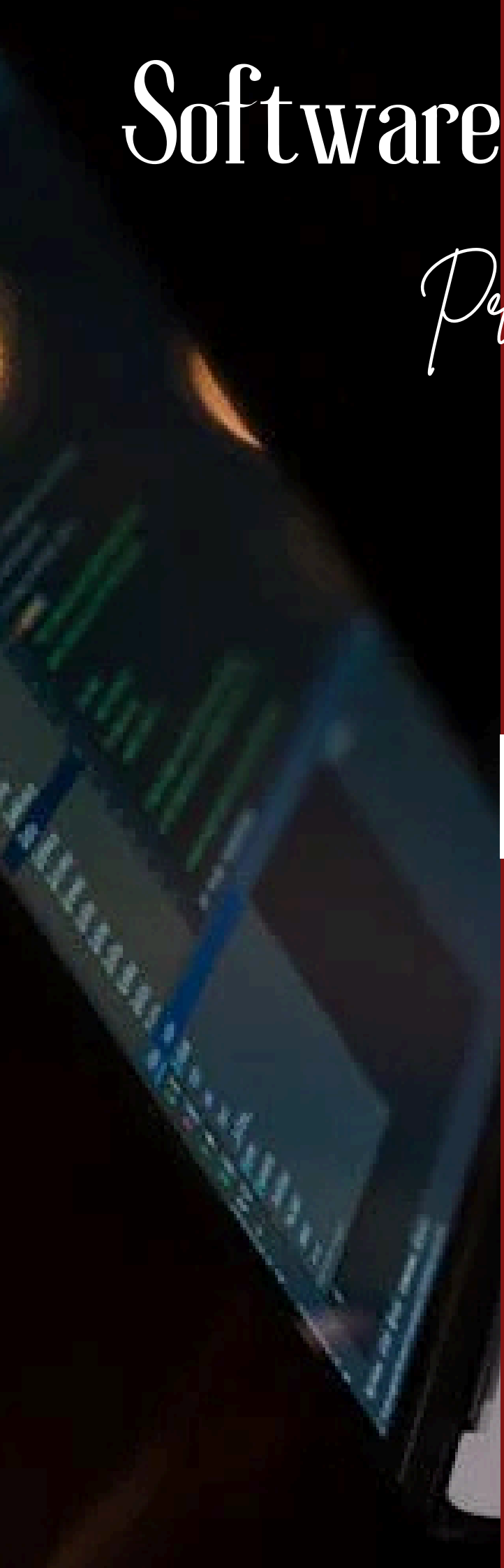


Software Engineering

Project

TEAM 16

Prepared By:

Anish Maity (21f3000417)
Achin Aggarwal (22f1001390)
Atharva Sarbhukan (22f1000533)
Abhishek Darji (22f1000678)
Sarath Sasidharan Pillai (22f1000152)
Rituparno Sen (21f1004275)
Dr. Ambrish Naik (22f2000424)

PyTest TESTING

INTRODUCTION

This document outlines the testing strategy for the AI-based academic guidance system using **pytest**, a powerful testing framework for **Python.** The system is designed to assist students by providing academic guidance, answering queries, and recommending relevant resources.

Given the AI-driven nature of the system, testing is crucial to ensure accuracy, reliability, and efficiency. Our approach involves **unit testing, integration testing, and functional testing** to validate different components, including:

- **AI Model Accuracy:** Ensuring responses are contextually correct and relevant.
- **API Integrations:** Verifying seamless communication with external data sources.
- **User Query Handling:** Testing various input scenarios for robustness.
- **Performance & Edge Cases:** Checking response times and system behavior under stress.

By leveraging **pytest**, we automate test execution, generate reports, and streamline debugging, ensuring the system meets quality standards before deployment.

TESTING FRAMEWORK

The testing framework for this project is built upon Python's **pytest**, a powerful and flexible testing tool known for its ease of use and scalability. This framework is designed to ensure the **accuracy, reliability, and efficiency** of the AI-based academic guidance system.

The testing strategy includes the following:

- **Unit Tests:** Verifying individual functions, such as text processing, query classification, and API interactions.
- **Integration Tests:** Ensuring seamless interactions between system components, including the AI model, database, and external APIs.
- **System Tests:** Validating the entire application workflow, from user input to AI-generated responses.
- **Performance Testing:** Evaluating response times and system behavior under high query loads.
- **Edge Case Handling:** Testing how the system responds to incomplete, ambiguous, or unexpected inputs.

To streamline testing, the framework integrates with **continuous integration (CI) pipelines**, allowing for automated test execution and early detection of issues. This ensures that the AI agent consistently delivers accurate and relevant academic guidance while maintaining high performance across different scenarios.

TEST CASES

1. ADMIN API's

a.) Admin Login (SUCCESS)

ENDPOINT : POST /login

- **Test Case Name :- test_admin_login**

INPUT :-

```
{
  "username": "admin@example.com",
  "password": "admin123"
}
```

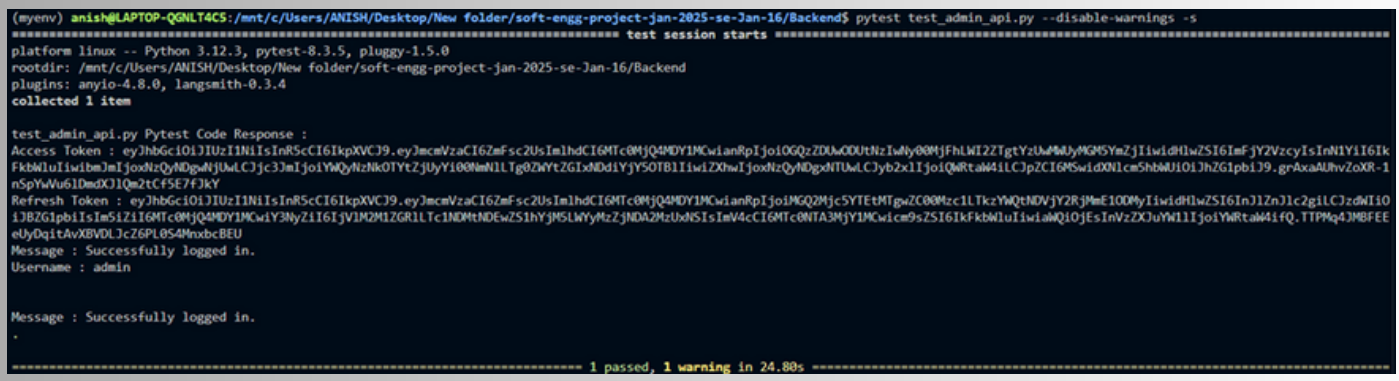
EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Successfully loged in." }

