МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП’ЮТЕРНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

**Звіт**

**з технологічної практики**

студента 4 курсу групи ІПЗ-19-1

спеціальності 121 «Інженерія програмного забезпечення»

Драка Тараса Сергійовича

(прізвище, ім’я та по-батькові)

Керівник

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії Кузьменко О.В.

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чижмотря О.Г.

(підпис) (прізвище та ініціали)

Локтікова Т.М. (підпис) (прізвище та ініціали)

Житомир – 2021

**ЗМІСТ**

[Розділ 1. Технічне завдання на розробку веб-орієнтованої системи для продажу авіаквитків 4](#_Toc75459511)

[**Функціональні вимоги щодо системи.** 6](#_Toc75459512)

[Розділ 2. Проектування структури та логіки роботи системи 8](#_Toc75459513)

[**Діаграма активності** 8](#_Toc75459514)

[**Діаграма моделей БД** 9](#_Toc75459515)

[**Діаграма варіантів використання** 12](#_Toc75459516)

[Розділ 3. Хід реалізації 14](#_Toc75459517)

[Розділ 4. Інструкція користувача 24](#_Toc75459518)

[Розділ 5. Тестування 34](#_Toc75459519)

[Висновки 36](#_Toc75459520)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 37](#_Toc75459521)

# **Розділ 1. Технічне завдання на розробку веб-орієнтованої системи для продажу авіаквитків**

**Постановка задачі:** Необхідно розробити, здійснити тестування і розгортання веб-орієнтованої багатокористувацької системи для *продажу авіаквитків*.

**Інструменти**: Python, Django, SQLLite,

**Що входить в проект:** розробка бази даних багатокористувацької системи для продажу авіаквитків.

**Функціональні можливості системи:**

• Перегляд новин: Користувачі можуть переглядати список новин, відсортованих за датою публікації або за категоріями (наприклад, Події тощо).

• Деталі новини: При натисканні на окрему новину, користувач отримує повний текст новини, разом з фотографіями та додатковими деталями.

• Категорії новин: Сайт має розділення новин на різні категорії, щоб допомогти користувачам знайти цікаві їм матеріали швидше і зручніше.

• Адміністративна панель: Ви, як розробник, використовуєте адміністративну панель Django для керування контентом сайту, таким як додавання нових новин, редагування існуючих матеріалів, керування категоріями тощо.редагування існуючих матеріалів, керування категоріями тощо.

**Загальні функції системи:**

* Наявність клієнтської (frontend) та адміністративної (backend) частини;
* Категорії новин;
* Реєстрація користувачів;
* Перегляд новин
* Можливість залишити коментар під постом

**Функції адміністративної частини:**

* Робота з категоріями: CRUD
* Робота з новинами CRUD
* Робота з користувачами: CRUD

**Функції клієнтської частини:**

* Стандартні функції користувачів: реєстрація, вхід.
* Перегляд новин.
* Можливість написати feedback до сайту
* Сортування новин за категоріями

## **Функціональні вимоги щодо системи.**

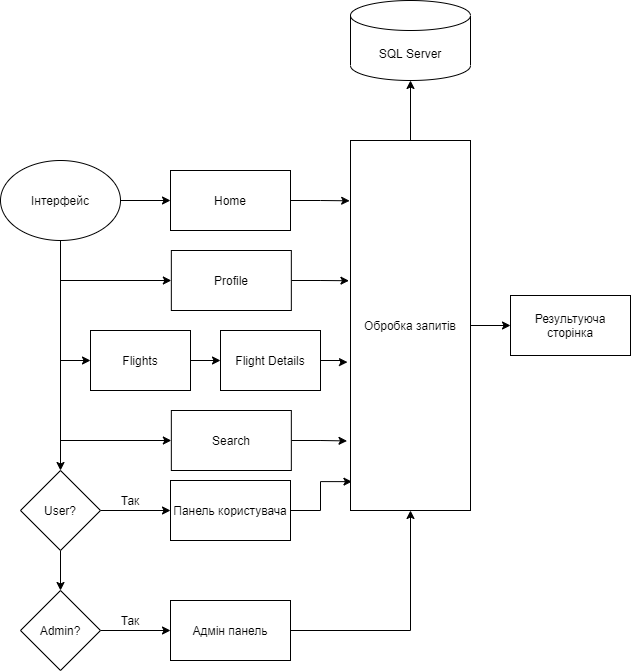
На етапі проектування потрібно скласти загальний алгоритм роботи програми (рис. 1.4).

Рис. 1.2. Загальна схема роботи програми.

**Загальні вимоги**

* В системі потрібно передбачити ролі “Користувач” та “Адмін”.

**Функціональні вимоги клієнтської частини:**

Після завантаження сайту з’являється головна сторінка.

Можна переключатись між вкладками.

**Вкладка «Flights»:**

* + - *Детальна інформація про кожен рейс* (“Flights”). Елементи: назва авіакомпанії, літака, місце вильоту, місце прибуття, час вильоту, час прибуття, тривалість польоту, кількість місць.
    - *Детальна інформація про кожен виліт* (“Details”). Елементи: дата вильоту, кількість доступних місць, кількість квитків, ціна квитка, попередні дані.

**Вкладка «Home»:**

* + - Можливість знайти потрібний маршрут (місце вильоту, місце прибуття).
    - Перехід на вкладки «Flights», «Login», «Register», «Profile», «Logout»

**Вкладка «Profile»:**

* + - Список куплених квитків.
    - Загальна інформація про користувача: ім’я, прізвище, електронна пошта, номер телефону, адреса

**Функціональні вимоги адміністративної частини:**

**Загальні вимоги щодо адміністративної частини:**

* + - Доступ користувачам з роллю Admin

**Сторінки для роботи з сутностями:**

* + - CRUD по рейсам
    - CRUD по польотам рейсів
    - Зв'язок між рейсами і квитками
    - Створення/обробка польотів за рейсом



Рис. 1.3. Вхід в систему.

# **Розділ 2. Проектування структури та логіки роботи системи**

Перед початком розробки додатку було проаналізовано технічне завдання і спроектовано модель БД системи – рис. 1.1. Структуру та логіку роботи зображено на наступних UML-діаграмах.

