LaSalle College
Montréal

Course Identification

Name of programs – Codes:	COMPUTER SCIENCE TECHNOLOGY– PROGRAMMING (420.BP) INFORMATION TECHNOLOGY PROGRAMMERANALYST (LEA.3Q)
Course title:	**SCRIPTING LANGUAGES**
Course number:	420-LS3-AS
Group:	7178
Teachers' names:	Renan Cavalcanti
Duration:	8 hours
Semester:	Fall 2023

Standard of the Evaluated Competencies

Statement of the evaluated competency – Code

Use programming languages - 00Q2

Evaluated elements of the competencies

1. Analyze the problem.
2. Translate the algorithm into a programming language.
3. Debug the code.
4. Implement the functional test plan.

Important Dates

1. November 17 – Project Implementation
2. November 20 – Project Implementation
3. November 24 - Project Implementation
4. November 27 - Project Implementation (Deadline)

Project Python

Your task for the project will be implement an entire backend application by creating an API using Flask framework and MongoDB.

Project requirements

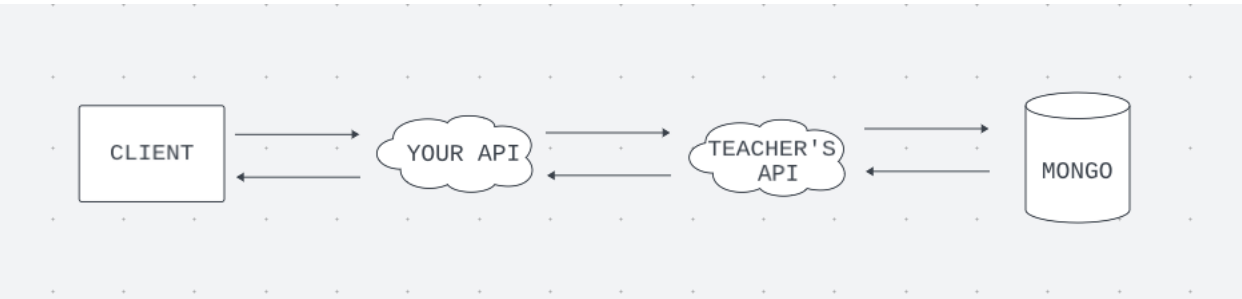
1. Endpoints:

   You will need to implement the following endpoints in your new verb view:

   - Get a Verb: “/verbs/” -> method: GET
   - Get Random Verbs: “/verbs /random/” -> method: GET
   - Add a favorite : “/verbs /favorites/“-> method: POST
   - Get one favorite “/verbs /favorites/<favoriteUid> /” -> method: GET
   - Get all favorites “/verbs /favorites/” -> method: GET
   - Delete a favorite : “/verbs /favorites/<favoriteUid> “ -> method: DELETE

To be able to access all the requests the user must have a valid token from the login, so you will start the project from the app built in class.

Some endpoints we will use a third party API from our API.



1.1)

Get a verb endpoint will be requested from another API .

First you will need to install “**pip install requests**”.

Import the library using “import requests” at the top of your file.

The view should check if the token is valid using the function that we created in class “validate_token”

In case of the function returns 400:

```
return jsonify({"error": 'Token is missing in the request, please try again'}), 401
```

In case of the function returns 401:

```
return jsonify({"error": 'Invalid authentication token, please login again'}), 403
```

If the token is OK, you will need to check if inside the body there is the key “verb”, this must be the verb that the user is looking for.

To make the request to the other API you must use the following code below:

```
external_api_url = 'https://lasalle-frenchverb-api-afpnl.ondigitalocean.app/v1/api/verb'
    response = requests.get(external_api_url, headers={'token':
'278ef2169b144e879aec4f48383dce28e654a009cacf46f8b6c03bbc9a4b9d11'}, json={'verb':
'habiter'})
```

This is one example using the verb “habiter”, of course your request must be dynamic. You must take the verb from the body of the request and use it instead of the constant “habiter”

The expected return for this function is a json with the key “verb” and the value **response.json()**

```
return jsonify({"verb": response.json()})
```

Note that because you are calling another API you must check if the status of the response is 200, you can check it using:

```
response.status_code == 200:
```

If the response is not 200, you should return the error message from the other API inside a key called “error”.

```
return jsonify({"error": response.json()["errorMessage"]})
```

Please, put your code inside try / except

1.2)

Get a random verb endpoint will be requested from another API .

