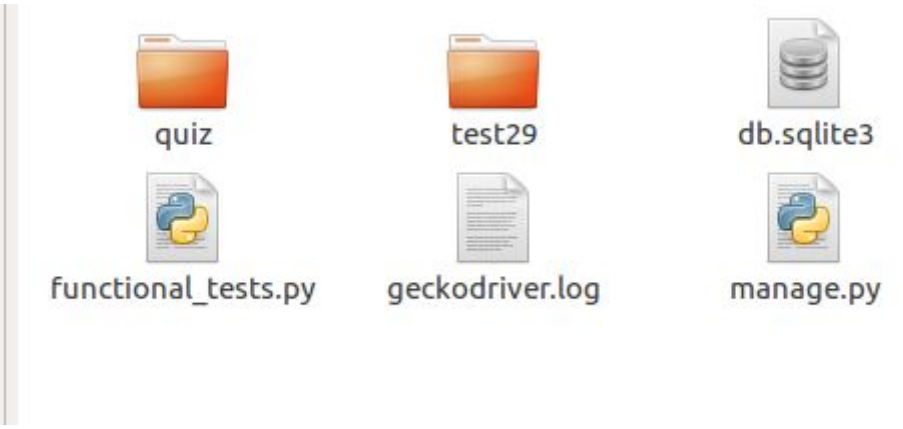
รายงาน test2 Software Development Practice II

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

Edit 06/7/2018

ทำการ add .gitignore เพื่อลบไฟล์ caches ที่ไม่จำเป็น เช่นไฟล์สกุล .pyc ของทั้ง Project 'test29' และของ App 'quiz'

จากนั้นทำการแก้ไข url เพื่อจะให้สามารถเรียกใช้งานได้ง่ายขึ้นจากเดิมที่ต้องมีการต่อท้าย zesstzero1.pythonanywhere.com/quiz/1

เป็น zesstzero1.pythonanywhere.com โดยได้ทำการเปลี่ยน url ใน settings.py จากเดิม ‘quiz/’ เป็น ‘ ’

```
from django.urls import path, include

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

จากนั้นทำการเพิ่มหน้า home.html โดยจะเป็นหน้าแรกสุดที่มีลิงค์ไปยังหน้าคำถาม

```
<html>
    <head>
        <title>Quiz App</title>
    </head>
    <body>
        <h1>Welcome to Quiz App</h1>
        <a href="{% url 'quiz:detail' 1 %}">Click here to do the Question</a>
    </body>
</html>
```

ทำการเพิ่มฟังก์ชั่น home_page() ใน views.py โดยจะทำการเรนเดอร์หน้า home.html ขึ้นมา

```
def home_page(request):
    return render(request,'quiz/home.html')
```

จากนั้นทำการแก้ไขเพิ่มเติม urls.py เพื่อกำหนดเส้นทางให้แก่ home_page()

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

]
```

ทำการแก้ไขให้ functional_tests.py เสร็จสมบูรณ์ โดยทำการเพิ่มให้มีการตรวจหน้าแรก(home.html) ว่ามี title ‘Quiz App’ หรือไม่ และตรวจสอบข้อความต้อนรับแบบ <h1> ว่าเป็น ‘Welcome to Quiz App’ หรือไม่ จากนั้นให้ทำการกดไปยังลิงค์ ‘Click here to do the Question’ ซึ่งจะพาไปยังหน้าคำถาม

```
def test_can_use_functions(self):

    # Paul known a online quiz app.
    self.browser.get('http://localhost:8000')
    # Paul notices the page title and header mention 'Quiz App'
    self.assertIn('Quiz App', self.browser.title)
    # Paul see 'Welcome to Quiz App'
    question = self.browser.find_element_by_tag_name('h1').text
    self.assertIn('Welcome to Quiz App', question)
    # Paul see Link 'Click here to do the Question' and clicked it
    h_link = self.browser.find_element_by_link_text('Click here to do the Question').click()
    time.sleep(1)
```