## **Діаграма активності**

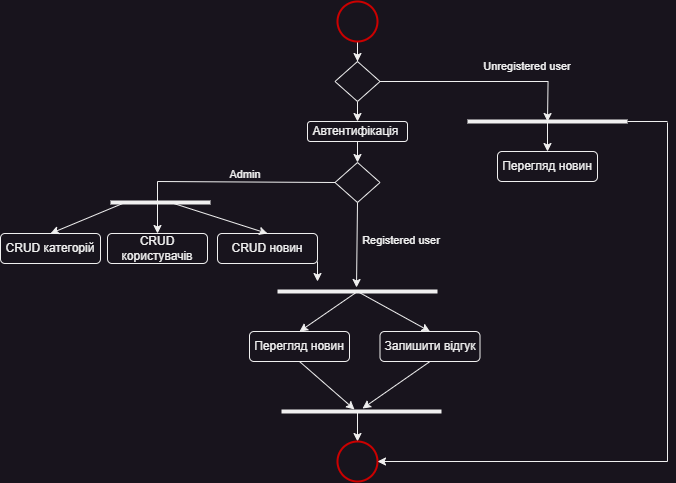


Рис. 2.1 Діаграма активності.

На діаграмі зображена можлива діяльність в додатку. Після запуску додатку користувач має можливість здійснити вхід чи реєстрацію. В разі успішної автентифікації користувач розглядається як адміністратор | користувач.

**Не зареєстрований Користувач** може переглядати дивитись лише новини

**Адміністратор** в свою чергу має можливість виконувати CRUD-операції

**Користувач** має можливість дивитись новини та залиши відгук.

## **Діаграма моделей БД**

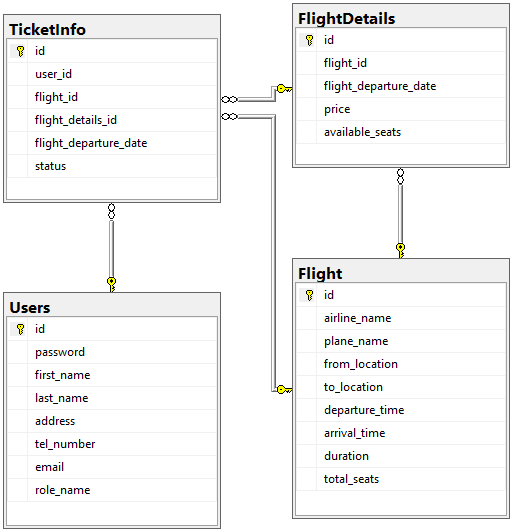


Рис. 2.2 Діаграма моделей БД.

## **Діаграма варіантів використання**

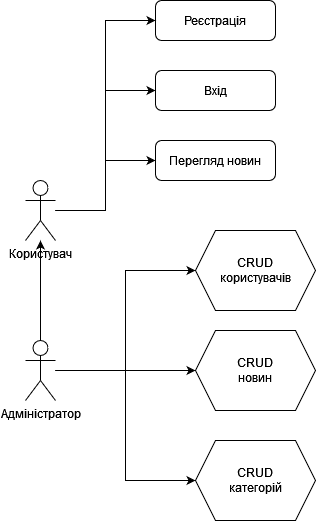


Рис. 2.3 Діаграма варіантів використання.

На діаграмі використання зображено відношення між користувачами та доступними їм функціям.

Користувач має можливість зареєструватися , війти до свого акаунту, та переглянути новини на сайті.

Адміністратор розширює можливості користувача та може виконувати CRUD-операції над новинами, категоріями та користувачами.

# **Розділ 3. Хід реалізації**

**2023-04-29**

**Формальні задачі:**

1. Встановлення програмного оточення:

–Visual StudioCode, Git, Python, Django, SQLLite

**Робота з проектом:**

1. Розробка шаблону веб-сторінки.
2. Створення каталогу з новинами.
3. Створення наступних компонентів: urls.py, manage

**2021-05-30**

**Формальні задачі:**

1. Побудова діаграм активності, моделей, варіантів використання.

2) Міграції: Генерація міграційних файлів на основі визначених моделей

3) Виконання міграцій: Застосування міграцій для створення таблиць бази даних.

4) Реєстрація моделей: Реєстрація моделей в адміністративному інтерфейсі Django.

5) Налаштування відображення моделей у панелі адміністратора Django.

**2023-05-31**

**Робота з користувачами:**

1. Можливість входу до системи з бек-частини.
2. Верстка сторінки входу.
3. Реалізація автентифікації, реєстрування.

**2023-06-01**

**Робота з рейсами:**

1. Реалізація контролерів новин

**2023-06-02**

Робота з формами: Створення форм для збору даних від користувачів та їх обробки на серверному боці.

**2023-06-03**

Робота з базою даних: Виконання операцій збереження, оновлення, видалення та отримання даних з бази даних, використовуючи моделі та запити Django ORM.

**2023-06-03**

Заповнення сайту данними

**2023-06-04**

Покрокове тестування: Перевірка роботи вашого проекту, включаючи тестування функціональності, валідації форм, обробки помилок та іншого.

