Software Engineering Project
Team – 08

Milestone – 5
Test Cases & Test Suite of the project

Testing User API

Test: To create a new user
API being tested: http://127.0.0.1:5000/api/user
Inputs

- Request Method: POST
- JSON: {“user_name”: “testuser”, “email”: “testuser@gmail.com”, “password”: “12345678”}
- Header:

Expected Output

- HTTP Status Code: 201
- JSON: {“username”: “testuser”}

Actual Output

- HTTP Status Code: 201
- JSON: {“username”: “testuser”}

Result: **PASSED**
Function Code:

```
def test_create_user_api(client,db_init):
    #Act
    user={
        "user_name":"testuser",
        "email":"testuser@gmail.com",
        "password":"12345678"
    }
    response = client.post('/api/user', json=user)

    #Assertion
    assert response.status_code==201
    assert response.json['username']=='testuser'
```

Test: To delete an existing user
API being tested: http://127.0.0.1:5000/api/user/1
Inputs

- Request Method: DELETE
- JSON: {“user”: “1”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“username”: “testuser”}

Actual Output

- HTTP Status Code: 200
- JSON: {“username”: “testuser”}

Result: **PASSED**

Function Code:

```
def test_delete_user_api(client,db_init):
    #Act
    response = client.delete('/api/user/1')

    #Assertion
    assert response.status_code==200
```

Test: To update user details

API being tested: http://127.0.0.1:5000/api/user

Inputs

- Request Method: POST/PUT
- JSON: {“user_name”: “testuser”, “email”: “testuser@gmail.com”, “password”: “12345678”}
- Header:

Expected Output

- HTTP Status Code: 201/200
- JSON: {“username”: “testuser”, “email”: “test@gmail.com”}

Actual Output

- HTTP Status Code: 201/200
- JSON: {“username”: “testuser”, “email”: “test@gmail.com”}

Result: **PASSED**

Function Code:

```python
def test_update_user_api(client,db_init):
    #Act
    user={
        "user_name":"testuser",
        "email":"testuser@gmail.com",
        "password":"12345678"
    }
    response = client.post('/api/user',json=user)

    #Assertion
    assert response.status_code==201
    assert response.json['username']=='testuser'

    user_update={
        "user_name":"testuser",
        "email":"test@gmail.com",
        "password":"12345678"
    }

    response = client.put('/api/user/1',json=user_update)
    assert response.status_code==200
    assert response.json['email']=='test@gmail.com'
```

Test: To get a user's detail
API being tested: http://127.0.0.1:5000/api/user
Inputs

- Request Method: GET
- JSON: {"user_name": "testuser", "email": "testuser@gmail.com", "password": "12345678"}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {"username": "testuser", "email": "testuser@gmail.com"}

Actual Output

- HTTP Status Code: 200
- JSON: {"username": "testuser", "email": "testuser@gmail.com"}

Result: PASSED
Function Code:

```python
def test_get_user_api(client,db_init):
    #Act
    response = client.get('/api/user')

    #Assertion
    assert response.status_code==200
    assert response.json[0]['user_name']=='testuser'
    assert response.json[0]['email']=='test@gmail.com'
```

Test: To create a new user without specifying username
API being tested: http://127.0.0.1:5000/api/user
Inputs

- Request Method: POST

- JSON: {“email”: “test@gmail.com”, “password”: “12345678”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “USER001”, “error_message”: “User_name is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “USER001”, “error_message”: “User_name is required”}

Result: PASSED

Function Code:

```
def test_post_user_api_nousername(client,db_init):
    #Act
    user={
        "email":"test@gmail.com",
        "password":"12345678"
    }
    response = client.post('/api/user', json=user)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=='USER001'
    assert response.json['error_message']=='User_name is required'
```

Test: To create a new user without specifying email

API being tested: http://127.0.0.1:5000/api/user

Inputs

- Request Method: POST
- JSON: {“user_name”: “testuser”, “password”: “12345678”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “USER002”, “error_message”: “Email is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “USER002”, “error_message”: “Email is required”}

Result: PASSED

Function Code:

```
def test_post_user_api_noemail(client,db_init):
    #Act
    user={
        "user_name":"testuser",
        "password":"12345678"
    }
    response = client.post('/api/user', json=user)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=='USER002'
    assert response.json['error_message']=='Email is required'
```

Test: To create a new user without specifying password
API being tested: http://127.0.0.1:5000/api/user
Inputs

- Request Method: POST
- JSON: {“user_name”: “testuser”, “email”: “test@gmail.com”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “USER003”, “error_message”: “Password is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “USER003”, “error_message”: “Password is required”}

Result: **PASSED**
Function Code:

```python
def test_post_user_api_nopassword(client,db_init):
    #Act
    user={
        "user_name":"testuser",
        "email":"test@gmail.com"
    }
    response = client.post('/api/user', json=user)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=='USER003'
    assert response.json['error_message']=='Password is required'
```

Test: To check for an already existing user
API being tested: http://127.0.0.1:5000/api/user
Inputs

- Request Method: POST
- JSON: {“user_name”: “testuser”, “email”: “test@gmail.com”, “password”: “12345678”}
- Header:

Expected Output

- HTTP Status Code: 409
- JSON: {“message”: “already exists”}

Actual Output

- HTTP Status Code: 409
- JSON: {“message”: “already exists”}

Result: **PASSED**
Function Code:

```python
def test_post_user_api_userexist(client,db_init):
    #Act
    user={
        "user_name":"testuser",
        "email":"test@gmail.com",
        "password":"12345678"
    }
    response = client.post('/api/user', json=user)

    #Assertion
    assert response.status_code==409
    assert response.json=='already exists'
```

Output:

```
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-7.4.3, pluggy-1.3.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /mnt/e/Software Engineering Project/milestone4/SE-PROJECT-MILESTONE-4
collected 15 items / 7 deselected / 8 selected

tests/unittests/test_UserAPI.py::test_create_user_api PASSED             [ 12%]
tests/unittests/test_UserAPI.py::test_delete_user_api PASSED             [ 25%]
tests/unittests/test_UserAPI.py::test_update_user_api PASSED             [ 37%]
tests/unittests/test_UserAPI.py::test_get_user_api PASSED                [ 50%]
tests/unittests/test_UserAPI.py::test_post_user_api_nousername PASSED    [ 62%]
tests/unittests/test_UserAPI.py::test_post_user_api_noemail PASSED       [ 75%]
tests/unittests/test_UserAPI.py::test_post_user_api_nopassword PASSED    [ 87%]
tests/unittests/test_UserAPI.py::test_post_user_api_userexist PASSED     [100%]
```

Testing Recommendation API

Test: To get recommendation from the system
API being tested: http://127.0.0.1:5000/api/recommendation
Inputs

- Request Method: GET
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {}

Actual Output

- HTTP Status Code: 200
- JSON: {}

Result: **PASSED**
Function Code:

```python
def test_get_recommendation_api(client,db_init):

    #Arrange
    studd = StudentDetails(user_id=1, roll_no="22f1000888",
    current_status="Enrolled", select_your_course="PDSA",
    commit_per_week=30, budget_per_term=30000,
    CGPA=8.5, interest="Programming", about="A passionate student",
    dob=datetime.strptime(str("2000-01-01"), '%Y-%m-%d').date())
    db.session.add(studd)
    db.session.commit()

    course = Courses(course_name="PDSA", level="UG",
    enrolled_this_term=100, enrolled_so_far=100,
    course_credit=4, course_description="Data Structures and Algorithms",
    teacher="Prof. ABC", pre_req1="DSA", pre_req2="None",
    fees=10000,toughness=7, avg_marks=80, success_rate=90)
    db.session.add(course)
    db.session.commit()
    #Act
    response = client.get('/api/recommendation')

    #Assertion
    assert response.status_code==200
```