PyTest Code :-

```python
def test_admin_login(client):
    """Test admin login API."""
    response = client.post("/login", json={"username": "admin@example.com", "password": "admin123"})

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Code Response : ')
    print('Access Token :',json_data['access_token'])
    print('Refresh Token :' ,json_data['refresh_token'])
    print('Message :',json_data['message'])
    print('Username :',json_data['username'])
    assert "access_token" in json_data
    assert json_data["message"] == "Successfully logged in."
    print('\n')
    print('Message :',json_data['message'])
```

Actual Output :-



b.) Admin Login (Missing Fields)

ENDPOINT : POST /login

- **Test Case Name :- test_login_invalid_password**

INPUT :-

```
{
  "username": "admin@example.com",
  "password": "wrongpassword"
}
```

EXPECTED OUTPUT :-

HTTP 200 , JSON { "message": "Invalid Credentials." }

PyTest Code :-

```python
def test_login_invalid_password(client):
    """Test login with incorrect password."""
    response = client.post("/login", json={"username": "admin@example.com", "password": "wrongpassword"})

    assert response.status_code == 200  # Your API returns 200 even for invalid credentials
    json_data = response.get_json()
    print('Pytest Code Response:\n',json_data ,'\n')
    assert json_data["message"] == "Invalid credentials"
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Pytest Code Response:
 {'message': 'Invalid credentials'}

.

============================== 1 passed, 1 warning in 22.68s ===============================
```

c.) Admin Login (Missing Fields)

ENDPOINT : POST /login

- **Test Case Name :- test_login_nonexistent_user**

INPUT :-

```
{
  "username": "nonexistent@example.com",
  "password": "wrongpassword"
}
```

EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Invalid Credentials." }

PyTest Code :-

```python
def test_login_nonexistent_user(client):
    """Test login with a non-existing username."""
    response = client.post("/login", json={"username": "nonexistent@example.com", "password": "admin123"})

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Code Response :\n',json_data ,'\n')
    assert json_data["message"] == "Invalid credentials"
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Pytest Code Response :
 {'message': 'Invalid credentials'}

.

============================== 1 passed, 1 warning in 22.01s ===============================
```

d.) Admin Login (Missing Fields)

ENDPOINT : POST /login

- **Test Case Name :- test_login_missing_username**

INPUT :-

```
{
   "password": "wrongpassword"
}
```

EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Invalid Credentials." }

PyTest Code :-

```python
def test_login_missing_username(client):
    """Test login with missing username."""
    response = client.post("/login", json={"password": "admin123"})

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Code Response :\n',json_data ,'\n')
    assert json_data["message"] == "Invalid credentials"
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4CS:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Pytest Code Response :
 {'message': 'Invalid credentials'}

.

======================== 1 passed, 1 warning in 24.18s =========================
```

e.) Admin Login (Missing Fields)

ENDPOINT : POST /login

- Test Case Name :- test_login_empty_credentials

INPUT :-

{ }

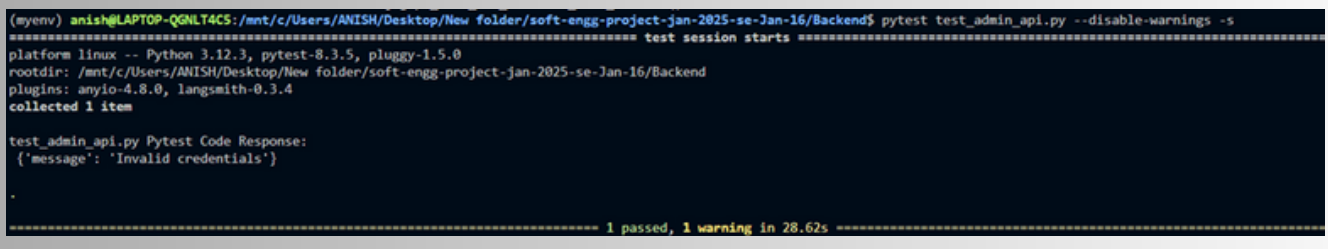
EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Invalid Credentials." }

PyTest Code :-

```python
def test_login_empty_credentials(client):
    """Test login with empty credentials."""
    response = client.post("/login", json={})

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Code Response:\n',json_data ,'\n')
    assert json_data["message"] == "Invalid credentials"
```

Actual Output :-



f.) Admin Login (Missing Fields)

ENDPOINT : POST /login

- **Test Case Name :- test_login_invalid_json**

INPUT :-

data = “invalid_json”, content_type = “application/json”

EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Invalid Credentials." }

PyTest Code :-

```python
def test_login_invalid_json(client):
    """Test login with invalid JSON format."""
    response = client.post("/login", data="invalid_json", content_type="application/json")

    assert response.status_code == 400  # Assuming Flask handles bad JSON format errors
    json_data = response.get_json()

    print('Pytest Code Response : \n',json_data ,'\n')
    assert "message" in json_data
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Pytest Code Response:
 {'message': 'Invalid credentials'}

.

======================== 1 passed, 1 warning in 28.62s =========================
```

g.) Admin Details (SUCCESS)

ENDPOINT : GET /admin_details

- **Test Case Name :- test_admin_details**

EXPECTED OUTPUT :-
HTTP 200

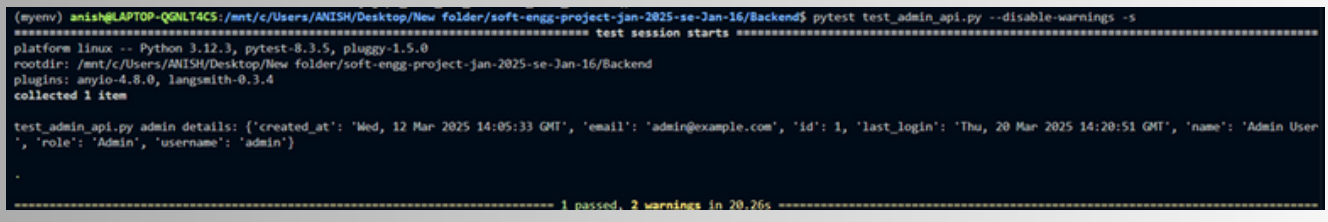
PyTest Code :-

```python
def test_admin_details(client, auth_header):
    """Test fetching admin details API."""
    response = client.get("/admin_details", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    print('admin details:',json_data,'\n')

    assert json_data["username"] == "admin"
    assert json_data["email"] == "admin@example.com"
    assert json_data["role"].lower() == "admin"
```

Actual Output :-



h.) Admin Details (SUCCESS)