**Приклад програмного коду:**

Моделі новин і категорій.

from django.db import models

from django.urls import reverse

# Create your models here.

class News(models.Model):

    title = models.CharField(max\_length=150)

    content = models.TextField(blank=True)

    created\_at = models.DateTimeField(auto\_now\_add=True)

    updated\_at = models.DateTimeField(auto\_now=True)

    photo = models.ImageField(upload\_to='photos/%Y/%m/%d/', blank=True)

    is\_published = models.BooleanField(default=True)

    category = models.ForeignKey('Category', on\_delete=models.CASCADE, null=True)

    views = models.IntegerField(default=0)

    def get\_absolute\_url(self):

        return reverse('view\_news', kwargs={'news\_id': self.pk})

    def \_\_str\_\_(self):

        return self.title

    class Meta:

        verbose\_name = 'News'

        verbose\_name\_plural = 'News'

        ordering = ['-created\_at']

class Category(models.Model):

    title = models.CharField(max\_length=150, db\_index=True, verbose\_name='Category name')

    def get\_absolute\_url(self):

        return reverse('category', kwargs={'category\_id': self.pk})

    def \_\_str\_\_(self):

        return self.title

    class Meta:

        verbose\_name = 'Category'

        verbose\_name\_plural = 'Categories'

        ordering = ['title']

Форми реєстрації, аутентифікації та новин.

from django import forms

from .models import News, Category

import re

from django.core.exceptions import ValidationError

from django.contrib.auth.forms import UserCreationForm, AuthenticationForm

from django.contrib.auth.models import User

from captcha.fields import CaptchaField

class ContactForm(forms.Form):

    subject = forms.CharField(label='Subject', widget=forms.TextInput(attrs={"class": "form-control"}))

    body = forms.CharField(label='Body', widget=forms.Textarea(attrs={"class": "form-control"}))

    # email = forms.EmailField(label='Email', widget=forms.EmailInput(attrs={"class": "form-control"}))

    captcha = CaptchaField()

class UserRegisterForm(UserCreationForm):

    email = forms.EmailField(label='Email', widget=forms.EmailInput(attrs={"class": "form-control"}))

    username = forms.CharField(label='Username', widget=forms.TextInput(attrs={"class": "form-control"}))

    password1 = forms.CharField(label='Password', help\_text='Password must contain at least 8 characters',

                                widget=forms.PasswordInput(attrs={"class": "form-control"}))

    password2 = forms.CharField(label='Repeat password', widget=forms.PasswordInput(attrs={"class": "form-control"}))

    class Meta:

        model = User

        fields = ['username', 'email', 'password1', 'password2']

class UserAuthenticationForm(AuthenticationForm):

    username = forms.CharField(label='Username', widget=forms.TextInput(attrs={"class": "form-control"}))

    password = forms.CharField(label='Password', widget=forms.PasswordInput(attrs={"class": "form-control"}))

class NewsForm(forms.ModelForm):

    class Meta:

        model = News

        # fields = '\_\_all\_\_'

        fields = ['title', 'content', 'is\_published', 'category']

        widgets = {

            'title': forms.TextInput(attrs={"class": "form-control"}),

            'content': forms.Textarea(attrs={"class": "form-control"}),

            'category': forms.Select(attrs={"class": "form-select"})

        }

    def clean\_title(self):

        title = self.cleaned\_data['title']

        if re.match(r'\d', title):

            raise ValidationError('Title can`t starts with number')

        return title

    '''

    title = forms.CharField(max\_length=150, label='Name', widget=forms.TextInput(attrs={"class": "form-control"}))

    content = forms.CharField(label='Description', required=False, widget=forms.Textarea(attrs={"class": "form-control"}))

    is\_published = forms.BooleanField(label='Published', initial=True, widget=forms.CheckboxInput(attrs={"class": "form-check-input"}))

    category = forms.ModelChoiceField(queryset=Category.objects.all(), label='Category', empty\_label='Choose category',

                                      widget=forms.Select(attrs={"class": "form-select"}))

                                      '''

class CategoriesForm(forms.ModelForm):

    class Meta:

        model = Category

        fields = ['title']

        widgets = {

            'title': forms.TextInput(attrs={"class": "form-control"})

        }

    def clean\_title(self):

        title = self.cleaned\_data['title']

        if re.match(r'\d', title):

            raise ValidationError('Category name can`t starts with number')

        return title

Компонент routes.js:

from django.urls import path

from .views import \*

from django.views.decorators.cache import cache\_page

urlpatterns = [

    #path('', index, name='home'),

    #path('', cache\_page(60)(HomeNews.as\_view()), name='home'),

    path('', HomeNews.as\_view(), name='home'),

    path('category/<int:category\_id>/', NewsCategoryList.as\_view(), name='category'),

    path('news/<int:news\_id>/', ViewNews.as\_view(), name='view\_news'),

    path('news/add-news', CreateNews.as\_view(), name='add\_news'),

    path('news/add-category', CreateCat.as\_view(), name='add\_category'),

    path('register/', register, name='register'),

    path('login/', user\_login, name='login'),

    path('logout/', user\_logout, name='logout'),

    path('feedback/', send\_email, name='feedback')

]

urls.py

from django.urls import path

from .views import \*

from django.views.decorators.cache import cache\_page

urlpatterns = [

    #path('', index, name='home'),

    #path('', cache\_page(60)(HomeNews.as\_view()), name='home'),

    path('', HomeNews.as\_view(), name='home'),

    path('category/<int:category\_id>/', NewsCategoryList.as\_view(), name='category'),

    path('news/<int:news\_id>/', ViewNews.as\_view(), name='view\_news'),

    path('news/add-news', CreateNews.as\_view(), name='add\_news'),

    path('news/add-category', CreateCat.as\_view(), name='add\_category'),

    path('register/', register, name='register'),

    path('login/', user\_login, name='login'),

    path('logout/', user\_logout, name='logout'),

    path('feedback/', send\_email, name='feedback')

]

Налаштування Django проекту

"""

Django settings for study\_site project.

Generated by 'django-admin startproject' using Django 3.2.8.

For more information on this file, see

https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/3.2/ref/settings/

"""

import os

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-k9\*f\*76lp1\_^d%msm$r23+28h=i(ojayusi7nx2mphg+28zp!o'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'debug\_toolbar',

    'news.apps.NewsConfig',

    'ckeditor',

    'ckeditor\_uploader',

    'captcha',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    #'whitenoise.middleware.WhiteNoiseMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

    'debug\_toolbar.middleware.DebugToolbarMiddleware',

]

ROOT\_URLCONF = 'study\_site.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [os.path.join(BASE\_DIR, 'templates')],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'study\_site.wsgi.application'

# Database

# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

# Password validation

# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

# Internationalization

# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC\_URL = '/static/'

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')

STATICFILES\_DIRS = [

    os.path.join(BASE\_DIR, 'study\_site/static'),

]

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

MEDIA\_URL = '/media/'

# Default primary key field type

# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

INTERNAL\_IPS = [

    '127.0.0.1'

