Schola - Scholarships For Everyone API

IT325 Web Services Final Project

by

Oussema Hassine

January 2023

BA/IT Senior Student

Information Technology Minor

Tunis Business School

Ben Arous,TUNISIA.

2022-2023

Declaration of Academic Ethics

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: January 8th, 2023 Oussema Hassine

Abstract

The availability of scholarships can greatly impact a student's ability to pursue higher education. However, finding and applying for scholarships can be a time-consuming and daunting task, especially with the vast number of options out there. That's where our Scholar comes in.

Our API simplifies the scholarship search process by allowing users to easily, generate the latest scholarships, access a wide range of its information with just a few simple queries. Using preferences specified by the user, such as field of study ,degree or location, the API returns a list of relevant scholarships, along with details on eligibility requirements and application deadlines.

Keywords - API, Provider, Scholarships.

Contents

Chapter 1

Introduction

My final Project for the IT325 Web Services course this semester consists of a RESTful API developed using Flask Python and Beautiful soup. The resources were stored thanks to the implementation of sqlite3.
I have also used technologies such as Postman, VSCode, and Git-Version Control to maximize the project's quality.
This project contains all the CRUD Operations, secured with session authentication and hashing passwords. Multiple Postman Snippets were also used in order to test this project's efficiency.
This projects aim is to facilitate the information sharing of scholarships for students or any scholarship seeker.
Its main function is returning a scholarship based on criteria set by the user and before its deadline. It generates an updated database of scholarships and it can send it the user via email.

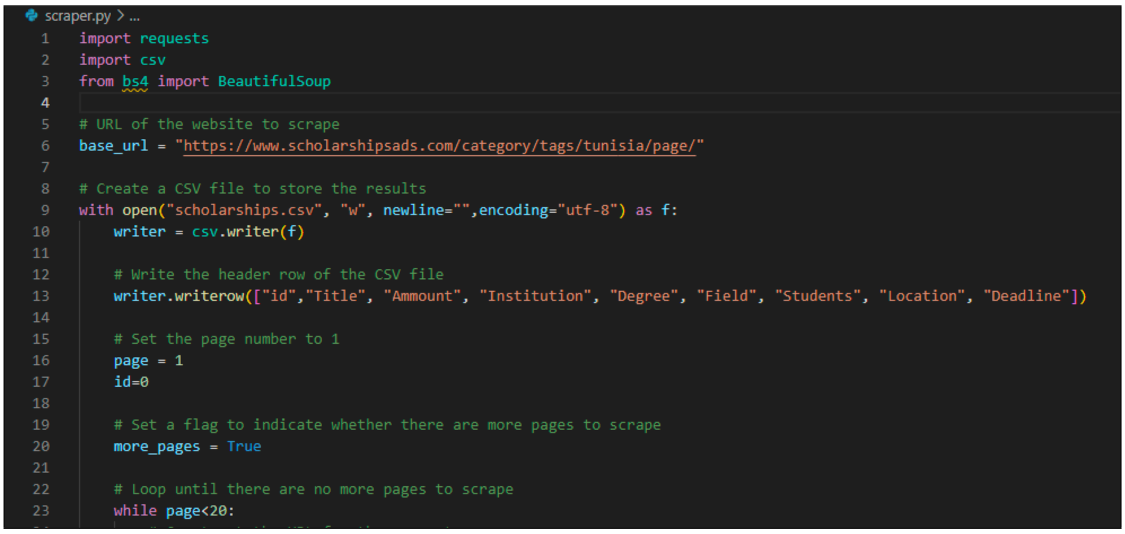
Chapter 2

Explanation of the work carried out

2.1 BeautifulSoup/Requests Contribution

Beautiful Soup is a Python library that is used for web scraping purposes. It allows a programmer to extract data from a website by parsing the HTML or XML of a webpage. Beautiful Soup provides a number of useful methods and attributes to navigate, search, and modify the parse tree, making it easier to extract the data you're interested in. [1] The requests library is a popular Python library for making HTTP requests. It allows you to send HTTP requests using Python, and it provides a number of useful methods for handling the response [2].

I have used BeautifulSoup/Requests to generate a .csv file that contains the scholarships from the latest 20 pages from ScholarshipsAds [**?**]
Implementation code :

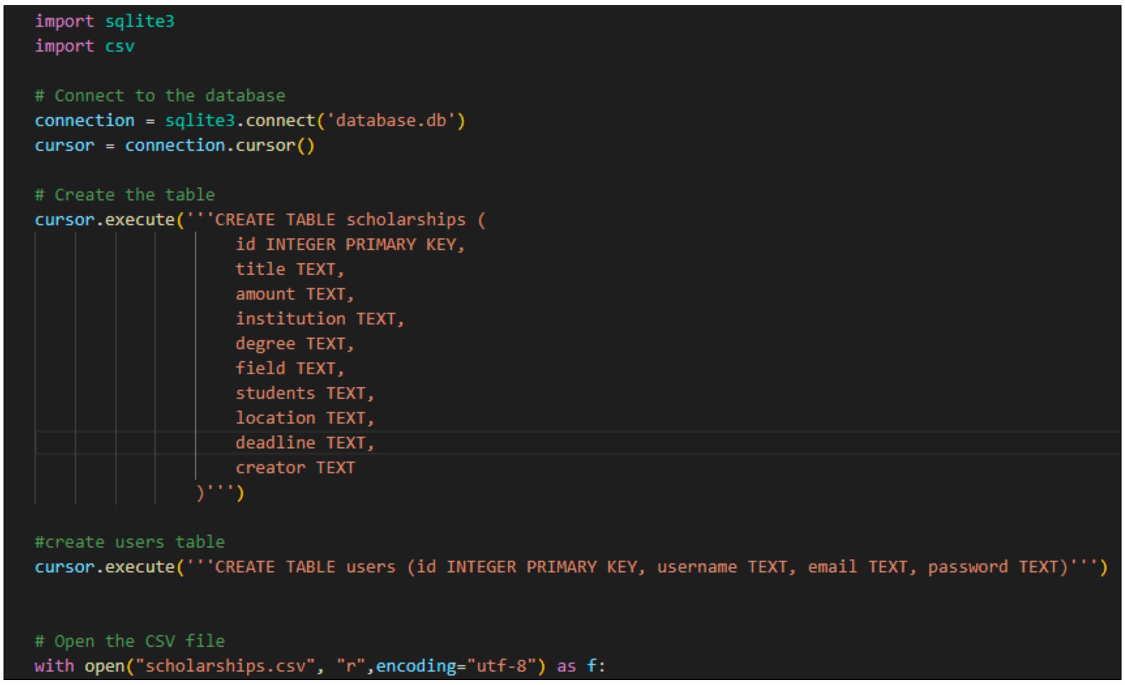

scraper.py > ...

```python
import requests
import csv
from bs4 import BeautifulSoup

# URL of the website to scrape
base_url = "https://www.scholarshipsads.com/category/tags/tunisia/page/"

# Create a CSV file to store the results
with open("scholarships.csv", "w", newline="",encoding="utf-8") as f:
    writer = csv.writer(f)

    # Write the header row of the CSV file
    writer.writerow(["id","Title", "Ammount", "Institution", "Degree", "Field", "Students", "Location", "Deadline"])

    # Set the page number to 1
    page = 1
    id=0

    # Set a flag to indicate whether there are more pages to scrape
    more_pages = True

    # Loop until there are no more pages to scrape
    while page<20:
```

2.2 Sqlite3 Contribution

SQLite is a self-contained, serverless, zero-configuration, transactional SQL database engine. It is designed to be embedded in applications, rather than used as a standalone database server. SQLite is the most widely deployed SQL database engine in the world. [3]

I have used the .csv scraped data to build a database containing the latest scholarships. Implementation code :



```
import sqlite3
import csv

# Connect to the database
connection = sqlite3.connect('database.db')
cursor = connection.cursor()

# Create the table
cursor.execute('''CREATE TABLE scholarships (
                    id INTEGER PRIMARY KEY,
                    title TEXT,
                    amount TEXT,
                    institution TEXT,
                    degree TEXT,
                    field TEXT,
                    students TEXT,
                    location TEXT,
                    deadline TEXT,
                    creator TEXT
                )''')

#create users table
cursor.execute('''CREATE TABLE users (id INTEGER PRIMARY KEY, username TEXT, email TEXT, password TEXT)''')


# Open the CSV file
with open("scholarships.csv", "r",encoding="utf-8") as f:
```

2.3 Flask Contribution

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. [4]

I used Flask to build my API using decorators as endpoints after connecting it to the database. I also used sessions to make sure that the user needs to login in order to be able to add a scholarship to the database, update it or delete it. Jsonify was also imported from flask which helped me convert Python dictionaries to JSON object and send it as a the response for an HTTP request.