ENDPOINT : GET /admin_details

- **Test Case Name :- test_admin_details_unauthorized**

EXPECTED OUTPUT :-
HTTP 401 , JSON { "message": "Missing Authorization Header." }

PyTest Code :-

```python
def test_admin_details_unauthorized(client):
    """Test fetching admin details without authentication."""
    response = client.get("/admin_details")  # No auth header provided

    assert response.status_code == 401  # Expected: Unauthorized
    json_data = response.get_json()
    print("Pytest Code Response :",json_data)
    assert "msg" in json_data  # Typical Flask-JWT error response
    assert json_data["msg"] == "Missing Authorization Header"
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Pytest Code Response : {'msg': 'Missing Authorization Header'}
.

============================== 1 passed, 1 warning in 23.49s ===============================
```

i.) Admin Details (Missing Fields)

ENDPOINT : GET /admin_details

- **Test Case Name :- test_admin_details_invalid_token**

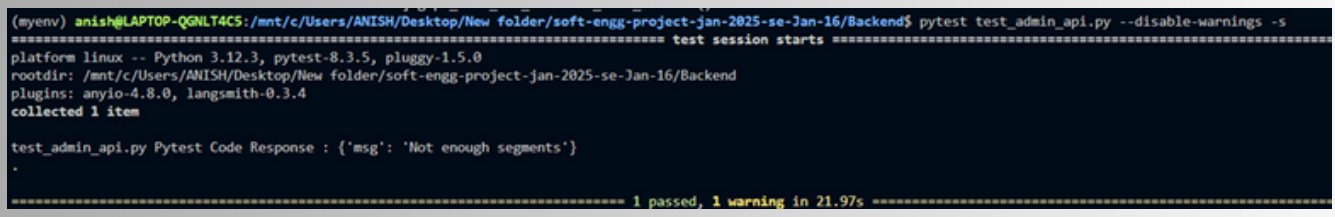
EXPECTED OUTPUT :-
HTTP 422 , JSON { "message": "Not enough segments." }

PyTest Code :-

```python
def test_admin_details_invalid_token(client):
    """Test fetching admin details with an invalid token."""
    response = client.get("/admin_details", headers={"Authorization": "Bearer invalid_token"})

    assert response.status_code == 422  # Expected: Unauthorized
    json_data = response.get_json()
    print("Pytest Code Response :", json_data)
```

Actual Output :-



j.) Admin Details (SUCCESS)

ENDPOINT : GET /admin_courses

- **Test Case Name :- test_get_all_courses**

EXPECTED OUTPUT :-
HTTP 200

PyTest Code :-

```python
def test_get_all_courses(client, auth_header):
    """Test fetching all courses API."""
    response = client.get("/admin_courses", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    print("Pytest Code Response :", json_data,'\n')

    assert "courses" in json_data
    assert isinstance(json_data["courses"], list)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4CS:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Pytest Code Response : {'courses': [{'created_at': 'Wed, 12 Mar 2025 14:13:17 GMT', 'description': 'This is a machine learning course.', 'id': 1, 'name': 'MLP', 'total_
assignments': 0, 'total_materials': 3, 'total_students': 1}, {'created_at': 'Wed, 12 Mar 2025 14:13:47 GMT', 'description': 'This is a Business management course.', 'id': 2, 'name': 'BDM
', 'total_assignments': 0, 'total_materials': 0, 'total_students': 1}, {'created_at': 'Wed, 12 Mar 2025 16:10:58 GMT', 'description': 'aaaaaaaaaaa', 'id': 3, 'name': 'aaaaaaa', 'total_as
signments': 0, 'total_materials': 0, 'total_students': 0}, {'created_at': 'Tue, 18 Mar 2025 11:12:37 GMT', 'description': 'h', 'id': 4, 'name': 'd', 'total_assignments': 0, 'total_materi
als': 0, 'total_students': 0}, {'created_at': 'Tue, 18 Mar 2025 11:21:56 GMT', 'description': 'gggggg', 'id': 5, 'name': 'ddddddddd', 'total_assignments': 0, 'total_materials': 2, 'tot
al_students': 0}, {'created_at': 'Tue, 18 Mar 2025 11:40:07 GMT', 'description': 'cccc', 'id': 6, 'name': 'Advanced Python', 'total_assignments': 0, 'total_materials': 2, 'total_students
': 0}, {'created_at': 'Thu, 20 Mar 2025 11:49:03 GMT', 'description': 'Intro to Python', 'id': 7, 'name': 'Python 101', 'total_assignments': 0, 'total_materials': 0, 'total_students': 0)
, {'created_at': 'Thu, 20 Mar 2025 11:49:18 GMT', 'description': 'Intro to Python', 'id': 8, 'name': 'Python 101', 'total_assignments': 0, 'total_materials': 0, 'total_students': 0}, {'c
reated_at': 'Thu, 20 Mar 2025 11:54:11 GMT', 'description': 'Intro to Python', 'id': 9, 'name': 'Python 101', 'total_assignments': 0, 'total_materials': 0, 'total_students': 0}, {'create
d_at': 'Thu, 20 Mar 2025 13:03:46 GMT', 'description': 'Intro to Python', 'id': 10, 'name': 'Python 101', 'total_assignments': 0, 'total_materials': 0, 'total_students': 0}, {'created_at
': 'Thu, 20 Mar 2025 13:10:58 GMT', 'description': 'Intro to Python', 'id': 11, 'name': 'Python 101', 'total_assignments': 0, 'total_materials': 0, 'total_students': 0}]}

.

============================== 1 passed, 1 warning in 21.25s ===============================
```

k.) Admin Details (SUCCESS)

ENDPOINT : GET /admin_students

- **Test Case Name :- test_get_all_students**

EXPECTED OUTPUT :-
HTTP 200

PyTest Code :-

```python
def test_get_all_students(client, auth_header):
    """Test fetching all students API (admin access only)."""
    response = client.get("/admin_students", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    print("Students:", json_data,'\n')

    assert "students" in json_data
    assert isinstance(json_data["students"], list)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Students: {'students': [{'created_at': 'Wed, 12 Mar 2025 14:27:39 GMT', 'email': 'student1@gmail.com', 'id': 2, 'name': 'Student1', 'total_courses': 2, 'username': 'stu
dent1'}]}

.

============================== 1 passed, 2 warnings in 23.71s ==============================
```

l.) Admin Details (SUCCESS)

ENDPOINT : POST /add_course

- **Test Case Name :- test_add_course**

INPUT :-

```
{
  "name": "Python 101",
  "description": "Intro to Python"
}
```

EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Course added successfully!" }