]

EMAIL\_HOST = 'smtp.ukr.net'

EMAIL\_PORT = 465

EMAIL\_HOST\_USER = 'work\_test813@ukr.net'

EMAIL\_HOST\_PASSWORD = 'VHDq5AKTM8HWPpAK'

EMAIL\_USE\_TLS = False

EMAIL\_USE\_SSL = True

CKEDITOR\_UPLOAD\_PATH = 'uploads/'

CKEDITOR\_CONFIGS = {

    'default': {

        'skin': 'moono-lisa',

        # 'skin': 'office2013',

        'toolbar\_Basic': [

            ['Source', '-', 'Bold', 'Italic']

        ],

        'toolbar\_YourCustomToolbarConfig': [

            {'name': 'document', 'items': ['Source', '-', 'Save', 'NewPage', 'Preview', 'Print', '-', 'Templates']},

            {'name': 'clipboard', 'items': ['Cut', 'Copy', 'Paste', 'PasteText', 'PasteFromWord', '-', 'Undo', 'Redo']},

            {'name': 'editing', 'items': ['Find', 'Replace', '-', 'SelectAll']},

            {'name': 'forms',

             'items': ['Form', 'Checkbox', 'Radio', 'TextField', 'Textarea', 'Select', 'Button', 'ImageButton',

                       'HiddenField']},

            '/',

            {'name': 'basicstyles',

             'items': ['Bold', 'Italic', 'Underline', 'Strike', 'Subscript', 'Superscript', '-', 'RemoveFormat']},

            {'name': 'paragraph',

             'items': ['NumberedList', 'BulletedList', '-', 'Outdent', 'Indent', '-', 'Blockquote', 'CreateDiv', '-',

                       'JustifyLeft', 'JustifyCenter', 'JustifyRight', 'JustifyBlock', '-', 'BidiLtr', 'BidiRtl',

                       'Language']},

            {'name': 'links', 'items': ['Link', 'Unlink', 'Anchor']},

            {'name': 'insert',

             'items': ['Image', 'Flash', 'Table', 'HorizontalRule', 'Smiley', 'SpecialChar', 'PageBreak', 'Iframe']},

            '/',

            {'name': 'styles', 'items': ['Styles', 'Format', 'Font', 'FontSize']},

            {'name': 'colors', 'items': ['TextColor', 'BGColor']},

            {'name': 'tools', 'items': ['Maximize', 'ShowBlocks']},

            {'name': 'about', 'items': ['About']},

            '/',  # put this to force next toolbar on new line

            {'name': 'yourcustomtools', 'items': [

                # put the name of your editor.ui.addButton here

                'Preview',

                'Maximize',

            ]},

        ],

        'toolbar': 'YourCustomToolbarConfig',  # put selected toolbar config here

        # 'toolbarGroups': [{ 'name': 'document', 'groups': [ 'mode', 'document', 'doctools' ] }],

        # 'height': 291,

        # 'width': '100%',

        # 'filebrowserWindowHeight': 725,

        # 'filebrowserWindowWidth': 940,

        # 'toolbarCanCollapse': True,

        # 'mathJaxLib': '//cdn.mathjax.org/mathjax/2.2-latest/MathJax.js?config=TeX-AMS\_HTML',

        'tabSpaces': 4,

        'extraPlugins': ','.join([

            'uploadimage', # the upload image feature

            # your extra plugins here

            'div',

            'autolink',

            'autoembed',

            'embedsemantic',

            'autogrow',

            # 'devtools',

            'widget',

            'lineutils',

            'clipboard',

            'dialog',

            'dialogui',

            'elementspath'

        ]),

    }

}

CACHES = {

    'default': {

        'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',

        'LOCATION': os.path.join(BASE\_DIR, 'django\_cache'),

    }

}

Код Django Views

from django.contrib import messages

from django.core.mail import send\_mail

from django.urls import reverse\_lazy

from django.views.generic import ListView, DetailView, CreateView

from django.db.models import \*

from .models import News, Category

from django.shortcuts import render, get\_object\_or\_404, redirect

from .forms import NewsForm, CategoriesForm, UserRegisterForm, UserAuthenticationForm, ContactForm

from django.contrib.auth.mixins import LoginRequiredMixin

from django.contrib.auth import login, logout

# Замена ф-ции index для отображения списка новостей

class HomeNews(ListView):

    paginate\_by = 5

    model = News

    template\_name = 'news/index.html'

    context\_object\_name = 'news'

    # extra\_context = {'title': 'News List'}

    def get\_context\_data(self, \*, object\_list=None, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        context['title'] = 'News List'

        return context

    def get\_queryset(self):

        return News.objects.filter(is\_published=True).select\_related('category')

'''

def index(request):

    news = News.objects.all()

    context = {

        'news': news,

        'title': 'News List',

    }

    return render(request, 'news/index.html', context=context)

'''

# Замена ф-ции get\_category для отображения списка новостей по категориям

class NewsCategoryList(ListView):

    paginate\_by = 5

    model = News

    template\_name = 'news/index.html'

    context\_object\_name = 'news'

    allow\_empty = False

    def get\_context\_data(self, \*, object\_list=None, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        context['title'] = Category.objects.get(pk=self.kwargs['category\_id'])

        return context

    def get\_queryset(self):

        return News.objects.filter(category\_id=self.kwargs['category\_id'], is\_published=True).select\_related('category')

'''

def get\_category(request, category\_id):

    news = News.objects.filter(category\_id=category\_id)

    category = Category.objects.get(pk=category\_id)

    return render(request, 'news/category.html', {'news': news, 'category': category})

'''

# Замена ф-ции view\_news для отображения одной новости

class ViewNews(DetailView):

    model = News

    pk\_url\_kwarg = 'news\_id'

    template\_name = 'news/view\_news.html'

    context\_object\_name = 'news\_item'

'''

def view\_news(request, news\_id):

    news\_item = get\_object\_or\_404(News, pk=news\_id)

    return render(request, 'news/view\_news.html', {'news\_item': news\_item})