ต่อมาทำการเพิ่มในส่วนของการตรวจสอบหน้า results.html ว่ามี title ‘result’ หรือไม่ และทำการตรวจสอบว่ามี <h1> ‘50 Dollas

more values than 10 Dollas?’ หรือไม่ จากนั้นทำการตรวจสอบผลลัพธ์ว่ามีการแสดง True และ False ขึ้นมาหรือไม่ โดยได้มีการเรียกตรวจจาก id ของ choice ซึ่ง ‘True’ จะมี id = 1 และ ‘False’ จะมี id = 2 จากนั้นจึงทำการปิด Browser เมื่อเสร็จสิ้นการใช้งาน

```
# Paul notices the page title and header mention 'result'
self.assertIn('result', self.browser.title)
# 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 see result of 'True'
true = self.browser.find_element_by_id('1').text
self.assertIn('True', true)

# Paul see result of 'False'
false = self.browser.find_element_by_id('2').text
self.assertIn('False', false)

# Paul do not want to do the question anymore then Paul close the browser
self.browser.quit()
```

ทำการแก้ไขในไฟล์ results.html โดยได้ทำการเพิ่มไอดีลงไป โดยให้มีการวนลูปเป็นตัวเลข เราจะได้ไอดีที่เป็นตัวเลขของ choice มา ซึ่งเราจะนำไอดีดังกล่าวไประบุเพื่อตรวจสอบใน functional_tests ได้

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

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

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

** Edit 17 July 2018 **

```python
def add_question(request):
    question = Question.objects.all
    new_question = Question.objects.create(question_text = request.POST['q_in'])
    return redirect('/')


def add_choice(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    question.choice_set.create(choice_text='True', votes=0)
    question.choice_set.create(choice_text='False', votes=0)
    return render(request, 'quiz/detail.html', {'question': question})


def del_question(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    question.delete()
    return redirect('/')
```

ทำการเพิ่มฟังก์ชั่นการทำงานใน views.py มา 3 ส่วนดังนี้

- add_question() สำหรับเพิ่มคำถามโดยให้มีการ objects.create() หรือสร้างคำถามตามที่ได้รับข้อมูลจากการ Post มาจาก Inputbox ที่มีชื่อว่า ‘q_in’ ก่อนจะทำการ redirect กลับไปยังหน้า home (หน้าแรก)
- add_choice() โดยในฟังก์ชั่นนี้จะมีการรับค่าพารามิเตอร์ที่เป็น question_id เพื่อจะได้ระบุได้ว่าเป็นการสร้างตัวเลือกให้กับคำถามข้อใด จากนั้นจะทำการสร้าง choice ที่เป็น ‘True’ และ ‘False’ ที่ถูกกำหนดค่า votes เริ่มต้นให้เป็น 0 จากนั้นจะทำการ return ไปยังหน้า detail พร้อมส่งค่าของ question_id ไปด้วย
- del_question() ทำงานคล้ายกับ add_choice() ที่จะต้องรู้ค่า question_id เพื่อที่จะระบุว่าคำถามใดที่จะถูกลบ ก่แนจะใช้คำสั่ง .delete() ในการลบข้อมูลทิ้ง แล้วจึง return กลับไปยังหน้าแรก (home)

```python
    path('add_question', views.add_question, name='add_question'),
    path('<int:question_id>/add_choice/', views.add_choice, name='add_choice'),
    path('<int:question_id>/del_question/', views.del_question, name='del_question'),
]
```

ต่อมาทำการเพิ่ม url ของแต่ละฟังก์ชั่นใน urls.py โดย add_choice() และ del_question() นั้นจำเป็นต้องรู้ค่า question_id

```
    <title>Quiz App</title>
</head>
<body>
    <h1>Welcome to Quiz App</h1>
    <br>
    <h2>Add your Question</h2>
    <form method="POST" id="add_q" action="/add_question">
        <input name="q_in" id="new_question" placeholder="Add your Question" />
        {% csrf_token %}
    </form>
    <button type="submit" form="add_q" value="Submit">Add</button>

    <h3>Question list</h3>
    <table id="q_table">
        {% for question_text in question %}
            <tr><td><a href="{% url 'quiz:detail' question_text.id %}" id="question_number {{ forloo
        {% endfor %}
```