PyTest Code :-

```python
def test_add_course(client, auth_header):
    """Test adding a new course."""
    response = client.post("/add_course", json={"name": "Python 101", "description": "Intro to Python"}, headers=auth_header)
    print("Pytest Response :",response.get_json(),'\n')
    assert response.status_code == 201
    assert response.get_json()["message"] == "Course added successfully!"
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4CS:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Pytest Response : {'message': 'Course added successfully!'}

.

======================== 1 passed, 2 warnings in 20.28s ========================
```

m.) Admin Details (SUCCESS)

ENDPOINT : PUT /edit_course/<int:course_id>

- **Test Case Name :- test_add_course**

INPUT :-

```
{
  "name": "Advanced Python Hack"
}
```

EXPECTED OUTPUT :-

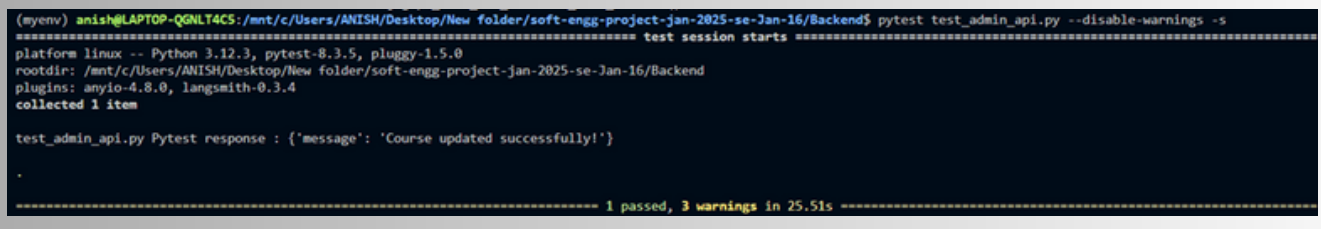
HTTP 200 , JSON { "message": "Course updated successfully!" }

PyTest Code :-

```python
def test_edit_course(client, auth_header):
    """Test editing an existing course."""
    response = client.put("/edit_course/6", json={"name": "Advanced Python Hack"}, headers=auth_header)
    print("Pytest response :",response.get_json(),'\n')

    assert response.status_code == 200
    assert response.get_json()["message"] == "Course updated successfully!"
```

Actual Output :-



n.) Admin Details (SUCCESS)

ENDPOINT : POST /add_course/<int:course_id>/material

- Test Case Name :- test_add_course

INPUT :-

```
{
  "week_number": 2,
  "title": "Introduction",
  "material_link": "http://example.com/video1"
}
```

EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Course material added successfully!" }

PyTest Code :-

```python
def test_add_course_material(client, auth_header):
    response = client.post("/add_course/5/material", json={"week_number": 2, "title": "Introduction", "material_link": "http://example.com/video1"}, headers=auth_header)
    print("Pytest response:",response.get_json(),'\n')

    assert response.status_code == 200
    assert response.get_json()["message"] == "Course material added successfully!"
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend$ pytest test_admin_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/Backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_admin_api.py Pytest response: {'message': 'Course material added successfully!'}

.

======================== 1 passed, 3 warnings in 23.26s ========================
```

2. STUDENT API's

a.) Student SignUp (SUCCESS)

ENDPOINT : POST /signup

- **Test Case Name :- test_successful_student_signup**

INPUT :-

```
{
  "name": "John",
  "email": "johndoe1@example.com",
  "password": "password123",
  "role": "Student"
}
```

EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Student registered successfully!" }

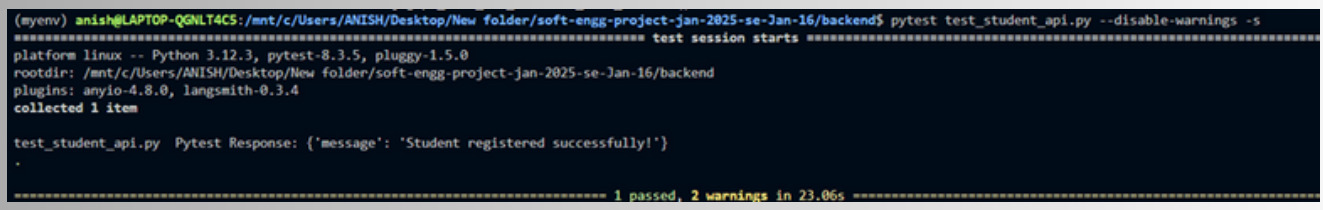
PyTest Code :-

```python
def test_successful_student_signup(client, db_session):
    """Test successful student signup"""
    payload = {
        "name": "John",
        "email": "johndoe1@example.com",
        "password": "password123",
        "role": "Student"
    }

    response = client.post("/signup", json=payload)
    print( " Pytest Response:", response.get_json())

    assert response.status_code == 200
    assert response.get_json()["message"] == "Student registered successfully!"
```

Actual Output :-



b.) Student SignUp (Missing Fields)

ENDPOINT : POST /signup

- **Test Case Name :- test_signup_missing_fields**

INPUT :-

```
{
  "email": "user@example.com",
  "password": "testpass",
  "role": "Student"
}
```

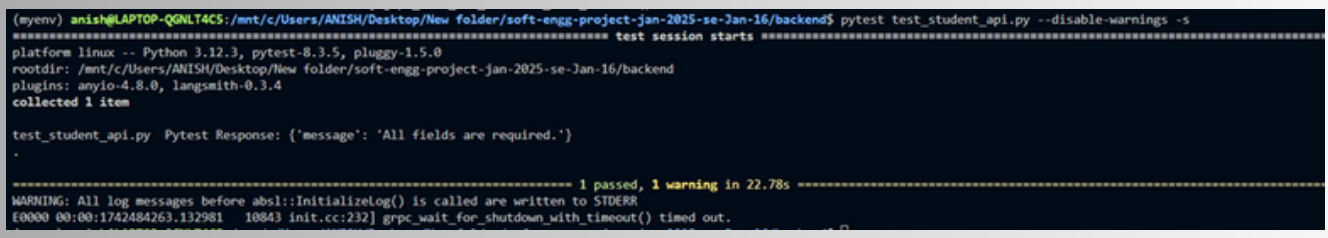
EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "All fields are required." }

PyTest Code :-

```python
def test_signup_missing_fields(client):
    """Test signup with missing required fields"""
    payload = {
        "email": "user@example.com",
        "password": "testpass",
        "role": "Student"
    }

    response = client.post("/signup", json=payload)
    print( " Pytest Response:", response.get_json())
    assert response.status_code == 200
    assert response.get_json()["message"] == "All fields are required."
```

Actual Output :-



c.) Student SignUp (Invalid Role)

