**Міністерство освіти і науки України**

**Національний університет «Запорізька політехніка»**

кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи № 4

з дисципліни «Проектування сучасних інформаційних систем» на тему:

**«ПРИНЦИПИ ТЕСТУВАННЯ У DJANGO»**

Виконали:

студенти групи КНТ-119м Е.В.Гунько

Р.І. Макаренко

Прийняла:

к.т.н., доцент Г.В. Табунщик

**1. Мета роботи**

Ознайомитися із принципами тестування у Djangо, навчитися створювати власні тести та аналізувати результати

**2. Завдання до роботи**

Протестувати роботу веб-застосування та виявити чи має застосування порушення у функціональності або містить помилки, а також виявити та виправити помилки, якщо вони були допущені при написанні тестів.

**3. Хід роботи**

Під час створення застосування, важливим етапом є його тестування, що дозволяє впевнитися в правильності та коректності його роботи. Під час написання коду тестів було навмисно створено помилкову ситуацію в невірному коді.Далі наведено код програми

from django.test import TestCase

from django.urls import reverse\_lazy

import datetime

import pytz

from djangoapp import models

class TestCalls(TestCase):

def setUp(self):

self.create\_url = 'create\_measurement'

self.list\_url = 'measurement\_list'

def test\_call\_view\_loads(self):

for url in [self.create\_url, self.list\_url]:

path = str(reverse\_lazy(url))

response = self.client.get(path)

self.assertEqual(response.status\_code, 404)

self.assertTemplateUsed(response, '{}.html'.format(url))

def test\_call\_view\_fails\_blank(self):

path = str(reverse\_lazy(self.create\_url))

response = self.client.post(path, {})

self.assertFormError(response, 'form', 'value',

u'This field is required.')

def test\_call\_view\_fails\_incorrect(self):

path = str(reverse\_lazy(self.create\_url))

value = '1'\*51

measurement\_params = {

'value': value,

}

response = self.client.post(path, measurement\_params)

self.assertFormError(response, 'form', 'value',

u'Ensure this value has at most 50 characters (it has 51).')

def test\_call\_view\_succeed(self):

path = str(reverse\_lazy(self.create\_url))

value = '1'\*5

description = 'measurement description'

time = datetime.datetime.now()

measurement\_params = {

'value': value,

'description': description,

'time': time

}

response = self.client.post(path, measurement\_params)

self.assertEqual(response.status\_code, 302)

measurement = models.Measurement.objects.all().reverse()[0]

measurement\_params['time'] = time.astimezone(pytz.UTC)

for field in measurement\_params.keys():

measurement\_field = getattr(measurement, field)

self.assertEqual(measurement\_field, measurement\_params[field])

def test\_call\_view\_succeed\_with\_blank\_description(self):

path = str(reverse\_lazy(self.create\_url))

value = '1'\*5

description = ''

time = datetime.datetime.now()

measurement\_params = {

'value': value,

'description': description,

'time': time

}

response = self.client.post(path, measurement\_params)

self.assertEqual(response.status\_code, 302)

Далі наведено створену діаграму класів веб-застосування

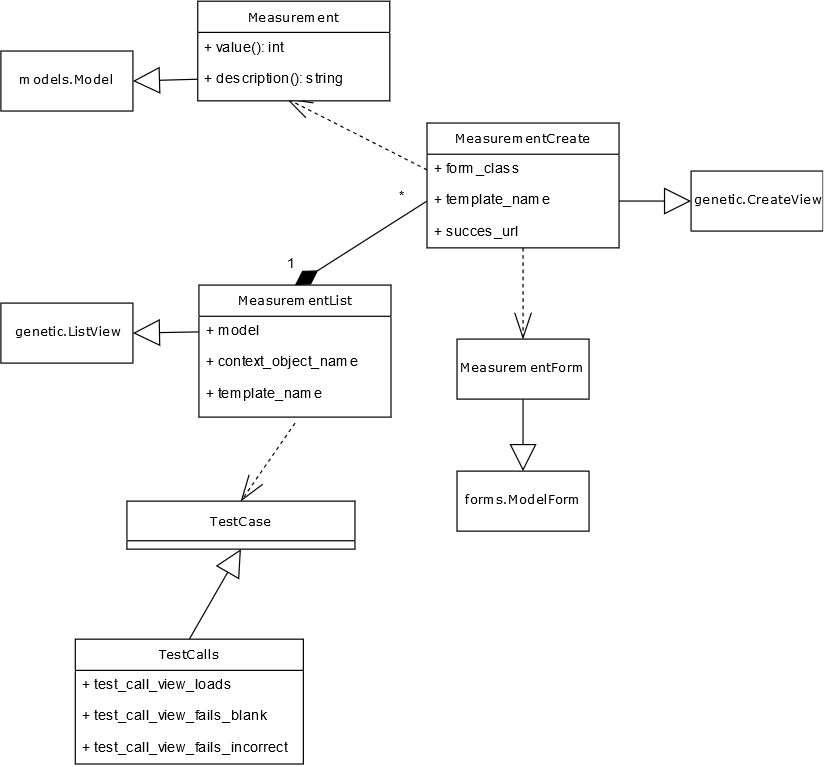


Рисунок 3.4 – Діаграма класів

**4. Висновок**

В цій лабораторній роботі було вивчено принципи застосування в Django та виконано тестування створеного веб-застосування