Deponentlanadigan materiallarning titul varag‘i

EHM uchun dastur (Ma’lumotlar bazasi) nomi:

“cyber-news.uz”

Huquq ega(lar)si :

1. Ro‘zmetov Temurbek Ixtiyor o‘g‘li

Muallif(lar):

1. Ro‘zmetov Temurbek Ixtiyor o‘g‘li

EHM uchun dasturni identifikatsiya qiluvchi materiallar dastlabki matni (Dastur kodi)

Saytning news_app.models qismi kodi:

```
from django.contrib.auth.models import User
from django.urls import reverse
from django.utils import timezone
from django.db import models


class PublishedManager(models.Manager):
    def get_queryset(self):
        return super().get_queryset().filter(News.Status.Published)


class Category(models.Model):
    name = models.CharField(max_length=150)

    def __str__(self):
        return self.name


class News(models.Model):
    class Status(models.TextChoices):
        Draft = 'DF', 'Draft'
        Published = 'PB', 'Published'

    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250)
    body = models.TextField()
    image = models.ImageField(upload_to='news/image')
    category = models.ForeignKey(Category,
                                 on_delete=models.CASCADE
                                 )
    publish_time = models.DateTimeField(default=timezone.now)
    created_time = models.DateTimeField(auto_now_add=True)
    updated_time = models.DateTimeField(auto_now=True)
    status = models.CharField(max_length=2,
                              choices=Status.choices,
                              default=Status.Draft
                              )
    objects = models.Manager()
    published = PublishedManager()

    class Meta:
        ordering = ["-publish_time"]
    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse("news_detail_page", args=[self.slug])


class Contact(models.Model):
    name = models.CharField(max_length=150)
    email = models.EmailField(max_length=150)
    message = models.TextField()

    def __str__(self):
        return self.email


class Comment(models.Model):
    news = models.ForeignKey(News,
                             on_delete=models.CASCADE,
                             related_name='comments'
                             )
    user = models.ForeignKey(User,
                             on_delete=models.CASCADE,
                             related_name='comments'
                             )
    body = models.TextField()
    created_time = models.DateTimeField(auto_now_add=True)
    active = models.BooleanField(default=True)

    class Meta:
        ordering = ['created_time']

    def __str__(self):
        return f"Comment - {self.body} by {self.user} "
```