2.4 Python Contribution

Python is a general-purpose language, which means it can be used to build just about anything, from web applications to desktop applications to scientific applications and data analysis. [5]

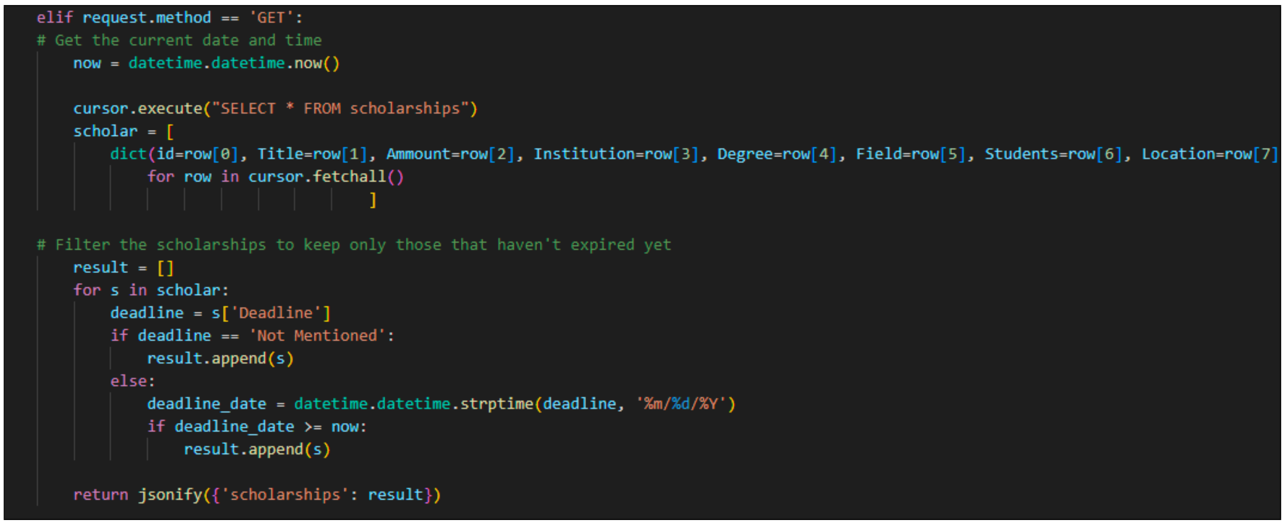
After choosing Flask as my web framework, creating a virtual environment, I used python to create my API's endpoints using specific URL. I added the code to handle requests and responses such as parsing request data, validating input, or interacting with a database.

2.5 All the HTTP Methods used

2.5.1 GET Requests

GET /scholarships

Get all the scholarships that haven't expired yet and all their information. No need to be logged in.



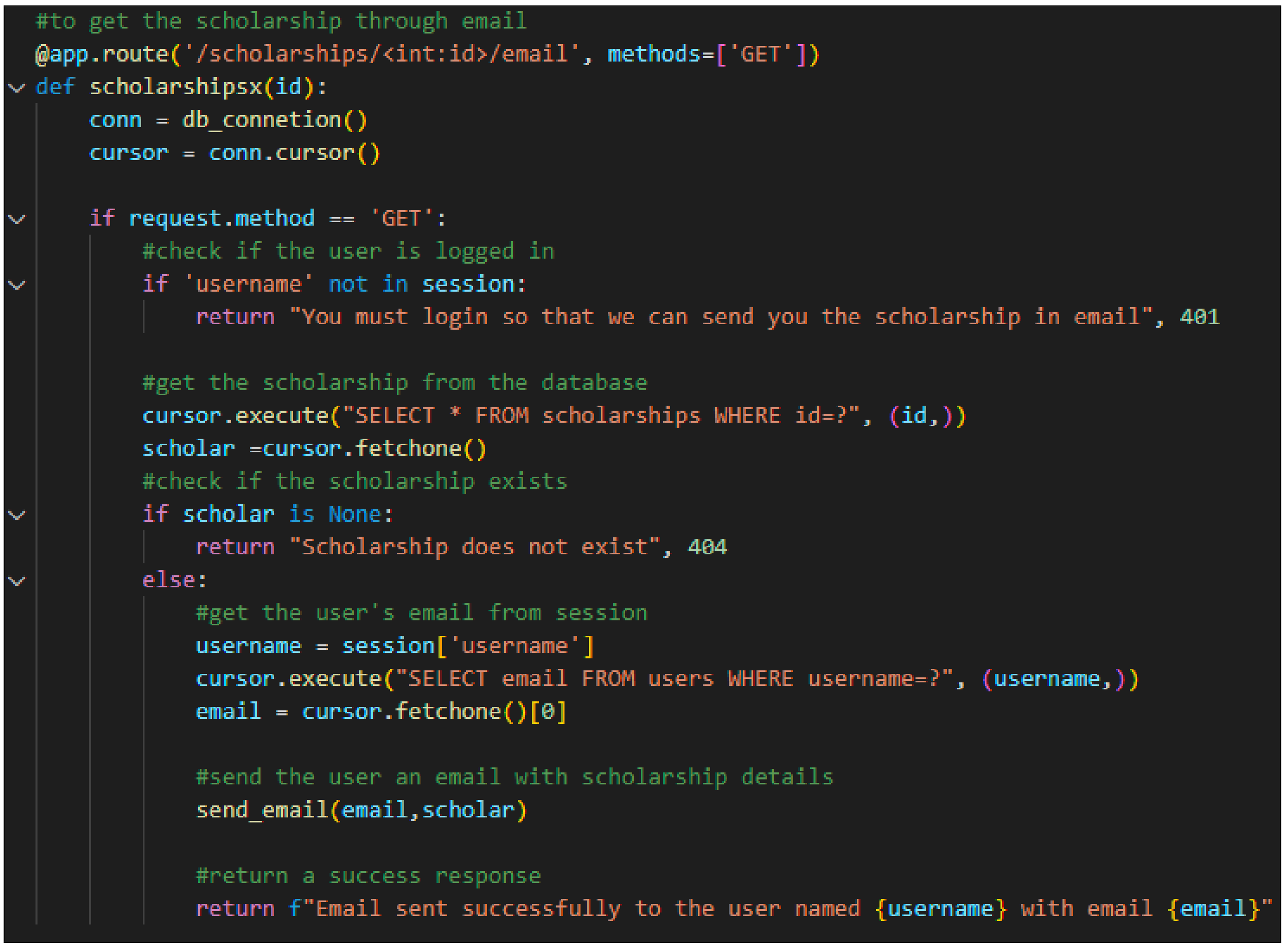
```
elif request.method == 'GET':
# Get the current date and time
    now = datetime.datetime.now()

    cursor.execute("SELECT * FROM scholarships")
    scholar = [
        dict(id=row[0], Title=row[1], Amount=row[2], Institution=row[3], Degree=row[4], Field=row[5], Students=row[6], Location=row[7]
            for row in cursor.fetchall()
                                    ]

# Filter the scholarships to keep only those that haven't expired yet
    result = []
    for s in scholar:
        deadline = s['Deadline']
        if deadline == 'Not Mentioned':
            result.append(s)
        else:
            deadline_date = datetime.datetime.strptime(deadline, '%m/%d/%Y')
            if deadline_date >= now:
                result.append(s)

    return jsonify({'scholarships': result})
```

GET /scholarships/<int:id>

Get all the information about a scholarship using its ID, ID is in the request URI. A login is required.

GET /scholarships/<int:id>/email

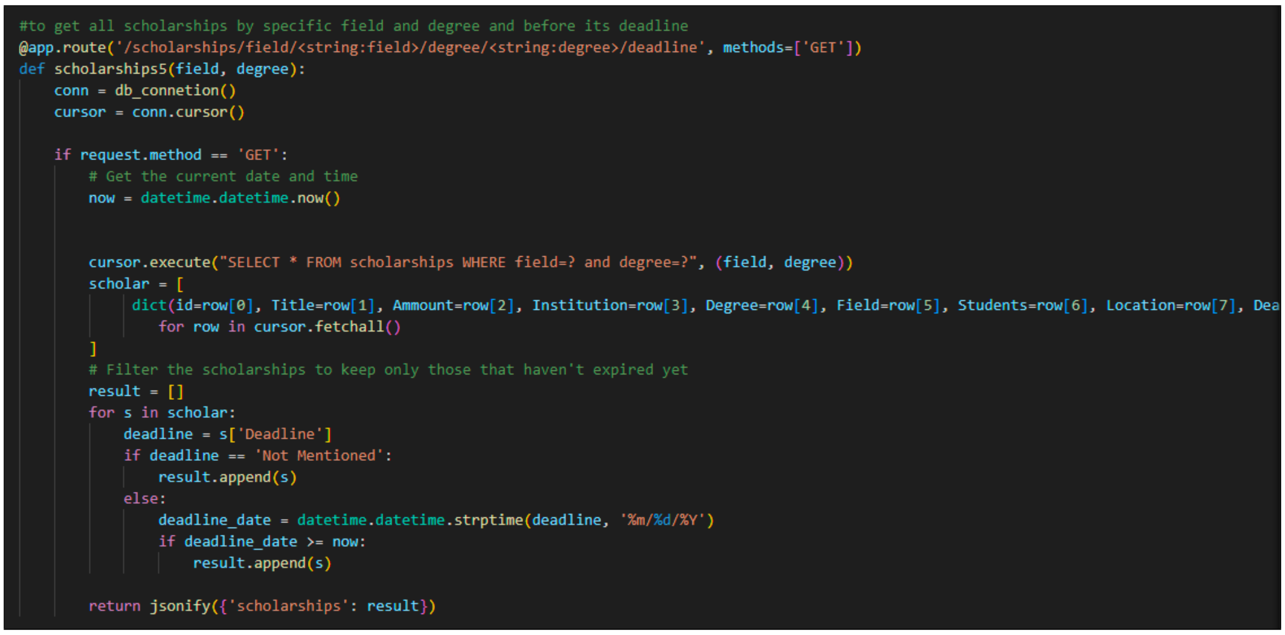
The user needs to login to access this endpoint. The session's user will receive an email containing all the information about the scholarship that have the ID mentioned in the URI.



```python
#to get the scholarship through email
@app.route('/scholarships/<int:id>/email', methods=['GET'])
def scholarshipsx(id):
    conn = db_connetion()
    cursor = conn.cursor()

    if request.method == 'GET':
        #check if the user is logged in
        if 'username' not in session:
            return "You must login so that we can send you the scholarship in email", 401

        #get the scholarship from the database
        cursor.execute("SELECT * FROM scholarships WHERE id=?", (id,))
        scholar =cursor.fetchone()
        #check if the scholarship exists
        if scholar is None:
            return "Scholarship does not exist", 404
        else:
            #get the user's email from session
            username = session['username']
            cursor.execute("SELECT email FROM users WHERE username=?", (username,))
            email = cursor.fetchone()[0]

            #send the user an email with scholarship details
            send_email(email,scholar)

            #return a success response
            return f"Email sent successfully to the user named {username} with email {email}"
```

