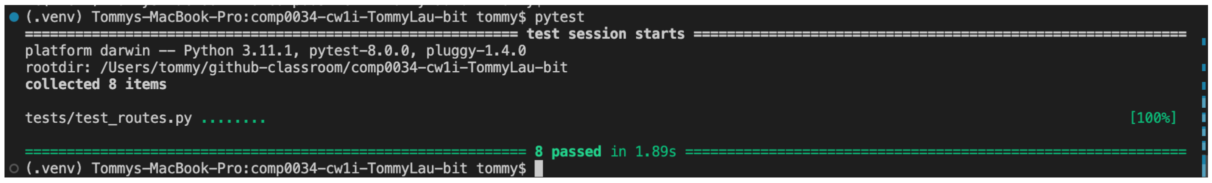
Testing:

```python
# Add a new User
def test_add_user(client):
    """Test creating a new user."""
    user_data = {"email": "newuser@example.com", "password": "securepassword"}
    response = client.post("/users", json=user_data)
    assert response.status_code == 201
    assert "id" in response.json

# Get User by ID
def test_get_user(client):
    """Test retrieving an existing user."""
    user_id = 1  # Assuming this user exists
    response = client.get(f"/users/{user_id}")
    assert response.status_code == 200
    assert response.json['email'] == "newuser@example.com"  # Example assertion

# Update User
def test_update_user(client):
    """Test updating an existing user."""
    user_id = 1  # Assuming this user exists
    update_data = {"email": "updateduser@example.com"}
    response = client.put(f"/users/{user_id}", json=update_data)
    assert response.status_code == 200
    assert response.json['email'] == "updateduser@example.com"

# Delete User
def test_delete_user(client):
    """Test deleting an existing user."""
    user_id = 1  # Assuming this user exists
    response = client.delete(f"/users/{user_id}")
    assert response.status_code == 200
    # Further assertions can check if the user is actually removed

# Add a new Case
def test_add_case(client):
    """Test creating a new case."""
    case_data = {"case_type": "Example", "status": "Open", "user_id": 1}  #
Assuming user_id 1 exists
    response = client.post("/cases", json=case_data)
    assert response.status_code == 201
    assert "id" in response.json

# Get Case by ID
def test_get_case(client):
    """Test retrieving an existing case."""
    case_id = 1  # Assuming this case exists
    response = client.get(f"/cases/{case_id}")
    assert response.status_code == 200
    assert response.json['case_type'] == "Example"  # Example assertion
```

```python
# Update Case
def test_update_case(client):
    """Test updating an existing case."""
    case_id = 1  # Assuming this case exists
    update_data = {"status": "Closed"}
    response = client.put(f"/cases/{case_id}", json=update_data)
    assert response.status_code == 200
    assert response.json['status'] == "Closed"

# Delete Case
def test_delete_case(client):
    """Test deleting an existing case."""
    case_id = 1  # Assuming this case exists
    response = client.delete(f"/cases/{case_id}")
    assert response.status_code == 200
    # Further assertions can check if the case is actually removed
```

Evidence:



```
(.venv) Tommys-MacBook-Pro:comp0034-cw1i-TommyLau-bit tommy$ pytest
============================================================ test session starts ============================================================
platform darwin -- Python 3.11.1, pytest-8.0.0, pluggy-1.4.0
rootdir: /Users/tommy/github-classroom/comp0034-cw1i-TommyLau-bit
collected 8 items

tests/test_routes.py ........                                                                                                        [100%]

============================================================= 8 passed in 1.89s =============================================================
(.venv) Tommys-MacBook-Pro:comp0034-cw1i-TommyLau-bit tommy$
```

Tools and technique:
URL: https://github.com/ucl-comp0035/comp0034-cw1i-TommyLau-bit.git

Reference:

- Use of AI for ‘from flask_migrate import Migrate’