Test: To post a recommendation
API being tested: http://127.0.0.1:5000/api/recommendation
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_one”: “PDSA”, “course_two”: “DBMS”, “course_three”: “MAD I”, “course_four”: “SC”, “status”: “Enrolled”}
- Header:

Expected Output

- HTTP Status Code: 201
- JSON: {“roll_no”: “22f1000888”}

Actual Output

- HTTP Status Code: 201
- JSON: {“roll_no”: “22f1000888”}

Result: PASSED
Function Code:

```python
def test_post_recommendation_api(client,db_init):
    recommendation={
    "roll_no":"22f1000888",
    "course_one":"PDSA",
    "course_two":"DBMS",
    "course_three":"MAD I",
    "course_four":"SC",
    "status":"Enrolled"
    }

    response = client.post('/api/recommendation',json=recommendation)
    assert response.status_code==201
    assert response.json['roll_no']=="22f1000888"
```

Test: To delete a recommendation
API being tested: http://127.0.0.1:5000/api/recommendation/1

Inputs

- Request Method: DELETE
- JSON: {“recommendation”: “1”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“message”: “recommendation deleted”}

Actual Output

- HTTP Status Code: 200
- JSON: {“message”: “recommendation deleted”}

Result: PASSED

Function Code:

```
def test_delete_recommendation_api(client,db_init):

    #Act
    response = client.delete('/api/recommendation/1')

    #Assertion
    assert response.status_code==200
```

Test: To update an existing recommendation

API being tested: http://127.0.0.1:5000/api/recommendation

Inputs

- Request Method: POST/PUT
- JSON: {“roll_no”: “22f1000888”, “course_one”: “PDSA”, “course_two”: “DBMS”, “course_three”: “MAD I”, “course_four”: “SC”, “status”: “Enrolled”}
- Header:

Expected Output

- HTTP Status Code: 201/200
- JSON: {“roll_no”: “22f1000888”}

Actual Output

- HTTP Status Code: 201/200
- JSON: {“roll_no”: “22f1000888”}

Result: PASSED

Function Code:

```python
def test_update_recommendation_api(client,db_init):
    #Act
    recommendation={
    "roll_no":"22f1000888",
    "course_one":"PDSA",
    "course_two":"DBMS",
    "course_three":"MAD I",
    "course_four":"SC",
    "status":"Enrolled"
    }

    response = client.post('/api/recommendation',json=recommendation)
    assert response.status_code==201
    assert response.json['roll_no']=="22f1000888"

    recommendation_update={
    "roll_no":"22f1000898",
    "course_one":"PDSA",
    "course_two":"DBMS",
    "course_three":"MAD I",
    "course_four":"SC",
    "status":"Enrolled"
    }

    response = client.put('/api/recommendation/1',json=recommendation_update)
    assert response.status_code==200
    assert response.json['roll_no']=="22f1000898"
```

Test: To post a recommendation without specifying roll number
API being tested: http://127.0.0.1:5000/api/recommendation
Inputs

- Request Method: POST
- JSON: {“course_one”: “PDSA”, “course_two”: “DBMS”, “course_three”: “MAD I”, “course_four”: “SC”, “status”: “Enrolled”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “R001”, “error_message”: “roll_no is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “R001”, “error_message”: “roll_no is required”}

Result: PASSED
Function Code:

```python
def test_post_recommendation_api_norollno(client,db_init):
    #Act
    recommendation={
    "course_one":"PDSA",
    "course_two":"DBMS",
    "course_three":"MAD I",
    "course_four":"SC",
    "status":"Enrolled"
    }

    response = client.post('/api/recommendation', json=recommendation)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="R001"
    assert response.json['error_message']=="roll_no is required"
```

Test: To post a recommendation without specifying courses
API being tested: http://127.0.0.1:5000/api/recommendation
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_two”: “DBMS”, “course_three”: “MAD I”, “course_four”: “SC”, “status”: “Enrolled”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “R002”, “error_message”: “course_one is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “R002”, “error_message”: “course_one is required”}

Result: **PASSED**
Function Code:

```python
def test_post_recommendation_api_nocourse(client,db_init):
    #Act
    recommendation={
    "roll_no":"22f1000888",
    "course_two":"DBMS",
    "course_three":"MAD I",
    "course_four":"SC",
    "status":"Enrolled"
    }

    response = client.post('/api/recommendation', json=recommendation)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="R002"
    assert response.json['error_message']=="course_one is required"
```

Test: To post a recommendation without specifying acceptance status
API being tested: http://127.0.0.1:5000/api/recommendation
Inputs

- Request Method: POST

- JSON: {“roll_no”: “22f1000888”, “course_one”: “PDSA”, “course_two”: “DBMS”, “course_three”: “MAD I”, “course_four”: “SC”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “R006”, “error_message”: “status is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “R006”, “error_message”: “status is required”}

Result: PASSED

Function Code:

```
def test_post_recommendation_api_nostatus(client,db_init):
    #Act
    recommendation={
    "roll_no":"22f1000888",
    "course_one":"PDSA",
    "course_two":"DBMS",
    "course_three":"MAD I",
    "course_four":"SC"
    }

    response = client.post('/api/recommendation', json=recommendation)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="R006"
    assert response.json['error_message']=="status is required"
```

Test: To post an already existing recommendation

API being tested: http://127.0.0.1:5000/api/recommendation

Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_one”: “PDSA”, “course_two”: “DBMS”, “course_three”: “MAD I”, “course_four”: “SC”, “status”: “Enrolled”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “R007”, “error_message”: “recommendation already exists”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “R007”, “error_message”: “recommendation already exists”}

Result: PASSED

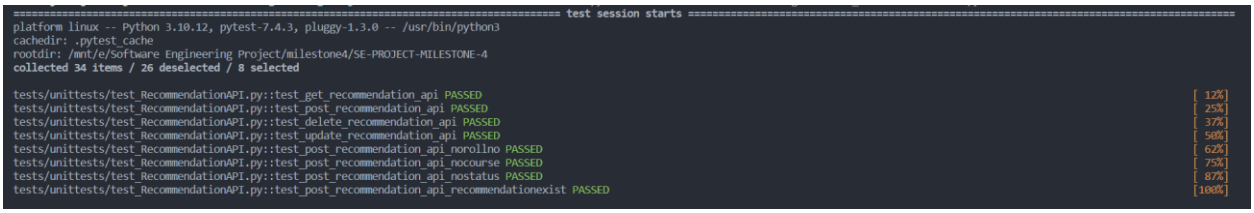
Function Code:

```python
def test_post_recommendation_api_recommendationexist(client,db_init):
    #Act
    recommendation={
    "roll_no":"22f1000888",
    "course_one":"PDSA",
    "course_two":"DBMS",
    "course_three":"MAD I",
    "course_four":"SC",
    "status":"Enrolled"
    }

    response = client.post('/api/recommendation', json=recommendation)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="R007"
    assert response.json['error_message']=="recommendation already exist"
```

Output:



Testing Feedback API

Test: To create new feedback
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 201
- JSON: {“feedback”: “testfeedback”}

Actual Output

- HTTP Status Code: 201
- JSON: {“feedback”: “testfeedback”}

Result: **PASSED**
Function Code:

```python
def test_create_feedback_api(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==201
    assert response.json['feedback']=='testfeedback'
```

Test: To get feedback
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: GET
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“feedback”: “testfeedback”}

Actual Output

- HTTP Status Code: 200
- JSON: {“feedback”: “testfeedback”}

Result: **PASSED**
Function Code:

```python
def test_get_feedback_api(client,db_init):
    #Act
    response = client.get('/api/feedback')

    #Assertion
    assert response.status_code==200
    assert response.json[0]['feedback']=='testfeedback'
```

Test: To delete a feedback
API being tested: http://127.0.0.1:5000/api/feedback/1
Inputs

- Request Method: DELETE
- JSON: {“feedback”: “1”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“message”: “feedback deleted”}

