandboxes(/user\_containers/current)

Check submission

 Get a sandbox

View results

 Tools(/dashboards/my\_tools)

 **13. Destroy session**  
Video on demand(/dashboards/videos)

mandatory

Score: 88.33% (Checks completed: 100.0%)

 Peers(/users/peers)

In this task, you will implement `Auth.destroy_session`. The method takes a single `user_id` integer argument and returns `None`.

 Disorders(/disorders/pom/impl)  
The method updates the corresponding user's session ID to `None`.

Remember to only use public methods of `self._db`.



Repo:

- My Profile(/users/my\_profile)
- GitHub repository: alx-backend-user-data

- Directory: 0x03-user\_authentication\_service



- File: auth.py

(/)

Check submission

>\_ Get a sandbox

View results



Home(/)

## 14. Log out

mandatory



My Planning(/planning/me)

Score: 82.5% (Checks completed: 100.0%)



Projects(/projects/current)

In this task, you will implement a `logout` function to respond to the `DELETE /sessions` route.

The request is expected to contain the session ID as a cookie with key `"session_id"`.



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

Find the user with the requested session ID. If the user exists destroy the session and redirect the user to `GET /`. If the user does not exist, respond with a 403 HTTP status.



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

### Repo:

- GitHub repository: alx-backend-user-data
- Curriculum: (/dashboards/my\_curriculums)
- Directory: 0x03-user\_authentication\_service
- File: app.py



Concepts(/concepts)

Check submission

>\_ Get a sandbox

View results



Conference rooms(/dashboards/video\_rooms)



Servers(/servers)

## 15. User profile

mandatory



Sandboxes(/user\_containers/current)

Score: 82.5% (Checks completed: 100.0%)



In this task, you will implement a `profile` function to respond to the `GET /profile` route.

Tools(/dashboards/my\_tools)

The request is expected to contain a `session_id` cookie. Use it to find the user. If the user exist, respond with a 200 HTTP status and the following JSON payload:



Video on demand(/dashboards/videos)

```
{"email": "<user email>"}
```

If the session ID is invalid or the user does not exist, respond with a 403 HTTP status.



Peers(/users/peers)



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



My Profile(/users/my\_profile)

```
bob@dylan:~$ curl -XPOST localhost:5000/sessions -d 'email=bob@bob.com' -d 'password=mySuperPwd' -v
Note: Unnecessary use of -X or --request, POST is already inferred.
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 5000 (#0)
> POST /sessions HTTP/1.1
> Host: localhost:5000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Length: 37
> Content-Type: application/x-www-form-urlencoded
* upload completely sent off: 37 out of 37 bytes
* HTTP/1.0 200 OK
< HTTP/1.0 200 OK
< Content-Type: application/json
< Content-Length: 46
< Set-Cookie: session_id=75c89af8-1729-44d9-a592-41b5e59de9a1; Path=/
< Server: Werkzeug/1.0.1 Python/3.7.3
< Date: Wed, 19 Aug 2020 00:15:57 GMT
<
{"email": "bob@bob.com", "message": "logged in"}
* Closing connection 0
bob@dylan:~$
bob@dylan:~$ curl -XGET localhost:5000/profile -b "session_id=75c89af8-1729-44d9-a592-41b5e59de9a1"
{"email": "bob@bob.com"}
bob@dylan:~$
bob@dylan:~$ curl -XGET localhost:5000/profile -b "session_id=nope" -v
Note: Unnecessary use of -X or --request, GET is already inferred.
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 5000 (#0)
> GET /profile HTTP/1.1
> Host: localhost:5000
> User-Agent: curl/7.58.0
> Accept: */*
> Cookie: session_id=75c89af8-1729-44d9-a592-41b5e59de9a1
>
* HTTP/1.0 403 FORBIDDEN
< HTTP/1.0 403 FORBIDDEN
< Content-Type: text/html; charset=utf-8
< Content-Length: 234
< Server: Werkzeug/1.0.1 Python/3.7.3
< Date: Wed, 19 Aug 2020 00:16:43 GMT
<
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>403 Forbidden</title>
<h1>Forbidden</h1>
<p>You don't have the permission to access the requested resource. It is either read-protected or not readable by the server.</p>
* Closing connection 0

bob@dylan:~$
```



Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File: app.py



Check submission

Get a sandbox

View results



## 16. Generate reset password token

mandatory

Projects(/projects/current)



Score: 76.67% (Checks completed: 100.0%)

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

In this task, you will implement the `Auth.get_reset_password_token` method. It take an `email` string argument and returns a string.

