COMP0034 2023/24 Coursework 1 specification (Individual)

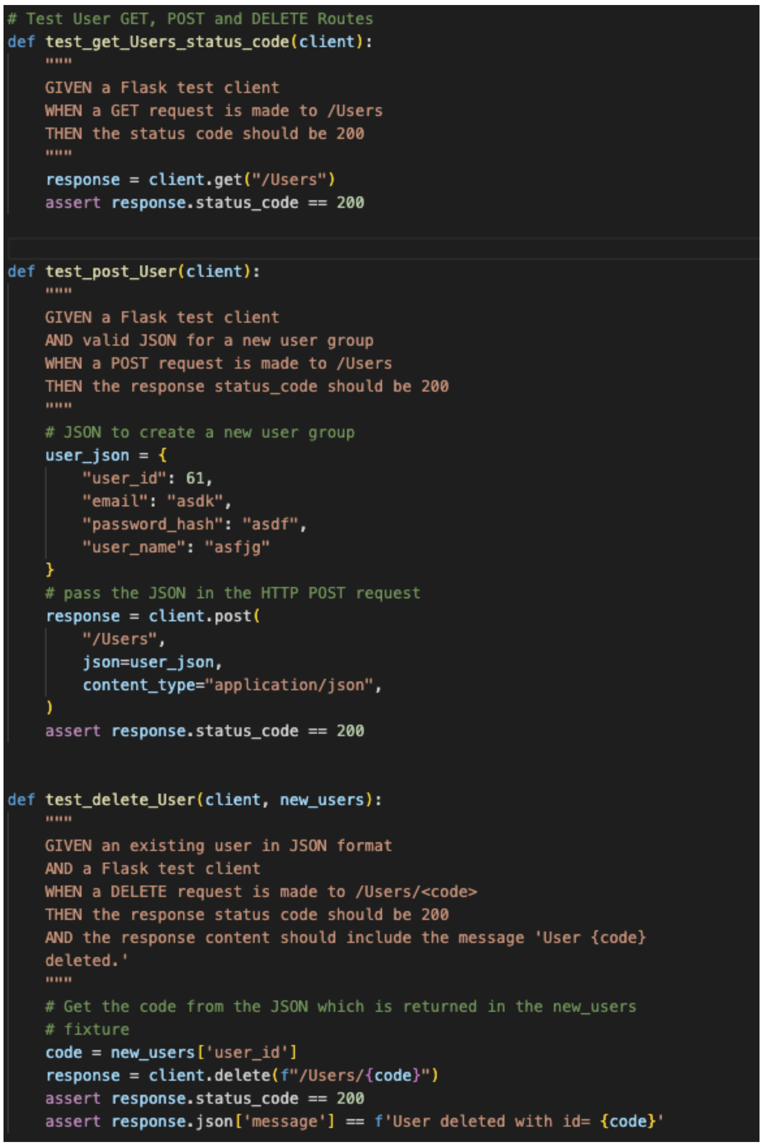
Coursework content

1. Test code

1.1. test_routes.py

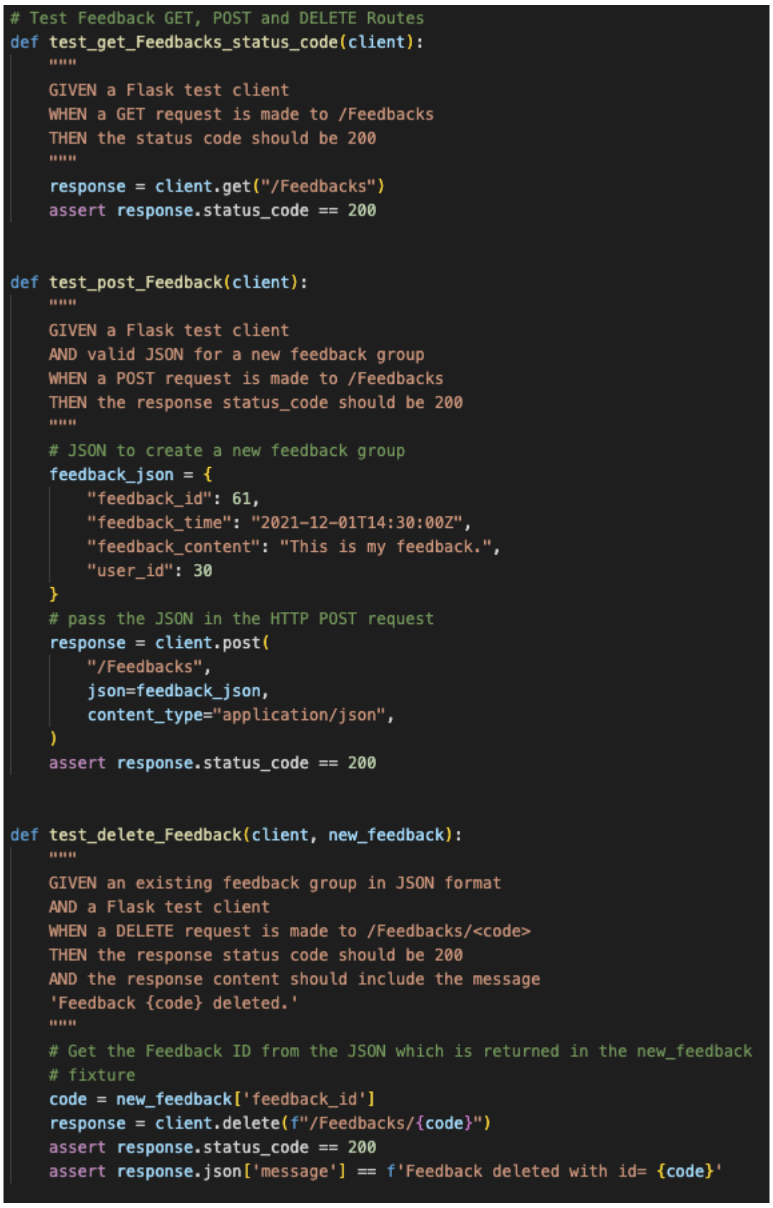
Testing each route such as GET, POST route in application code is necessary. I have nine class: User, Feedback, Age_group, Gender, Ethnicity, Employment, Course_level, Disability and Teacher.

The first one is User. These three test functions are designed to verify the functionality of a Flask application's REST API concerning user management. Each test simulates a different HTTP method (GET, POST, and DELETE) to interact with the /Users endpoint. They use Flask's test client to make these requests without the need for the server to be running, providing a streamlined way to ensure the API behaves as expected under various conditions.