Actual Output

- HTTP Status Code: 200
- JSON: {“message”: “feedback deleted”}

Result: **PASSED**

Function Code:

```
def test_delete_feedback_api(client,db_init):
    #Act
    response = client.delete('/api/feedback/1')

    #Assertion
    assert response.status_code==200
```

Test: To update an existing feedback

API being tested: http://127.0.0.1:5000/api/feedback

Inputs

- Request Method: POST/PUT
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 201/200
- JSON: {“feedback_question”: “testquestion”, “feedback”: “testfeedback”}

Actual Output

- HTTP Status Code: 201/200
- JSON: {“feedback_question”: “testquestion”, “feedback”: “testfeedback”}

Result: **PASSED**

Function Code:

```python
def test_update_feedback_api(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback',json=feedback)

    #Assertion
    assert response.status_code==201
    assert response.json['feedback']=='testfeedback'
```

```python
feedback_update={
    "roll_no":"22f1000888",
    "course_id":1,
    "teacher":1,
    "assignment":1,
    "exams":"test",
    "content":1,
    "toughness":1,
    "overall":1,
    "grade":"A",
    "feedback_question":"testquestion",
    "feedback":"testfeedback"
}

response = client.put('/api/feedback/1',json=feedback_update)
assert response.status_code==200
assert response.json['feedback_question']=='testquestion'
```

Test: To create new feedback without specifying roll number
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB001”, “error_message”: “roll_no is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB001”, “error_message”: “roll_no is required”}

Result: **PASSED**

Function Code:

```
def test_post_feedback_api_norollno(client,db_init):
    #Act
    feedback={
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB001"
    assert response.json['error_message']=='roll_no is required'
```

Test: To create new feedback without specifying the course_id

API being tested: http://127.0.0.1:5000/api/feedback

Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB002”, “error_message”: “course_id is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB002”, “error_message”: “course_id is required”}

Result: **PASSED**

Function Code:

```python
def test_post_feedback_api_nocourseid(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB002"
    assert response.json['error_message']=='course_id is required'
```

Test: To create new feedback without specifying the teacher
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB003”, “error_message”: “teacher is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB003”, “error_message”: “teacher is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_noteacher(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB003"
    assert response.json['error_message']=='teacher is required'
```

Test: To create new feedback without specifying the assignment
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB004”, “error_message”: “assignment is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB004”, “error_message”: “assignment is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_noassignment(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB004"
    assert response.json['error_message']=='assignment is required'
```

Test: To create new feedback without specifying exams
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB005”, “error_message”: “exams is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB005”, “error_message”: “exams is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_noexams(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB005"
    assert response.json['error_message']=='exams is required'
```

Test: To create new feedback without specifying the content
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB006”, “error_message”: “content is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB006”, “error_message”: “content is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_nocontent(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB006"
    assert response.json['error_message']=='content is required'
```

Test: To create new feedback without specifying the toughness
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB007”, “error_message”: “toughness is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB007”, “error_message”: “toughness is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_notoughness(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB007"
    assert response.json['error_message']=='toughness is required'
```

Test: To create new feedback without specifying overall
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB008”, “error_message”: “overall is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB008”, “error_message”: “overall is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_nooverall(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB008"
    assert response.json['error_message']=='overall is required'
```

Test: To create new feedback without specifying grade
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB009”, “error_message”: “grade is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB009”, “error_message”: “grade is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_nograde(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB009"
    assert response.json['error_message']=='grade is required'
```

Test: To create new feedback without specifying feedback question
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB010”, “error_message”: “feedback_question is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB010”, “error_message”: “feedback_question is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_nofeedback_question(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback":"testfeedback"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB010"
    assert response.json['error_message']=="feedback_question is required"
```

Test: To create new feedback without specifying the feedback details
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “1”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB011”, “error_message”: “feedback is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB011”, “error_message”: “feedback is required”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_nofeedback(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":1,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion"
    }
    response = client.post('/api/feedback', json=feedback)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code']=="FB011"
    assert response.json['error_message']=="feedback is required"
```

Test: To create an already existing feedback
API being tested: http://127.0.0.1:5000/api/feedback
Inputs

- Request Method: POST
- JSON: {“roll_no”: “22f1000888”, “course_id”: “1”, “teacher”: “2”, “assignment”: “1”, “exams”: “test”, “content”: “1”, “toughness”: “1”, “overall”: “1”, “grade”: “A”, “feedback_question”: “testquestion”, “feedback”: “testfeedback”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB012”, “error_message”: “feedback already exists”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “FB012”, “error_message”: “feedback already exists”}

Result: **PASSED**
Function Code:

```python
def test_post_feedback_api_feedbackexist(client,db_init):
    #Act
    feedback={
        "roll_no":"22f1000888",
        "course_id":1,
        "teacher":2,
        "assignment":1,
        "exams":"test",
        "content":1,
        "toughness":1,
        "overall":1,
        "grade":"A",
        "feedback_question":"testquestion",
        "feedback":"testfeedback"
    }

    response = client.post('/api/feedback', json=feedback)
    assert response.status_code==400
    assert response.json['error_code']=="FB012"
    assert response.json['error_message']=="feedback already exist"
```

Output:

```
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-7.4.3, pluggy-1.3.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /mnt/e/Software Engineering Project/milestone4/SE-PROJECT-MILESTONE-4
collected 34 items / 18 deselected / 16 selected

tests/unittests/test_FeedBackAPI.py::test_create_feedback_api PASSED     [  6%]
tests/unittests/test_FeedBackAPI.py::test_get_feedback_api PASSED        [ 12%]
tests/unittests/test_FeedBackAPI.py::test_delete_feedback_api PASSED     [ 18%]
tests/unittests/test_FeedBackAPI.py::test_update_feedback_api PASSED     [ 25%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_norollno PASSED [ 31%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_nocourseid PASSED [ 37%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_noteacher PASSED [ 43%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_noassignment PASSED [ 50%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_noexams PASSED [ 56%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_nocontent PASSED [ 62%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_notoughness PASSED [ 68%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_nooverall PASSED [ 75%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_nograde PASSED [ 81%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_nofeedback_question PASSED [ 87%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_nofeedback PASSED [ 93%]
tests/unittests/test_FeedBackAPI.py::test_post_feedback_api_feedbackexist PASSED [100%]
```

Testing Student Details API

Test: To post student details
API being tested: http://127.0.0.1:5000/api/studentdetails
Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {}

Actual Output

- HTTP Status Code: 200

- JSON: {}

Result: PASSED

Function Code:

```
def test_studentdetails_api_post(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==200
```

Test: To retrieve student details

API being tested: http://127.0.0.1:5000/api/studentdetails

Inputs

- Request Method: GET
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”}

Actual Output

- HTTP Status Code: 200
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”}

Result: PASSED

Function Code:

```python
def test_studentdetails_api_get(client,db_init):
    #Act
    response = client.get('/api/studentdetails')

    #Assertion
    assert response.status_code==200
    assert response.json[0]['user_id']==1
    assert response.json[0]['roll_no']=='22f1000888'
    assert response.json[0]['current_status']=='Enrolled'
    assert response.json[0]['select_your_course']=='PDSA'
    assert response.json[0]['commit_per_week']==30
    assert response.json[0]['budget_per_term']==30000
    assert response.json[0]['CGPA']==8.5
    assert response.json[0]['interest']=='Programming'
    assert response.json[0]['about']=='A passionate student'
```

Test: To put student details
API being tested: http://127.0.0.1:5000/api/studentdetails/1
Inputs

- Request Method: PUT
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “22f1000888”}

Actual Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “22f1000888”}

Result: **PASSED**
Function Code:

```python
def test_studentdetails_api_put(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000898",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.put('/api/studentdetails/1', json=data)