'''

class CreateNews(LoginRequiredMixin, CreateView):

    form\_class = NewsForm

    template\_name = 'news/add\_news.html'

'''

def add\_news(request):

    if request.method == 'POST':

        form\_news = NewsForm(request.POST)

        if form\_news.is\_valid():

            # news = News.objects.create(\*\*form.cleaned\_data)

            news = form\_news.save()

            return redirect(news)

    else:

        form\_news = NewsForm()

    return render(request, 'news/add\_news.html', {'form': form\_news})

'''

class CreateCat(LoginRequiredMixin, CreateView):

    form\_class = CategoriesForm

    template\_name = 'news/add\_category.html'

    success\_url = reverse\_lazy('home')

'''

def add\_category(request):

    if request.method == 'POST':

        form\_cat = CategoriesForm(request.POST)

        if form\_cat.is\_valid():

            form\_cat.save()

            return redirect('home')

    else:

        form\_cat = CategoriesForm()

    return render(request, 'news/add\_category.html', {'form': form\_cat})

'''

def register(request):

    if request.method == 'POST':

        login\_form = UserRegisterForm(request.POST)

        if login\_form.is\_valid():

            user = login\_form.save()

            login(request, user)

            messages.success(request, 'Register is succesfull')

            return redirect('home')

        else:

            messages.error(request, 'Error')

    else:

        login\_form = UserRegisterForm()

    return render(request, 'news/register.html', {'form': login\_form})

def user\_login(request):

    if request.method == 'POST':

        auth\_form = UserAuthenticationForm(data=request.POST)

        if auth\_form.is\_valid():

            user = auth\_form.get\_user()

            login(request, user)

            return redirect('home')

    else:

        auth\_form = UserAuthenticationForm()

    return render(request, 'news/login.html', {'form': auth\_form})

def user\_logout(request):

    logout(request)

    return redirect('login')

def send\_email(request):

    if request.method == 'POST':

        feedback\_form = ContactForm(request.POST)

        if feedback\_form.is\_valid():

            mail = send\_mail(feedback\_form.cleaned\_data['subject'], feedback\_form.cleaned\_data['body'],

                             'work\_test813@ukr.net', ['Drak@gmail.com'], fail\_silently=True)

            if mail:

                messages.success(request, 'E-mail sended')

                return redirect('home')

            else:

                messages.error(request, 'Error')

    else:

        feedback\_form = ContactForm()

    return render(request, 'news/feedback.html', {'form': feedback\_form})

# **Розділ 4. Інструкція користувача**

Після входу на сайт з’являється головна сторінка (рис. 4.1), де є новини

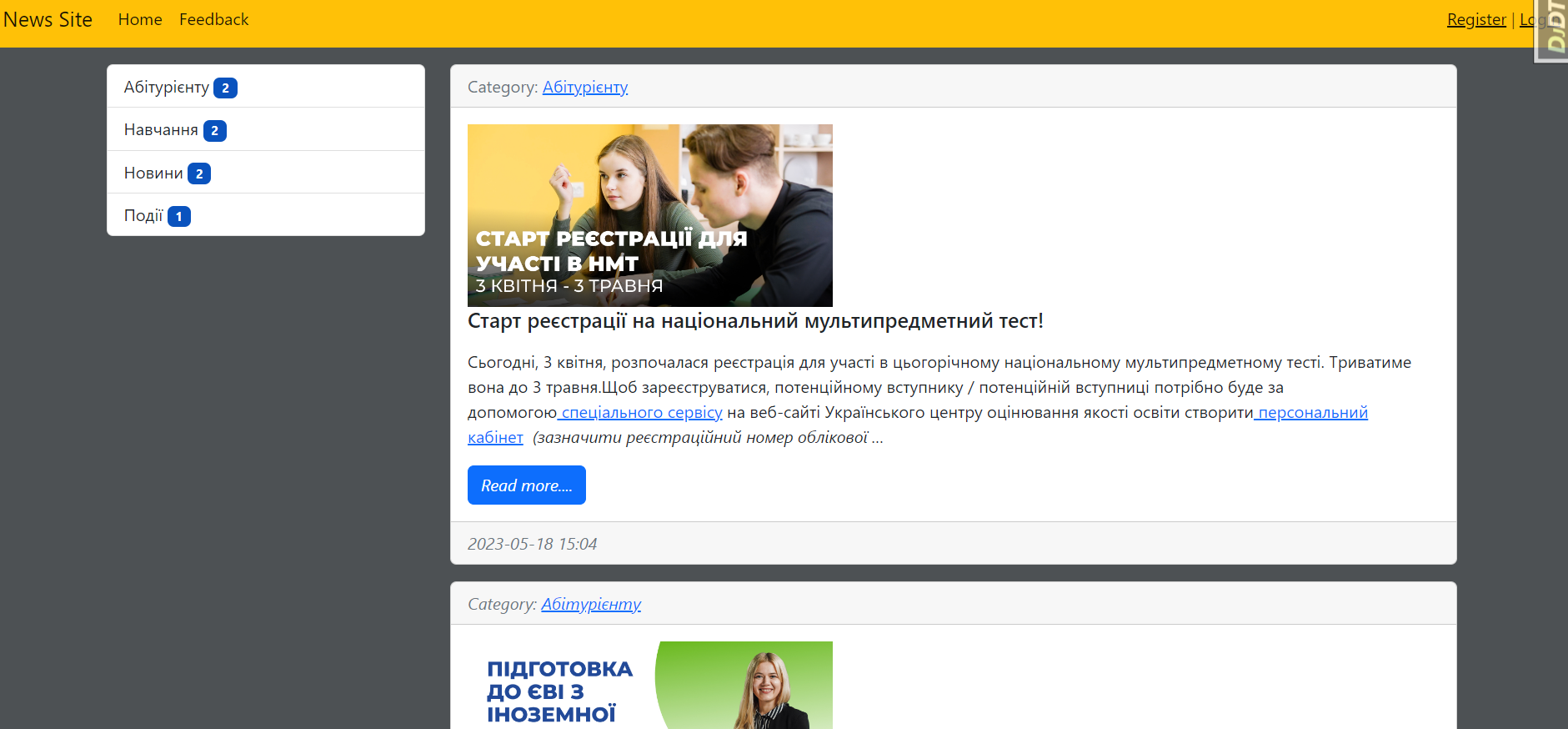


Рис 4.1. Головна сторінка.