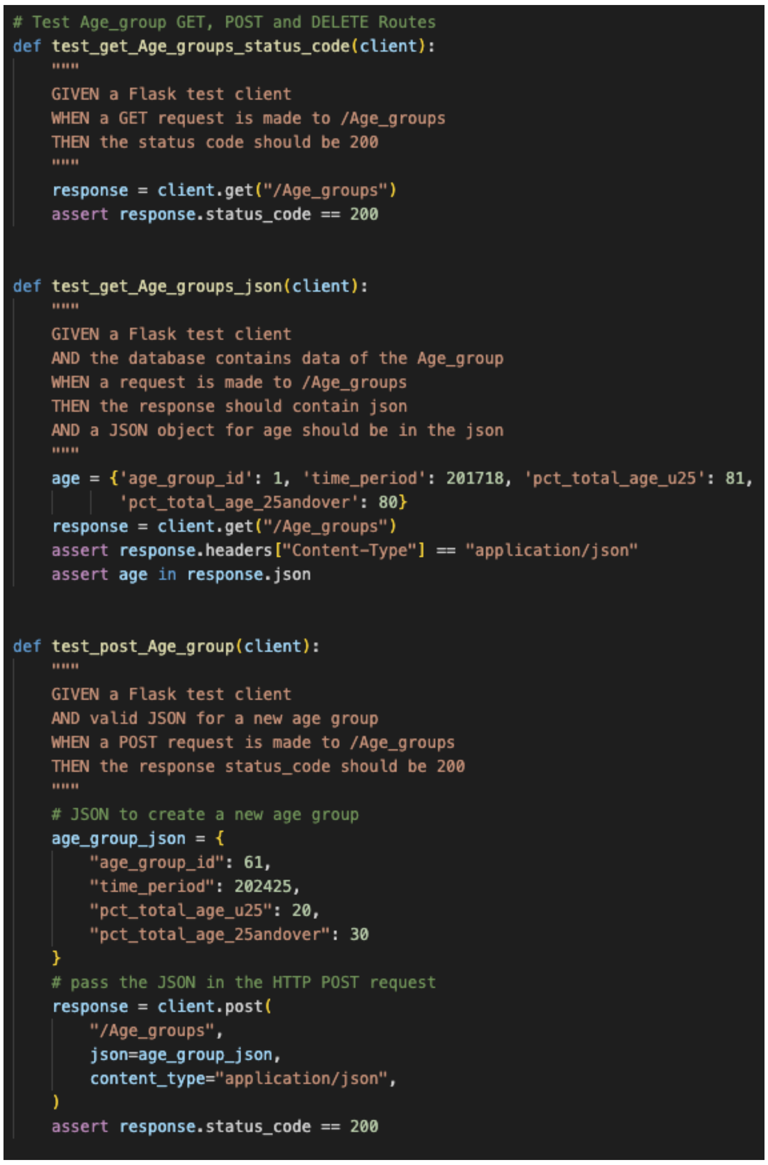
```python
# Test User GET, POST and DELETE Routes
def test_get_Users_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Users
    THEN the status code should be 200
    """
    response = client.get("/Users")
    assert response.status_code == 200


def test_post_User(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new user group
    WHEN a POST request is made to /Users
    THEN the response status_code should be 200
    """
    # JSON to create a new user group
    user_json = {
        "user_id": 61,
        "email": "asdk",
        "password_hash": "asdf",
        "user_name": "asfjg"
    }
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Users",
        json=user_json,
        content_type="application/json",
    )
    assert response.status_code == 200


def test_delete_User(client, new_users):
    """
    GIVEN an existing user in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Users/<code>
    THEN the response status code should be 200
    AND the response content should include the message 'User {code}
    deleted.'
    """
    # Get the code from the JSON which is returned in the new_users
    # fixture
    code = new_users['user_id']
    response = client.delete(f"/Users/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'User deleted with id= {code}'
```

Then it’s Feedback:

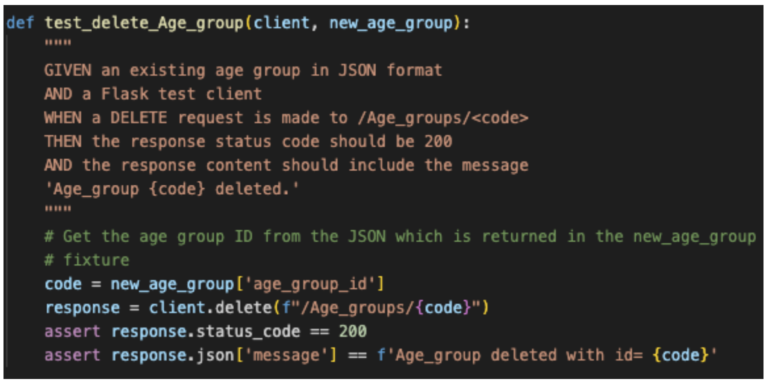


```python
# Test Feedback GET, POST and DELETE Routes
def test_get_Feedbacks_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Feedbacks
    THEN the status code should be 200
    """
    response = client.get("/Feedbacks")
    assert response.status_code == 200


def test_post_Feedback(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new feedback group
    WHEN a POST request is made to /Feedbacks
    THEN the response status_code should be 200
    """
    # JSON to create a new feedback group
    feedback_json = {
        "feedback_id": 61,
        "feedback_time": "2021-12-01T14:30:00Z",
        "feedback_content": "This is my feedback.",
        "user_id": 30
    }
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Feedbacks",
        json=feedback_json,
        content_type="application/json",
    )
    assert response.status_code == 200


def test_delete_Feedback(client, new_feedback):
    """
    GIVEN an existing feedback group in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Feedbacks/<code>
    THEN the response status code should be 200
    AND the response content should include the message
    'Feedback {code} deleted.'
    """
    # Get the Feedback ID from the JSON which is returned in the new_feedback
    # fixture
    code = new_feedback['feedback_id']
    response = client.delete(f"/Feedbacks/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'Feedback deleted with id= {code}'
```

Age_group:

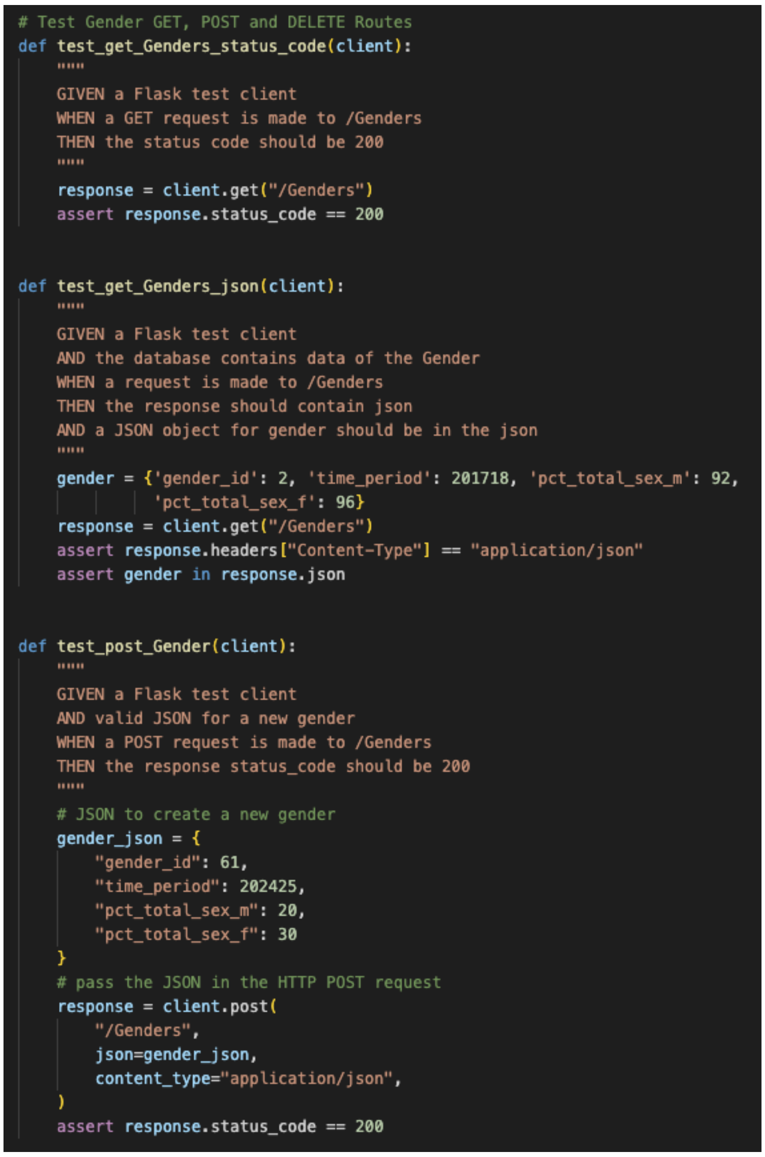


```python
# Test Age_group GET, POST and DELETE Routes
def test_get_Age_groups_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Age_groups
    THEN the status code should be 200
    """
    response = client.get("/Age_groups")
    assert response.status_code == 200


def test_get_Age_groups_json(client):
    """
    GIVEN a Flask test client
    AND the database contains data of the Age_group
    WHEN a request is made to /Age_groups
    THEN the response should contain json
    AND a JSON object for age should be in the json
    """
    age = {'age_group_id': 1, 'time_period': 201718, 'pct_total_age_u25': 81,
           'pct_total_age_25andover': 80}
    response = client.get("/Age_groups")
    assert response.headers["Content-Type"] == "application/json"
    assert age in response.json


def test_post_Age_group(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new age group
    WHEN a POST request is made to /Age_groups
    THEN the response status_code should be 200
    """
    # JSON to create a new age group
    age_group_json = {
        "age_group_id": 61,
        "time_period": 202425,
        "pct_total_age_u25": 20,
        "pct_total_age_25andover": 30
    }
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Age_groups",
        json=age_group_json,
        content_type="application/json",
    )
    assert response.status_code == 200
```

