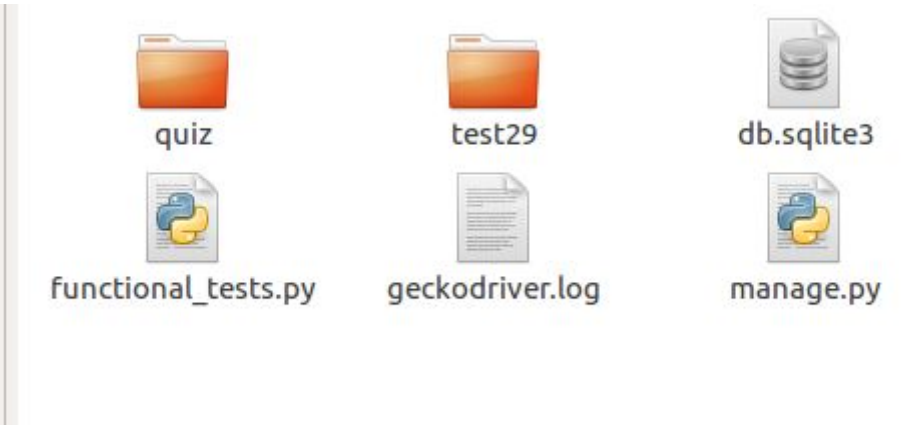
รายงาน test2 Software Development Practice II

เริ่มต้นด้วยการสร้าง Project ที่มีชื่อว่า test29 และสร้าง app ที่มีชื่อว่า quiz



จากนั้นจึงทำการเข้าไป installed apps ในไฟล์ settings.py โดยทำการติดตั้ง app ‘quiz’ ลงไปซึ่งจะเป็นการยืนยันให้ app ‘quiz ใช้งานกับฐานข้อมูลได้

```
INSTALLED_APPS = [
    'quiz.apps.QuizConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
```

หลังจากติดตั้ง app ‘quiz’ เรียบร้อยแล้วจึงทำการกำหนด url สำหรับ app ‘quiz’

```
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('quiz/', include('quiz.urls')),
]
```

ต่อมาทำการสร้างไฟล์ functional_tests.py และทำการเขียนการทดสอบดังภาพด้านล่าง

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
import unittest


class NewVisitorTest(unittest.TestCase):


    def setUp(self):
        self.browser = webdriver.Firefox()

    def tearDown(self):
        self.browser.quit()

    def test_can_use_functions(self):

        # Paul known a online quiz app.
        self.browser.get('http://localhost:8000/quiz/1/')
        # Paul notices the page title and header mention 'quiz'
        self.assertIn('quiz', self.browser.title)
```

- โดยได้กำหนดให้มีการเปิดหน้าเว็บที่ฟังก์ชั่น setUp() ซึ่งจะใช้ Firefox เป็น Browser
- กำหนดให้มีการ tearDown() สำหรับตอนจบเทส
- ทำการเริ่มทดสอบโดยมีฟังก์ชั่น test_can_use_functions()
    - เริ่มต้นด้วยการเข้าไปที่หน้าเว็บ app ‘quiz’
    - ตรวจสอบว่าเป็นเว็บ ‘quiz’ ด้วยการเทียบ title บนหน้าเว็บ browser

ต่อมาทำการ migrate ด้วยคำสั่ง python3 manage.py migrate สำหรับการเตรียมพร้อมที่จะสร้างฐานข้อมูล และทำการสร้างไฟล์ models.py ในโฟลเดอร์ quiz

```
import datetime
from django.db import models
from django.utils import timezone

