

## Emilia Code Fix

### Task-1:

"👤: The first step of a meaningful conversation is a friendly greeting. Therefore, your first task will be to improve our greeting functionality at `/task1/greet/{name}`. Currently, this route is only available in English GB. But not all of our customers speak English. Could you add a query parameter `language` to also support German DE and Spanish ES? Valid values should be `de`, `en`, and `es`! In case the user passes a different language, we should respond, that we don't speak it. Just go to `emilia.py` and implement the requested changes. The exact requirements for all tasks are defined by the test in `test_emilia.py`. Run `pytest -xsk task1` to verify you solved this task correctly ✓.\n"

### Task Solution:

Now in the first task, Emilia greets the users in 3 available languages so the code is added using the 'if-else' statements, where a return statement is given for each of the languages selected by the user and also there is a default case.

The code passes all the test cases given, using the command `poetry run pytest -xsk task1`, as the code is fixed for the `emilia.py` python file, the output is given as:

```
Terminal: Local x Local (2) x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\katya\python-challenge> poetry run pytest -xsk task1
===== test session starts =====
platform win32 -- Python 3.9.9, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\katya\python-challenge, configfile: pyproject.toml
plugins: anyio-3.4.0, asyncio-0.16.0, depends-1.0.1
collected 17 items / 15 deselected / 2 selected

test_emilia.py . ♦ Congratulations! You solved the first task. Go to '/task2' to solve the next one.
.
===== 2 passed, 15 deselected in 1.44s =====
PS C:\Users\katya\python-challenge> 
```

The Task-1 browser output for the FastAPI application is given as,

```
← → ↺ ⓘ 127.0.0.1:8000/task1/greet/Felix
"Hallo Felix, ich bin Emilia."
```

---

### Task-2:

"👤: Sometimes best practices can be annoying. Python uses 'snake\_case' 🐍 for variables names, but from a typical API you might expect keys in 'camelCase' 🐪. Can you write a 'camelize' function that transforms a snake\_case-string to a camelCase-string? The boilerplate code is already there! This function will be exposed at `/task2/camelize` and expects a JSON object as payload.\n"

## Task Solution:

In this task the input string in `snake\_case` format changes to `camelCase`. To this the approach used is, splitting the string over the delimiter underscore and then joining the string by Capitalizing the starting letter of the word after the delimiter.

The code passes all the test cases given, using the command `poetry run pytest -xsk task2`, as the code is fixed for the 'emilia.py' python file, the output is given as:

```
Terminal: Local x Local (2) x + v
PS C:\Users\katya\python-challenge> poetry run pytest -xsk task2
===== test session starts =====
platform win32 -- Python 3.9.9, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\katya\python-challenge, configfile: pyproject.toml
plugins: anyio-3.4.0, asyncio-0.16.0, depends-1.0.1
collected 17 items / 15 deselected / 2 selected

test_emilia.py . ♦ Wow, keep going! Let's see if you solve the next one too. You can find it at `/task3`.
.

===== 2 passed, 15 deselected in 1.99s =====
PS C:\Users\katya\python-challenge> 
```

The task-2 output for the FastAPI application is given as,

The screenshot displays a REST client interface with the following details:

- Request body:** A JSON object `{"company_name": "Emilia"}`.
- Execute button:** A blue button to execute the request.
- Responses section:**
  - Curl:** `curl -X 'POST' \ 'http://127.0.0.1:8000/task2/camelize' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{"company_name": "Emilia"}'`
  - Request URL:** `http://127.0.0.1:8000/task2/camelize`
  - Server response:**
    - Code:** 200
    - Response body:** `{"companyName": "Emilia"}`
    - Response headers:**
      - `content-length: 24`
      - `content-type: application/json`
      - `date: Sat, 26 Feb 2022 22:44:29 GMT`
      - `server: uvicorn`
- Responses table:** A table with columns 'Code' and 'Description'.

### Task-3:

"👤: Our goal is to increase the quality of life of elderly people by encouraging them to have more contact with other people through a rich and varied daily schedule ✅. Therefore we must understand the intentions of our users. For example, a user might want to call a family member or wants to set a reminder to have some tea with an old friend. We have different handlers to respond to the various user actions. Unfortunately, our code is broken ☐ as it uses a random handler for every incoming action. On top of that our responses are very rude 🙄 at the moment. Therefore, the first part of your task is to implement an intent recognition mechanism, which forwards the incoming actions to the right action handler. Afterward, you need to implement these action handlers. Make sure the responses are friendlier than they are now ☐. As always the exact requirements are set by the tests in `test\_emilia.py`. The number of tests is slightly higher than in the previous tasks, but don't let that discourage you!\n"

### Task Solution:

Now in this task, the user actions are handled by Emilia, these actions can be related to calling, reminder, timer, and unknown actions. In each action, the action request is passed to which a message response is received as output. Here the input calls are managed by the handler as there are multiple functions under a single endpoint.