ในหน้า home.html ได้ทำการเพิ่ม inputbox สำหรับ add_question() โดยได้เพิ่ม header tage <h2> ‘Add your Question’ เพื่อให้รู้ว่าเป็นส่วนสำหรับเพิ่มคำถาม โดยในช่อง inputbox ก็มีการใส่ข้อความประเภท placeholder ว่า “Add your Question” เช่นกัน ถัดมาจะเป็นการสร้างปุ่ม submit ที่จะทำงานกับ inputbox ซึ่งระบุโดย id “add_q” โดยได้ตั้งชื่อปุ่มว่า “Add”

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

        <h1>{{ question.question_text }}</h1>
        <a href="{% url 'quiz:add_choice' question.id %}">Add True and False</a>
        <br><br>
        {% 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" />
        <br><br>
        </form>
        <br><br>
        <a href="{% url 'quiz:del_question' question.id %}">delete this question</a>
    </body>
</html>
```

ในหน้า detail.html ได้เพิ่มลิงค์สำหรับเรียกใช้งาน add_choice() ใช้ชื่อว่า “Add True and False”
และเพิ่มลิงค์ด้านล่างสุดสำหรับเรียกใช้งาน del_question() ใช้ชื่อว่า “delete this question”

functional_tests.py

```
# Paul known a online quiz app.
self.browser.get('http://localhost:8000')
# Paul notices the page title and header mention 'Quiz App'
self.assertIn('Quiz App', self.browser.title)
# Paul see 'Welcome to Quiz App'
header1 = self.browser.find_element_by_tag_name('h1').text
self.assertIn('Welcome to Quiz App', header1)
# Paul see 'Add your Question'
header2 = self.browser.find_element_by_tag_name('h2').text
self.assertIn('Add your Question', header2)

time.sleep(1)
# Paul add the question 'Is this a Quiz App?' to inputbox
inputbox = self.browser.find_element_by_id('new_question')
inputbox.send_keys('Is this a Quiz App?')
time.sleep(1)
# Paul enter 'Add' button
inputbox.send_keys(Keys.ENTER)
time.sleep(1)
```

ทำการเพิ่มเทสในส่วนของ add_question() โดยเริ่มมาจะทำการตรวจสอบ header tag <h2> ก่อนว่ามีคำว่า ‘Add your Question’ หรือไม่ จากนั้นกำหนดตัวแปร inputbox ให้เป็นelement ที่มีไอดี ‘new_question’ ซึ่งเป็นไอดีของฟอร์ม inputbox ในหน้า home.html ก่อนจะสั่งให้ทำการป้อนข้อความ ‘Is this a Quiz App?’ แล้วทำการกด submit ยืนยัน จึงเสร็จสิ้นการทดสอบการเพิ่มคำถาม

```
# Paul see 'Question List' below
header3 = self.browser.find_element_by_tag_name('h3').text
self.assertIn('Question list', header3)

# Paul see his question that's he made 'Is this a Quiz App?' and clicked it
question_of_Paul = self.browser.find_element_by_link_text('Is this a Quiz App?').click()
time.sleep(2)

# Paul notices the page title and header mention 'quiz'
self.assertIn('quiz', self.browser.title)
# Paul see question 'Is this a Quiz App?'
question = self.browser.find_element_by_tag_name('h1').text
self.assertIn('Is this a Quiz App?', question)

# Paul see 'Add True and False' for create choices and clicked it
create_choice = self.browser.find_element_by_link_text('Add True and False').click()
time.sleep(1)
```

ต่อมาทำการตรวจสอบด้านล่างซึ่งเป็น header tag <h3> ว่าเป็นส่วนของ ‘Question list’ หรือไม่
ก่อนจะใช้การ find_element_by_link_text() หาคำถาม ‘Is this a Quiz App?’ แล้วจึงสั่ง click()
แล้ว browser จะพามาที่หน้า detail ที่มี title ‘quiz’ โดยได้ทำการตรวจสอบว่ามี title ‘quiz’ ตรงกันหรือไม่ แล้วมีการแสดงชื่อคำถามที่มี header tag <h1> ‘Is this a Quiz App?’ หรือไม่
จากนั้นจึงทำการกดไปยังลิงค์ ที่มีชื่อว่า ‘Add True and False’