# Create your models here.

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    def __str__(self):
        return self.question_text

    def was_published_recently(self):
        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)

class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
    def __str__(self):
        return self.choice_text
```

โดยจะทำการกำหนด class Question() และ class Choice() ขึ้นมาซึ่งจะกำหนดให้มีขนาดสำหรับเก็บ text ที่เป็นตัวอักษรขนาด 200 ตัวขึ้นมาสำหรับทั้ง Question() และ Choice()
เนื่องจาก Choice() จะมีการนับจำนวนที่กดจึงทำการเพิ่มพื้นที่สำหรับ votes ซึ่งเป็นตัวเลขจำนวนเต็ม หรือ Integer ที่มีค่าเริ่มต้นที่ 0

จากนั้นทำการสร้าง table ด้วยคำสั่ง python3 manage.py sqlmigrate quiz 0001 และทำการ migrate อีกครั้งเพื่อสร้างฐานข้อมูล

จากนั้นทำการเพิ่ม question และ choice ลงไปผ่าน shell ด้วยคำสั่งด้านล่างนี้

- เริ่มจากการ import ในส่วนของ models และ library ที่ต้องใช้
    - from quiz.models import Question, Choice
    - from django.utils import timezone
  - สำหรับเพิ่มคำถามลงไป
    - q = Question(question_text=”50 Dollas more values than 10 Dollas?”,
      pub_date=timezone.now())
    - q.save()
- ทำการสร้าง choice
  - c = Question.objects.get(pk=1) #ให้ c เป็นตัวเลือกสำหรับคำถามที่มี primary key เป็น 1 ซึ่งก็คือ ‘50 Dollas more values than 10 Dollas?’
- c.choice_set.create(choice_text='True', votes=0)
- c.choice_set.create(choice_text='False', votes=0)

ต่อมาทำการสร้าง views.py ในโฟลเดอร์ quiz และทำการเขียนฟังก์ชันลงไป

```python
from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect
from django.urls import reverse
from django.views import generic

from .models import Choice, Question

class DetailView(generic.DetailView):
    model = Question
    template_name = 'quiz/detail.html'


class ResultsView(generic.DetailView):
    model = Question
    template_name = 'quiz/results.html'


def vote(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    try:
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        # Redisplay the question voting form.
        return render(request, 'quiz/detail.html', {
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:
        selected_choice.votes += 1
        selected_choice.save()
        # Always return an HttpResponseRedirect after successfully dealing
        # with POST data. This prevents data from being posted twice if a
        # user hits the Back button.
        return HttpResponseRedirect(reverse('quiz:results', args=(question.id,)))
```

ซึ่งจะทำการ import libraries และ models ที่ต้องใช้งานเข้ามาก่อน โดยใน class DetailView() และ class ResultView() จะเป็นการกำหนดให้ใช้ model ของ Question และใช้ template ที่ชื่อว่า detail.html สำหรับ DetailView() และ result.html สำหรับ ResultView() เพื่อทำการเรนเดอร์แสดงผล

และในฟังก์ชั่น vote() จะทำการตรวจสอบก่อนว่าเรียกเจอคำถามตาม Primary key หรือไม่ ถ้าไม่เจอจะให้แสดงหน้า error 404 จากนั้นจะทำการกำหนดให้ selected_choice มีค่าเป็น choice ที่ถูกเลือกหลังจากที่มีการส่งค่าหรือ POST มา ซึ่งถ้าเกิดว่ามีการส่งค่าหรือ POST โดยที่ไม่มีการเลือก choice จะทำการเรนเดอร์หน้าเดิมคือ detail.html พร้อมแสดงคำเพิ่มขึ้นมาว่า “You didn’t select a choice” แต่หากมีการเลือก choice และส่งค่ามาจะทำการเพิ่มจำนวน

Vote ไปทีละ 1 ลงใน table model ของ choice ที่เป็น Interger สำหรับจำนวนคนโหวต และทำการ save() จากนั้นจะทำการ return เพื่อเรนเดอร์แสดงผลหน้า result.html ขึ้นมา

ทำการสร้าง template ‘detail.html’ ใน quiz/templates/quiz/