The code passes all the test cases given, using the command `poetry run pytest -xsk task3`, as the code is fixed for the `emila.py` python file, the output is given as:

```
Terminal: Local x Local (2) + v
PS C:\Users\katya\python-challenge> poetry run pytest -xsk task3
===== test session starts =====
platform win32 -- Python 3.9.9, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\katya\python-challenge, configfile: pyproject.toml
plugins: anyio-3.4.0, asyncio-0.16.0, depends-1.0.1
collected 17 items / 10 deselected / 7 selected

test_emilia.py ..... * This was really hard, congratulations! You're awesome 🥳 If you really wanna impress us, there is an optional bonus task at `task4`. But if you're short on time
you can already create a pull request at https://github.com/mit-emilia/hiring!
.

===== 7 passed, 10 deselected in 1.31s =====
PS C:\Users\katya\python-challenge>
```

The task-3 output for the FastAPI application is given as,

Request body required

application/json

{"username": "Matthias", "action": "Call my friend Sahar."}

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/task3/action' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{"username": "Matthias", "action": "Call my friend Sahar."}'
```

Request URL

```
http://127.0.0.1:8000/task3/action
```

Server response

Code

Details

200

Response body

```
{
  "message": "🔥 Calling Sahar ..."
}
```

Download

Response headers

```
content-length: 36
content-type: application/json
date: Sat, 26 Feb 2022 23:08:04 GMT
server: uvicorn
```

Responses

Code

Description

Links

## Task-4:

"(Bonus) 🧑: In our last task we have two users Stefan and Felix. Both of them have a secret stored in our database 🗄️. Currently, our application is not very secure! An imposter 🧑 could easily switch out the `username` field to pretend to be somebody else. Let's change that! Ideally, a secret should be accessible to the user 🧑 it belongs to. Therefore if a user 🧑 requests a secret from `/users/{username}/secret` route, we should verify that the user 🧑 is logged in and that the `username` matches. For the login process, we created some boilerplate code and a (not fully working) login route at `/task4/token`. Your task is to implement the user verification logic and to make sure to return the right HTTP status in case a user 🧑 is not authenticated or authorized. As always, check if you solved the task correctly by running `pytest -xsk task4`. Have fun, you're almost there 🏁!\n"

## Task Solution:

The code is fixed by modifying the login() function functionality where now the password is also checked using the `verify\_password` functionality and only those users that are registered and have valid login details can login to the system else the Emilia system raises an error. Now the system also verifies that the current\_user and login\_user both are the same and can only access their own secrets to which the function `get\_current\_user()` and `read\_user\_secret()` are modified in the code.

The code passes all the test cases given, using the command `poetry run pytest -xsk task4`, as the code is fixed in the `emila.py` python file, the output is given as:

```
Terminal: Local - Local (2) + v
PS C:\Users\katya\python-challenge> clear

PS C:\Users\katya\python-challenge> poetry run pytest -xsk task4
===== test session starts =====
platform win32 -- Python 3.9.9, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\katya\python-challenge, configfile: pyproject.toml
plugins: anyio-3.4.0, asyncio-0.16.0, depends-1.0.1
collected 17 items / 11 deselected / 6 selected

test_emila.py ..... 🎉 Amazing! This is really impressive. Here is your prize 🎁 We would love to get in touch with you 🎁 Therefore, create a pull request at https://github.com/mit-emilia/hiring!

===== 6 passed, 11 deselected in 2.70s =====
PS C:\Users\katya\python-challenge> 
```

The task-4 output for the FastAPI application is given as,

POST

/task4/token

Allows registered users to obtain a bearer token.

Parameters

No parameters

Request body

application/x-www-form-urlencoded

grant\_type

string

pattern: password

☒ Send empty value

username

string

password

string

☒ Send empty value

scope

string

☒ Send empty value

client\_id

string

☒ Send empty value

client\_secret

string

☒ Send empty value

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/task4/token' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'grant_type=password&username=felix&password=elm-javascript&scope=&client_id=&client_secret='
```

Request URL

http://127.0.0.1:8000/task4/token

Server response

Code

Details

200

Response body

```
{
  "access_token": "eyJ3bG6c10L3JlZl1h1s1nR5c161kpWVCJ9.eyJ2Zm4lI0I3m2NpcC1s1nR4c1RNTYWN1k1NTQzQ000.E3P7E5ctDkCnL1F-1E96ApCIL-CT7uhsFcq5SA0s",
  "token_type": "bearer"
}
```

Download

Response headers

```
content-length: 145
content-type: application/json
date: Sun, 27 Feb 2022 08:13:53 GMT
server: uvicorn
```

Now all the 4 tasks are done. So, the test cases are checked for the complete code by running the command `poetry run pytest` so this runs all the test cases for the Emilia and the result for this is,

```
Terminal: Local x Local (2) x Local (3) x + x
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\katya\python-challenge> poetry run pytest
===== test session starts =====
platform win32 -- Python 3.9.9, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: C:\Users\katya\python-challenge, configfile: pyproject.toml
plugins: anyio-3.4.0, asyncio-0.16.0, depends-1.0.1
collected 17 items

test_emilia.py ..... [100%]

===== 17 passed in 2.87s =====
PS C:\Users\katya\python-challenge>
```

All the test cases are passed for the Code. So, the code is fixed for Emilia. (Also added comments to the code for better understanding of the code)