GET /scholarships/field/<string:field>/degree/<string:degree>/deadline

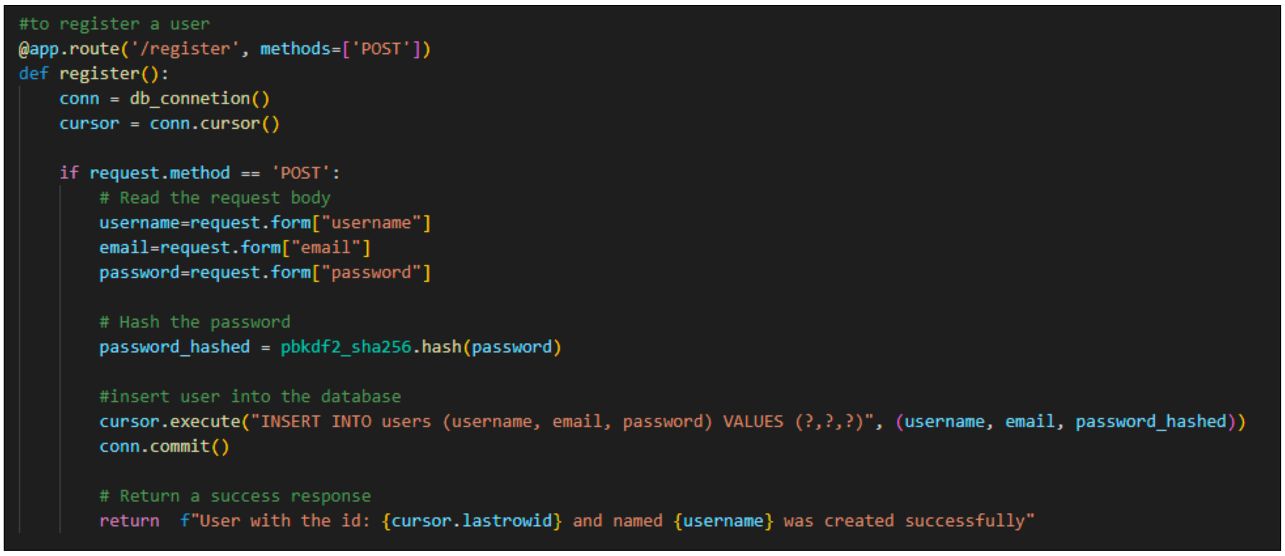
The user will get all the scholarships that offer the degree wanted, haven't expired yet, are related to the field mentioned and located in the location mentioned in the URI. No need to be logged in.


```
#to get all scholarships by specific field and degree and before its deadline
@app.route('/scholarships/field/<string:field>/degree/<string:degree>/deadline', methods=['GET'])
def scholarships5(field, degree):
    conn = db_connetion()
    cursor = conn.cursor()

    if request.method == 'GET':
        # Get the current date and time
        now = datetime.datetime.now()


        cursor.execute("SELECT * FROM scholarships WHERE field=? and degree=?", (field, degree))
        scholar = [
            dict(id=row[0], Title=row[1], Ammount=row[2], Institution=row[3], Degree=row[4], Field=row[5], Students=row[6], Location=row[7], Dea
                for row in cursor.fetchall()
        ]
        # Filter the scholarships to keep only those that haven't expired yet
        result = []
        for s in scholar:
            deadline = s['Deadline']
            if deadline == 'Not Mentioned':
                result.append(s)
            else:
                deadline_date = datetime.datetime.strptime(deadline, '%m/%d/%Y')
                if deadline_date >= now:
                    result.append(s)

        return jsonify({'scholarships': result})
```

2.5.2 POST Requests

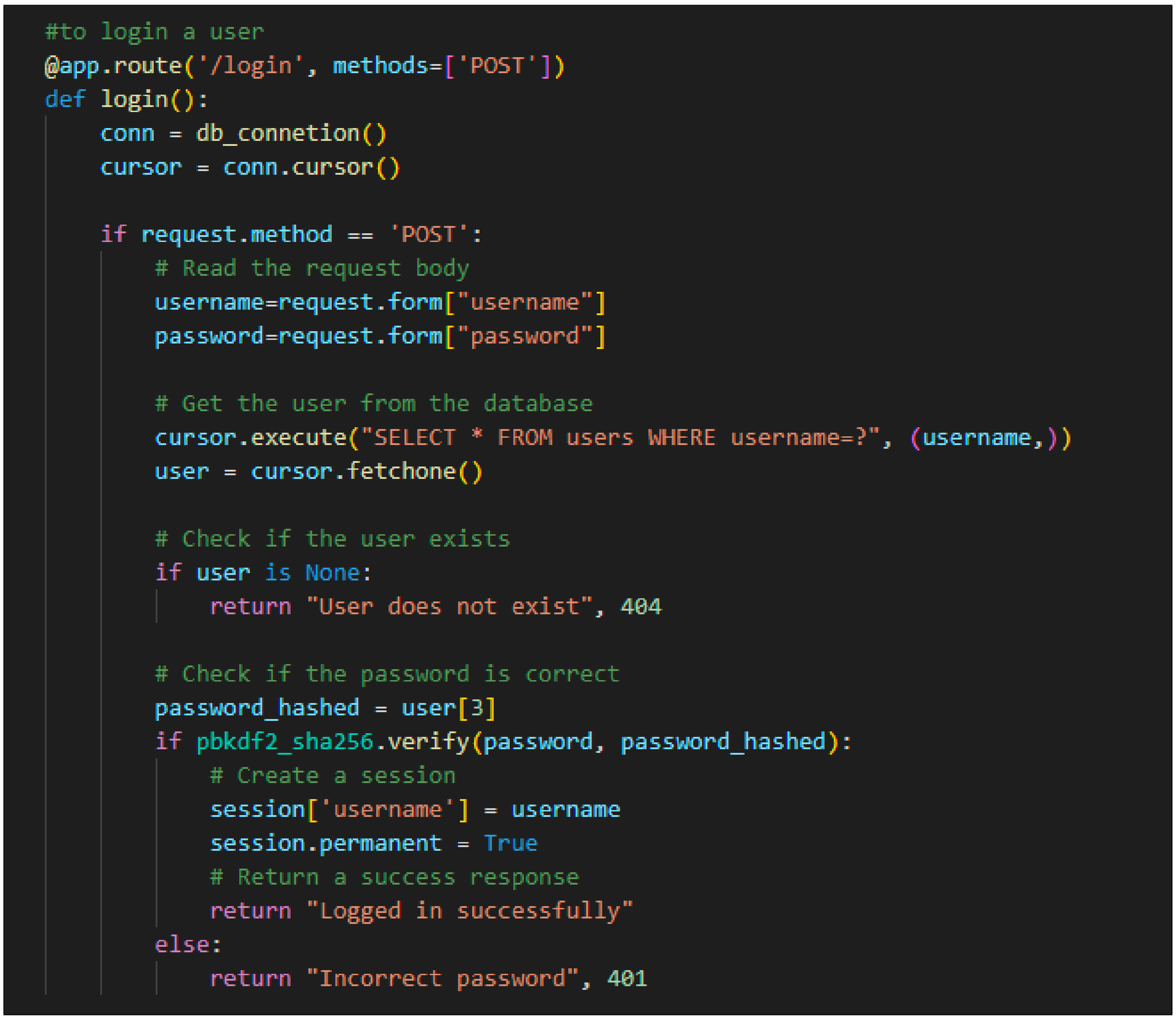
POST /register

Register in the API after specifying a valid username, password and email in the body of the request. The password entered will be hashed and stored in the database using passlib.hash library in Python.


```
#to register a user
@app.route('/register', methods=['POST'])
def register():
    conn = db_connetion()
    cursor = conn.cursor()

    if request.method == 'POST':
        # Read the request body
        username=request.form["username"]
        email=request.form["email"]
        password=request.form["password"]

        # Hash the password
        password_hashed = pbkdf2_sha256.hash(password)

        #insert user into the database
        cursor.execute("INSERT INTO users (username, email, password) VALUES (?,?,?)", (username, email, password_hashed))
        conn.commit()

        # Return a success response
        return  f"User with the id: {cursor.lastrowid} and named {username} was created successfully"
```