Saytning news_app.views qismi kodi:

```
from django.contrib.auth.decorators import login_required, user_passes_test
from django.contrib.auth.models import User
from django.db.models import Q
from django.http import HttpResponse
from django.shortcuts import render, get_object_or_404
from django.urls import reverse_lazy
from django.views.generic import TemplateView, ListView, UpdateView, DeleteView,
CreateView
```

```python
from django.contrib.auth.mixins import LoginRequiredMixin
from hitcount.utils import get_hitcount_model

from .models import News, Category
from .forms import ContactForm, CommentForm
from news_project.custom_permissions import OnlyLoggedSuperUser


def news_list(request):
    news_list = News.objects.filter(status=News.Status.Published)
    # news_list = News.objects.all()
    context = {
        "news_list": news_list
    }

    return render(request, "news/news_list.html", context)


from hitcount.views import HitCountDetailView, HitCountMixin


def news_detail(request, news):
    news = get_object_or_404(News, slug=news, status=News.Status.Published)
    context = {}
    # hitcount logic
    hit_count = get_hitcount_model().objects.get_for_object(news)
    hits = hit_count.hits
    hitcontext = context['hitcount'] = {'pk': hit_count.pk}
    hit_count_response = HitCountMixin.hit_count(request, hit_count)
    if hit_count_response.hit_counted:
        hits = hits + 1
        hitcontext['hit_counted'] = hit_count_response.hit_counted
        hitcontext['hit_message'] = hit_count_response.hit_message
        hitcontext['total_hits'] = hits

    comments = news.comments.filter(active = True)
    comment_count = comments.count()

    comments = news.comments.filter(active=True)
    new_comment = None
    if request.method == 'POST':
        comment_form = CommentForm(data=request.POST)
        if comment_form.is_valid():
            new_comment = comment_form.save(commit=False)
            new_comment.news = news
            new_comment.user = request.user
            new_comment.save()
            comment_form = CommentForm()
    else:
        comment_form = CommentForm()

    context = {
        "news": news,
        "comments": comments,
        "new_comment": new_comment,
        "comment_form": comment_form,
        "comment_count": comment_count
    }

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


def homePageView(request):
    categories = Category.objects.all()
    news_list = News.objects.filter(status=News.Status.Published).order_by('-
publish_time')[:5]
    cybersport_news_one = News.objects.filter(status=News.Status.Published,
category__name='KiberSport').order_by(
        '-publish_time')[:1]
    cybersport_news = News.objects.filter(status=News.Status.Published,
category__name='KiberSport').order_by(
        '-publish_time')[1:6]
    context = {
        'news_list': news_list,
        'categories': categories,
        'cybersport_news_one': cybersport_news_one,
        'cybersport_news': cybersport_news
    }

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


class HomePageView(ListView):
    model = News
    template_name = 'news/home.html'
    context_object_name = 'name'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['categories'] = Category.objects.all()
        context['news_list'] =
News.objects.filter(status=News.Status.Published).order_by('-publish_time')[:4]
        # context['cybersport_news_one'] =
News.objects.filter(status=News.Status.Published,
        #
 category__name='KiberSport').order_by('-publish_time')[:1]
        context['kibersport_xabarlar'] =
News.objects.filter(status=News.Status.Published,

 category__name='KiberSport').order_by('-publish_time')[:5]
        context['ctf'] = News.objects.filter(status=News.Status.Published,
                                             category__name='CTF').order_by('-
publish_time')[:5]
        context['kiberjinoyatlar'] =
News.objects.filter(status=News.Status.Published,

 category__name='KiberJinoyatchilik').order_by('-publish_time')[
                                     :5]
        context['texnologiyalar'] =
News.objects.filter(status=News.Status.Published,

category__name='Texnologiyalar').order_by('-publish_time')[:5]
        return context

class ContactPageView(TemplateView):
    template_name = 'news/contact.html'

    def get(self, request, *args, **kwargs):
        form = ContactForm()
        context = {
            "form": form
        }
        return render(request, 'news/contact.html', context)

    def post(self, request, *args, **kwargs):
        form = ContactForm(request.POST)
        if request.method == 'POST' and form.is_valid():
            form.save()
            return HttpResponse("<h2> Biz bilan bog'langaningiz uchun
tashakkur!</h2>")
        context = {
            "form": form
        }
        return render(request, 'news/contact.html', context)


def errorPageView(request):
    context = {

    }
    return render(request, 'news/404.html', context)


class KiberSportView(ListView):
    model = News
    template_name = 'news/kibersport.html'
    context_object_name = 'kibersportnews'

    def get_queryset(self):
        news = News.objects.filter(status=News.Status.Published,
category__name='KiberSport')
        return news


class CtfViews(ListView):
    model = News
    template_name = 'news/ctf.html'
    context_object_name = 'ctfnews'

    def get_queryset(self):
        news = News.objects.filter(status=News.Status.Published,
category__name='CTF')
        return news


class TexnologiyalarViews(ListView):
    model = News
    template_name = 'news/texnologiyalar.html'
    context_object_name = 'texnologiyalarnews'

    def get_queryset(self):
        news = News.objects.filter(status=News.Status.Published,
category__name='Texnologiyalar')
        return news


class KiberjinoyatlarViews(ListView):
    model = News
    template_name = 'news/kiberjinoyatlar.html'
    context_object_name = 'kiberjinoyatlarnews'

    def get_queryset(self):
        news = News.objects.filter(status=News.Status.Published,
category__name='KiberJinoyatchilik')
        return news

class NewsUpdateView(OnlyLoggedSuperUser, UpdateView):
```

```python
    model = News
    fields = ('title', 'body', 'image', 'category', 'image')
    template_name = 'crud/news_edit.html'


class NewsDeleteView(OnlyLoggedSuperUser, DeleteView):
    model = News
    template_name = 'crud/news_delete.html'
    success_url = reverse_lazy('home_page')


class NewsCreateView(OnlyLoggedSuperUser, CreateView):
    model = News
    template_name = 'crud/news_create.html'
    fields = ('title', 'title_uz', 'title_en', 'title_ru', 'slug',
              'body', 'body_uz', 'body_en', 'body_ru', 'image',
              'category', 'status')


@login_required
@user_passes_test(lambda u: u.is_superuser)
def admin_page_view(request):
    admin_users = User.objects.filter(is_superuser=True)
    context = {
        'admin_users': admin_users
    }
    return render(request, 'pages/admin_page.html', context)


class SearchResultsList(ListView):
    model = News
    template_name = 'news/search_result.html'
    context_object_name = 'barcha_yangiliklar'

    def get_queryset(self):
        query = self.request.GET.get('q')
        return News.objects.filter(
            Q(title__icontains=query) | Q(body__icontains=query)
        )
```

Saytning accounts.models qismi kodi:

```python
from django.contrib.auth.models import User
from django.db import models
```

```
class Profile(models.Model):
    user = models.OneToOneField(User,
                                on_delete=models.CASCADE
                                )
    photo = models.ImageField(upload_to='users/', blank=True, null=True)
    date_of_birth = models.DateField(blank=True, null=True)

    def __str__(self):
        return f"{self.user.username} profili"
```