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

Find the user corresponding to the email. If the user does not exist, raise a `ValueError` exception. If it exists, generate a UUID and update the user's `reset_token` database field. Return the token.



Curriculums(/dashboards/my\_curriculums)

Repo:



- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File: auth.py



Conference rooms(/dashboards/video\_rooms)



Check submission

Get a sandbox

View results

Servers(/servers)



Sandboxes(/user\_containers/current)

## 17. Get reset password token

mandatory



Tools(/dashboards/my\_tools)

Score: 82.5% (Checks completed: 100.0%)



Video rooms(/dashboards/video\_rooms)

In this task, you will implement the `Auth.get_reset_password_token` function to respond to the `POST /reset_password` route.

The request is expected to contain form data with the `"email"` field.



Users(/users/people)

If the email is not registered, respond with a 403 status code. Otherwise, generate a token and respond with a 200 HTTP status and the following JSON payload:



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

```
{ "email": "user_email", "reset_token": "<reset token>" }
```



Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service

- File: app.py



(/)

Check submission

> Get a sandbox

View results



Home(/)

mandatory



Score: 100.0% (1/1 checks completed)



In this task, you will implement the `Auth.update_password` method. It takes `reset_token` string argument and a `password` string argument and returns `None`.



Use the `reset_token` to find the corresponding user. If it does not exist, raise a `ValueError` exception.

Otherwise, hash the password and update the user's `hashed_password` field with the new hashed password and the `reset_token` field to `None`.



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

Repo:



- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File: auth.py



Concepts(/concepts)

Check submission

> Get a sandbox

View results



Conference rooms(/dashboards/video\_rooms)



Servers(/servers)

mandatory



Score: 79.0% (Checks completed: 100.0%)



In this task you will implement the `update_password` function in the `app` module to respond to the `PUT /reset_password` route.



The request is expected to contain form data with fields `"email"`, `"reset_token"` and `"new_password"`.

Update the password. If the token is invalid, catch the exception and respond with a 403 HTTP code.



If the token is valid, respond with a 200 HTTP code and the following JSON payload:

```
{"email": "<user email>", "message": "Password updated"}
```



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

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x03-user\_authentication\_service
- File: my\_profile.py



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

## 20. End-to-end integration test

#advanced

[Home\(/\)](#)

Score: 100.0% (Checks completed: 100.0%)



Start your app. Open a new terminal window.

[My Planning\(/planning/me\)](#)

Create a new module called `main.py`. Create one function for each of the following tasks. Use the

[Projects\(/projects/current\)](#)

requests module to query your web server for the corresponding end-point. Use `assert` to validate the response's expected status code and payload (if any) for each task.



- `register_user(email: str, password: str) -> None`
- `log_in_wrong_password(email: str, password: str) -> None`
- `log_in(email: str, password: str) -> str`



- `profile_logged(session_id: str) -> None`
- `log_out(session_id: str) -> None`
- `reset_password_token(email: str) -> str`



- `update_password(email: str, reset_token: str, new_password: str) -> None`

Then copy the following code at the end of the `main` module:

[Concepts\(/concepts\)](#)

EMAIL = "guillaume@holberton.io"

PASSWD = "b4l0u"



NEW\_PASSWD = "b4r0n1f10t13"

[Conference rooms\(/dashboards/video\\_rooms\)](#)

if \_\_name\_\_ == "\_\_main\_\_":

register\_user(EMAIL, PASSWD)

log\_in\_wrong\_password(EMAIL, NEW\_PASSWD)

profile\_unlogged()

session\_id = log\_in(EMAIL, PASSWD)

profile\_logged(session\_id)

log\_out(session\_id)

reset\_token = reset\_password\_token(EMAIL)

update\_password(EMAIL, reset\_token, NEW\_PASSWD)

log\_in(EMAIL, NEW\_PASSWD)

Run `python main.py`. If everything is correct, you should see no output.

[Peers\(/users/peers\)](#)**Repo:**

- `Directory: 0x03-user_authentication_service`
- `File: main.py`

[Check submission](#)[Get a sandbox](#)[View results](#)[My Profile\(/users/my\\_profile\)](#)





(/)

Copyright © 2024 ALX, All rights reserved.



Home(/)



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)