    #Assertion
    assert response.status_code==200
    assert response.json['roll_no']=='22f1000898'
```

Test: To delete a student's detail
API being tested: http://127.0.0.1:5000/api/studentdetails/1
Inputs

- Request Method: POST
- JSON: {"user_id":"1"}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {"message": "Student details deleted successfully"}

Actual Output

- HTTP Status Code: 200
- JSON: {"message": "Student details deleted successfully"}

Result: **PASSED**
Function Code:

```
def test_studentdetails_api_delete(client,db_init):
    #Act
    response = client.delete('/api/studentdetails/1')

    #Assertion
    assert response.status_code==200
    assert response.json['message']=='Student details deleted successfully'
```

Test: To post student details without specifying roll number
API being tested: http://127.0.0.1:5000/api/studentdetails
Inputs

- Request Method: POST
- JSON: {"user_id":"1", "current_status": "Enrolled", "select_your_course": "PDSA", "commit_per_week": "30", "budget_per_term": "30000", "CGPA": "8.5", "interest": "Programming", "about": "A passionate student", "dob": "2000-01-01"}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {"error_message": "roll_no is required"}

Actual Output

- HTTP Status Code: 400
- JSON: {"error_message": "roll_no is required"}

Result: **PASSED**
Function Code:

```python
def test_studentdetails_api_post_norollno(client,db_init):
    #Act
    data={
        "user_id": 1,
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='roll_no is required'
```

Test: To post student details without specifying current status
API being tested: http://127.0.0.1:5000/api/studentdetails
Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “current_status is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “current_status is required”}

Result: **PASSED**
Function Code:

```python
def test_studentdetails_api_post_nocurrentstatus(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='current_status is required'
```

Test: To post student details without specifying the course
API being tested: http://127.0.0.1:5000/api/studentdetails
Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “select_your_course is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “select_your_course is required”}

Result: **PASSED**
Function Code:

```
def test_studentdetails_api_post_nocourse(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='select_your_course is required'
```

Test: To post student details without specifying the commitment hours
API being tested: http://127.0.0.1:5000/api/studentdetails
Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “commit_per_week is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “commit_per_week is required”}

Result: **PASSED**

Function Code:

```
def test_studentdetails_api_post_nocommit(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='commit_per_week is required'
```

Test: To post student details without specifying the budget per term
API being tested: http://127.0.0.1:5000/api/studentdetails
Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “budget_per_term is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “budget_per_term is required”}

Result: PASSED
Function Code:

```
def test_studentdetails_api_post_nobudget(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='budget_per_term is required'
```

Test: To post student details without specifying the CGPA
API being tested: http://127.0.0.1:5000/api/studentdetails

Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “interest”: “Programming”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “CGPA is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “CGPA is required”}

Result: PASSED

Function Code:

```
def test_studentdetails_api_post_nocgpa(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='CGPA is required'
```

Test: To post student details without specifying interest

API being tested: http://127.0.0.1:5000/api/studentdetails

Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “interest is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “interest is required”}

Result: PASSED

Function Code:

```python
def test_studentdetails_api_post_nointerest(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='interest is required'
```

Test: To post student details without specifying about field
API being tested: http://127.0.0.1:5000/api/studentdetails
Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “about is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “about is required”}

Result: PASSED
Function Code:

```python
def test_studentdetails_api_post_noabout(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='about is required'
```

Test: To post student details without specifying the date of birth
API being tested: http://127.0.0.1:5000/api/studentdetails
Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “dob is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “dob is required”}

Result: PASSED

Function Code:

```
def test_studentdetails_api_post_nodob(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student"
        }
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='dob is required'
```

Test: To post an already existing student details

API being tested: http://127.0.0.1:5000/api/studentdetails

Inputs

- Request Method: POST
- JSON: {“user_id”:”1”, “roll_no”: “22f1000888”, “current_status”: “Enrolled”, “select_your_course”: “PDSA”, “commit_per_week”: “30”, “budget_per_term”: “30000”, “CGPA”: “8.5”, “interest”: “Programming”, “about”: “A passionate student”, “dob”: “2000-01-01”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_message”: “student details already exists”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_message”: “student details already exists”}

Result: PASSED

Function Code:

```python
def test_studentdetails_api_post_detailsexist(client,db_init):
    #Act
    data={
        "user_id": 1,
        "roll_no": "22f1000888",
        "current_status": "Enrolled",
        "select_your_course": "PDSA",
        "commit_per_week": 30,
        "budget_per_term": 30000,
        "CGPA": 8.5,
        "interest": "Programming",
        "about": "A passionate student",
        "dob": "2000-01-01"
        }
    response = client.post('/api/studentdetails', json=data)
    response = client.post('/api/studentdetails', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_message']=='student details already exist'
```

Output:

```
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-7.4.3, pluggy-1.3.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /mnt/e/Software Engineering Project/milestone4/SE-PROJECT-MILESTONE-4
collected 79 items / 65 deselected / 14 selected

tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post PASSED [  7%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_get PASSED [ 14%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_put PASSED [ 21%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_delete PASSED [ 28%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_norollno PASSED [ 35%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_nocurrentstatus PASSED [ 42%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_nocourse PASSED [ 50%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_nocommit PASSED [ 57%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_nobudget PASSED [ 64%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_nocgpa PASSED [ 71%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_nointerest PASSED [ 78%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_noabout PASSED [ 85%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_nodob PASSED [ 92%]
tests/unittests/test_studentdetailsAPI.py::test_studentdetails_api_post_detailsexist PASSED [100%]
```

Testing Courses API

Test: To post course details
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “20000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“course_name”: “PDSA”, “level”: “Diploma”}

Actual Output

- HTTP Status Code: 200
- JSON: {“course_name”: “PDSA”, “level”: “Diploma”}

Result: **PASSED**
Function Code:

```python
def test_courses_api(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==200
    assert response.json['course_name'] == "PDSA"
    assert response.json['level'] == "Diploma"
```

Test: To get course details
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: GET
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“course_name”: “PDSA”, “level”: “Diploma”}

Actual Output

- HTTP Status Code: 200
- JSON: {“course_name”: “PDSA”, “level”: “Diploma”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_get(client,db_init):
    #Act
    response = client.get('/api/courses')

    #Assertion
    assert response.status_code==200
    assert response.json[0]['course_name'] == "PDSA"
    assert response.json[0]['level'] == "Diploma"
```

Test: To put course details
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: PUT

- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“course_name”: “PDSA”, “level”: “Diploma”, “enrolled_so_far”: “10000”}

Actual Output

- HTTP Status Code: 200
- JSON: {“course_name”: “PDSA”, “level”: “Diploma”, “enrolled_so_far”: “10000”}

Result: **PASSED**

Function Code:

```
def test_courses_api_put(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 10000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.put('/api/courses/1', json=data)

    #Assertion
    assert response.status_code==200
    assert response.json['course_name'] == "PDSA"
    assert response.json['level'] == "Diploma"
    assert response.json['enrolled_so_far'] == 10000
```

Test: To delete course details

API being tested: http://127.0.0.1:5000/api/courses/1

Inputs

- Request Method: DELETE
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“message”: “Course deleted successfully”}

Actual Output

- HTTP Status Code: 200
- JSON: {“message”: “Course deleted successfully”}

Result: **PASSED**

Function Code:

```
def test_courses_api_delete(client,db_init):
    #Act
    response = client.delete('/api/courses/1')

    #Assertion
    assert response.status_code==200
    assert response.json['message'] == "Course deleted successfully"
```

Test: To post course details without specifying course name
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “20000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C002”, “error_message”: “course_name is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C002”, “error_message”: “course_name is required”}

Result: **PASSED**
Function Code:

```
def test_courses_api_nocoursename(client,db_init):
    #Act
    data={
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C002"
    assert response.json['error_message'] == "course_name is required"
```

Test: To post course details without specifying course level
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”: “PDSA”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “20000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C003”, “error_message”: “level is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C003”, “error_message”: “level is required”}