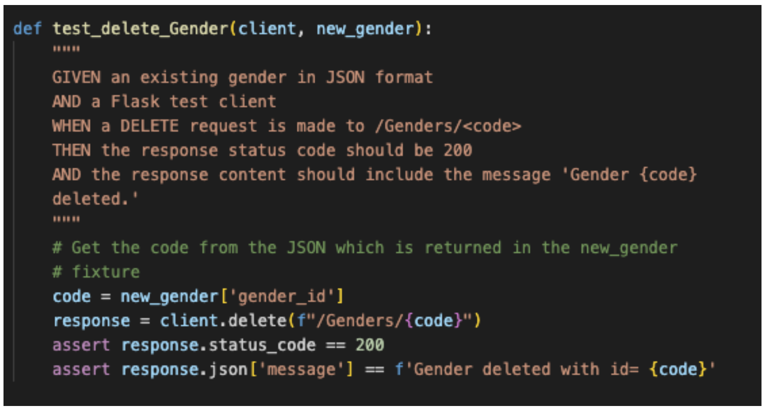


```python
def test_delete_Age_group(client, new_age_group):
    """
    GIVEN an existing age group in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Age_groups/<code>
    THEN the response status code should be 200
    AND the response content should include the message
    'Age_group {code} deleted.'
    """
    # Get the age group ID from the JSON which is returned in the new_age_group
    # fixture
    code = new_age_group['age_group_id']
    response = client.delete(f"/Age_groups/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'Age_group deleted with id= {code}'
```

Gender:

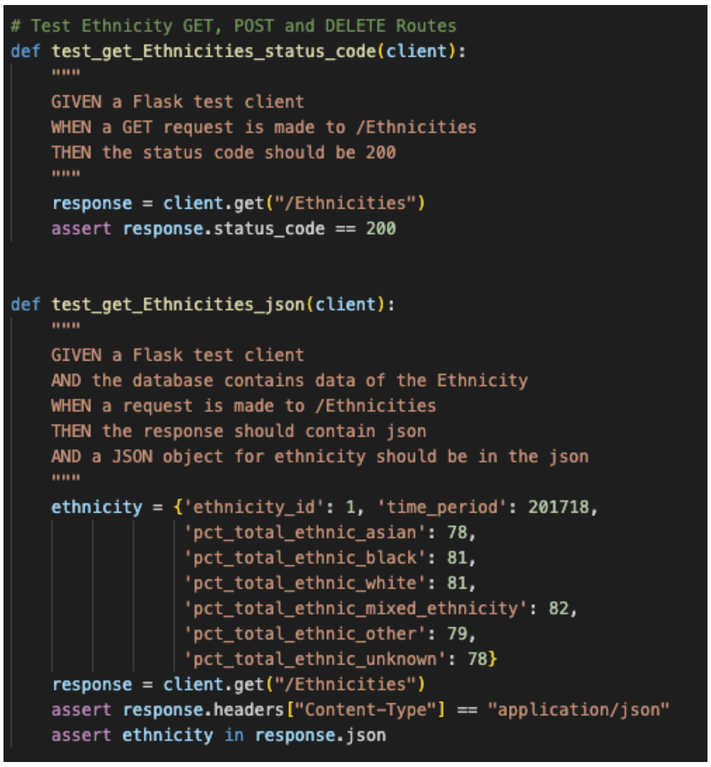


```python
# Test Gender GET, POST and DELETE Routes
def test_get_Genders_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Genders
    THEN the status code should be 200
    """
    response = client.get("/Genders")
    assert response.status_code == 200


def test_get_Genders_json(client):
    """
    GIVEN a Flask test client
    AND the database contains data of the Gender
    WHEN a request is made to /Genders
    THEN the response should contain json
    AND a JSON object for gender should be in the json
    """
    gender = {'gender_id': 2, 'time_period': 201718, 'pct_total_sex_m': 92,
              'pct_total_sex_f': 96}
    response = client.get("/Genders")
    assert response.headers["Content-Type"] == "application/json"
    assert gender in response.json


def test_post_Gender(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new gender
    WHEN a POST request is made to /Genders
    THEN the response status_code should be 200
    """
    # JSON to create a new gender
    gender_json = {
        "gender_id": 61,
        "time_period": 202425,
        "pct_total_sex_m": 20,
        "pct_total_sex_f": 30
    }
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Genders",
        json=gender_json,
        content_type="application/json",
    )
    assert response.status_code == 200
```

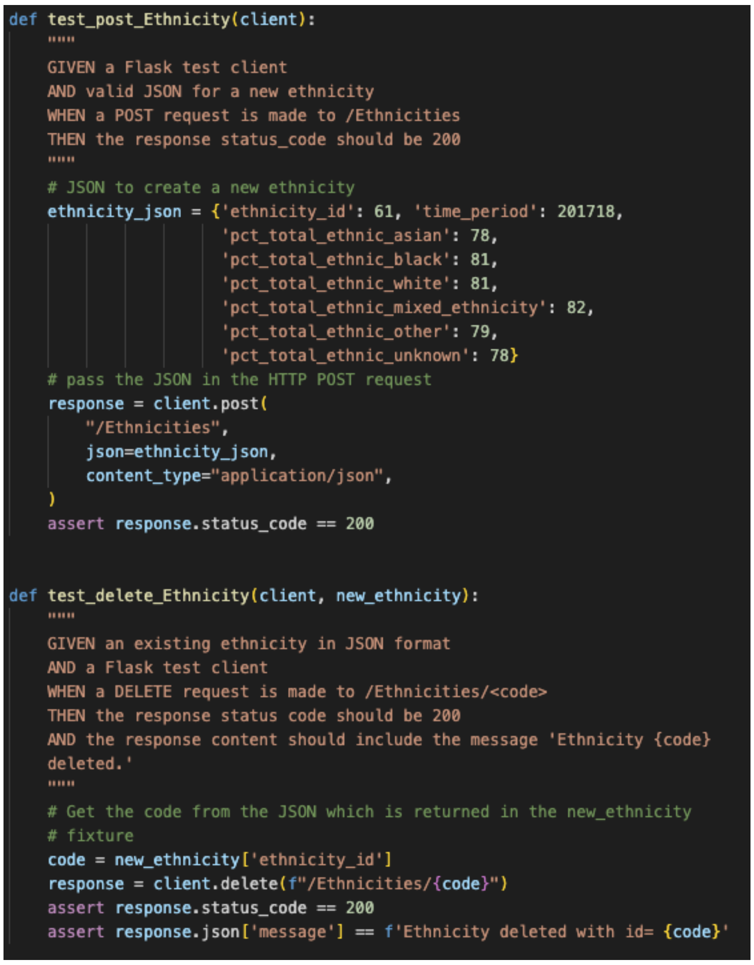


```python
def test_delete_Gender(client, new_gender):
    """
    GIVEN an existing gender in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Genders/<code>
    THEN the response status code should be 200
    AND the response content should include the message 'Gender {code}
    deleted.'
    """
    # Get the code from the JSON which is returned in the new_gender
    # fixture
    code = new_gender['gender_id']
    response = client.delete(f"/Genders/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'Gender deleted with id= {code}'
```

Ethnicity:

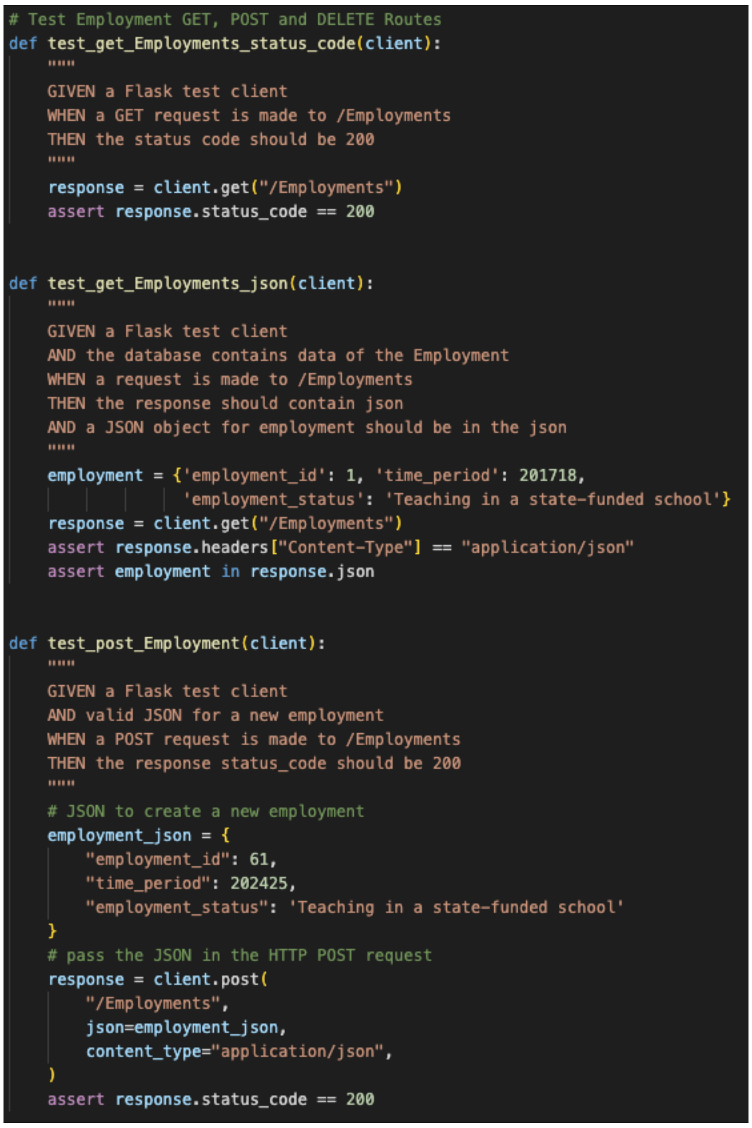


```python
# Test Ethnicity GET, POST and DELETE Routes
def test_get_Ethnicities_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Ethnicities
    THEN the status code should be 200
    """
    response = client.get("/Ethnicities")
    assert response.status_code == 200


def test_get_Ethnicities_json(client):
    """
    GIVEN a Flask test client
    AND the database contains data of the Ethnicity
    WHEN a request is made to /Ethnicities
    THEN the response should contain json
    AND a JSON object for ethnicity should be in the json
    """
    ethnicity = {'ethnicity_id': 1, 'time_period': 201718,
                 'pct_total_ethnic_asian': 78,
                 'pct_total_ethnic_black': 81,
                 'pct_total_ethnic_white': 81,
                 'pct_total_ethnic_mixed_ethnicity': 82,
                 'pct_total_ethnic_other': 79,
                 'pct_total_ethnic_unknown': 78}
    response = client.get("/Ethnicities")
    assert response.headers["Content-Type"] == "application/json"
    assert ethnicity in response.json
```

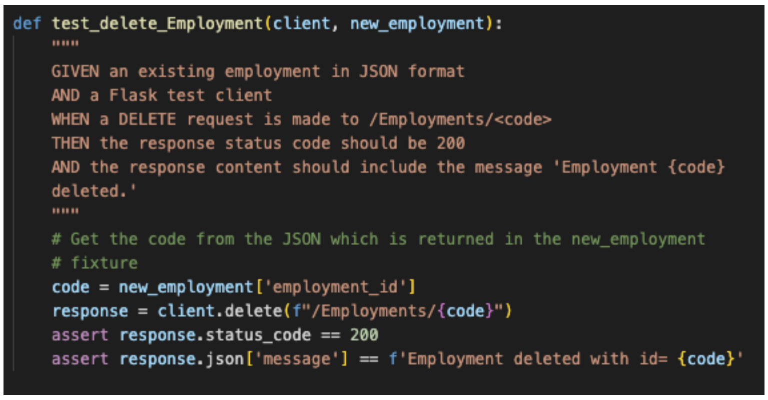


```python
def test_post_Ethnicity(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new ethnicity
    WHEN a POST request is made to /Ethnicities
    THEN the response status_code should be 200
    """
    # JSON to create a new ethnicity
    ethnicity_json = {'ethnicity_id': 61, 'time_period': 201718,
                      'pct_total_ethnic_asian': 78,
                      'pct_total_ethnic_black': 81,
                      'pct_total_ethnic_white': 81,
                      'pct_total_ethnic_mixed_ethnicity': 82,
                      'pct_total_ethnic_other': 79,
                      'pct_total_ethnic_unknown': 78}
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Ethnicities",
        json=ethnicity_json,
        content_type="application/json",
    )
    assert response.status_code == 200


def test_delete_Ethnicity(client, new_ethnicity):
    """
    GIVEN an existing ethnicity in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Ethnicities/<code>
    THEN the response status code should be 200
    AND the response content should include the message 'Ethnicity {code}
    deleted.'
    """
    # Get the code from the JSON which is returned in the new_ethnicity
    # fixture
    code = new_ethnicity['ethnicity_id']
    response = client.delete(f"/Ethnicities/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'Ethnicity deleted with id= {code}'
```

Employment:

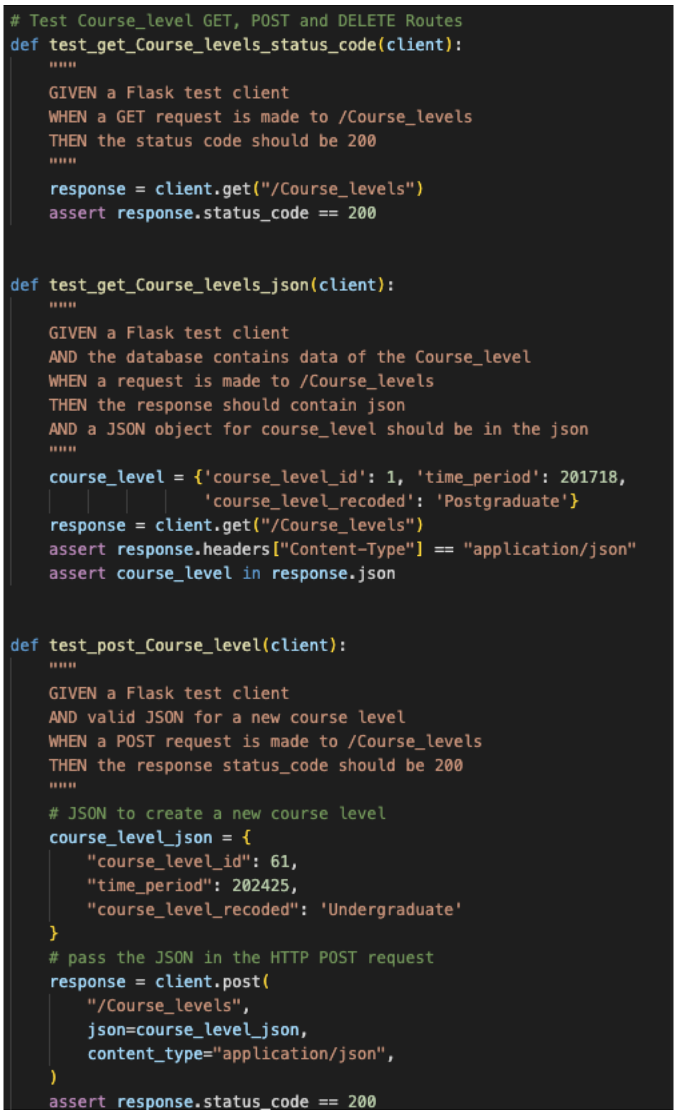


```python
# Test Employment GET, POST and DELETE Routes
def test_get_Employments_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Employments
    THEN the status code should be 200
    """
    response = client.get("/Employments")
    assert response.status_code == 200


def test_get_Employments_json(client):
    """
    GIVEN a Flask test client
    AND the database contains data of the Employment
    WHEN a request is made to /Employments
    THEN the response should contain json
    AND a JSON object for employment should be in the json
    """
    employment = {'employment_id': 1, 'time_period': 201718,
                  'employment_status': 'Teaching in a state-funded school'}
    response = client.get("/Employments")
    assert response.headers["Content-Type"] == "application/json"
    assert employment in response.json


def test_post_Employment(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new employment
    WHEN a POST request is made to /Employments
    THEN the response status_code should be 200
    """
    # JSON to create a new employment
    employment_json = {
        "employment_id": 61,
        "time_period": 202425,
        "employment_status": 'Teaching in a state-funded school'
    }
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Employments",
        json=employment_json,
        content_type="application/json",
    )
    assert response.status_code == 200
```