Функціонал веб-сервісу доступний для всіх, написати відгук потрібно провести регістрацію на сайті

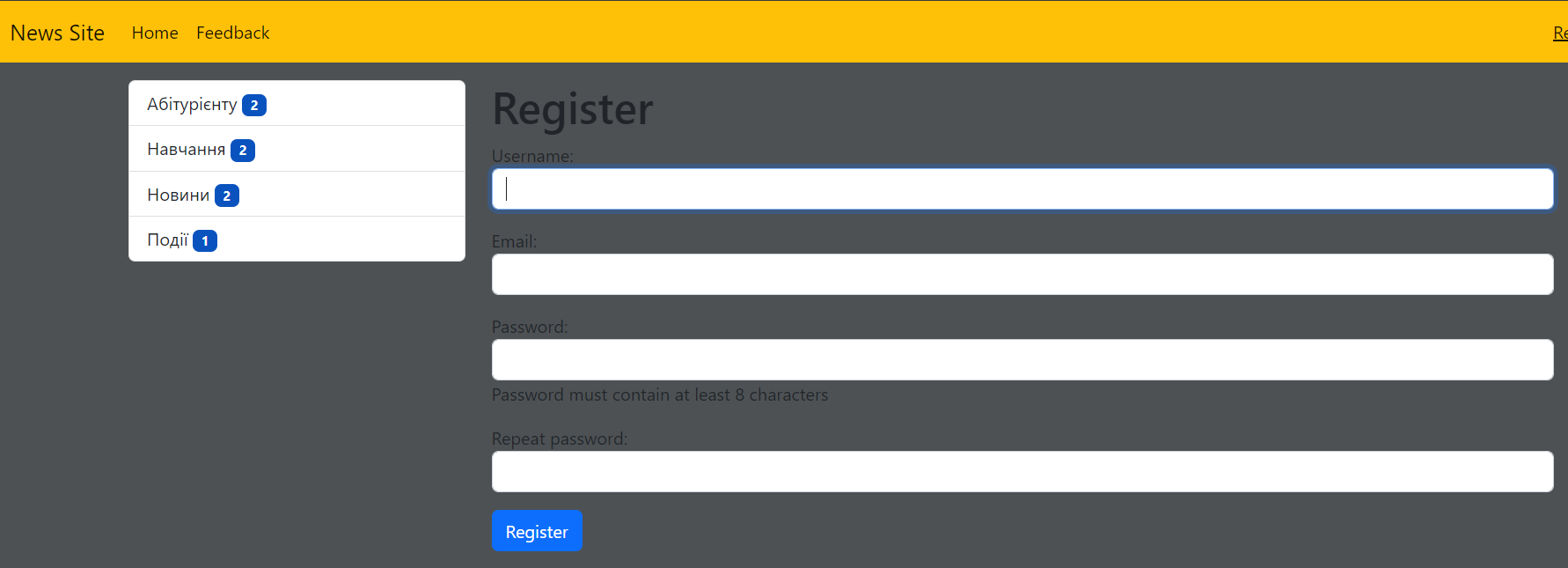


Рис 4.1. Сторінка реєстрації.

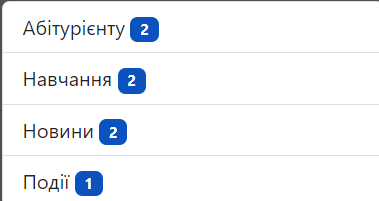


Рис 4.2. Категорії.

