Nhóm 23

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

Họ và tên: Hà Vĩ Khang

MSSV: 20110657

Milestone 4

Task 4: Xây dựng test case và tiến hành test cho chức năng tạo project (p2)

**5**. Kiểm tra uplaod video không thể đặt mã lưu trữ (id)

@pytest.mark.parametrize('filestreams', [('sample\_0.mp4',)], indirect=True)

@mock.patch('pymongo.collection.Collection.insert\_one',

side\_effect=ServerSelectionTimeoutError('Timeout error'))

def test\_upload\_project\_cant\_set\_storage\_id(mock\_insert\_one, test\_app, client, filestreams):

mp4\_stream = filestreams[0]

filename = 'sample\_0.mp4'

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

url = url\_for('projects.list\_upload\_project')

resp = client.post(

url,

data={

'file': (BytesIO(mp4\_stream), filename)

},

content\_type='multipart/form-data'

)

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

assert resp.status == '500 INTERNAL SERVER ERROR'

assert resp\_data == {'error': 'Timeout error'}

assert list(test\_app.mongo.db.projects.find()) == []

**6**. Kiểm tra upload liên tiếp nhiều video

@pytest.mark.parametrize('filestreams', [('sample\_0.mp4',)], indirect=True)

def test\_list\_projects(test\_app, client, filestreams):

mp4\_stream = filestreams[0]

filename = 'sample\_0.mp4'

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

url = url\_for('projects.list\_upload\_project')

resp = client.get(url)

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

assert resp\_data == {

'\_items': [],

'\_meta': {

'page': 1,

'max\_results': test\_app.config.get('ITEMS\_PER\_PAGE'),

'total': 0

}

}

# create 3 projects

for i in range(3):

client.post(

url,

data={

'file': (BytesIO(mp4\_stream), filename)

},

content\_type='multipart/form-data'

)

# list projects (1st page)

resp = client.get(url)

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

assert resp.status == '200 OK'

assert resp\_data['\_meta'] == {

'page': 1,

'max\_results': test\_app.config.get('ITEMS\_PER\_PAGE'),

'total': 3

}

assert len(resp\_data['\_items']) == test\_app.config.get('ITEMS\_PER\_PAGE')

assert '\_id' in resp\_data['\_items'][0]

assert 'filename' in resp\_data['\_items'][0]

assert 'storage\_id' in resp\_data['\_items'][0]

assert 'create\_time' in resp\_data['\_items'][0]

assert resp\_data['\_items'][0]['mime\_type'] == 'video/mp4'

assert resp\_data['\_items'][0]['request\_address'] == '127.0.0.1'

assert resp\_data['\_items'][0]['original\_filename'] == filename

assert resp\_data['\_items'][0]['version'] == 1

assert resp\_data['\_items'][0]['parent'] is None

assert resp\_data['\_items'][0]['processing'] == {

'video': False,

'thumbnail\_preview': False,

'thumbnails\_timeline': False

}

assert resp\_data['\_items'][0]['thumbnails'] == {'timeline': [], 'preview': {}}

assert resp\_data['\_items'][0]['url'] == url\_for('projects.get\_raw\_video',

project\_id=resp\_data["\_items"][0]["\_id"], \_external=True)

# list 1nd page explicitly

resp = client.get(url, query\_string={'page': 1})

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

assert resp\_data['\_meta'] == {

'page': 1,

'max\_results': test\_app.config.get('ITEMS\_PER\_PAGE'),

'total': 3

}

# list 2nd page explicitly

resp = client.get(url, query\_string={'page': 2})

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

assert len(resp\_data['\_items']) == 1

assert resp\_data['\_meta'] == {

'page': 2,

'max\_results': test\_app.config.get('ITEMS\_PER\_PAGE'),

'total': 3

}