```
# Paul done the quiz but he wants to delete his question 'Is this a Quiz App?'
# so he came back to detail page by clicking 'Vote again?' below
vote_again = self.browser.find_element_by_link_text('Vote again?').click()
time.sleep(1)

# at the bottom there is a 'delete this question'
# Paul clicked it to delete 'Is this a Quiz App'
del_q = self.browser.find_element_by_link_text('delete this question').click()
time.sleep(1)

# browser bring Paul back to home page
# Paul do not want to do the question anymore then Paul close the browser
self.browser.quit()
```

ถัดมาได้เพิ่มให้มีการ ลบคำถาม โดยหลังจากที่ตรวจดูผลลัพธ์การโหวตแล้วจากเดิมที่จะจบการทำงานโดยการปิด browser คราวนี้จะทำการกดไปยังลิงค์ ‘Vote again?’ แทน จากนั้นระบบจะพากลับไปยังหน้าสำหรับตอบคำถาม ก่อนจะกดลิงค์ delete this question จากนั้นจะถูกพากลับไปยังหน้าแรก ก่อนจะทำการปิด browser

*** Edit 7 July 2018 part 2 ***

```
        return redirect( / )


    def add_choice(request, question_id):
        question = get_object_or_404(Question, pk=question_id)
        question.choice_set.create(choice_text = request.POST['c_in'], votes=0)
        return render(request,'quiz/detail.html', {'question': question})
```

ทำการแก้ไข add_choice() ใน views.py โดยเปลี่ยนให้มีการสร้าง choice จาก ‘c_in’ ซึ่งเป็นค่าที่รับมาจาก html form <inputbox> ใน detail.html

```
<h1>{{ question.question_text }}</h1>
<br>
<h2>Add your Choice</h2>
<form method="POST" id="add_c" action="/{{ question.id }}/add_choice">
    <input name="c_in" id="new_choice" placeholder="Add your Choice" />
    {% csrf_token %}
</form>
<button type="submit" form="add_c" value="Submit">Add</button>

<br><br>
```

โดยใน detail.html นั้นได้ทำการนำลิงค์เดิม (Add True and False) ออก แล้วทำการสร้าง inputbox ขึ้นมาโดยกำหนดชื่อตัวแปร ‘c_in’ สำหรับนำค่าที่ได้ Post ไปสร้างเป็นตัวเลือก
และได้กำหนด action ให้เรียกใช้งาน add_choice() ซึ่งจะมีการกำหนดค่า id ของคำถามลงไปด้วย เพราะจะต้องรู้ว่าเป็นตัวเลือกของคำถามใด จากนั้นจึงทำปุ่มกด Add สำหรับส่งข้อมูลแบบ Post

functional_tests.py

```
        # Paul add the 1st choice 'True' to inputbox
        input_choice1 = self.browser.find_element_by_id('new_choice')
        input_choice1.send_keys('True')
        time.sleep(1)
        # Paul enter 'Add' button
        input_choice1.send_keys(Keys.ENTER)
        time.sleep(3)

        # Paul add the 1st choice 'True' to inputbox
        input_choice2 = self.browser.find_element_by_id('new_choice')
        input_choice2.send_keys('False')
        time.sleep(1)
        # Paul enter 'Add' button
        input_choice2.send_keys(Keys.ENTER)
```

ในส่วนนี้นั้นเปลี่ยนจากเดิมที่ทำการกดไปที่ลิงค์ แล้วทำการสร้าง True กับ False ให้เปลี่ยนเป็นทำการกรอกข้อมูลลงใน inputbox ที่มี id = ‘new_choice’ ซึ่งสร้างเอาไว้ใน detail.html จากนั้นจึงทำการใส่ตัวเลือก ‘True’ ก่อนจะกดยืนยัน และทำการรอโดยมี time.sleep(3) เพื่อให้ข้อมูลได้ส่งไป ก่อนจะทำการสร้างตัวเลือกอีกครั้งหนึ่งโดยครั้งนี้ให้เป็นตัวเลือก ‘False’