ENDPOINT : POST /signup

- **Test Case Name :- test_signup_invalid_role**

INPUT :-

```
{
  "name": "Unknown Role",
  "email": "unknown@example.com",
  "password": "testpass",
  "role": "Manager"
}
```

EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Role 'Manager' not found in database " }

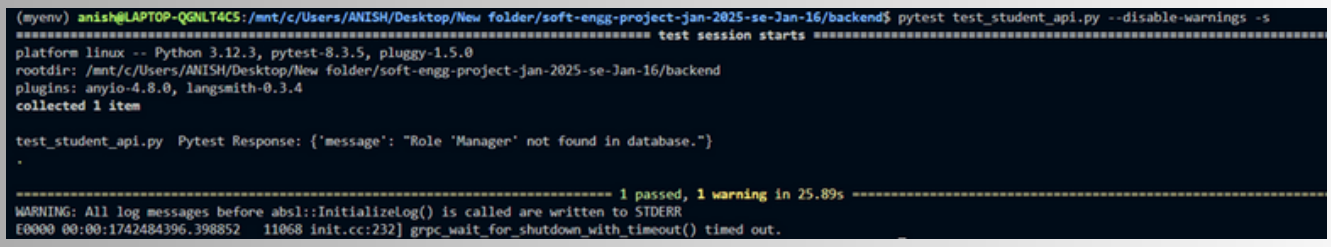
PyTest Code :-

```python
def test_signup_invalid_role(client):
    """Test signup with an invalid role"""
    payload = {
        "name": "Unknown Role",
        "email": "unknown@example.com",
        "password": "testpass",
        "role": "Manager"
    }

    response = client.post("/signup", json=payload)
    print( " Pytest Response:", response.get_json())

    assert response.status_code == 200
    assert response.get_json()["message"] == "Role 'Manager' not found in database."
```

Actual Output :-



d.) Student LogIn (SUCCESS)

ENDPOINT : POST /login

- **Test Case Name :- test_student_login**

INPUT :-

```
{
  "username": "student1@gmail.com",
  "password": "123456"
}
```

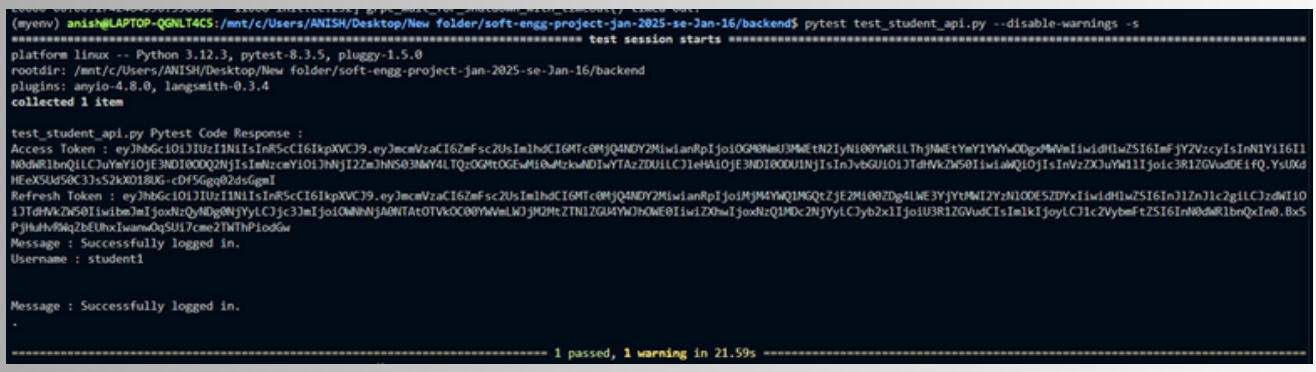
EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Successfully logged in." }

PyTest Code :-

```python
def test_student_login(client):
    """Test admin login API."""
    response = client.post("/login", json={"username": "student1@gmail.com", "password": "123456"})

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Code Response : ')
    print('Access Token :',json_data['access_token'])
    print('Refresh Token :' ,json_data['refresh_token'])
    print('Message :',json_data['message'])
    print('Username :',json_data['username'])
    assert "access_token" in json_data
    assert json_data["message"] == "Successfully logged in."
    print('\n')
    print('Message :',json_data['message'])
```

Actual Output :-



e.) Student Profile (SUCCESS)

ENDPOINT : GET /student_profile

- **Test Case Name :- test_student_profile**

EXPECTED OUTPUT :-

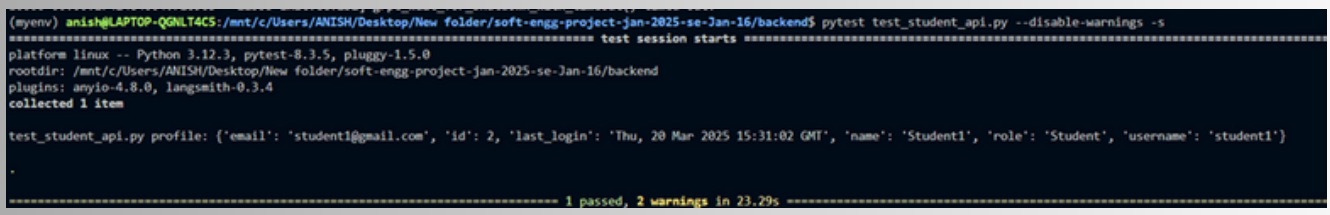
HTTP 200

PyTest Code :-

```python
def test_student_profile(client, auth_header):
    """Test the Student Profile API."""
    response = client.get("/student_profile", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    print("profile:",json_data ,'\n')
```

Actual Output :-



f.) Student Dashboard (SUCCESS)

ENDPOINT : GET /student_dashboard

- **Test Case Name :- test_student_dashboard**

EXPECTED OUTPUT :-
HTTP 200

PyTest Code :-

```python
def test_student_dashboard(client, auth_header):
    """Test the Student Dashboard API."""
    response = client.get("/student_dashboard", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    print("dashbord",json_data)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_student_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_student_api.py dashbord {'courses_data': [{'assignments': [], 'created_at': 'Wed, 12 Mar 2025 14:13:17 GMT', 'description': 'This is a machine learning course.', 'id': 1, 'live_sess
ions': [], 'materials': [{'created_at': 'Wed, 12 Mar 2025 14:20:40 GMT', 'id': 1, 'title': 'Pandas Series and DataFrame 1', 'video_link': 'https://youtu.be/MdnmbjKM7a0', 'week_number': 1
}, {'created_at': 'Wed, 12 Mar 2025 14:20:40 GMT', 'id': 2, 'title': 'Pandas Series and DataFrame 1', 'video_link': 'https://youtu.be/MdnmbjKM7a0', 'week_number': 1}, {'created_at': 'Wed
, 12 Mar 2025 14:26:36 GMT', 'id': 3, 'title': 'Pandas Series and DataFrame 1', 'video_link': 'https://youtu.be/MdnmbjKM7a0', 'week_number': 1}], 'name': 'MLP', 'supplementary_materials'
: []}, {'assignments': [], 'created_at': 'Wed, 12 Mar 2025 14:13:47 GMT', 'description': 'This is a Business management course.', 'id': 2, 'live_sessions': [], 'materials': [], 'name': '
BDM', 'supplementary_materials': []}]}
.

============================== 1 passed, 4 warnings in 24.52s ==============================
```

g.) Student Refresh Token (SUCCESS)