Saytning accounts.views qismi kodi:

```
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.shortcuts import render
from django.http import HttpResponse
from django.contrib.auth import authenticate, login
from django.contrib.auth.forms import UserCreationForm
from django.template import RequestContext
from django.views import View
from django.shortcuts import redirect
from .forms import LoginForm, UserRegistrationForm, UserEditForm, ProfileEditForm
from django.views.generic import CreateView
from django.urls import reverse_lazy
from .models import Profile


def user_login(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            data = form.cleaned_data
            user = authenticate(request,
                                username=data['username'],
                                password=data['password']
                                )

            if user is not None:
                if user.is_active:
                    login(request, user)
                    return HttpResponse('Muvaffaqiyatli login amalga oshirildi!')
                else:
                    return HttpResponse('Sizning profilingiz faol holatda emas!')
            else:
                return HttpResponse('Login va parolda hatolik bor!')
```

```python
    else:
        form = LoginForm()
        context = {
            'form': form
        }
    return render(request, 'registration/login.html', {'form': form})


@login_required
def dashboard_views(request):
    user = request.user
    profil_info = Profile.objects.get(user=user)
    context = {
        'user': user,
        'profil_info': profil_info
    }
    return render(request, 'pages/user-profile.html', context)


def user_register(request):
    if request.method == 'POST':
        user_form = UserRegistrationForm(request.POST)
        if user_form.is_valid():
            new_user = user_form.save(commit=False)
            new_user.set_password(
                user_form.cleaned_data['password']
            )
            new_user.save()
            Profile.objects.create(user=new_user)
            context = {
                'new_user': new_user
            }
            return render(request, 'account/register_done.html', context)
    else:
        user_form = UserRegistrationForm()
        context = {
            'user_form': user_form
        }
        return render(request, 'account/register.html', context)


class SignUpView(CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy('login')
    template_name = 'account/register.html'
```

```
@login_required
def edit_user(request):
    if request.method == 'POST':
        user_form = UserEditForm(instance=request.user, data=request.POST)
        profile_form = ProfileEditForm(instance=request.user.profile,
                                       data=request.POST,
                                       files=request.FILES
                                       )
        if user_form.is_valid() and profile_form.is_valid():
            user_form.save()
            profile_form.save()
            return redirect('user_profile')
    else:
        user_form = UserEditForm(instance=request.user)
        profile_form = ProfileEditForm(instance=request.user.profile)

    return render(request, 'account/profile_edit.html', {"user_form": user_form,
"profile_form": profile_form})


class EditUserView(LoginRequiredMixin, View):

    def get(self, request):
        user_form = UserEditForm(instance=request.user)
        profile_form = ProfileEditForm(instance=request.user.profile)

        return render(request, 'account/profile_edit.html', {"user_form":
user_form, "profile_form": profile_form})

    def post(self, request):
        user_form = UserEditForm(instance=request.user, data=request.POST)
        profile_form = ProfileEditForm(instance=request.user.profile,
                                       data=request.POST,
                                       files=request.FILES
                                       )
        if user_form.is_valid() and profile_form.is_valid():
            user_form.save()
            profile_form.save()
            return redirect('user_profile')
```

Saytning asosiy settings.py qismi:

```
from pathlib import Path
from decouple import config
BASE_DIR = Path(__file__).resolve().parent.parent


SECRET_KEY = config('SECRET_KEY')

DEBUG = config('DEBUG', default=False, cast=bool)
ALLOWED_HOSTS = ["127.0.0.1", "cyber-news.uz", "www.cyber-news.uz"]

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'news_app',
    'accounts',
    'hitcount',
    'modeltranslation',
    'whitenoise.runserver_nostatic',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    "django.middleware.locale.LocaleMiddleware",
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'news_project.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [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',
                'news_app.context_processor.latest_news',
            ],
        },
    },
]

WSGI_APPLICATION = 'news_project.wsgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

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',
    },
]

LANGUAGE_CODE = 'uz'

TIME_ZONE = 'Asia/Tashkent'

USE_I18N = True

USE_TZ = True

from django.utils.translation import gettext_lazy as _

LANGUAGES = [
    ('uz', _("Uzbek")),
    ('en', _("English")),
    ('ru', _("Russian"))
]

MODELTRANSLATION_DEFAULT_LANGUAGE = 'en'
LOCALE_PATH = BASE_DIR, 'locale'


STATIC_URL = 'static/'

STATIC_ROOT = '/home/cybernew/cyber-news.uz/django/staticfiles'
STATICFILES_DIRS = ('/home/cybernew/cyber-news.uz/django/static', )


STATICFILES_FINDERS = [
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder'
]

MEDIA_URL = 'media/'

MEDIA_ROOT = '/home/cybernew/cyber-news.uz/django/media'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

LOGIN_REDIRECT_URL = 'home_page'

EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'

LOGIN_URL = 'login'

STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
```