POST /login

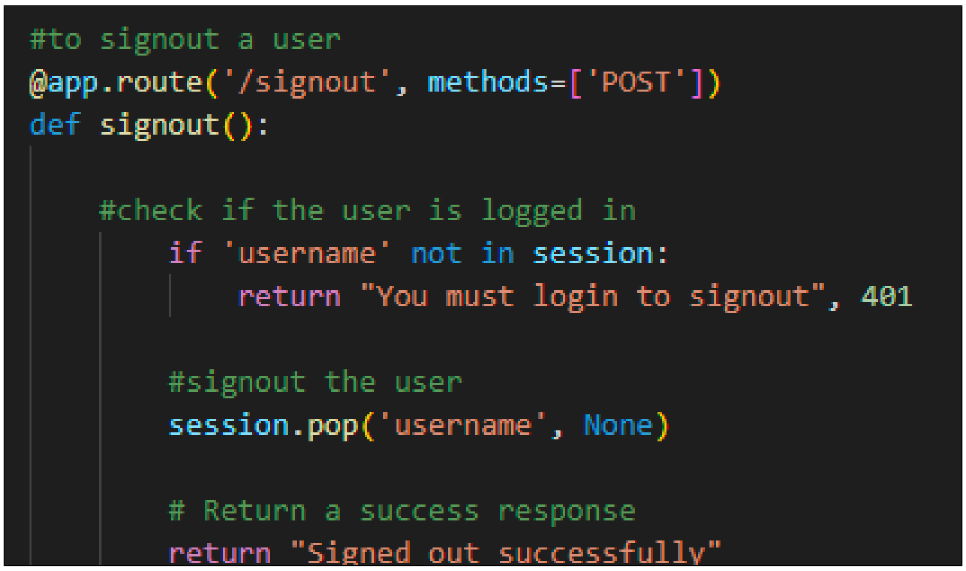
Use the credentials you entered when registering (username and password) to login to the API. This will create a session with username of the user.



```
#to login a user
@app.route('/login', methods=['POST'])
def login():
    conn = db_connetion()
    cursor = conn.cursor()

    if request.method == 'POST':
        # Read the request body
        username=request.form["username"]
        password=request.form["password"]

        # Get the user from the database
        cursor.execute("SELECT * FROM users WHERE username=?", (username,))
        user = cursor.fetchone()

        # Check if the user exists
        if user is None:
            return "User does not exist", 404

        # Check if the password is correct
        password_hashed = user[3]
        if pbkdf2_sha256.verify(password, password_hashed):
            # Create a session
            session['username'] = username
            session.permanent = True
            # Return a success response
            return "Logged in successfully"
        else:
            return "Incorrect password", 401
```

POST /signout

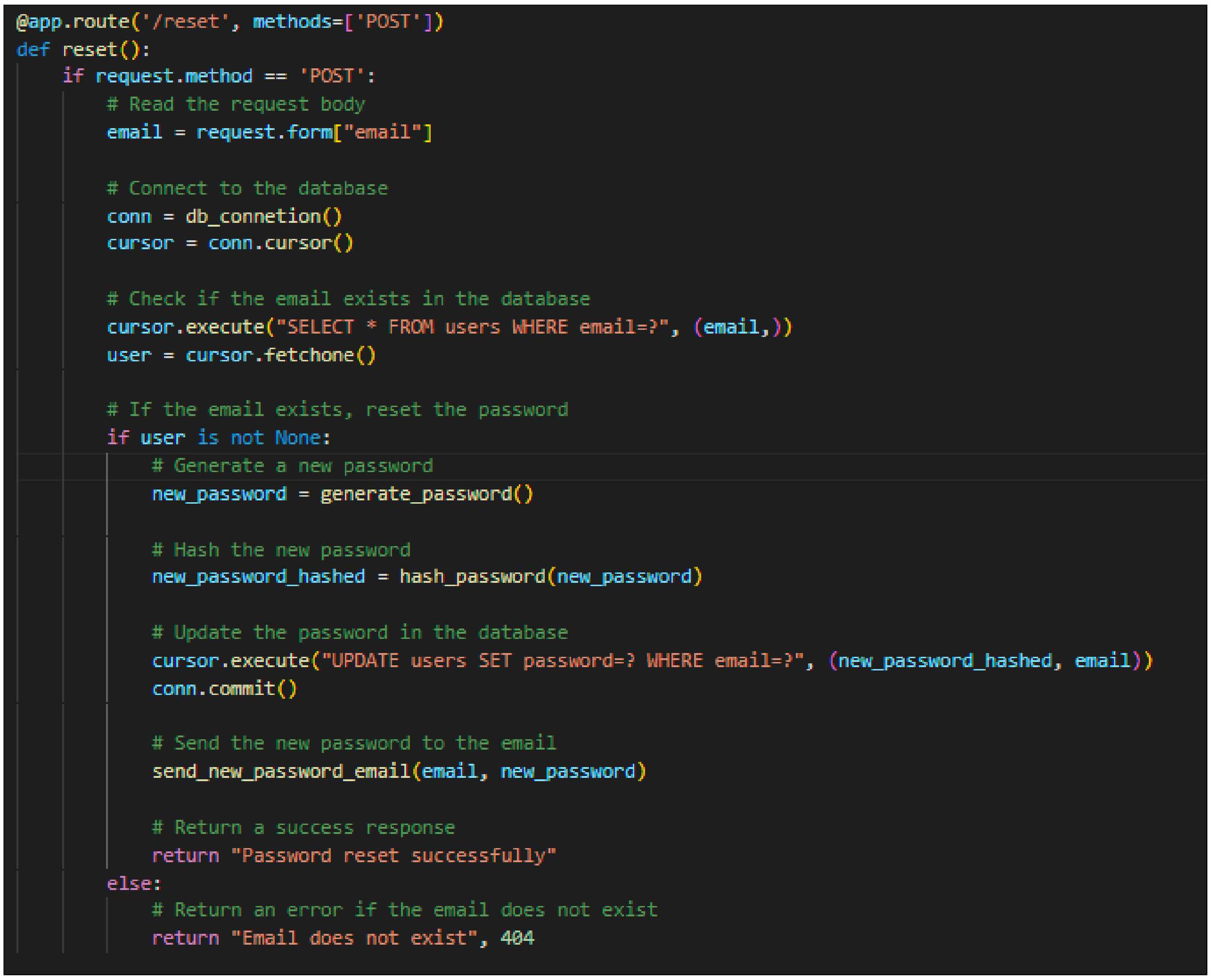
Signout from the current session.



```python
#to signout a user
@app.route('/signout', methods=['POST'])
def signout():

    #check if the user is logged in
        if 'username' not in session:
            return "You must login to signout", 401

        #signout the user
        session.pop('username', None)

        # Return a success response
        return "Signed out successfully"
```

POST /reset

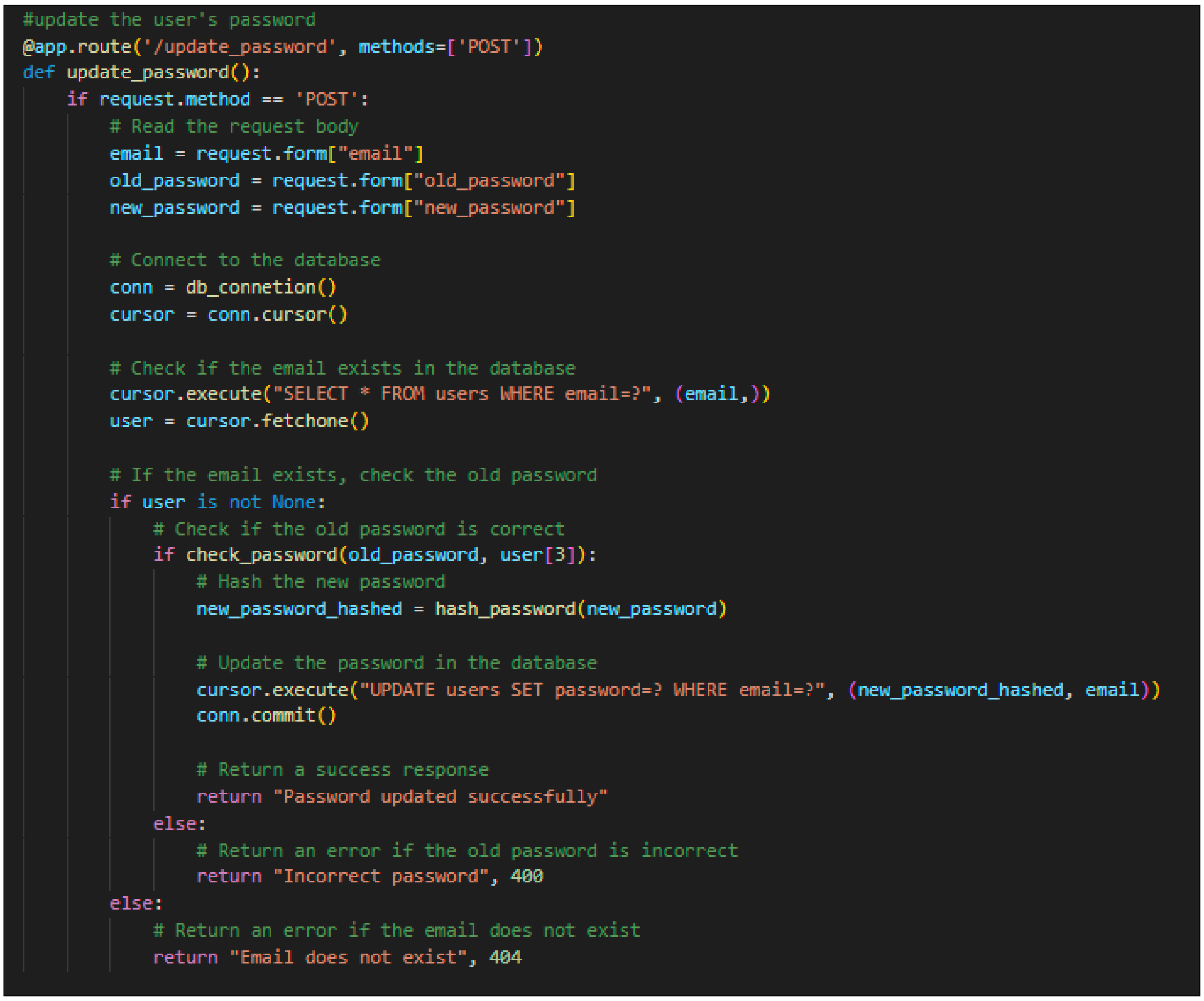
Get through email you signed up with a new password that you could use to sign in.



```python
@app.route('/reset', methods=['POST'])
def reset():
    if request.method == 'POST':
        # Read the request body
        email = request.form["email"]

        # Connect to the database
        conn = db_connetion()
        cursor = conn.cursor()

        # Check if the email exists in the database
        cursor.execute("SELECT * FROM users WHERE email=?", (email,))
        user = cursor.fetchone()

        # If the email exists, reset the password
        if user is not None:
            # Generate a new password
            new_password = generate_password()

            # Hash the new password
            new_password_hashed = hash_password(new_password)

            # Update the password in the database
            cursor.execute("UPDATE users SET password=? WHERE email=?", (new_password_hashed, email))
            conn.commit()

            # Send the new password to the email
            send_new_password_email(email, new_password)

            # Return a success response
            return "Password reset successfully"
        else:
            # Return an error if the email does not exist
            return "Email does not exist", 404
```