ENDPOINT : POST /token_refresh

- **Test Case Name :- test_refresh_token**

EXPECTED OUTPUT :-
HTTP 200

PyTest Code :-

```python
def test_refresh_token(client,auth_header):
    """Test the Refresh Token API."""
    response = client.post("/token_refresh", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    assert "access_token" in json_data
    print("New Access Token:", json_data["access_token"])
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4CS:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_student_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_student_api.py New Access Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTc0MjQ4NjI0MiwianRpIjoiNmQ0OWQyNGItOWI0OS00Yzc2LTk3ZTEtMjE3NzNkOWM2ZTkwIiwidHlwZSI6ImFjY2VzcyIsInN1YiI6IlN0dWR1bnQiLCJuYmYiOjE3NDI0ODYyNDIsImNzcmYiOiIwZjU4NDV1Zi0yNWU5LTQzZGItYWM4YS0yYmM3OTAwZGNkNjgiLCJleHAiOjE3NDI0ODcxNDIsInJvbGUiOiJTdHVkZW50IiwiaWQiOjIsInVzZXJuYW1lIjoic3R1ZGVudDEifQ.GbH-o5wDU0rqCJy661vkcp3TCuyQkbRGB8e1P5gEs14
.

============================== 1 passed, 2 warnings in 18.29s ==============================
```

h.) Student LogOut (SUCCESS)

ENDPOINT : POST /logout

- **Test Case Name :- test_refresh_token**

EXPECTED OUTPUT :-
HTTP 200, JSON { "message": "Logout successful" }

PyTest Code :-

```python
def test_logout(client, auth_header):
    """Test the Logout API."""
    response = client.post("/logout", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    assert json_data["message"] == "Logout successful"
    print("Logout Response:", json_data)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4CS:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_student_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_student_api.py Logout Response: {'message': 'Logout successful'}
.

============================== 1 passed, 1 warning in 21.21s ===============================
```

3. PROFESSOR API's

a.) Professor LogIn (SUCCESS)

ENDPOINT : POST /login

- **Test Case Name :- test_professor_login**

INPUT :-

```
{
  "username": "professor1@gmail.com",
  "password": "123456"
}
```

EXPECTED OUTPUT :-

HTTP 200 , JSON { "message": "Successful logged in." }

PyTest Code :-

```python
def test_professor_login(client):
    """Test admin login API."""
    response = client.post("/login", json={"username": "professor1@gmail.com", "password": "123456"})

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Code Response : ')
    print('Access Token :',json_data['access_token'])
    print('Refresh Token :' ,json_data['refresh_token'])
    print('Message :',json_data['message'])
    print('Username :',json_data['username'])
    assert "access_token" in json_data
    assert json_data["message"] == "Successfully logged in."
    print('\n')
    print('Message :',json_data['message'])
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4CS:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_professor_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_professor_api.py Pytest Code Response :
Access Token : eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTc0MjQ4ODAwOCwianRpIjoiN2YyNjVmODktOTcwNy00YWE0LTgyNmItODM0ZWQzMDM1YjAwIiwidHlwZSI6ImFjY2VzcyIsInN1YiI6IlByb2Zlc3NvciIsIm5iZiI6MTc0MjQ4ODAwOCwiY3NyZiI6ImJmMTRlMGE0LTY5NzQtNDRiMi04NjgSLTE2MWIwNTE5NjQ4NSIsImV4cCI6MTc0MjQ4ODkwOCwicm9sZSI6IlByb2Zlc3NvciIsImlkIjozLCJ1c2VybmFtZSI6InByb2Zlc3NvcjEifQ.L2H0pyvzsjhZpIwdqLFJ0GvSN8SlhXmtGKniYY-qIXk
Refresh Token : eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTc0MjQ4ODAwOCwianRpIjoiNTk5NWFiY2MtNjQ5OS00NjlmLThhYTMtMjJlNTk5MzVjNDFhIiwidHlwZSI6InJlZnJlc2giLCJzdWIiOiJQcm9mZXNzb3IiLCJuYmYiOjE3NDI0ODgwMDgsImNzcmYiOiJiMTM0Y2UwZi04ZWQzLTQ5NTAtODFjZC04ZjBjMmE5YjA5NDkiLCJleHAiOjE3NDUwODA4MDgsInJvbGUiOiJQcm9mZXNzb3IiLCJpZCI6MywidXNlcm5hbWUiOiJwcm9mZXNzb3IxIn0.WzpGXbLbwRbSMMUvSA-eF-u6rrAWp2OpqS1LFC0-Sk4
Message : Successfully logged in.
Username : professor1


Message : Successfully logged in.
.

============================== 1 passed, 1 warning in 18.97s ===============================
```

b.) Professor Details (SUCCESS)

ENDPOINT : GET /professor_details

- **Test Case Name :- test_professor_details**

EXPECTED OUTPUT :-
HTTP 200

PyTest Code :-

```python
def test_professor_details(client, auth_header):
    """Test fetching admin details API."""
    response = client.get("/professor_details", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Response:',json_data,'\n')
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_professor_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_professor_api.py Pytest Response: {'created_at': 'Tue, 18 Mar 2025 14:47:27 GMT', 'email': 'professor1@gmail.com', 'id': 3, 'last_login': 'Thu, 20 Mar 2025 13:16:14 GMT', 'name': 'P
rofessor User', 'role': 'Professor', 'username': 'professor1'}

.

======================== 1 passed, 2 warnings in 18.24s ========================
```

c.) Professor - Pending Instructors (SUCCESS)