****



Рис 4.3. Новина у категорії Навчання

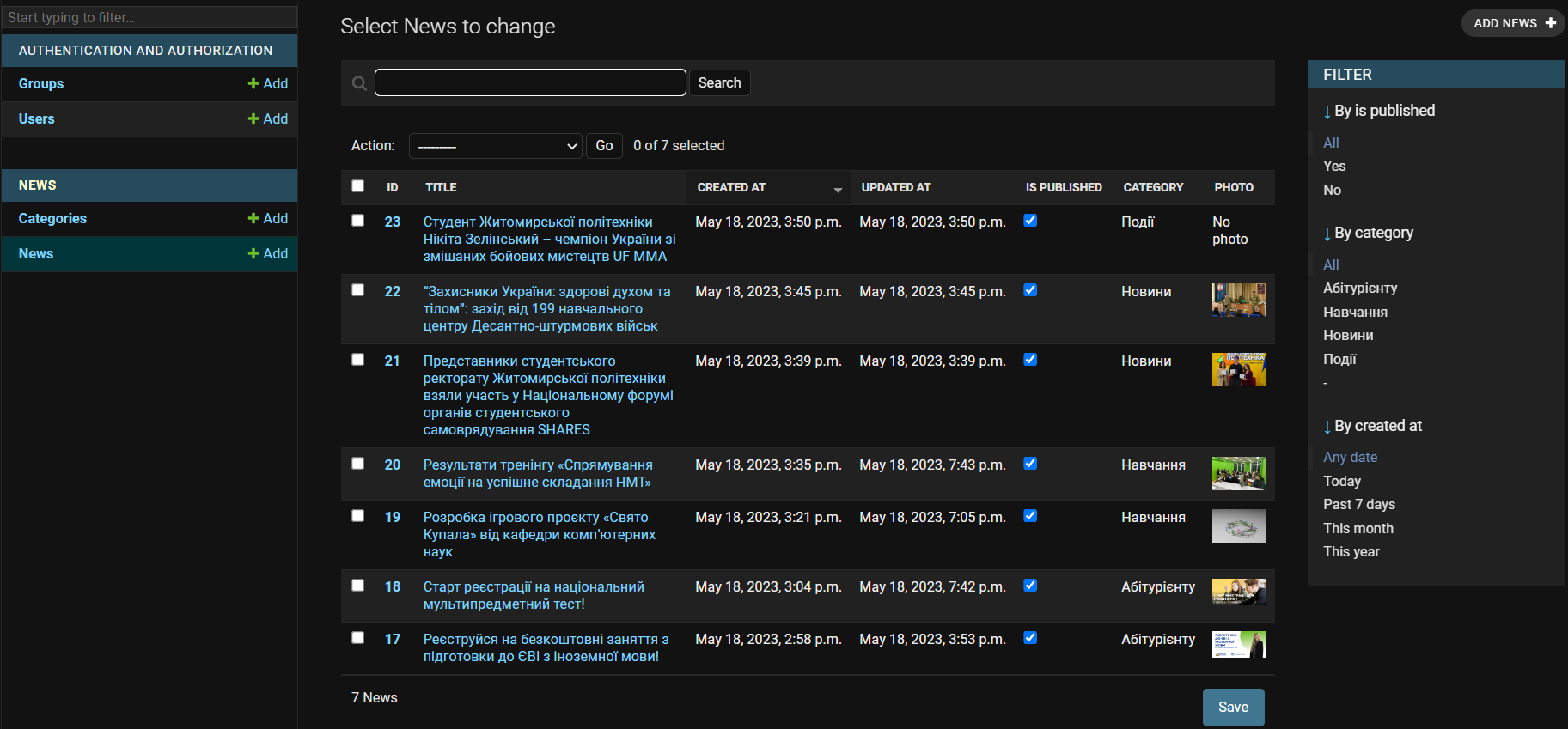
****

Рис. 4.4. Сторінка News, адмін-панель

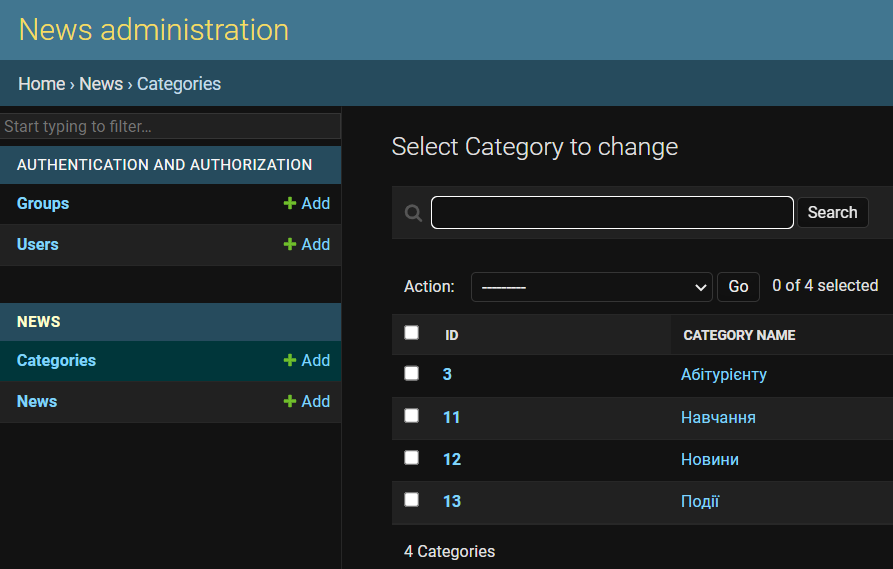
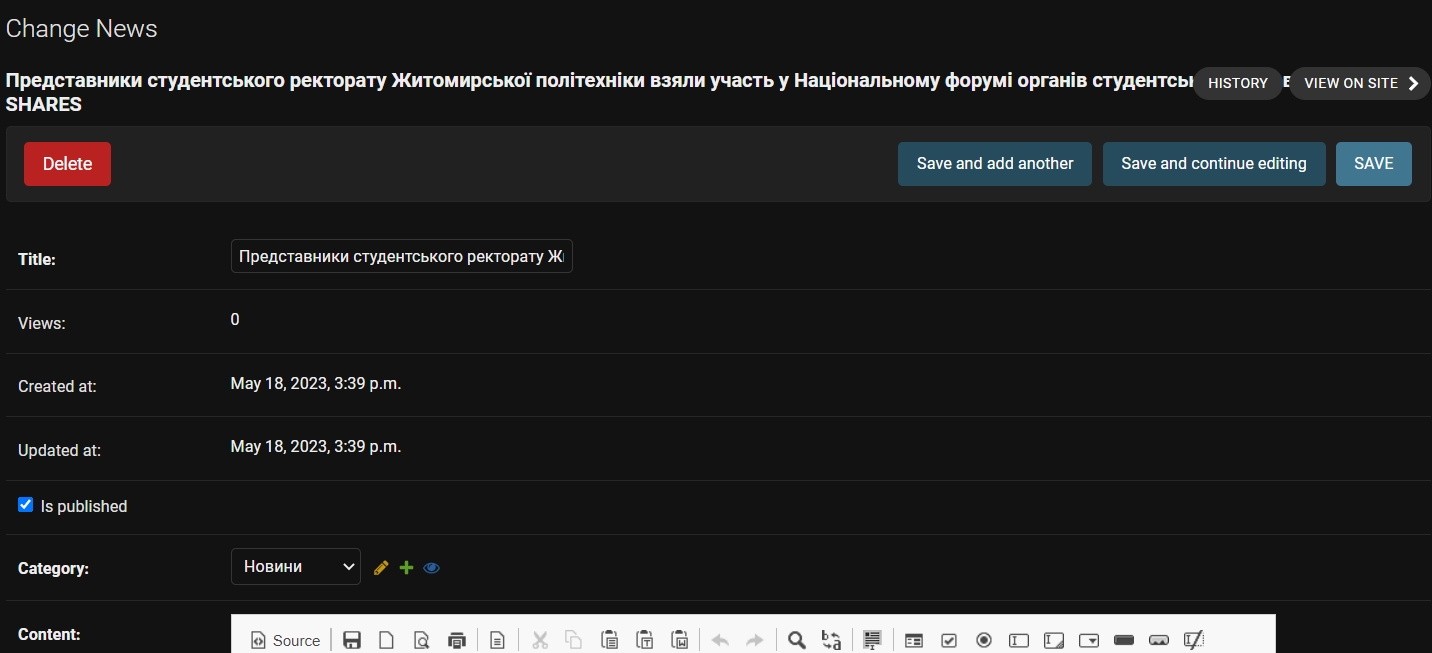