POST /update$_password$

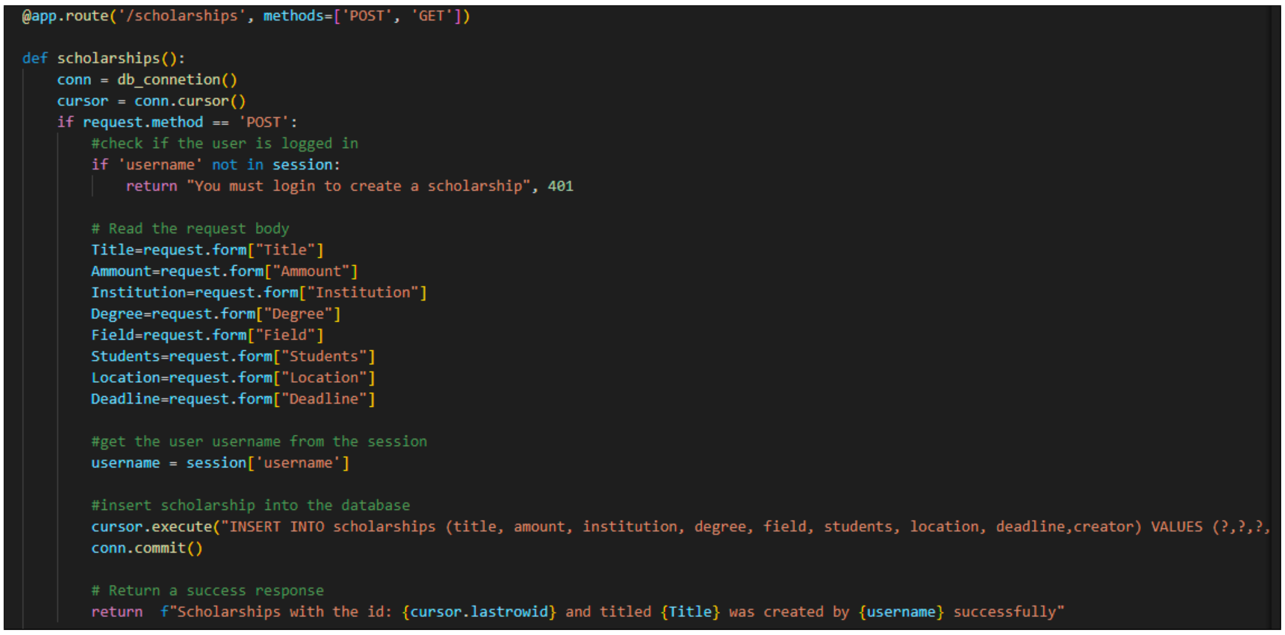
Update your old password to a new one. Parameters are passed in the body of the request.



```
#update the user's password
@app.route('/update_password', methods=['POST'])
def update_password():
    if request.method == 'POST':
        # Read the request body
        email = request.form["email"]
        old_password = request.form["old_password"]
        new_password = request.form["new_password"]

        # Connect to the database
        conn = db_connetion()
        cursor = conn.cursor()

        # Check if the email exists in the database
        cursor.execute("SELECT * FROM users WHERE email=?", (email,))
        user = cursor.fetchone()

        # If the email exists, check the old password
        if user is not None:
            # Check if the old password is correct
            if check_password(old_password, user[3]):
                # Hash the new password
                new_password_hashed = hash_password(new_password)

                # Update the password in the database
                cursor.execute("UPDATE users SET password=? WHERE email=?", (new_password_hashed, email))
                conn.commit()

                # Return a success response
                return "Password updated successfully"
            else:
                # Return an error if the old password is incorrect
                return "Incorrect password", 400
        else:
            # Return an error if the email does not exist
            return "Email does not exist", 404
```

POST /scholarships

Create a new scholarship. A log in is required in order to insert a new scholarship and get you username assigned to it. Parameters such Title, Degree, Institution and Location are in the body request.

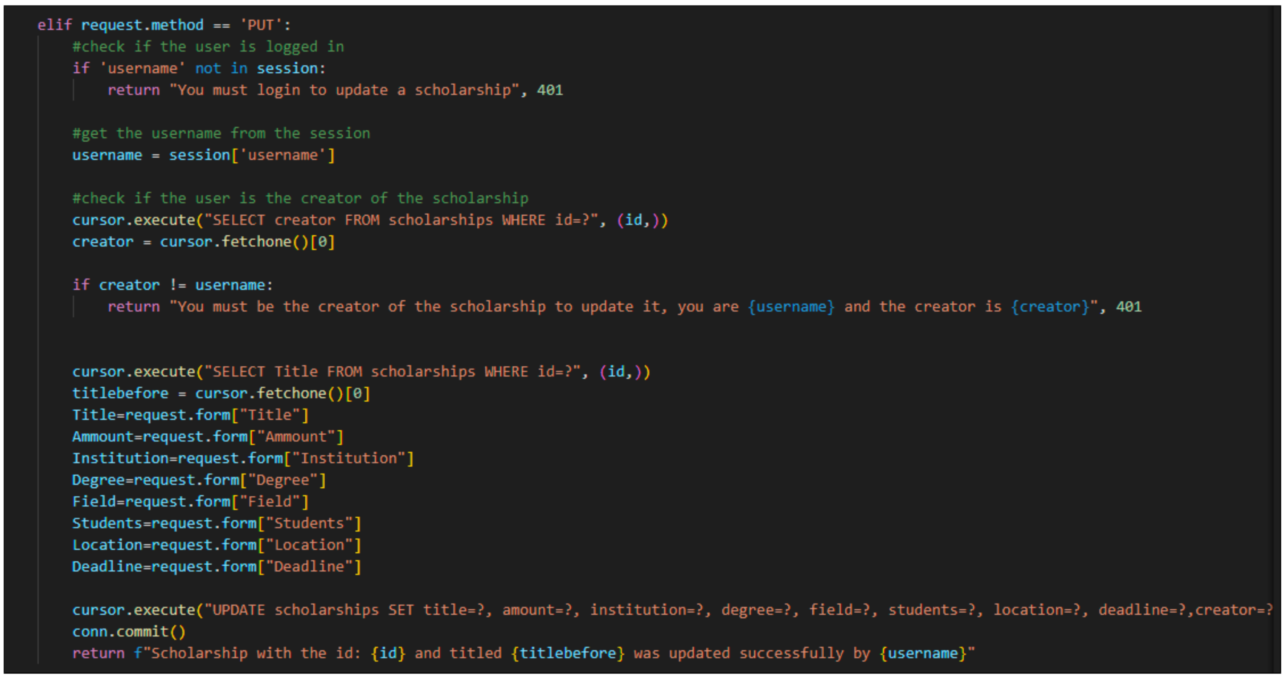


```
@app.route('/scholarships', methods=['POST', 'GET'])

def scholarships():
    conn = db_connetion()
    cursor = conn.cursor()
    if request.method == 'POST':
        #check if the user is logged in
        if 'username' not in session:
            return "You must login to create a scholarship", 401

        # Read the request body
        Title=request.form["Title"]
        Ammount=request.form["Ammount"]
        Institution=request.form["Institution"]
        Degree=request.form["Degree"]
        Field=request.form["Field"]
        Students=request.form["Students"]
        Location=request.form["Location"]
        Deadline=request.form["Deadline"]

        #get the user username from the session
        username = session['username']

        #insert scholarship into the database
        cursor.execute("INSERT INTO scholarships (title, amount, institution, degree, field, students, location, deadline,creator) VALUES (?,?,?,
        conn.commit()

        # Return a success response
        return  f"Scholarships with the id: {cursor.lastrowid} and titled {Title} was created by {username} successfully"
```

2.5.3 PUT Requests

PUT /scholarships/<int:id>

A login is required for this route. The user will be able to update any information related to only a scholarship he created. The user cannot update any scholarship that he did not create. Parameters are passed in the body request.