ENDPOINT : GET /pending_instructors

- **Test Case Name :- test_pending_instructors**

EXPECTED OUTPUT :-
HTTP 200

PyTest Code :-

```python
def test_pending_instructors(client, auth_header):
    """Test the Pending Instructors API."""
    response = client.get("/pending_instructor", headers=auth_header)

    assert response.status_code in [200, 404]  # Either success or professor not found
    json_data = response.get_json()

    if response.status_code == 200:
        assert isinstance(json_data, list)  # The response should be a list
        if json_data:  # If there are pending instructors
            for req in json_data:
                assert "id" in req
                assert "instructor_id" in req
                assert "status" in req
                assert "created_at" in req
        print("Pending Instructors:", json_data)
    else:
        assert json_data["message"] == "Professor not found"
        print("Professor Not Found:", json_data)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4CS:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_professor_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_professor_api.py Pending Instructors: [{'created_at': 'Thu, 20 Mar 2025 15:12:42 GMT', 'id': 6, 'instructor_id': 13, 'status': 'Pending'}, {'created_at': 'Thu, 20 Mar 2025 15:21:35
GMT', 'id': 7, 'instructor_id': 16, 'status': 'Pending'}]
.

======================== 1 passed, 2 warnings in 18.52s ========================
```

d.) Professor - Approve Instructors (SUCCESS)

ENDPOINT : PUT /approve_instructor/<int:request_id>'

- **Test Case Name :- test_approve_instructor**

INPUT :-

```
{
  "status": "Approved"
}
```

EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Instructor request approved successfully " }

PyTest Code :-

```python
def test_approve_instructor(client, auth_header):
    """Test the Approve Instructor API."""
    payload = {"status": "Approved"}  # Change to "Rejected" to test rejection flow
    response = client.put(f"/approve_instructor/6", json=payload, headers=auth_header)

    assert response.status_code in [200, 400, 404]  # Expect success, bad request, or not found
    json_data = response.get_json()

    if response.status_code == 200:
        assert "message" in json_data
        assert json_data["message"] in [f"Instructor request approved successfully",
                                        f"Instructor request rejected successfully"]
        print("Approve Instructor Response:", json_data)

    elif response.status_code == 400:
        assert json_data["message"] == "Invalid status, use 'Approved' or 'Rejected'"
        print("Invalid Status Error:", json_data)

    elif response.status_code == 404:
        assert json_data["message"] in ["Professor not found", "Instructor request not found"]
        print("Not Found Error:", json_data)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_professor_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_professor_api.py Approve Instructor Response: {'message': 'Instructor request approved successfully'}
.

============================== 1 passed, 3 warnings in 21.18s ==============================
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1742488620.626662   16592 init.cc:232] grpc_wait_for_shutdown_with_timeout() timed out.
```

e.) Professor - Add Lesson (SUCCESS)

ENDPOINT : POST /add_suplementary

- **Test Case Name :- test_add_lesson**

INPUT :-

```
{
  "course_id": 1,
  "material_type": "Video",
  "content": "https://example.com/lesson.mp4"
}
```

EXPECTED OUTPUT :-

HTTP 200 , JSON { "message": "Lesson added successfully " }

PyTest Code :-

```python
def test_add_lesson(client, auth_header):
    """Test the Add Lesson API."""
    payload = {
        "course_id": 1,  # Replace with a valid course_id from your test database
        "material_type": "Video",  # Example material type
        "content": "https://example.com/lesson.mp4"  # Example content (URL or text)
    }

    response = client.post("/add_suplementary", json=payload, headers=auth_header)

    assert response.status_code in [201, 400]  # Expect success or bad request
    json_data = response.get_json()

    if response.status_code == 201:
        assert json_data["message"] == "Lesson added successfully"
        print("Lesson Added Successfully:", json_data)

    elif response.status_code == 400:
        assert json_data["message"] == "Missing required fields"
        print("Missing Fields Error:", json_data)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_professor_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_professor_api.py Lesson Added Successfully: {'message': 'Lesson added successfully'}
.

============================== 1 passed, 1 warning in 16.84s ===============================
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1742488883.093717   17115 init.cc:232] grpc_wait_for_shutdown_with_timeout() timed out.
```

4. INSTRUCTOR API's

a.) Instructor SignUp (SUCCESS)

ENDPOINT : POST /signup

- **Test Case Name :- test_successful_instructor_signup**

INPUT :-

```
{
  "name": "Jane Smith1",
  "email": "janesmit1h@example.com",
  "password": "securepass",
  "role": "Instructor"
}
```

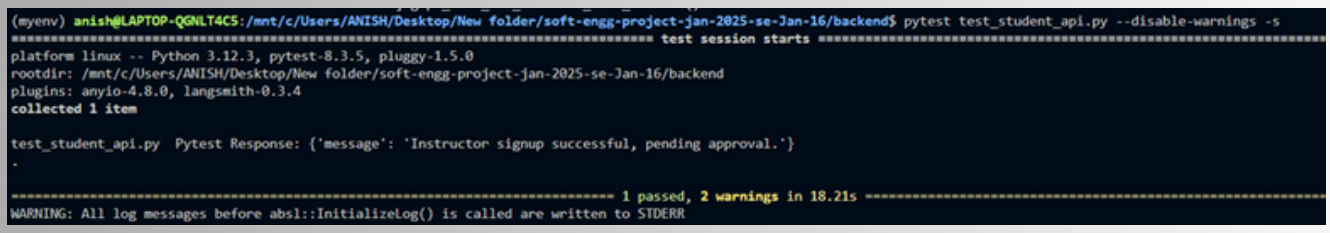
EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Instructor signup successful, pending approval." }

PyTest Code :-

```python
def test_successful_instructor_signup(client, db_session):
    """Test successful instructor signup (pending approval)"""
    payload = {
        "name": "Jane Smith1",
        "email": "janesmit1h@example.com",
        "password": "securepass",
        "role": "Instructor"
    }

    response = client.post("/signup", json=payload)
    print( " Pytest Response:", response.get_json())
    assert response.status_code == 200
    assert response.get_json()["message"] == "Instructor signup successful, pending approval."
```

Actual Output :-



b.) Instructor LogIn (SUCCESS)