Result: PASSED

Function Code:

```
def test_courses_api_nolevel(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C003"
    assert response.json['error_message'] == "level is required"
```

Test: To post course details without specifying enrolled this term

API being tested: http://127.0.0.1:5000/api/courses

Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C004”, “error_message”: “enrolled_this_term is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C004”, “error_message”: “enrolled_this_term is required”}

Result: **PASSED**

Function Code:

```
def test_courses_api_noenrolled_this_term(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C004"
    assert response.json['error_message'] == "enrolled_this_term is required"
```

Test: To post course details without specifying enrolled so far

API being tested: http://127.0.0.1:5000/api/courses

Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C005”, “error_message”: “enrolled_so_far is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C005”, “error_message”: “enrolled_so_far is required”}

Result: **PASSED**

Function Code:

```python
def test_courses_api_noenrolled_so_far(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C005"
    assert response.json['error_message'] == "enrolled_so_far is required"
```

Test: To post course details without specifying course credit
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C006”, “error_message”: “course_credit is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C006”, “error_message”: “course_credit is required”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_nocourse_credit(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C006"
    assert response.json['error_message'] == "course_credit is required"
```

Test: To post course details without specifying course description
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C007”, “error_message”: “course_description is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C007”, “error_message”: “course_description is required”}

Result: PASSED
Function Code:

```python
def test_courses_api_nocourse_description(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C007"
    assert response.json['error_message'] == "course_description is required"
```

Test: To post course details without specifying the teacher
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C008”, “error_message”: “teacher is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C008”, “error_message”: “teacher is required”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_noteacher(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C008"
    assert response.json['error_message'] == "teacher is required"
```

Test: To post course details without specifying pre-requisite subject-1
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C009”, “error_message”: “pre_req1 is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C009”, “error_message”: “pre_req1 is required”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_nopre_req1(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C009"
    assert response.json['error_message'] == "pre_req1 is required"
```

Test: To post course details without specifying pre-requisite subject-2
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C010”, “error_message”: “pre_req2 is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C010”, “error_message”: “pre_req2 is required”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_nopre_req2(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C010"
    assert response.json['error_message'] == "pre_req2 is required"
```

Test: To post course details without specifying the fees
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C011”, “error_message”: “fees is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C011”, “error_message”: “fees is required”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_nofees(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90,
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C011"
    assert response.json['error_message'] == "fees is required"
```

Test: To post course details without specifying course toughness
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C012”, “error_message”: “toughness is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C012”, “error_message”: “toughness is required”}

Result: PASSED
Function Code:

```python
def test_courses_api_notoughness(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C012"
    assert response.json['error_message'] == "toughness is required"
```

Test: To post course details without specifying course average marks
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C013”, “error_message”: “avg_marks is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C013”, “error_message”: “avg_marks is required”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_noavg_marks(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C013"
    assert response.json['error_message'] == "avg_marks is required"
```

Test: To post course details without specifying course success rate
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C014”, “error_message”: “success_rate is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C014”, “error_message”: “success_rate is required”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_nosuccess_rate(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Diploma",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "programming",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 10000,
        "toughness": 8,
        "avg_marks": 85.5
        }

    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C014"
    assert response.json['error_message'] == "success_rate is required"
```

Test: To post already existing course details
API being tested: http://127.0.0.1:5000/api/courses
Inputs

- Request Method: POST
- JSON: {“course_name”:”PDSA”, “level”: “Diploma”, “enrolled_this_term”: “5000”, “enrolled_so_far”: “10000”, “course_credit”: “4”, “course_description”: “programming”, “teacher”: “John Doe”, “pre_req1”: “NIL”, “pre_req2”: “DBMS”, “fees”: “10000”, “toughness”: “8”, “avg_marks”: “85.5”, “success_rate”: “90”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C015”, “error_message”: “course already exists”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “C015”, “error_message”: “course already exists”}

Result: **PASSED**
Function Code:

```python
def test_courses_api_coursealreadyexist(client,db_init):
    #Act
    data={
        "course_name": "PDSA",
        "level": "Degree",
        "enrolled_this_term": 5000,
        "enrolled_so_far": 20000,
        "course_credit": 4,
        "course_description": "data structures",
        "teacher": "John Doe",
        "pre_req1": "NIL",
        "pre_req2": "DBMS",
        "fees": 20000,
        "toughness": 8,
        "avg_marks": 85.5,
        "success_rate": 90
        }

    response = client.post('/api/courses', json=data)
    response = client.post('/api/courses', json=data)

    #Assertion
    assert response.status_code==400
    assert response.json['error_code'] == "C015"
    assert response.json['error_message'] == "course already exist"
```

Output:

```
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-7.4.3, pluggy-1.3.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /mnt/e/Software Engineering Project/milestone4/SE-PROJECT-MILESTONE-4
collected 79 items / 61 deselected / 18 selected

tests/unittests/test_courseAPI.py::test_courses_api PASSED               [  5%]
tests/unittests/test_courseAPI.py::test_courses_api_get PASSED           [ 11%]
tests/unittests/test_courseAPI.py::test_courses_api_put PASSED           [ 16%]
tests/unittests/test_courseAPI.py::test_courses_api_delete PASSED        [ 22%]
tests/unittests/test_courseAPI.py::test_courses_api_nocoursename PASSED  [ 27%]
tests/unittests/test_courseAPI.py::test_courses_api_nolevel PASSED       [ 33%]
tests/unittests/test_courseAPI.py::test_courses_api_noenrolled_this_term PASSED [ 38%]
tests/unittests/test_courseAPI.py::test_courses_api_noenrolled_so_far PASSED [ 44%]
tests/unittests/test_courseAPI.py::test_courses_api_nocourse_credit PASSED [ 50%]
tests/unittests/test_courseAPI.py::test_courses_api_nocourse_description PASSED [ 55%]
tests/unittests/test_courseAPI.py::test_courses_api_noteacher PASSED     [ 61%]
tests/unittests/test_courseAPI.py::test_courses_api_nopre_req1 PASSED    [ 66%]
tests/unittests/test_courseAPI.py::test_courses_api_nopre_req2 PASSED    [ 72%]
tests/unittests/test_courseAPI.py::test_courses_api_nofees PASSED        [ 77%]
tests/unittests/test_courseAPI.py::test_courses_api_notoughness PASSED   [ 83%]
tests/unittests/test_courseAPI.py::test_courses_api_noavg_marks PASSED   [ 88%]
tests/unittests/test_courseAPI.py::test_courses_api_nosuccess_rate PASSED [ 94%]
tests/unittests/test_courseAPI.py::test_courses_api_coursealreadyexist PASSED [100%]
```

Testing Term API

Test: To post term details
API being tested: http://127.0.0.1:5000/api/term
Inputs

- Request Method: POST
- JSON: {“term”: “Spring 2022”, “start_date”: “datetime.now()”, “end_date”: “datetime.now()”, “total_students_enrolled”: “100”, “status”: “Open”}
- Header:

Expected Output

- HTTP Status Code: 201
- JSON: {“term”: “Spring 2022”, “total_students_enrolled”: “100”, “status”: “Open”}

Actual Output

- HTTP Status Code: 201
- JSON: {“term”: “Spring 2022”, “total_students_enrolled”: “100”, “status”: “Open”}

Result: **PASSED**

Function Code:

```
def test_term_api_post(client, db_init):
    # Arrange
    data = {
        "term": "Spring 2022",
        "start_date": datetime.now().isoformat(),
        "end_date": (datetime.now() + timedelta(days=90)).isoformat(),
        "total_student_enrolled": 100,
        "status": "Open"
    }

    # Act
    response = client.post('/api/term', json=data)

    # Assertion
    assert response.status_code == 201
    assert response.json['term'] == "Spring 2022"
    assert response.json['total_student_enrolled'] == 100
    assert response.json['status'] == "Open"
```

Test: To get term details

API being tested: http://127.0.0.1:5000/api/term

Inputs

- Request Method: GET
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {}

Actual Output

- HTTP Status Code: 200
- JSON: {}

Result: **PASSED**

Function Code:

```
def test_term_api_get(client, db_init):
    # Act
    response = client.get('/api/term')

    # Assertion
    assert response.status_code == 200
```

Test: To update term details

API being tested: http://127.0.0.1:5000/api/term

Inputs

- Request Method: PUT

- JSON: {“term”: “Spring 2022”, “start_date”: “datetime.now()”, “end_date”: “datetime.now()”, “total_students_enrolled”: “200”, “status”: “Closed”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“term”: “Spring 2022”, “total_students_enrolled”: “200”, “status”: “Closed”}

Actual Output

- HTTP Status Code: 200
- JSON: {“term”: “Spring 2022”, “total_students_enrolled”: “200”, “status”: “Closed”}

Result: **PASSED**

Function Code:

```
def test_term_api_put(client, db_init):
    # Arrange
    data = {
        "term": "Spring 2022",
        "start_date": datetime.now().isoformat(),
        "end_date": (datetime.now() + timedelta(days=90)).isoformat(),
        "total_student_enrolled": 200,
        "status": "Closed"
    }

    # Act
    response = client.put('/api/term/1', json=data)

    # Assertion
    assert response.status_code == 200
    assert response.json['term'] == "Spring 2022"
    assert response.json['total_student_enrolled'] == 200
    assert response.json['status'] == "Closed"
```

Test: To delete term details

API being tested: http://127.0.0.1:5000/api/term/1

Inputs

- Request Method: DELETE
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“message”: “term deleted”}

Actual Output

- HTTP Status Code: 200
- JSON: {“message”: “term deleted”}

Result: **PASSED**

Function Code:

```
def test_term_api_delete(client, db_init):
    # Act
    response = client.delete('/api/term/1')

    # Assertion
    assert response.status_code == 200
```

Test: Trying to post term details with missing fields
API being tested: http://127.0.0.1:5000/api/term
Inputs

- Request Method: POST
- JSON: {“term”: “Spring 2022”, “start_date”: “datetime.now()”, “end_date”: “datetime.now()”, “status”: “Open”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {}

Actual Output

- HTTP Status Code: 400
- JSON: {}

Result: **PASSED**
Function Code:

```
def test_post_term_missing_fields(client):
    data = {
        "term": "Spring 2022",
        "start_date": datetime.now().isoformat(),
        "end_date": (datetime.now() + timedelta(days=90)).isoformat(),
        "status": "Open"
    }
    response = client.post('/api/term', data=json.dumps(data), content_type='application/json')
    assert response.status_code == 400
```

Test: Trying to update term details that do not exist
API being tested: http://127.0.0.1:5000/api/term/9999
Inputs

- Request Method: PUT
- JSON: {“term”: “Spring 2022”, “start_date”: “datetime.now()”, “end_date”: “datetime.now()”, “total_students_enrolled”: “200”, “status”: “Closed”}
- Header:

Expected Output

- HTTP Status Code: 404
- JSON: {}

Actual Output

- HTTP Status Code: 404
- JSON: {}

Result: **PASSED**
Function Code:

```
def test_put_term_not_found(client):
    data = {
        "term": "Spring 2022",
        "start_date": datetime.now().isoformat(),
        "end_date": (datetime.now() + timedelta(days=90)).isoformat(),
        "total_student_enrolled": 200,
        "status": "Closed"
    }
    response = client.put('/api/term/9999', data=json.dumps(data), content_type='application/json')
    assert response.status_code == 404
```

Test: Trying to delete term details that do not exist
API being tested: http://127.0.0.1:5000/api/term/9999
Inputs

- Request Method: DELETE
- JSON: {“term_id”: “9999”}
- Header:

Expected Output

- HTTP Status Code: 404
- JSON: {}

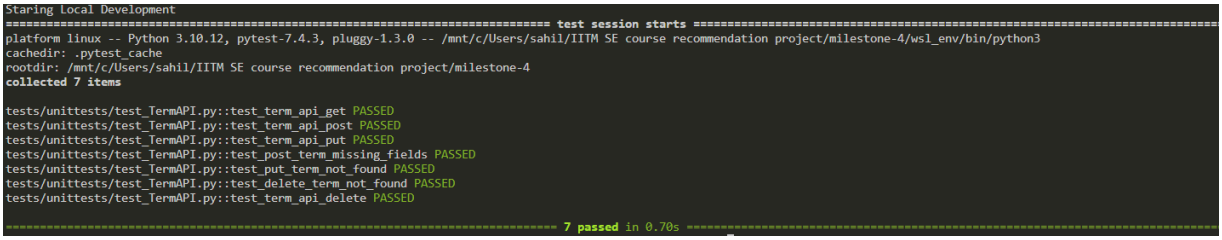
Actual Output

- HTTP Status Code: 404
- JSON: {}

Result: **PASSED**
Function Code:

```
def test_delete_term_not_found(client):
    response = client.delete('/api/term/9999')
    assert response.status_code == 404
```

Output:



Testing Queries API

Test: To post queries details
API being tested: http://127.0.0.1:5000/api/queries
Inputs

- Request Method: POST
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”, “date”: “datetime.now()”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”}

Actual Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”}

Result: **PASSED**
Function Code:

```python
def test_queries_api_post(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "query": "Test query",
        "reply": "Test reply",
        "status": "Open",
        "date": datetime.now().isoformat()
    }

    # Act
    response = client.post('/api/queries', json=data)

    # Assertion
    assert response.status_code == 200
    assert response.json['roll_no'] == "123"
    assert response.json['query'] == "Test query"
    assert response.json['reply'] == "Test reply"
    assert response.json['status'] == "Open"
```

Test: To get queries details
API being tested: http://127.0.0.1:5000/api/queries
Inputs

- Request Method: GET
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {}

Actual Output

- HTTP Status Code: 200
- JSON: {}

Result: **PASSED**
Function Code:

```python
def test_queries_api_get(client, db_init):
    # Act
    response = client.get('/api/queries')

    # Assertion
    assert response.status_code == 200
```

Test: To update queries details
API being tested: http://127.0.0.1:5000/api/queries/1
Inputs

- Request Method: PUT
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”, “date”: “datetime.now()”}
- Header:

Expected Output

- HTTP Status Code: 200

- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”, “date”: “datetime.now()”}

Actual Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”}

Result: PASSED

Function Code:

```
def test_queries_api_put(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "query": "Test query",
        "reply": "Test reply",
        "status": "Open",
        "date": datetime.now().isoformat()
    }

    # Act
    response = client.put('/api/queries/1', json=data)

    # Assertion
    assert response.status_code == 200
    assert response.json['roll_no'] == "123"
    assert response.json['query'] == "Test query"
    assert response.json['reply'] == "Test reply"
    assert response.json['status'] == "Open"
```

Test: To post queries details with missing fields

API being tested: http://127.0.0.1:5000/api/queries

Inputs

- Request Method: POST
- JSON: {“roll_no”: “123”, “query”: “Test query”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “Q003”, “error_message”: “reply, status & date is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “Q003”, “error_message”: “reply, status & date is required”}

Result: PASSED

Function Code:

```python
def test_queries_api_post_missing_fields(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "query": "Test query",
        # Missing reply, status, and date
    }

    # Act
    response = client.post('/api/queries', json=data)

    # Assertion
    assert response.status_code == 400
    assert 'error_code' in response.json
    assert response.json['error_code'] == "Q003"  # reply is required
```

Test: To update queries details with missing fields
API being tested: http://127.0.0.1:5000/api/queries
Inputs

- Request Method: PUT
- JSON: {“roll_no”: “123”, “query”: “Test query”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “Q003”, “error_message”: “reply, status & date is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “Q003”, “error_message”: “reply, status & date is required”}

Result: **PASSED**
Function Code:

```python
def test_queries_api_put_missing_fields(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "query": "Test query",
        # Missing reply, status, and date
    }

    # Act
    response = client.put('/api/queries/1', json=data)

    # Assertion
    assert response.status_code == 400
    assert 'error_code' in response.json
    assert response.json['error_code'] == "Q003"  # reply is required
```

Test: To update non-existing queries details
API being tested: http://127.0.0.1:5000/api/queries/999
Inputs

- Request Method: PUT
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”, “date”: “datetime.now()”}

- Header:

Expected Output

- HTTP Status Code: 404
- JSON: {}

Actual Output

- HTTP Status Code: 404
- JSON: {}

Result: **PASSED**

Function Code:

```
def test_queries_api_put_nonexistent_query(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "query": "Test query",
        "reply": "Test reply",
        "status": "Open",
        "date": datetime.now().isoformat()
    }

    # Act
    response = client.put('/api/queries/999', json=data)  # Nonexistent query id

    # Assertion
    assert response.status_code == 404
```

Test: To delete non-existing queries details

API being tested: http://127.0.0.1:5000/api/queries/999

Inputs

- Request Method: DELETE
- JSON: {“query”: “999”}
- Header:

Expected Output

- HTTP Status Code: 404
- JSON: {}

Actual Output

- HTTP Status Code: 404
- JSON: {}

Result: **PASSED**

Function Code:

```
def test_queries_api_delete_nonexistent_query(client, db_init):
    # Act
    response = client.delete('/api/queries/999')  # Nonexistent query id

    # Assertion
    assert response.status_code == 404
```

Test: To post queries details

API being tested: http://127.0.0.1:5000/api/queries

Inputs

- Request Method: POST
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”, “date”: “datetime.now()”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”}

Actual Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “123”, “query”: “Test query”, “reply”: “Test reply”, “status”: “Open”}

Result: PASSED

Function Code:

```python
def test_queries_api_delete(client, db_init):
    # Act
    response = client.delete('/api/queries/1')

    # Assertion
    assert response.status_code == 200
```

Output:

```
Staring Local Development
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-7.4.3, pluggy-1.3.0 -- /mnt/c/Users/sahil/IITM SE course recommendation project/milestone-4/wsl_env
/bin/python3
cachedir: .pytest_cache
rootdir: /mnt/c/Users/sahil/IITM SE course recommendation project/milestone-4
collected 8 items

tests/unittests/test_QueriesAPI.py::test_queries_api_post PASSED
tests/unittests/test_QueriesAPI.py::test_queries_api_get PASSED
tests/unittests/test_QueriesAPI.py::test_queries_api_put PASSED
tests/unittests/test_QueriesAPI.py::test_queries_api_post_missing_fields PASSED
tests/unittests/test_QueriesAPI.py::test_queries_api_put_missing_fields PASSED
tests/unittests/test_QueriesAPI.py::test_queries_api_put_nonexistent_query PASSED
tests/unittests/test_QueriesAPI.py::test_queries_api_delete_nonexistent_query PASSED
tests/unittests/test_QueriesAPI.py::test_queries_api_delete PASSED

============================== 8 passed in 0.51s ===============================
```

Testing Student Enrollment API

Test: To post student enrollment

API being tested: http://127.0.0.1:5000/api/studentenrollment

Inputs

- Request Method: POST
- JSON: {“term”: “Fall 2021”, “total_students_enrolled”: “100”, “total_registered_students”: “80”, “students_with_full_profile”: “70”, “Average_score”: “85”, “high_sub_rate”: “0.8”}
- Header:

Expected Output

- HTTP Status Code: 201
- JSON: {“term”: “Fall 2021”}

Actual Output

- HTTP Status Code: 201

- JSON: {“term”: “Fall 2021”}

Result: **PASSED**

Function Code:

```
def test_post_student_enrollment_api(client, db_init):
    # Act
    student_enrollment = {
        "term": "Fall 2021",
        "total_students_enrolled": 100,
        "total_registered_students": 80,
        "students_with_full_profile": 70,
        "Average_score": 85,
        "high_sub_rate": 0.8
    }
    response = client.post('/api/studentenrollment', json=student_enrollment)

    # Assertion
    assert response.status_code == 201
    assert response.json['term'] == 'Fall 2021'
```

Test: To get student enrollment

API being tested: http://127.0.0.1:5000/api/studentenrollment

Inputs

- Request Method: GET
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {}

Actual Output

- HTTP Status Code: 200
- JSON: {}

Result: **PASSED**

Function Code:

```
def test_get_student_enrollment_api(client, db_init):
    # Act
    response = client.get('/api/studentenrollment')

    # Assertion
    assert response.status_code == 200
```

Test: To update student enrollment

API being tested: http://127.0.0.1:5000/api/studentenrollment/1

Inputs

- Request Method: PUT
- JSON: {“term”: “Fall 2021”, “total_students_enrolled”: “120”, “total_registered_students”: “90”, “students_with_full_profile”: “80”, “Average_score”: “88”, “high_sub_rate”: “0.85”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“total_students_enrolled”: “120”}

Actual Output

- HTTP Status Code: 200
- JSON: {“total_students_enrolled”: “120”}

Result: PASSED

Function Code:

```
def test_put_student_enrollment_api(client, db_init):
    # Act
    student_enrollment_update = {
        "term": "Fall 2021",
        "total_students_enrolled": 120,
        "total_registered_students": 90,
        "students_with_full_profile": 80,
        "Average_score": 88,
        "high_sub_rate": 0.85
    }
    response = client.put('/api/studentenrollment/1', json=student_enrollment_update)

    # Assertion
    assert response.status_code == 200
    assert response.json['total_students_enrolled'] == 120
```

Test: To post student enrollment without specifying the term

API being tested: http://127.0.0.1:5000/api/studentenrollment

Inputs

- Request Method: POST
- JSON: {“total_students_enrolled”: “100”, “total_registered_students”: “80”, “students_with_full_profile”: “70”, “Average_score”: “85”, “high_sub_rate”: “0.8”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “SE001”, “error_message”: “term is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “SE001”, “error_message”: “term is required”}

Result: PASSED

Function Code:

```
def test_post_student_enrollment_api_no_term(client, db_init):
    # Act
    student_enrollment = {
        "total_students_enrolled": 100,
        "total_registered_students": 80,
        "students_with_full_profile": 70,
        "Average_score": 85,
        "high_sub_rate": 0.8
    }
    response = client.post('/api/studentenrollment', json=student_enrollment)

    # Assertion
    assert response.status_code == 400
    assert response.json['error_code'] == 'SE001'
    assert response.json['error_message'] == 'term is required'
```

Test: To delete student enrollment

API being tested: http://127.0.0.1:5000/api/studentenrollment/1

Inputs

- Request Method: DELETE
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {}

Actual Output

- HTTP Status Code: 200
- JSON: {}

Result: PASSED

Function Code:

```
def test_delete_student_enrollment_api(client, db_init):
    # Act
    response = client.delete('/api/studentenrollment/1')