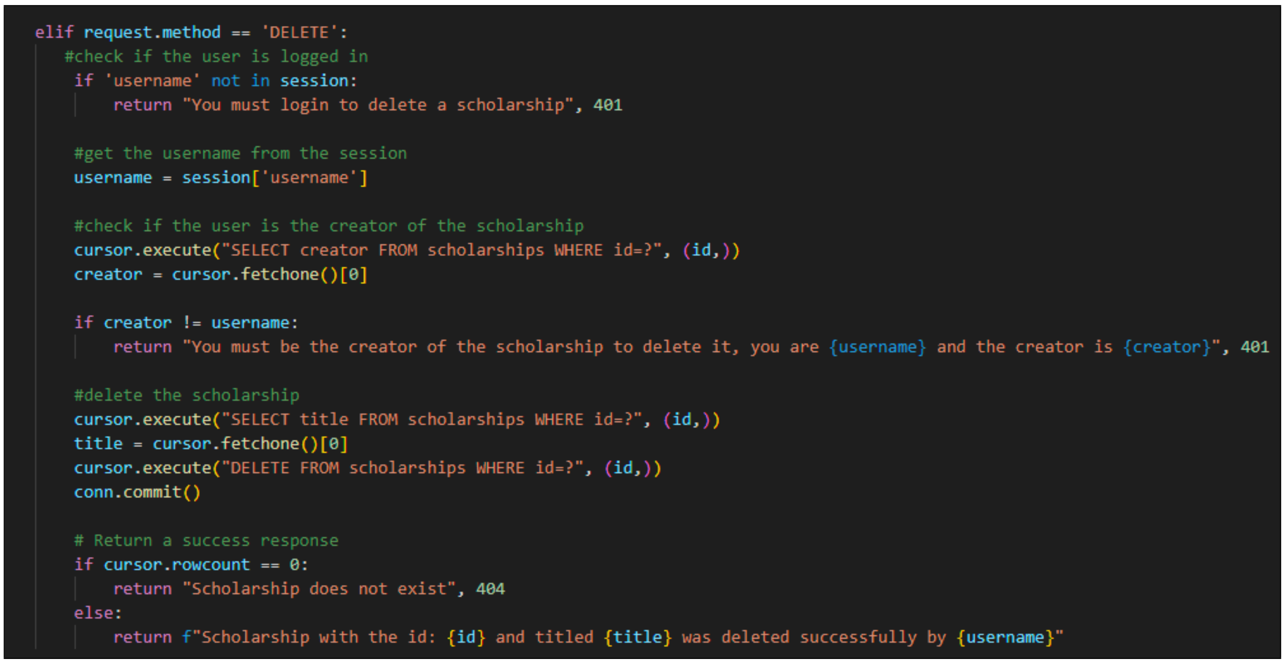


```
elif request.method == 'PUT':
    #check if the user is logged in
    if 'username' not in session:
        return "You must login to update a scholarship", 401

    #get the username from the session
    username = session['username']

    #check if the user is the creator of the scholarship
    cursor.execute("SELECT creator FROM scholarships WHERE id=?", (id,))
    creator = cursor.fetchone()[0]

    if creator != username:
        return "You must be the creator of the scholarship to update it, you are {username} and the creator is {creator}", 401


    cursor.execute("SELECT Title FROM scholarships WHERE id=?", (id,))
    titlebefore = cursor.fetchone()[0]
    Title=request.form["Title"]
    Ammount=request.form["Ammount"]
    Institution=request.form["Institution"]
    Degree=request.form["Degree"]
    Field=request.form["Field"]
    Students=request.form["Students"]
    Location=request.form["Location"]
    Deadline=request.form["Deadline"]

    cursor.execute("UPDATE scholarships SET title=?, amount=?, institution=?, degree=?, field=?, students=?, location=?, deadline=?,creator=?
    conn.commit()
    return f"Scholarship with the id: {id} and titled {titlebefore} was updated successfully by {username}"
```

2.5.4 DELETE Requests

DELETE /scholarships/<int:id>

A login is required. The user will be able to delete only scholarships he created by passing its ID in the URI.

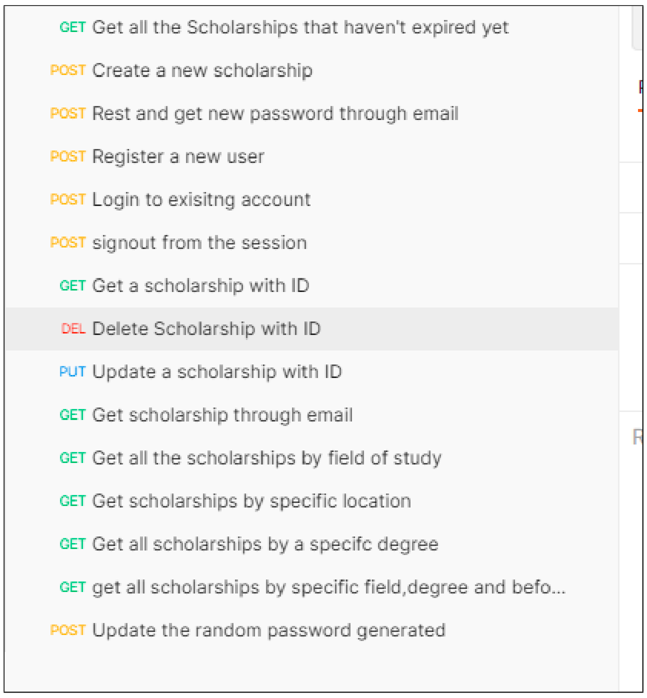


```python
elif request.method == 'DELETE':
    #check if the user is logged in
    if 'username' not in session:
        return "You must login to delete a scholarship", 401

    #get the username from the session
    username = session['username']

    #check if the user is the creator of the scholarship
    cursor.execute("SELECT creator FROM scholarships WHERE id=?", (id,))
    creator = cursor.fetchone()[0]

    if creator != username:
        return "You must be the creator of the scholarship to delete it, you are {username} and the creator is {creator}", 401

    #delete the scholarship
    cursor.execute("SELECT title FROM scholarships WHERE id=?", (id,))
    title = cursor.fetchone()[0]
    cursor.execute("DELETE FROM scholarships WHERE id=?", (id,))
    conn.commit()

    # Return a success response
    if cursor.rowcount == 0:
        return "Scholarship does not exist", 404
    else:
        return f"Scholarship with the id: {id} and titled {title} was deleted successfully by {username}"
```

2.6 Postman Contribution

Postman is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated. [6]

I have used Postman for the automatic testing of my API requests, as well as some snippets such as "Response Time less than 200ms","Status Code is 200"..etc.

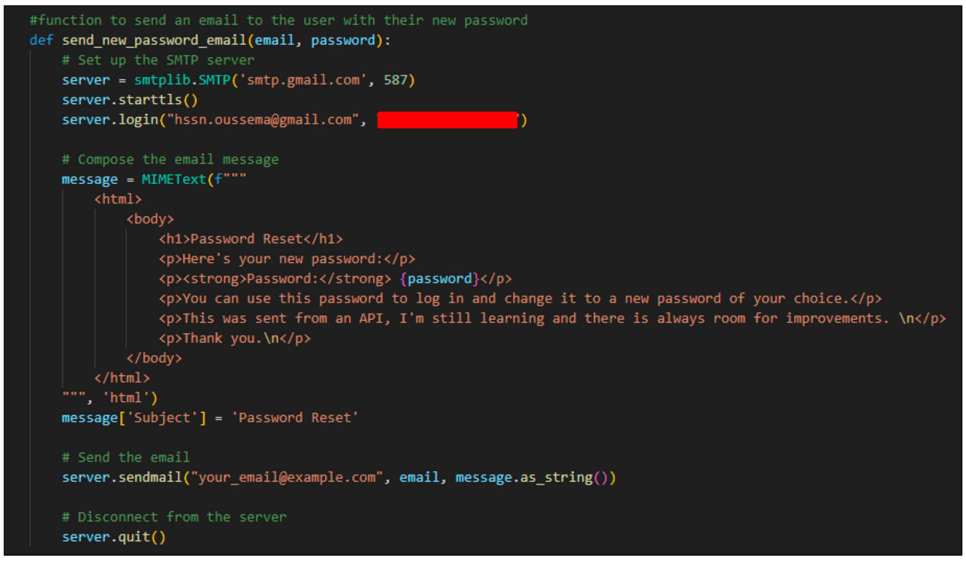


GET Get all the Scholarships that haven't expired yet
POST Create a new scholarship
POST Rest and get new password through email
POST Register a new user
POST Login to exisitng account
POST signout from the session
GET Get a scholarship with ID
DEL Delete Scholarship with ID
PUT Update a scholarship with ID
GET Get scholarship through email
GET Get all the scholarships by field of study
GET Get scholarships by specific location
GET Get all scholarships by a specifc degree
GET get all scholarships by specific field,degree and befo...
POST Update the random password generated