ENDPOINT : POST /login

- **Test Case Name :- test_inst_login**

INPUT :-

```
{
  "username": "int5@gmail.com",
  "password": "123456"
}
```

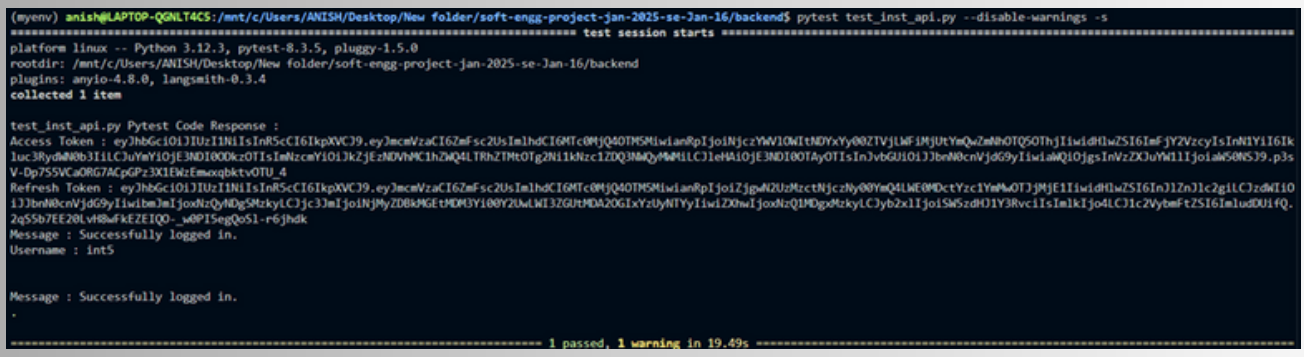
EXPECTED OUTPUT :-
HTTP 200 , JSON { "message": "Successfully logged in ." }

PyTest Code :-

```python
def test_inst_login(client):
    """Test admin login API."""
    response = client.post("/login", json={"username": "int5@gmail.com", "password": "123456"})

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Code Response : ')
    print('Access Token :',json_data['access_token'])
    print('Refresh Token :' ,json_data['refresh_token'])
    print('Message :',json_data['message'])
    print('Username :',json_data['username'])
    assert "access_token" in json_data
    assert json_data["message"] == "Successfully logged in."
    print('\n')
    print('Message :',json_data['message'])
```

Actual Output :-



c.) Instructor Details (SUCCESS)

ENDPOINT : GET /instructor_details

- **Test Case Name :- test_inst_details**

EXPECTED OUTPUT :-

HTTP 200

PyTest Code :-

```python
def test_inst_details(client, auth_header):
    """Test fetching admin details API."""
    response = client.get("/instructor_details", headers=auth_header)

    assert response.status_code == 200
    json_data = response.get_json()
    print('Pytest Response:',json_data,'\n')
```

Actual Output :-

d.) Instructor - Top Support Queries (SUCCESS)

ENDPOINT : GET /topquery

- **Test Case Name :- test_top_support_queries**

EXPECTED OUTPUT :-
HTTP 200

PyTest Code :-

```python
def test_top_support_queries(client, auth_header):
    """Test the Top Support Queries API."""
    response = client.get("/topquery", headers=auth_header)

    assert response.status_code in [200, 404]  # Expect success or no queries found
    json_data = response.get_json()

    if response.status_code == 200:
        assert isinstance(json_data, list)  # The response should be a list of queries
        assert all("query_text" in query and "count" in query for query in json_data)
        print("Top Support Queries Response:", json_data)

    elif response.status_code == 404:
        assert json_data["message"] == "No queries found"
        print("No Queries Found:", json_data)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_inst_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_inst_api.py Top Support Queries Response: [{'count': 3, 'query_text': 'axios'}, {'count': 2, 'query_text': 'what is axios'}, {'count': 2, 'query_text': 'how to print in python'}, {'count': 1, 'query_text': 'what is profit'}, {'count': 1, 'query_text': 'what is decorator function'}]
.

======================== 1 passed, 1 warning in 18.96s =========================
```

e.) Instructor - Add Live Lessons (SUCCESS)

ENDPOINT : POST /add_livesession

- Test Case Name :- test_top_support_queries

INPUT :-

```
{
  "course_id": 1,
  "yt_link": "https://www.youtube.com/watch?v=dQw4w9WgXcQ",
  "description": "Live session on Python Basics"
}
```

EXPECTED OUTPUT :-
HTTP 200, JSON { "message": "Live session added successfully" }

PyTest Code :-

```python
def test_add_live_session(client, auth_header):
    """Test the Add Live Session API."""
    payload = {
        "course_id": 1,
        "yt_link": "https://www.youtube.com/watch?v=dQw4w9WgXcQ",
        "description": "Live session on Python Basics"
    }

    response = client.post("/add_livesession", json=payload, headers=auth_header)

    assert response.status_code in [201, 400]  # Expect success or missing fields error
    json_data = response.get_json()

    if response.status_code == 201:
        assert json_data["message"] == "Live session added successfully"
        print("Live Session Added:", json_data)

    elif response.status_code == 400:
        assert json_data["message"] == "Missing required fields"
        print("Missing Fields Error:", json_data)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4C5:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_inst_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_inst_api.py Live Session Added: {'message': 'Live session added successfully'}
.

============================== 1 passed, 1 warning in 18.54s ===============================
```

5. CHATBOT API's

a.) ChatBot (SUCCESS)

ENDPOINT : POST /chat

- **Test Case Name :- test_chatbot_api**

INPUT :-

```
{
  "query": "What is recursion?"
}
```

EXPECTED OUTPUT :-
HTTP 200

PyTest Code :-

```python
def test_chatbot_api(client, auth_header):
    """Test the Chatbot API."""
    payload = {"query": "What is recursion?"}

    response = client.post("/chat", json=payload, headers=auth_header)

    assert response.status_code in [200, 400, 500]  # Expect success, bad request, or server error
    json_data = response.get_json()

    if response.status_code == 200:
        assert json_data["query"] == payload["query"]
        assert "response" in json_data
        assert "references" in json_data
        assert isinstance(json_data["references"], list)
        print("Chatbot Response:", json_data)

    elif response.status_code == 400:
        assert json_data["error"] == "Query cannot be empty"
        print("Empty Query Error:", json_data)

    elif response.status_code == 500:
        print("Server Error:", json_data)
```

Actual Output :-

```
(myenv) anish@LAPTOP-QGNLT4CS:/mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend$ pytest test_student_api.py --disable-warnings -s
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /mnt/c/Users/ANISH/Desktop/New folder/soft-engg-project-jan-2025-se-Jan-16/backend
plugins: anyio-4.8.0, langsmith-0.3.4
collected 1 item

test_student_api.py Chatbot Response: {'query': 'What is recursion?', 'references': [{'doc_type': 'Transcript', 'subject': 'MAD2', 'week': 'Week-2'}, {'doc_type': 'Transcript', 'subject'
: 'MAD1', 'week': 'Week-5'}, {'doc_type': 'Lecture', 'subject': 'MAD1', 'week': 'Week-6'}], 'response': 'Hint:  Recursion is a programming concept where a function calls itself within it
s own definition.\n\nResource Guide: Check MAD1 Week-5 for more details.  The provided transcript discusses functions, return values, and mentions a "recursive function without a base ca
se," which suggests this week\'s material might offer a more complete explanation of recursion.'}
.

============================== 1 passed, 2 warnings in 24.17s ==============================
```