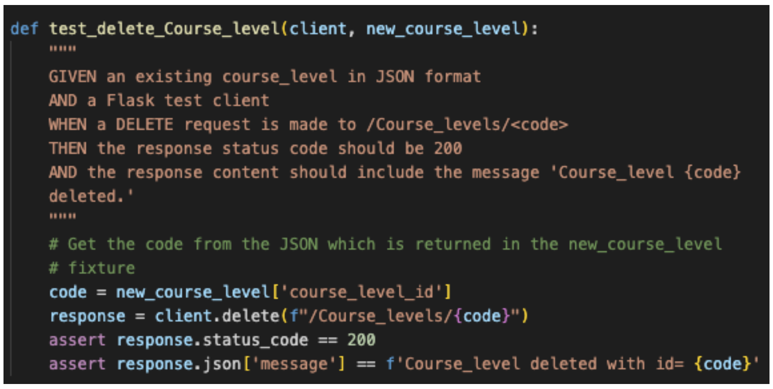


```python
def test_delete_Employment(client, new_employment):
    """
    GIVEN an existing employment in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Employments/<code>
    THEN the response status code should be 200
    AND the response content should include the message 'Employment {code}
    deleted.'
    """
    # Get the code from the JSON which is returned in the new_employment
    # fixture
    code = new_employment['employment_id']
    response = client.delete(f"/Employments/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'Employment deleted with id= {code}'
```

Course_level:

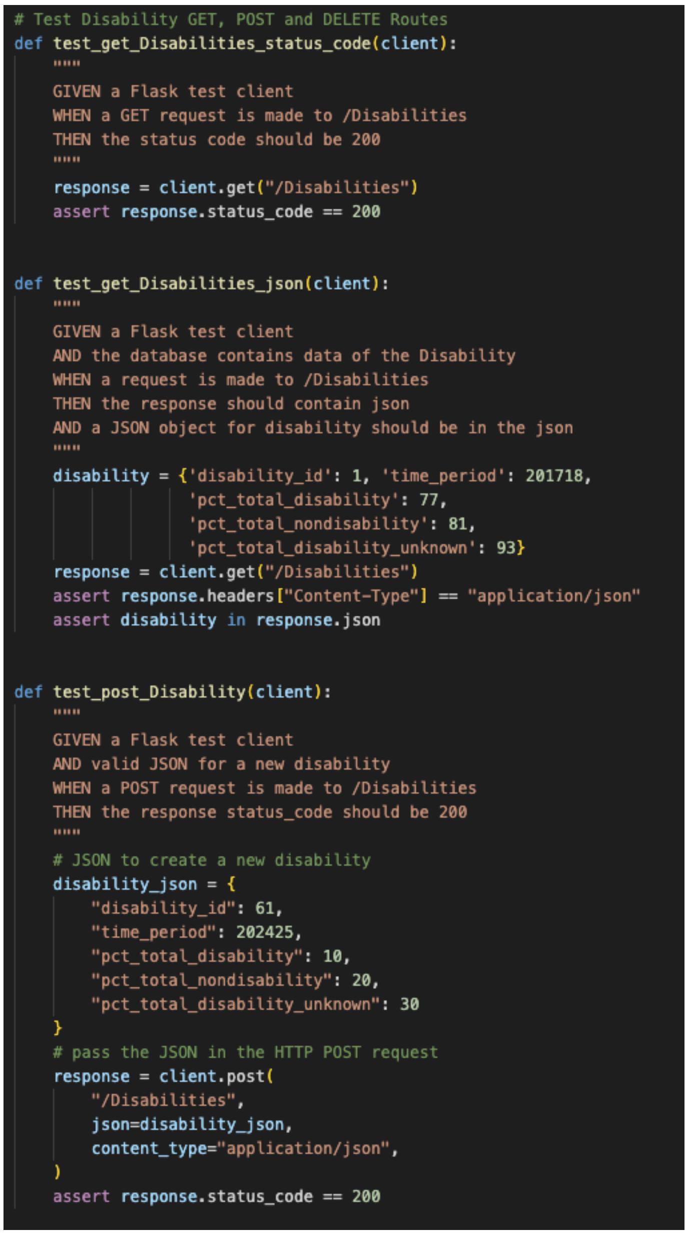


```python
# Test Course_level GET, POST and DELETE Routes
def test_get_Course_levels_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Course_levels
    THEN the status code should be 200
    """
    response = client.get("/Course_levels")
    assert response.status_code == 200


def test_get_Course_levels_json(client):
    """
    GIVEN a Flask test client
    AND the database contains data of the Course_level
    WHEN a request is made to /Course_levels
    THEN the response should contain json
    AND a JSON object for course_level should be in the json
    """
    course_level = {'course_level_id': 1, 'time_period': 201718,
                    'course_level_recoded': 'Postgraduate'}
    response = client.get("/Course_levels")
    assert response.headers["Content-Type"] == "application/json"
    assert course_level in response.json


def test_post_Course_level(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new course level
    WHEN a POST request is made to /Course_levels
    THEN the response status_code should be 200
    """
    # JSON to create a new course level
    course_level_json = {
        "course_level_id": 61,
        "time_period": 202425,
        "course_level_recoded": 'Undergraduate'
    }
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Course_levels",
        json=course_level_json,
        content_type="application/json",
    )
    assert response.status_code == 200
```

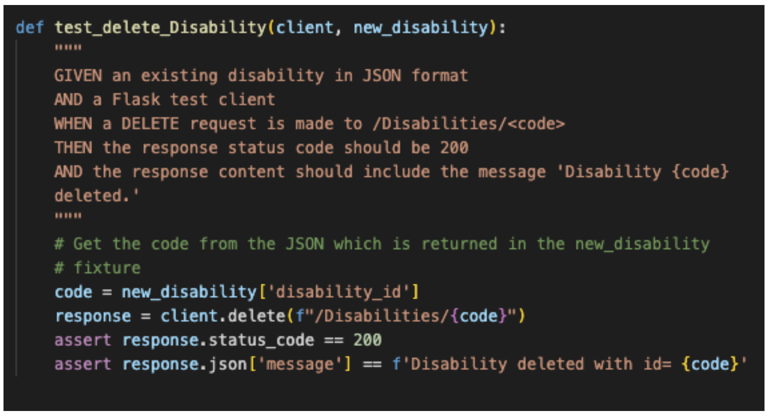


```python
def test_delete_Course_level(client, new_course_level):
    """
    GIVEN an existing course_level in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Course_levels/<code>
    THEN the response status code should be 200
    AND the response content should include the message 'Course_level {code}
    deleted.'
    """
    # Get the code from the JSON which is returned in the new_course_level
    # fixture
    code = new_course_level['course_level_id']
    response = client.delete(f"/Course_levels/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'Course_level deleted with id= {code}'
```

Disability:

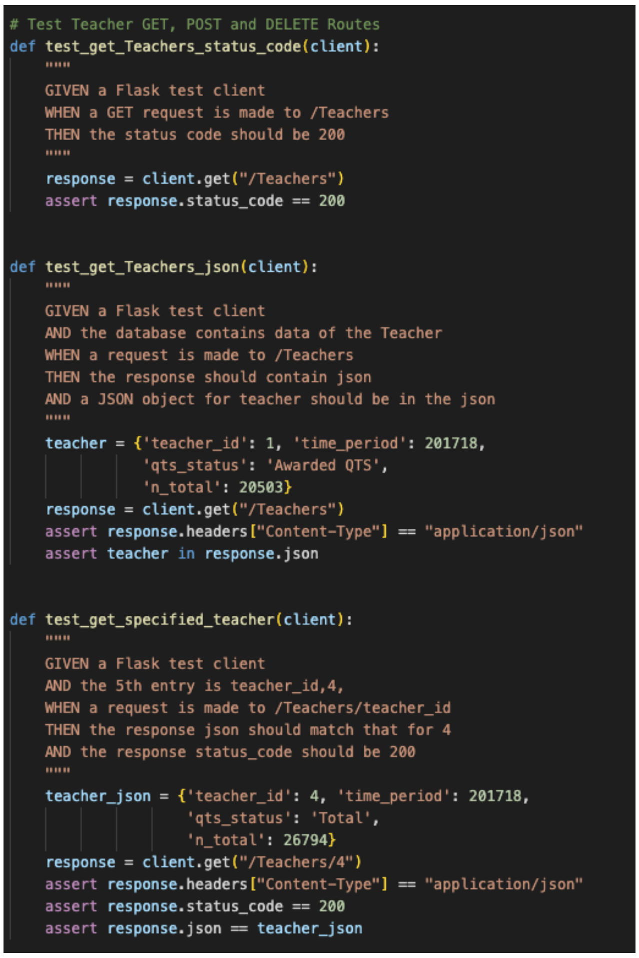


```python
# Test Disability GET, POST and DELETE Routes
def test_get_Disabilities_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Disabilities
    THEN the status code should be 200
    """
    response = client.get("/Disabilities")
    assert response.status_code == 200


def test_get_Disabilities_json(client):
    """
    GIVEN a Flask test client
    AND the database contains data of the Disability
    WHEN a request is made to /Disabilities
    THEN the response should contain json
    AND a JSON object for disability should be in the json
    """
    disability = {'disability_id': 1, 'time_period': 201718,
                  'pct_total_disability': 77,
                  'pct_total_nondisability': 81,
                  'pct_total_disability_unknown': 93}
    response = client.get("/Disabilities")
    assert response.headers["Content-Type"] == "application/json"
    assert disability in response.json


def test_post_Disability(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new disability
    WHEN a POST request is made to /Disabilities
    THEN the response status_code should be 200
    """
    # JSON to create a new disability
    disability_json = {
        "disability_id": 61,
        "time_period": 202425,
        "pct_total_disability": 10,
        "pct_total_nondisability": 20,
        "pct_total_disability_unknown": 30
    }
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Disabilities",
        json=disability_json,
        content_type="application/json",
    )
    assert response.status_code == 200
```

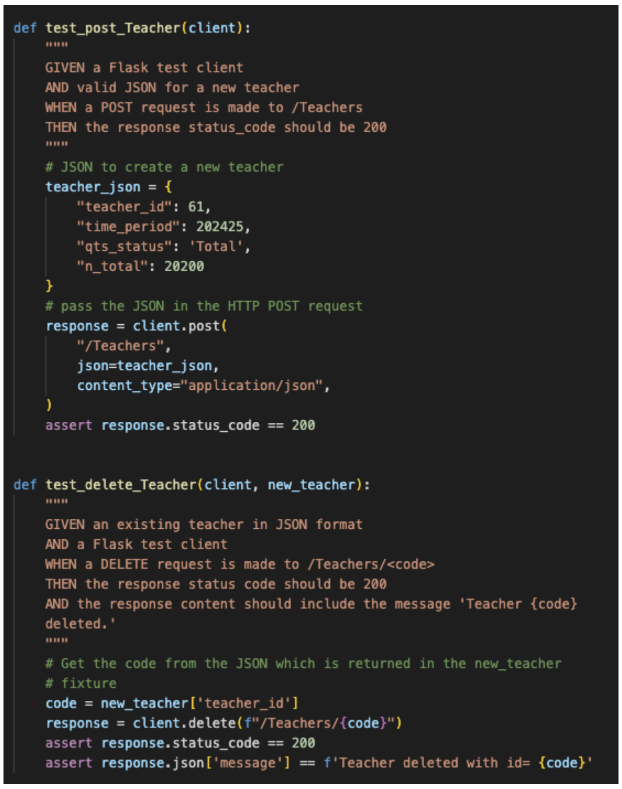


```python
def test_delete_Disability(client, new_disability):
    """
    GIVEN an existing disability in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Disabilities/<code>
    THEN the response status code should be 200
    AND the response content should include the message 'Disability {code}
    deleted.'
    """
    # Get the code from the JSON which is returned in the new_disability
    # fixture
    code = new_disability['disability_id']
    response = client.delete(f"/Disabilities/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'Disability deleted with id= {code}'
```

Teacher:



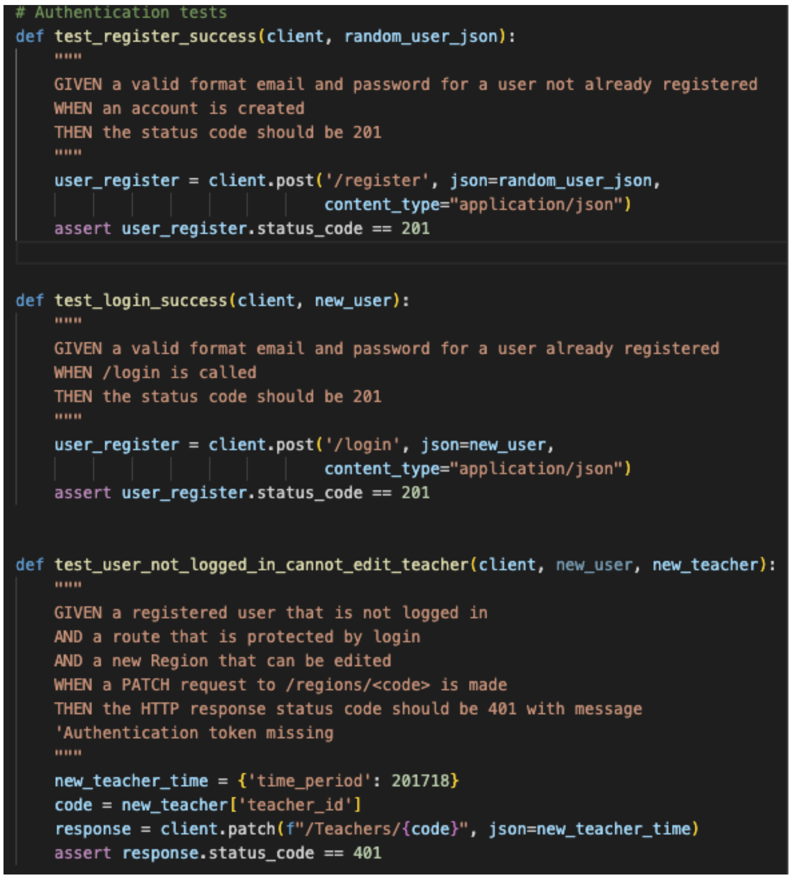
```python
# Test Teacher GET, POST and DELETE Routes
def test_get_Teachers_status_code(client):
    """
    GIVEN a Flask test client
    WHEN a GET request is made to /Teachers
    THEN the status code should be 200
    """
    response = client.get("/Teachers")
    assert response.status_code == 200


def test_get_Teachers_json(client):
    """
    GIVEN a Flask test client
    AND the database contains data of the Teacher
    WHEN a request is made to /Teachers
    THEN the response should contain json
    AND a JSON object for teacher should be in the json
    """
    teacher = {'teacher_id': 1, 'time_period': 201718,
               'qts_status': 'Awarded QTS',
               'n_total': 20503}
    response = client.get("/Teachers")
    assert response.headers["Content-Type"] == "application/json"
    assert teacher in response.json


def test_get_specified_teacher(client):
    """
    GIVEN a Flask test client
    AND the 5th entry is teacher_id,4,
    WHEN a request is made to /Teachers/teacher_id
    THEN the response json should match that for 4
    AND the response status_code should be 200
    """
    teacher_json = {'teacher_id': 4, 'time_period': 201718,
                    'qts_status': 'Total',
                    'n_total': 26794}
    response = client.get("/Teachers/4")
    assert response.headers["Content-Type"] == "application/json"
    assert response.status_code == 200
    assert response.json == teacher_json
```