2.7 SMTP Contribution

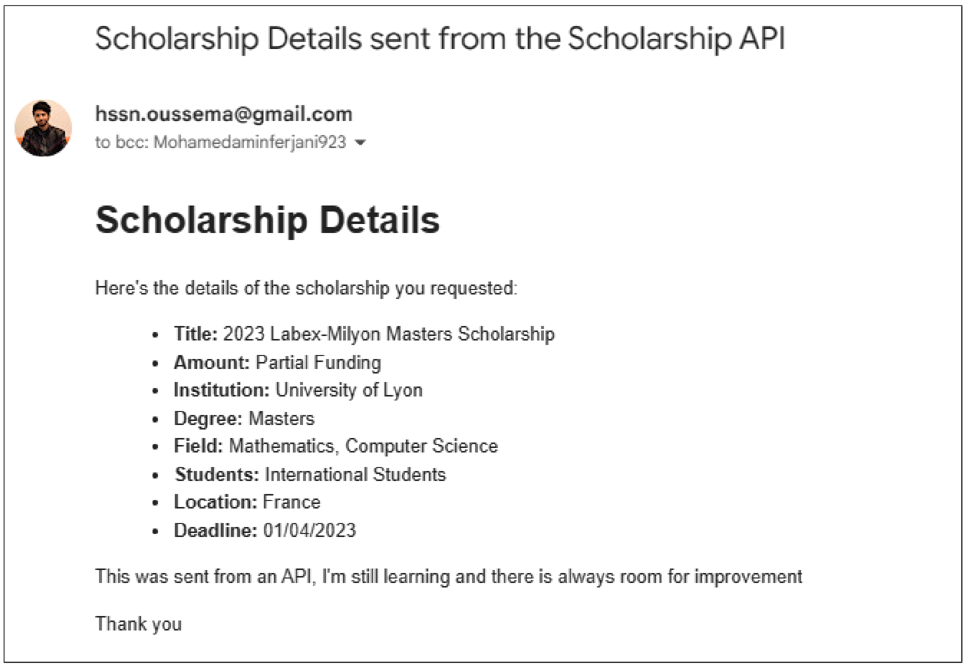
The Simple Mail Transfer Protocol is an Internet standard communication protocol for electronic mail transmission. Mail servers and other message transfer agents use SMTP to send and receive mail messages. [7]

I was able to send email to users who requested full information about a scholarship or those who lost their passwords and requested a new one. I also used MIMEText [**?**] to transform .json



```
#function to send an email to the user with their new password
def send_new_password_email(email, password):
    # Set up the SMTP server
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login("hssn.oussema@gmail.com", [redacted])

    # Compose the email message
    message = MIMEText(f"""
        <html>
            <body>
                <h1>Password Reset</h1>
                <p>Here's your new password:</p>
                <p><strong>Password:</strong> {password}</p>
                <p>You can use this password to log in and change it to a new password of your choice.</p>
                <p>This was sent from an API, I'm still learning and there is always room for improvements. \n</p>
                <p>Thank you.\n</p>
            </body>
        </html>
    """, 'html')
    message['Subject'] = 'Password Reset'

    # Send the email
    server.sendmail("your_email@example.com", email, message.as_string())

    # Disconnect from the server
    server.quit()
```

file into more aesthetic format (HTML) in the email sent.


Scholarship Details sent from the Scholarship API

hssn.oussema@gmail.com
to bcc: Mohamedaminferjani923

Scholarship Details

Here's the details of the scholarship you requested:

- **Title:** 2023 Labex-Milyon Masters Scholarship
- **Amount:** Partial Funding
- **Institution:** University of Lyon
- **Degree:** Masters
- **Field:** Mathematics, Computer Science
- **Students:** International Students
- **Location:** France
- **Deadline:** 01/04/2023

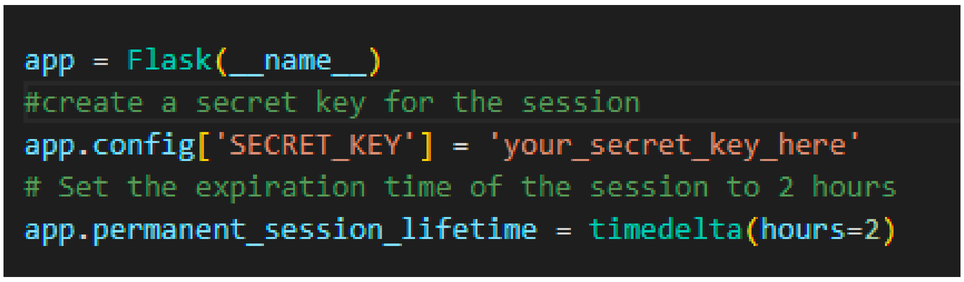
This was sent from an API, I'm still learning and there is always room for improvement

Thank you

2.8 Flask-Session Contribution

Flask-Session is an extension for Flask that supports Server-side Session to your application

I used sessions to remember each user when they log in. Each session has a unique username, each username can PUT and DELETE only the scholarships that he POSTED. This way, the authenticity of the scholarships is maintained. This also allowed me assign usernames to scholarships created manually, without a session, you would have to specify explicitly who is the one that made the scholarship which can result in misleading. [8]


```
app = Flask(__name__)
#create a secret key for the session
app.config['SECRET_KEY'] = 'your_secret_key_here'
# Set the expiration time of the session to 2 hours
app.permanent_session_lifetime = timedelta(hours=2)
```

2.9 Database Structure

Database is scraped from the latest 20 pages from scholarshipsads.com .
The database consists of 2 tables which no relations between them.

2.9.1 Tables

Table "Scholarships"

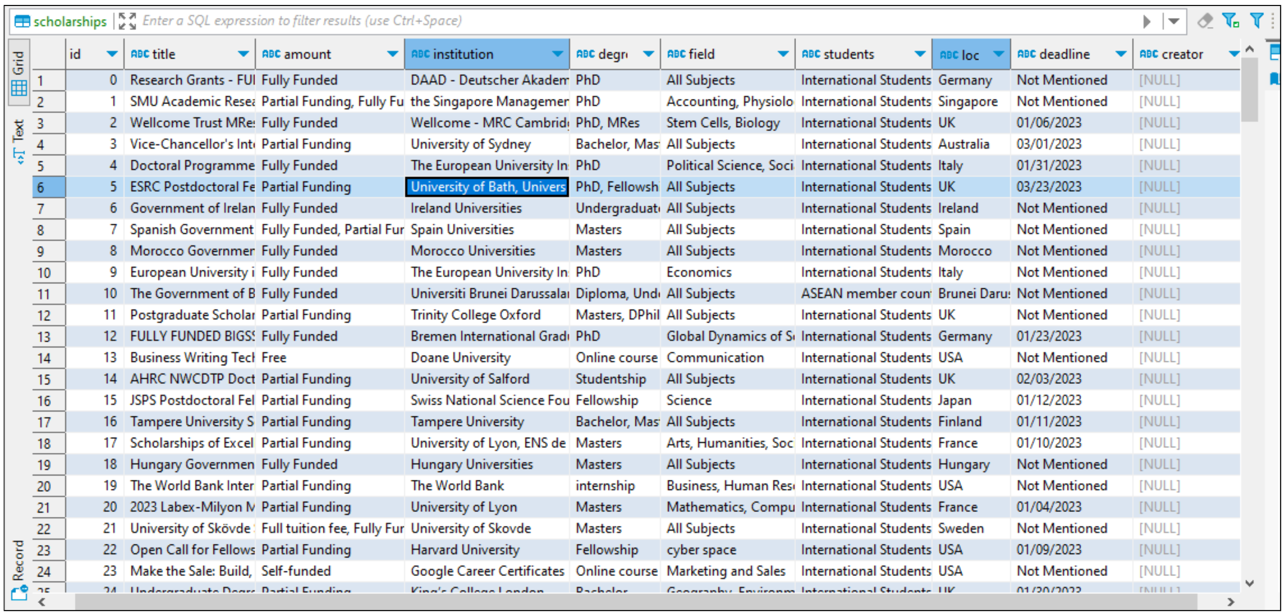
containing 10 columns :

1. Id : ID of the scholarship
2. title : Title of the scholarship
3. amount : Fully/Partially funded
4. institution : Name of the insitution
5. degree : MSc/Phd/Bachelor
6. field : Field of study
7. students : Targeted students
8. location : Location of the institution
9. deadline : Deadline of the Scholarship mm/dd/yyyy
10. creator : username of the user who created the scholarship

Table "users"

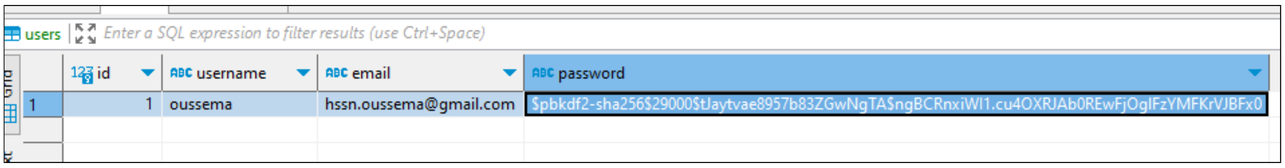
containing 3 columns :