```
<html>
    <head>
        <title>quiz</title>
    </head>
    <body>

        <h1>{{ question.question_text }}</h1>

        {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}

        <form action="{% url 'quiz:vote' question.id %}" method="post">
        {% csrf_token %}
        {% for choice in question.choice_set.all %}
            <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}" />
            <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br />
        {% endfor %}
        <input type="submit" value="Vote" />
        </form>
    </body>
</html>
```

ทำการกำหนด title ว่า quiz และกำหนด headder ขนาด h1 ให้เป็นคำถามที่เลือกเอาไว้
ตามมาด้วยการใส่ error_message ซึ่งเป็นการส่งผ่านค่าเป็นตัวแปลภาษา python ซึ่งจะแสดงผลตามเงื่อนไขในฟังก์ชั่น vote() ใน views.py
จากนั้นจึงทำการสร้างฟอร์มสำหรับตัวเลือกโดยมีการป้องกันเบื้องต้นด้วย csrf_token และทำการวนลูปเพื่อเรียกตัวเลือกจากไอดี ในกรณีนี้มี 2 ไอดีคือ ‘1’ True และ ‘2’ False
จากนั้นทำการสร้างปุ่ม submit สำหรับยืนยันการส่งคำตอบ

ต่อมาทำการสร้าง template ‘result.html’ ใน quiz/templates/quiz/

```
<html>
    <head>
        <title>result</title>
    </head>
    <body>
        <h1>{{ question.question_text }}</h1>

        <ul>
        {% for choice in question.choice_set.all %}
            <li>{{ choice.choice_text }} -- {{ choice.votes }} vote{{ choice.votes|pluralize }}</li>
        {% endfor %}
        </ul>

        <a href="{% url 'quiz:detail' question.id %}">Vote again?</a>
    </body>
</html>
```

ทำการกำหนด title ว่า ‘result’ และกำหนด headder ขนาด h1 ให้เป็นคำถามที่เลือกเอาไว้
ทำการแสดงตัวเลือกทั้งหมดโดยการวนลูปทุกๆตัวเลือก โดยให้แสดงผลเป็น <choice> -- <จำนวนที่มีการกดเข้ามา>
แล้วทำการลิงค์ไปยังหน้าคำถามเมื่อต้องการโหวตอีกครั้ง

ทำการตั้งค่า url ใน quiz/urls.py

```
from django.urls import path

from . import views

app_name = 'quiz'
urlpatterns = [
    # ex: /quiz/5/
    path('<int:pk>/', views.DetailView.as_view(), name='detail'),
    # ex: /pquiz/5/results/
    path('<int:pk>/results/', views.ResultsView.as_view(), name='results'),
    # ex: /quiz/5/vote/
    path('<int:question_id>/vote/', views.vote, name='vote'),
]
```

จากนั้นทำการแก้ไข functional_tests.py

```
# Paul see question '50 Dollas more values than 10 Dollas?'
question = self.browser.find_element_by_tag_name('h1').text
self.assertIn('50 Dollas more values than 10 Dollas?', question)

# Paul knew the answer then clicked on the choice 'True'
time.sleep(1)
choice = self.browser.find_element_by_xpath(".//input[@type='radio' and @value='1']").click()
time.sleep(1)

# Paul accepted the answer 'True' by clicking the "Submit" button
submit = self.browser.find_element_by_xpath(".//input[@type='submit' and @value='Vote']").click()
time.sleep(3)
```

ทำการตรวจเทียบคำถามว่าเป็น ‘50 Dollas more values than 10 Dollas?’ หรือไม่ จากนั้นทำการเลือก choice โดยการใช้ find_element_by_xpath() โดยกำหนดให้เป็นรูปแบบ input จากนั้นทำการใช้คำสั่ง click() เพื่อเลือกตัวเลือกที่ 1 โดย @value คือไอดีของตัวเลือกนั้น ดังนั้นในกรณีนี้คือ ‘True’ จากนั้นจึงใช้หลักการเดียวกันในการกดปุ่ม submit

เอกสารอ้างอิงการใช้งาน find_element_by_xpath() :

1 :

http://selenium-python.readthedocs.io/locating-elements.html

2 :

https://stackoverflow.com/questions/32779563/how-can-i-click-submit-button