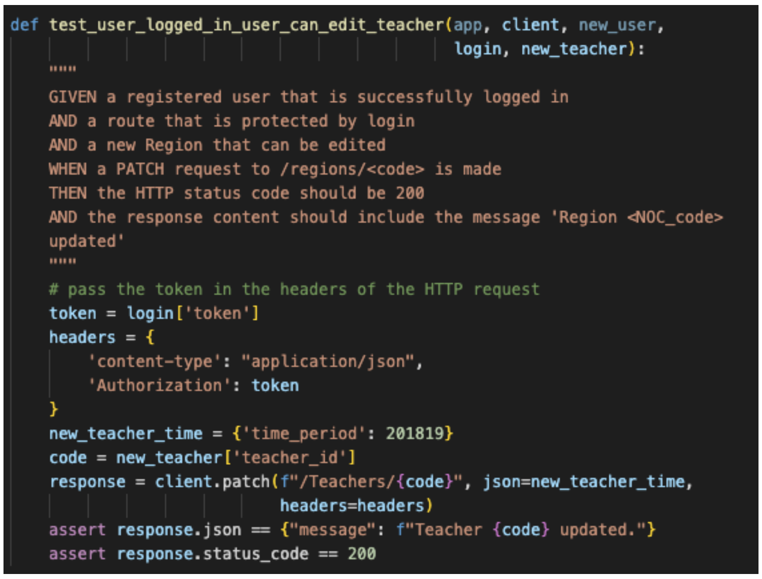
```python
def test_post_Teacher(client):
    """
    GIVEN a Flask test client
    AND valid JSON for a new teacher
    WHEN a POST request is made to /Teachers
    THEN the response status_code should be 200
    """
    # JSON to create a new teacher
    teacher_json = {
        "teacher_id": 61,
        "time_period": 202425,
        "qts_status": 'Total',
        "n_total": 20200
    }
    # pass the JSON in the HTTP POST request
    response = client.post(
        "/Teachers",
        json=teacher_json,
        content_type="application/json",
    )
    assert response.status_code == 200


def test_delete_Teacher(client, new_teacher):
    """
    GIVEN an existing teacher in JSON format
    AND a Flask test client
    WHEN a DELETE request is made to /Teachers/<code>
    THEN the response status code should be 200
    AND the response content should include the message 'Teacher {code}
    deleted.'
    """
    # Get the code from the JSON which is returned in the new_teacher
    # fixture
    code = new_teacher['teacher_id']
    response = client.delete(f"/Teachers/{code}")
    assert response.status_code == 200
    assert response.json['message'] == f'Teacher deleted with id= {code}'
```

1.2. test_auth.py

These tests are designed to validate critical functionalities related to user management and access control in a web application, specifically focusing on registration, login, and conditional access to editing resources based on authentication status.



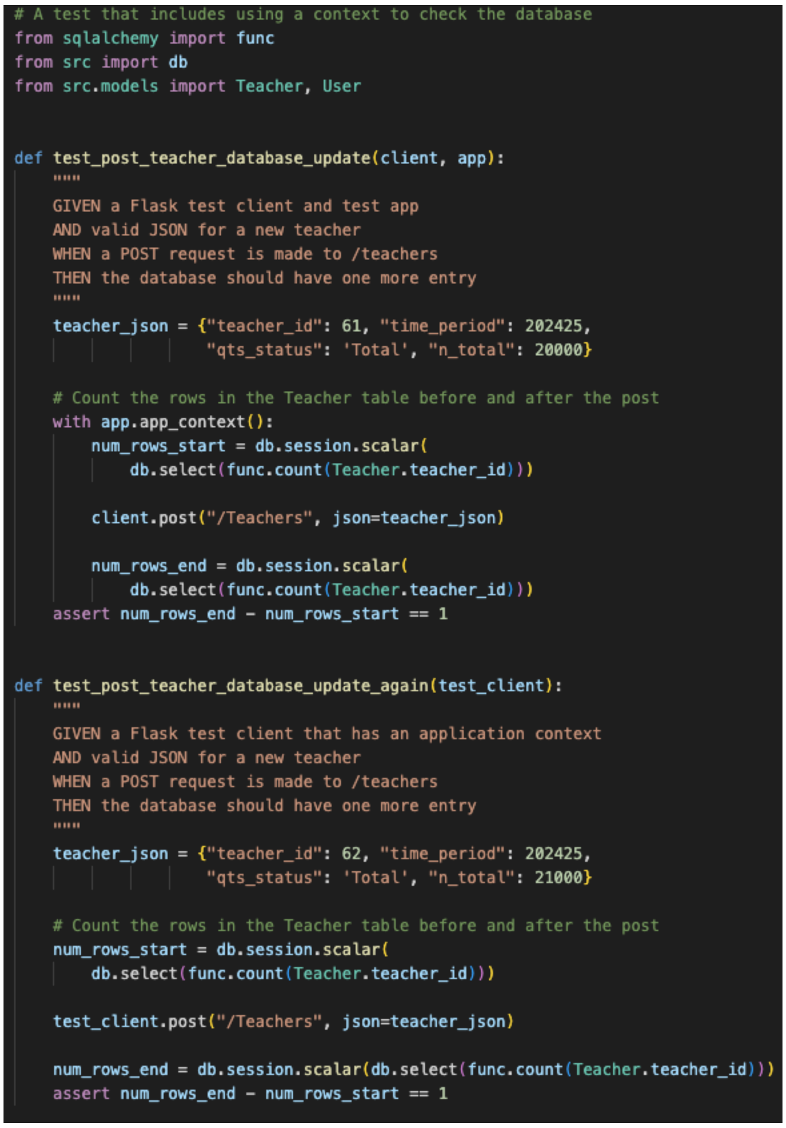
```python
# Authentication tests
def test_register_success(client, random_user_json):
    """
    GIVEN a valid format email and password for a user not already registered
    WHEN an account is created
    THEN the status code should be 201
    """
    user_register = client.post('/register', json=random_user_json,
                                content_type="application/json")
    assert user_register.status_code == 201


def test_login_success(client, new_user):
    """
    GIVEN a valid format email and password for a user already registered
    WHEN /login is called
    THEN the status code should be 201
    """
    user_register = client.post('/login', json=new_user,
                                content_type="application/json")
    assert user_register.status_code == 201


def test_user_not_logged_in_cannot_edit_teacher(client, new_user, new_teacher):
    """
    GIVEN a registered user that is not logged in
    AND a route that is protected by login
    AND a new Region that can be edited
    WHEN a PATCH request to /regions/<code> is made
    THEN the HTTP response status code should be 401 with message
    'Authentication token missing
    """
    new_teacher_time = {'time_period': 201718}
    code = new_teacher['teacher_id']
    response = client.patch(f"/Teachers/{code}", json=new_teacher_time)
    assert response.status_code == 401
```