1. username : Name of the user
2. password : Password of the user hashed
3. email: Email of the user



scholarships | Enter a SQL expression to filter results (use Ctrl+Space)

	id	title	amount	institution	degr	field	students	loc	deadline	creator
1	0	Research Grants - FUI	Fully Funded	DAAD - Deutscher Akadem	PhD	All Subjects	International Students	Germany	Not Mentioned	[NULL]
2	1	SMU Academic Resea	Partial Funding, Fully Fu	the Singapore Managemer	PhD	Accounting, Physiolo	International Students	Singapore	Not Mentioned	[NULL]
3	2	Wellcome Trust MRe:	Fully Funded	Wellcome - MRC Cambrid	PhD, MRes	Stem Cells, Biology	International Students	UK	01/06/2023	[NULL]
4	3	Vice-Chancellor's Int	Partial Funding	University of Sydney	Bachelor, Mas	All Subjects	International Students	Australia	03/01/2023	[NULL]
5	4	Doctoral Programme	Fully Funded	The European University In	PhD	Political Science, Soci	International Students	Italy	01/31/2023	[NULL]
6	5	ESRC Postdoctoral Fe	Partial Funding	University of Bath, Univers	PhD, Fellowsh	All Subjects	International Students	UK	03/23/2023	[NULL]
7	6	Government of Irelan	Fully Funded	Ireland Universities	Undergraduat	All Subjects	International Students	Ireland	Not Mentioned	[NULL]
8	7	Spanish Government	Fully Funded, Partial Fur	Spain Universities	Masters	All Subjects	International Students	Spain	Not Mentioned	[NULL]
9	8	Morocco Governmer	Fully Funded	Morocco Universities	Masters	All Subjects	International Students	Morocco	Not Mentioned	[NULL]
10	9	European University i	Fully Funded	The European University In	PhD	Economics	International Students	Italy	Not Mentioned	[NULL]
11	10	The Government of B	Fully Funded	Universiti Brunei Darussalai	Diploma, Und	All Subjects	ASEAN member coun	Brunei Daru:	Not Mentioned	[NULL]
12	11	Postgraduate Scholar	Partial Funding	Trinity College Oxford	Masters, DPhil	All Subjects	International Students	UK	Not Mentioned	[NULL]
13	12	FULLY FUNDED BIGSS	Fully Funded	Bremen International Grad	PhD	Global Dynamics of S	International Students	Germany	01/23/2023	[NULL]
14	13	Business Writing Tech	Free	Doane University	Online course	Communication	International Students	USA	Not Mentioned	[NULL]
15	14	AHRC NWCDTP Doct	Partial Funding	University of Salford	Studentship	All Subjects	International Students	UK	02/03/2023	[NULL]
16	15	JSPS Postdoctoral Fel	Partial Funding	Swiss National Science Fou	Fellowship	Science	International Students	Japan	01/12/2023	[NULL]
17	16	Tampere University S	Partial Funding	Tampere University	Bachelor, Mas	All Subjects	International Students	Finland	01/11/2023	[NULL]
18	17	Scholarships of Excel	Partial Funding	University of Lyon, ENS de	Masters	Arts, Humanities, Soc	International Students	France	01/10/2023	[NULL]
19	18	Hungary Governmen	Fully Funded	Hungary Universities	Masters	All Subjects	International Students	Hungary	Not Mentioned	[NULL]
20	19	The World Bank Inter	Partial Funding	The World Bank	internship	Business, Human Res	International Students	USA	Not Mentioned	[NULL]
21	20	2023 Labex-Milyon M	Partial Funding	University of Lyon	Masters	Mathematics, Compu	International Students	France	01/04/2023	[NULL]
22	21	University of Skövde	Full tuition fee, Fully Fur	University of Skovde	Masters	All Subjects	International Students	Sweden	Not Mentioned	[NULL]
23	22	Open Call for Fellows	Partial Funding	Harvard University	Fellowship	cyber space	International Students	USA	01/09/2023	[NULL]
24	23	Make the Sale: Build,	Self-funded	Google Career Certificates	Online course	Marketing and Sales	International Students	USA	Not Mentioned	[NULL]
25	24	Undergraduate Degr	Partial Funding	King's College London	Bachelor	Geography, Environm	International Students	UK	01/30/2023	[NULL]



users | Enter a SQL expression to filter results (use Ctrl+Space)

	id	username	email	password
1	1	oussema	hssn.oussema@gmail.com	$pbkdf2-sha256$29000$tJaytvae8957b83ZGwNgTA$ngBCRnxiWI1.cu4OXRJAb0REwFjOgIFzYMFKrVJBFx0

Chapter 3

Conclusion

I am currently struggling in my search for a scholarship for a master's degree. As I was researching, I had an idea to build this API that could generate up-to-date scholarships and sort them according to user's preferences. This is how I came up with the idea for my WebService project. By creating this API, I hope to make the scholarship search process easier for others who are in a similar situation to mine. The API will utilize the latest information to match students with the scholarships that best fit their needs and qualifications.
I had fun working on this project because I related to the problem proposed and I knew what was missing. Althought I faced more problems than I imagined, each successful compilation gave me a dose of dopamine and I discovered a lot of new tricks/life hacks that could help me in the future in my professional career.
At first, I thought that this course is unnecessary and would not help me much in becoming a data scientist but later on, I realized that APIs are fundamentals to know and that they facilitate a lot of work. Thank you Dr.Montassar for helping us in our learning journey.

Oussema Hassine

TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Appendix A

Response examples

A.1 GET Requests

A.1.1 GET /scholarships/1

```
{
  "scholarship": {
    "Ammount": "Partial Funding, Fully Funded",
    "Creator": null,
    "Deadline": "Not Mentioned",
    "Degree": "PhD",
    "Field": "Accounting, Physiology, Economics, Business,
       Information Technology, Computer Science +4 More, Law,
       Commerce, Technology, Asian Urbanism",
    "Institution": "the Singapore Management University",
    "Location": "Singapore",
    "Students": "International Students",
    "Title": "SMU Academic Research PhD Programmes 2023",
    "id": 1
  }
}
```

A.1.2 GET /scholarship/1/email

```
Email sent successfully to the user named oussema with email
    hssn.oussema@gmail.com
```

A.1.3 GET /scholarships/location/France

```
{
  "scholarships": [
    {
      "Ammount": "Partial Funding",
      "Deadline": "01/10/2023",
      "Degree": "Masters",
      "Field": "Arts, Humanities, Social Sciences, Mathematics,
          Computer Science, Physics +2 More, Chemistry, Science
        ",
      "Institution": "University of Lyon, ENS de Lyon",
      "Location": "France",
      "Students": "International Students",
      "Title": "Scholarships of Excellence 2023 France",
      "id": 17
    },
    {
      "Ammount": "Partial Funding",
      "Deadline": "01/04/2023",
      "Degree": "Masters",
      "Field": "Mathematics, Computer Science",
      "Institution": "University of Lyon",
      "Location": "France",
      "Students": "International Students",
      "Title": "2023 Labex-Milyon Masters Scholarship",
      "id": 20
    },
    {
      "Ammount": "Partial Funding",
      "Deadline": "01/16/2023",
      "Degree": "Fellowship",
      "Field": "Climate Change",
      "Institution": "Campus France",
      "Location": "France",
      "Students": "International Students",
      "Title": "MOPGA 2023 - VISITNG FELLOWSHIP PROGRAM FOR
        YOUNG RESEARCHERS",
      "id": 32
    },
    {
      "Ammount": "Fully Funded
                                                      Monthly
        grant for three years and accomodation",
      "Deadline": "Not Mentioned",
      "Degree": "Masters, Diploma",
      "Field": "Humanities, Arts and Humanities, Arts",
      "Institution": "\u00c9cole Normale Sup\u00e9rieure",
      "Location": "France",
      "Students": "International Students",
      "Title": "\u00c9cole Normale Sup\u00e9rieure
        International Selection Scholarship",
      "id": 110
    }
  ]
}
```

A.2 POST Requests

A.2.1 POST /login

```
Logged in successfully
```

A.2.2 POST /register

```
User with the id: 2 and named Mohammed was created successfully
```

A.2.3 POST /scholarships

```
Scholarships with the id: 285 and titled TBS SCHOLARSHIP was
    created by oussema successfully
```

A.2.4 POST /reset

```
Password reset successfully and sent to your email hssn.
    oussema@gmail.com
```

A.3 PUT Requests

A.3.1 PUT /scholarships/1

```
You must be the creator of the scholarship to update it, you
    are oussema and the creator is None
```

A.4 DELETE Requests

A.4.1 DELETE /scholarships/285

```
Scholarship with the id: 285 and titled TBS SCHOLARSHIP was
    deleted successfully by oussema
```

Bibliography

[1] L. Richardson, “Beautiful soup documentation,” *Dosegljivo: https://www. crummy. com/-software/BeautifulSoup/bs4/doc/.[Dostopano: 7. 7. 2018]*, 2007.

[2] D. F. Ningtyas and N. Setiyawati, “Implementasi flask framework pada pembangunan aplikasi purchasing approval request,” *Jurnal Janitra Informatika Dan Sistem Informasi*, vol. 1, no. 1, pp. 19–34, 2021.

[3] M. Prammer, S. S. Rajesh, J. Chen, and J. Patel, “Introducing a query acceleration path for analytics in sqlite3,” in *12th Annual Conference on Innovative Data Systems Research (CIDR’22)*, 2022.

[4] T. T. Tidwell, “Wilhelm schlenk: the man behind the flask,” *Angewandte Chemie International Edition*, vol. 40, no. 2, pp. 331–337, 2001.

[5] W. Python, “Python,” *Python Releases for Windows*, vol. 24, 2021.

[6] G. Romero, “What is postman api test,” *encora*, p. 1, June 2021.

[7] P. Hoffman, “Smtp service extension for secure smtp over transport layer security,” tech. rep., 2002.

[8] M. Grinberg, “Flask-socketio documentation,” *línea]. Disponible en: https://flask-socketio. readthedocs. org/en/latest/.[Último acceso: 2015]*, 2019.