Рис. 4.5. Сторінка Саtegoty, адмін-панель



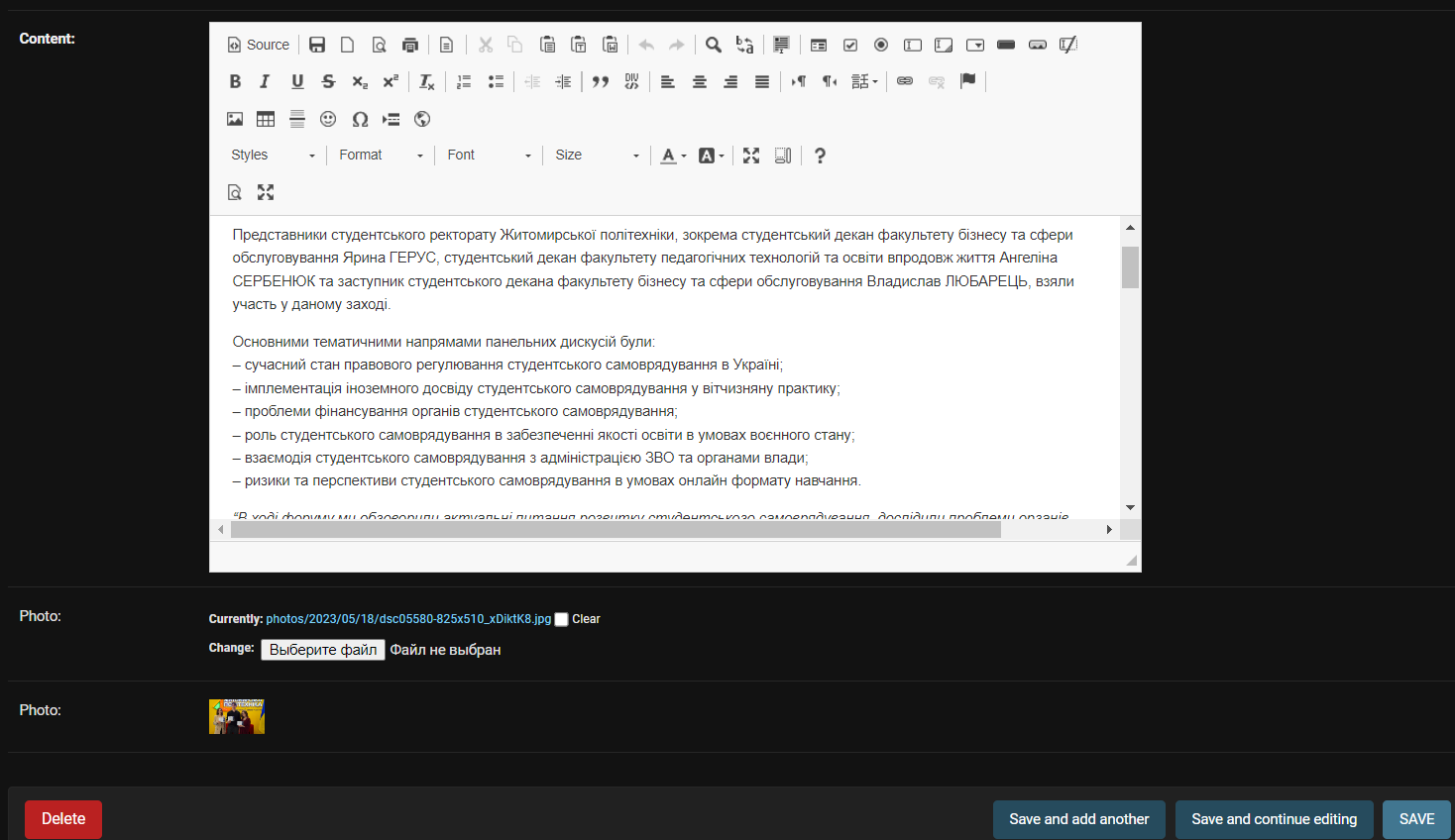


Рис. 4.6. Сторінка редагування поста, адмін-панель

# **Розділ 5. Тестування**

Тепер спробуємо зайти до сторінки логіну. За результатом, не вдалося через те, що користувач вже автентифікований. Те ж саме і на сторінці реєстрації.



Рис. 5.2. Перехід до сторінки логіну з вже автентифікованим користувачем



Рис. 5.3. Перехід до сторінки реєстрації з вже автентифікованим користувачем

При спробі перейти до неіснуючих маршрутів, ми отримаємо помилку 404.



Рис. 5.1. Перехід до неіснуючої сторінки, та результат дії

# **Висновки**

Під час виконання роботи у якості проекту з технологічної практики, було розроблено, здійснено тестування і розгортання веб-cайту новин.

У першому розділі було поставлено завдання, сформовано технічне завдання та спроектовано план реалізації проекту.

У другому розділі було спроектовано структуру бази даних системи, структуру та логіку роботи додатку було зображено за допомогою UML-діаграм відповідно до поставленого технічного завдання.

У третьому розділі було покроково описано щоденний план ходу реалізації додатку, наведено приклад програмного коду.

У четвертому розділі було описано інструкцію експлуатації для користувача, описано можливі дії користувачів з різними правами та їхні наслідки.

У п’ятому розділі було перевірено працездатність системи за допомогою ручного тестування.

В результаті роботи було отримано готовий, завершений додаток, що відповідає поставленому ТЗ.

## **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Форум з питаннями та відповідями [Електронний ресурс] – Режим доступу до ресурсу: <https://stackoverflow.com/>.
2. Документація Django– Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.2/
3. Документація Express – Режим доступу до ресурсу:  [https://expressjs.com/en/guide/routing.html](https://ru.reactjs.org/docs/getting-started.html)
4. Документація Python – Режим доступу до ресурсу:

https://docs.python.org/uk/3/

1. Документація SQLite Server – Режим доступу до ресурсу:   
   https://www.sqlite.org/docs.html