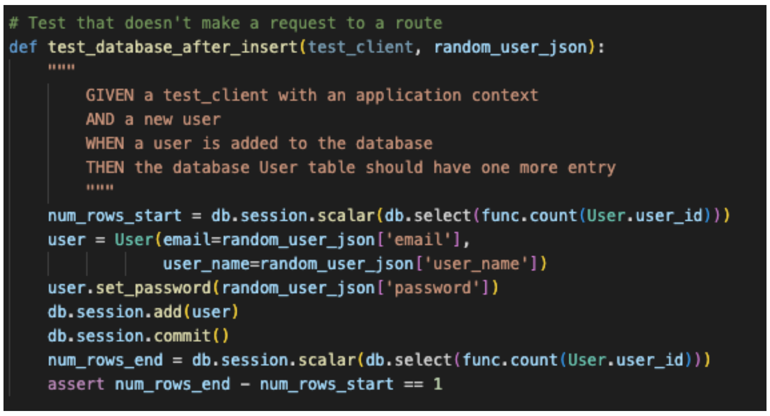
```python
def test_user_logged_in_user_can_edit_teacher(app, client, new_user,
                                              login, new_teacher):
    """
    GIVEN a registered user that is successfully logged in
    AND a route that is protected by login
    AND a new Region that can be edited
    WHEN a PATCH request to /regions/<code> is made
    THEN the HTTP status code should be 200
    AND the response content should include the message 'Region <NOC_code>
    updated'
    """
    # pass the token in the headers of the HTTP request
    token = login['token']
    headers = {
        'content-type': "application/json",
        'Authorization': token
    }
    new_teacher_time = {'time_period': 201819}
    code = new_teacher['teacher_id']
    response = client.patch(f"/Teachers/{code}", json=new_teacher_time,
                            headers=headers)
    assert response.json == {"message": f"Teacher {code} updated."}
    assert response.status_code == 200
```

1.3. test_database.py

These tests are designed to validate the functionality and integrity of a web application's database operations related to adding new entries, specifically focusing on teachers and users. They are performed within the context of a Flask application using SQLAlchemy for ORM (Object-Relational Mapping) and Flask's test client for simulating requests.



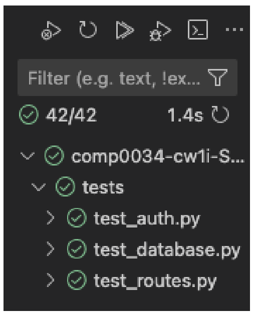
```python
# A test that includes using a context to check the database
from sqlalchemy import func
from src import db
from src.models import Teacher, User


def test_post_teacher_database_update(client, app):
    """
    GIVEN a Flask test client and test app
    AND valid JSON for a new teacher
    WHEN a POST request is made to /teachers
    THEN the database should have one more entry
    """
    teacher_json = {"teacher_id": 61, "time_period": 202425,
                    "qts_status": 'Total', "n_total": 20000}

    # Count the rows in the Teacher table before and after the post
    with app.app_context():
        num_rows_start = db.session.scalar(
            db.select(func.count(Teacher.teacher_id)))

        client.post("/Teachers", json=teacher_json)

        num_rows_end = db.session.scalar(
            db.select(func.count(Teacher.teacher_id)))
    assert num_rows_end - num_rows_start == 1


def test_post_teacher_database_update_again(test_client):
    """
    GIVEN a Flask test client that has an application context
    AND valid JSON for a new teacher
    WHEN a POST request is made to /teachers
    THEN the database should have one more entry
    """
    teacher_json = {"teacher_id": 62, "time_period": 202425,
                    "qts_status": 'Total', "n_total": 21000}

    # Count the rows in the Teacher table before and after the post
    num_rows_start = db.session.scalar(
        db.select(func.count(Teacher.teacher_id)))

    test_client.post("/Teachers", json=teacher_json)

    num_rows_end = db.session.scalar(db.select(func.count(Teacher.teacher_id)))
    assert num_rows_end - num_rows_start == 1
```



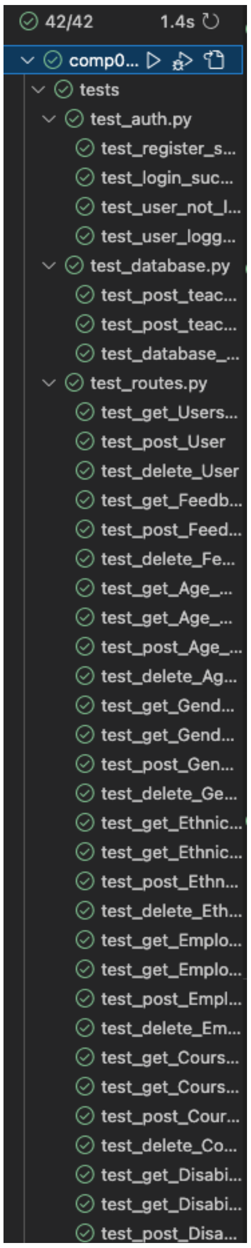
```python
# Test that doesn't make a request to a route
def test_database_after_insert(test_client, random_user_json):
    """
        GIVEN a test_client with an application context
        AND a new user
        WHEN a user is added to the database
        THEN the database User table should have one more entry
        """
    num_rows_start = db.session.scalar(db.select(func.count(User.user_id)))
    user = User(email=random_user_json['email'],
                user_name=random_user_json['user_name'])
    user.set_password(random_user_json['password'])
    db.session.add(user)
    db.session.commit()
    num_rows_end = db.session.scalar(db.select(func.count(User.user_id)))
    assert num_rows_end - num_rows_start == 1
```

1.4. Evidence that the tests have been run (screenshot)

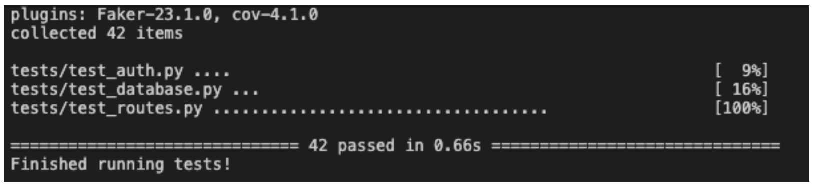
By using pytest:



```
Filter (e.g. text, !ex...
42/42          1.4s
comp0034-cw1i-S...
  tests
    test_auth.py
    test_database.py
    test_routes.py
```



```
42/42          1.4s
comp0...
  tests
    test_auth.py
      test_register_s...
      test_login_suc...
      test_user_not_l...
      test_user_logg...
    test_database.py
      test_post_teac...
      test_post_teac...
      test_database_...
    test_routes.py
      test_get_Users...
      test_post_User
      test_delete_User
      test_get_Feedb...
      test_post_Feed...
      test_delete_Fe...
      test_get_Age_...
      test_get_Age_...
      test_post_Age_...
      test_delete_Ag...
      test_get_Gend...
      test_get_Gend...
      test_post_Gen...
      test_delete_Ge...
      test_get_Ethnic...
      test_get_Ethnic...
      test_post_Ethn...
      test_delete_Eth...
      test_get_Emplo...
      test_get_Emplo...
      test_post_Empl...
      test_delete_Em...
      test_get_Cours...
      test_get_Cours...
      test_post_Cour...
      test_delete_Co...
      test_get_Disabi...
      test_get_Disabi...
      test_post_Disa...
```



```
plugins: Faker-23.1.0, cov-4.1.0
collected 42 items

tests/test_auth.py ....                                                    [  9%]
tests/test_database.py ...                                                 [ 16%]
tests/test_routes.py ...................................                   [100%]

============================== 42 passed in 0.66s ==============================
Finished running tests!
```

2. Use of tools and techniques

2.1. URL of my GitHub repository

https://github.com/ucl-comp0035/comp0034-cw1i-SHOX1ie.git

2.2. README.md

```
# COMP0034 Coursework 1 2023/24
COMP0034 Coursework 1 starter repository

## Prerequisites

Before you begin, ensure you have met the following requirements:
* You have installed the latest version of [Python](https://www.python.org/downloads/) (the version your application is compatible with, e.g., Python 3.11.1).
* You have a Windows/Linux/Mac machine.
* You have read [any prerequisites or related documentation].

## Installing

To install , follow these steps:

Linux and macOS:
git clone:
https://github.com/ucl-comp0035/comp0034-cw1i-SHOX1ie.git

python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt

## Run the app
flask --app src run --debug

## Test the codes by pytest
```

2.3. requirements.txt

```
Flask
Flask-SQLAlchemy
Flask-Marshmallow
marshmallow-sqlalchemy
bcrypt
pandas
```

```
pytest
selenium
pytest-cov
PyJWT
faker
pyarrow
```

3. References

Acknowledgement of the use of AI for suggestions about coding errors.

Data set is same as used for COMP0035 coursework.

Code references refer to the tutorials on Moodle.

No books, papers, websites etc used.