Nhóm 23

Đề tài: Phát triển service xử lý và lưu trữ video

Họ và tên: Hà Vĩ Khang

MSSV: 20110657

Milestone 4

Task 3: Xây dựng test case và tiến hành test cho chức năng cắt nội dung video

1. Kiểm tra cắt nội dung video với lỗi xung đột

@pytest.mark.parametrize('projects', [({'file': 'sample\_0.mp4', 'duplicate': False},)], indirect=True)

def test\_edit\_project\_409\_response(test\_app, client, projects):

project = projects[0]

with test\_app.test\_request\_context():

# since we use CELERY\_TASK\_ALWAYS\_EAGER, task will be executed immediately,

# it means next request will return a finshed result,

# since we want to test 409 response, we must set processing flag in db directly

test\_app.mongo.db.projects.find\_one\_and\_update(

{'\_id': ObjectId(project['\_id'])},

{'$set': {'processing.video': True}}

)

# edit request

url = url\_for('projects.retrieve\_edit\_destroy\_project', project\_id=project['\_id'])

trim = '2.0,6.0'

resp = client.put(

url,

data=json.dumps({

"trim": trim

}),

content\_type='application/json'

)

assert resp.status == '409 CONFLICT'

1. Kiểm tra cắt nội dung video với lỗi không cung cấp thông tin thay đổi

@pytest.mark.parametrize('projects', [({'file': 'sample\_0.mp4', 'duplicate': True},)], indirect=True)

def test\_edit\_project\_no\_edit\_rules\_provided(test\_app, client, projects):

project = projects[0]

with test\_app.test\_request\_context():

# edit request

url = url\_for('projects.retrieve\_edit\_destroy\_project', project\_id=project['\_id'])

resp = client.put(

url,

data=json.dumps({}),

content\_type='application/json'

)

assert resp.status == '400 BAD REQUEST'

1. Kiểm tra cắt nội dung video với lỗi sai request

@pytest.mark.parametrize('projects', [({'file': 'sample\_0.mp4', 'duplicate': False},)], indirect=True)

def test\_edit\_project\_version\_1(test\_app, client, projects):

project = projects[0]

with test\_app.test\_request\_context():

# edit request

url = url\_for('projects.retrieve\_edit\_destroy\_project', project\_id=project['\_id'])

trim = '2.0,6.0'

resp = client.put(

url,

data=json.dumps({

"trim": trim

}),

content\_type='application/json'

)

assert resp.status == '400 BAD REQUEST'

1. Kiểm tra cắt nội dung video thành công

@pytest.mark.parametrize('projects', [({'file': 'sample\_0.mp4', 'duplicate': True},)], indirect=True)

def test\_edit\_project\_trim\_success(test\_app, client, projects):

project = projects[0]

with test\_app.test\_request\_context():

# edit request

url = url\_for('projects.retrieve\_edit\_destroy\_project', project\_id=project['\_id'])

start = 2.0

end = 6.0

resp = client.put(

url,

data=json.dumps({

"trim": "%s,%s" % (start, end)

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '202 ACCEPTED'

assert resp\_data == {'processing': True}

# get details

resp = client.get(url)

resp\_data = json.loads(resp.data)

assert not resp\_data['processing']['video']

assert resp\_data['metadata']['duration'] == end - start

# edit request have trim end greater than duration

old\_duration = resp\_data['metadata']['duration']

url = url\_for('projects.retrieve\_edit\_destroy\_project', project\_id=project['\_id'])

start = 2.0

end = 6.0

resp = client.put(

url,

data=json.dumps({

"trim": "%s,%s" % (start, end)

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '202 ACCEPTED'

assert resp\_data == {'processing': True}

# get details

resp = client.get(url)

resp\_data = json.loads(resp.data)

assert not resp\_data['processing']['video']

assert resp\_data['metadata']['duration'] == old\_duration – start

1. Kiểm tra cắt nội dung video thất bại với các lỗi sau: (sai định dạng, thời gian bắt đầu lớn hơn kết thúc, thời gian cắt nhỏ hơn 2 giây, thời gian cắt bằng toàn bộ video, thời gian bắt đầu nhỏ hơn 0, thời gian kết thúc bé hơn hoặc bằng 1)

@pytest.mark.parametrize('projects', [({'file': 'sample\_0.mp4', 'duplicate': True},)], indirect=True)

def test\_edit\_project\_trim\_fail(test\_app, client, projects):

project = projects[0]

with test\_app.test\_request\_context():

url = url\_for('projects.retrieve\_edit\_destroy\_project', project\_id=project['\_id'])

# incorrect type

resp = client.put(

url,

data=json.dumps({

"trim": 123

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '400 BAD REQUEST'

assert resp\_data == {'trim': ["must be of string type"]}

# malformed format

resp = client.put(

url,

data=json.dumps({

"trim": "x,y"

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '400 BAD REQUEST'

assert resp\_data == {'trim': ["value does not match regex '^\\d+\\.?\\d\*,\\d+\\.?\\d\*$'"]}

# edit request

resp = client.put(

url,

data=json.dumps({

"trim": "6.0,2.0"

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '400 BAD REQUEST'

assert resp\_data == {'trim': ["'start' value must be less than 'end' value"]}

# edit request

resp = client.put(

url,

data=json.dumps({

"trim": "0,1"

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '400 BAD REQUEST'

assert resp\_data == {'trim': ['Trimmed video duration must be at least 2 seconds']}

# edit request

resp = client.put(

url,

data=json.dumps({

"trim": "0,15"

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '400 BAD REQUEST'

assert resp\_data == {'trim': ["'end' value of trim is duplicating an entire video"]}

# edit request

resp = client.put(

url,

data=json.dumps({

"trim": "-1,3"

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '400 BAD REQUEST'

assert resp\_data == {'trim': ['start time must be greater than 0']}

# edit request

resp = client.put(

url,

data=json.dumps({

"trim": "0,0"

}),

content\_type='application/json'

)

resp\_data = json.loads(resp.data)

assert resp.status == '400 BAD REQUEST'

assert resp\_data == {'trim': ['end time must be greater than 1']}