    # Assertion
    assert response.status_code == 200
```

Output:

```
Staring Local Development
============================================================ test session starts ============================================================
platform linux -- Python 3.10.12, pytest-7.4.3, pluggy-1.3.0 -- /mnt/c/Users/sahil/IITM SE course recommendation project/milestone-4/wsl_env
n/python3
cachedir: .pytest_cache
rootdir: /mnt/c/Users/sahil/IITM SE course recommendation project/milestone-4
collected 5 items

tests/unittests/test_StudentEnrolmentAPI.py::test_get_student_enrollment_api PASSED
tests/unittests/test_StudentEnrolmentAPI.py::test_post_student_enrollment_api PASSED
tests/unittests/test_StudentEnrolmentAPI.py::test_put_student_enrollment_api PASSED
tests/unittests/test_StudentEnrolmentAPI.py::test_post_student_enrollment_api_no_term PASSED
tests/unittests/test_StudentEnrolmentAPI.py::test_delete_student_enrollment_api PASSED
```

Testing Student Course Details API

Test: To post student course details

API being tested: http://127.0.0.1:5000/api/studentcoursedetails

Inputs

- Request Method: POST
- JSON: {“roll_no”: “123”, “course_id”: “CS101”, “course_term”: “Fall 2021”, “course_status”: “Completed”, “grade”: “A”}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “123”, “course_id”: “CS101”, “course_term”: “Fall 2021”, “course_status”: “Completed”, “grade”: “A”}

Actual Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “123”, “course_id”: “CS101”, “course_term”: “Fall 2021”, “course_status”: “Completed”, “grade”: “A”}

Result: PASSED

Function Code:

```python
def test_student_course_details_api_post(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "course_id": "CS101",
        "course_term": "Fall 2021",
        "course_status": "Completed",
        "grade": "A"
    }

    # Act
    response = client.post('/api/studentcoursedetails', json=data)

    # Assertion
    assert response.status_code == 200
    assert response.json['roll_no'] == "123"
    assert response.json['course_id'] == "CS101"
    assert response.json['course_term'] == "Fall 2021"
    assert response.json['course_status'] == "Completed"
    assert response.json['grade'] == "A"
```

Test: To get student course details
API being tested: http://127.0.0.1:5000/api/studentcoursedetails
Inputs

- Request Method: GET
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {}

Actual Output

- HTTP Status Code: 200
- JSON: {}

Result: PASSED
Function Code:

```python
def test_student_course_details_api_get(client, db_init):
    # Act
    response = client.get('/api/studentcoursedetails')

    # Assertion
    assert response.status_code == 200
```

Test: To update student course details
API being tested: http://127.0.0.1:5000/api/studentcoursedetails
Inputs

- Request Method: PUT
- JSON: {“roll_no”: “123”, “course_id”: “CS101”, “course_term”: “Fall 2021”, “course_status”: “Completed”, “grade”: “A”}
- Header:

Expected Output

- HTTP Status Code: 200

- JSON: {“roll_no”: “123”, “course_id”: “CS101”, “course_term”: “Fall 2021”, “course_status”: “Completed”, “grade”: “A”}

Actual Output

- HTTP Status Code: 200
- JSON: {“roll_no”: “123”, “course_id”: “CS101”, “course_term”: “Fall 2021”, “course_status”: “Completed”, “grade”: “A”}

Result: PASSED

Function Code:

```python
def test_student_course_details_api_put(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "course_id": "CS101",
        "course_term": "Fall 2021",
        "course_status": "Completed",
        "grade": "A"
    }

    # Act
    response = client.put('/api/studentcoursedetails/1', json=data)

    # Assertion
    assert response.status_code == 200
    assert response.json['roll_no'] == "123"
    assert response.json['course_id'] == "CS101"
    assert response.json['course_term'] == "Fall 2021"
    assert response.json['course_status'] == "Completed"
    assert response.json['grade'] == "A"
```

Test: To post student course details with missing fields

API being tested: http://127.0.0.1:5000/api/studentcoursedetails

Inputs

- Request Method: POST
- JSON: {“roll_no”: “123”, “course_id”: “CS101”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “SCD003”, “error_message”: “course_term is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “SCD003”, “error_message”: “course_term is required”}

Result: PASSED

Function Code:

```python
def test_student_course_details_api_post_missing_fields(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "course_id": "CS101",
        # Missing course_term, course_status, and grade
    }

    # Act
    response = client.post('/api/studentcoursedetails', json=data)

    # Assertion
    assert response.status_code == 400
    assert 'error_code' in response.json
    assert response.json['error_code'] == "SCD003"  # course_term is required
```

Test: To update student course details with missing fields
API being tested: http://127.0.0.1:5000/api/studentcoursedetails/1
Inputs

- Request Method: PUT
- JSON: {“roll_no”: “123”, “course_id”: “CS101”}
- Header:

Expected Output

- HTTP Status Code: 400
- JSON: {“error_code”: “SCD003”, “error_message”: “course_term is required”}

Actual Output

- HTTP Status Code: 400
- JSON: {“error_code”: “SCD003”, “error_message”: “course_term is required”}

Result: PASSED
Function Code:

```python
def test_student_course_details_api_put_missing_fields(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "course_id": "CS101",
        # Missing course_term, course_status, and grade
    }

    # Act
    response = client.put('/api/studentcoursedetails/1', json=data)

    # Assertion
    assert response.status_code == 400
    assert 'error_code' in response.json
    assert response.json['error_code'] == "SCD003"  # course_term is required
```

Test: Trying to update non-existing student course details
API being tested: http://127.0.0.1:5000/api/studentcoursedetails/999
Inputs

- Request Method: PUT
- JSON: {“roll_no”: “123”, “course_id”: “CS101”, “course_term”: “Fall 2021”, “course_status”: “Completed”, “grade”: “A”}
- Header:

Expected Output

- HTTP Status Code: 404
- JSON: {}

Actual Output

- HTTP Status Code: 404
- JSON: {}

Result: **PASSED**

Function Code:

```
def test_student_course_details_api_put_nonexistent_record(client, db_init):
    # Arrange
    data = {
        "roll_no": "123",
        "course_id": "CS101",
        "course_term": "Fall 2021",
        "course_status": "Completed",
        "grade": "A"
    }

    # Act
    response = client.put('/api/studentcoursedetails/999', json=data)  # Nonexistent record id

    # Assertion
    assert response.status_code == 404
```

Test: Trying to delete non-existing student course details

API being tested: http://127.0.0.1:5000/api/studentcoursedetails/999

Inputs

- Request Method: DELETE
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 404
- JSON: {}

Actual Output

- HTTP Status Code: 404
- JSON: {}

Result: **PASSED**

Function Code:

```
def test_student_course_details_api_delete_nonexistent_record(client, db_init):
    # Act
    response = client.delete('/api/studentcoursedetails/999')  # Nonexistent record id

    # Assertion
    assert response.status_code == 404
```

Test: To delete student course details

API being tested: http://127.0.0.1:5000/api/studentcoursedetails/1

Inputs

- Request Method: DELETE
- JSON: {}
- Header:

Expected Output

- HTTP Status Code: 200
- JSON: {}

Actual Output

- HTTP Status Code: 200
- JSON: {}

Result: PASSED

Function Code:

```python
def test_student_course_details_api_delete(client, db_init):
    # Act
    response = client.delete('/api/studentcoursedetails/1')

    # Assertion
    assert response.status_code == 200
```

Output:

```
Staring Local Development
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-7.4.3, pluggy-1.3.0 -- /mnt/c/Users/sahil/IITM SE course recommendation project/milestone-4/wsl_env/bi
n/python3
cachedir: .pytest_cache
rootdir: /mnt/c/Users/sahil/IITM SE course recommendation project/milestone-4
collected 8 items

tests/unittests/test_StudentCourseDetailsAPI.py::test_student_course_details_api_post PASSED
tests/unittests/test_StudentCourseDetailsAPI.py::test_student_course_details_api_get PASSED
tests/unittests/test_StudentCourseDetailsAPI.py::test_student_course_details_api_put PASSED
tests/unittests/test_StudentCourseDetailsAPI.py::test_student_course_details_api_post_missing_fields PASSED
tests/unittests/test_StudentCourseDetailsAPI.py::test_student_course_details_api_put_missing_fields PASSED
tests/unittests/test_StudentCourseDetailsAPI.py::test_student_course_details_api_put_nonexistent_record PASSED
tests/unittests/test_StudentCourseDetailsAPI.py::test_student_course_details_api_delete_nonexistent_record PASSED
tests/unittests/test_StudentCourseDetailsAPI.py::test_student_course_details_api_delete PASSED
```