The view should check if the token is valid using the function that we created in class “validate_token”

In case of the function returns 400:

```
return jsonify({"error": 'Token is missing in the request, please try again'}), 401
```

In case of the function returns 401:

```
return jsonify({"error": 'Invalid authentication token, please login again'}), 403
```

If the token is OK, you will need to check if inside the body there is the key “quantity”, this must be the amount of random verbs to be returned.

To make the request to the other API you must use the following code below:

```
external_api_url = 'https://lasalle-frenchverb-api-
afpnl.ondigitalocean.app/v1/api/verb/random'
    response = requests.get(external_api_url, headers={'token':
'278ef2169b144e879aec4f48383dce28e654a009cacf46f8b6c03bbc9a4b9d11'}, json={'quantity': 5})
```

This is one example using the quantity “5”, of course your request must be dynamic. You must take the quantity from the body of the request and use it instead of the constant “5”

The expected return for this function is a json with the key “verbs” and the value **response.json()**

```
return jsonify({"verb": response.json()})
```

Note that because you are calling another API you must check if the status of the response is 200, you can check it using:

```
response.status_code == 200:
```

If the response is not 200, you should return the error message from the other API inside a key called “error”.

```
return jsonify({"error": response.json()["errorMessage"]})
```

Please, put your code inside try / except

1.3)

Add a favorite verb will save inside your MongoDB one verb.

To do that, you will need to validate if the token is in the request by using the function validate_token:

In case of the function returns 400:

```
return jsonify({"error": 'Token is missing in the request, please try again'}), 401
```

In case of the function returns 401:

```
return jsonify({"error": 'Invalid authentication token, please login again'}), 403
```

Your endpoint must check if there is inside the body the key called “verb”

If there is no this key return the error to the user.

From the token, take the user ID and you must save the new favorite verb in the new database collection “verbs” with the following structure:

{
“owner”: ID from the token inside request
“verb”: The verb inside the body of the request
}

1.4)

Get a favorite verb will return a verb from your MongoDB verbs collection.

To do that, you will need to validate if the token is in the request by using the function validate_token:

In case of the function returns 400:

```
return jsonify({"error": 'Token is missing in the request, please try again'}), 401
```

In case of the function returns 401:

```
return jsonify({"error": 'Invalid authentication token, please login again'}), 403
```

Your endpoint must receive the ID of the verb saved inside the collection. We saw that every time that we create a new object inside Mongo, the object has an unique ID.

This ID must be received from the URL of the request, for example:

You must add the parameter after the route, and use the same name in the parameter of the function for this endpoint.

Using this UID, you can find inside the mongo the favorite verb that the user is trying to access.

This endpoint must return the entire verb information in a json format inside a key called “verb

1.5)

Get all favorites verbs will return all verbs for the user that is making the request.

To do that, you will need to validate if the token is in the request by using the function validate_token:

In case of the function returns 400:

```
return jsonify({"error": 'Token is missing in the request, please try again'}), 401
```

In case of the function returns 401:

```
return jsonify({"error": 'Invalid authentication token, please login again'}), 403
```

You must now take the information from the function validate_token, we saw in class that after decode we will have access to the User ID that is making the request.

When we save the favorites verbs, we are adding a key called owner that is exactly the the ID for the person that is saving the verb as favorite.

So, this endpoint must return all the favorite verbs for the user that is making the request.

1.6)

Delete a favorite verb will delete a favorite verb from the MongoDB.

To do that, you will need to validate if the token is in the request by using the function validate_token:

In case of the function returns 400:

```
return jsonify({"error": 'Token is missing in the request, please try again'}), 401
```

In case of the function returns 401:

```
return jsonify({"error": 'Invalid authentication token, please login again'}), 403
```

Your endpoint must receive the ID of the verb saved inside the collection. We saw that every time that we create a new object inside Mongo, the object has a unique ID.

This ID must be received from the URL of the request, for example:

Using this ID from the request, we can access the Mongo and delete the verb.

You can use the function delete_one() from pyMongo to delete the object from MongoDB.

In case of the deletion is succeed, return the amount of verbs affected:

```
{
    "verbs_affected": 1
}
```

You can take this information from the Mongo response when you delete the verb:

```
return jsonify({'verbs_affected': verbDeleteAttempt.deleted_count}), 200
```

GOOD
LUCK


It always seems
impossible until
it's done.
Nelson Mandela
BrainyQuote