**ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ**

**СӘТБАЕВ УНИВЕРСИТЕТІ**

|  |
| --- |
|  |

Автоматтандыру және ақпараттық технологиялар институты

Программалық инженерия кафедрасы

**Отчет**

Тақырыбы: Сайт (Өнер галериясы)

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Жұмыс сапасы** | **Бағалау диапазоны** | **Алған бағасы%/балл** |
| 1 | Орындалмаған | 0% |  |
| 2 | Орындалған | 0-50% |  |
| 3 | Материалды өздігінен ұйымдастыру | 0-10% |  |
| 4 | Қажетті көлемді және белгіленген мерзімде аяқтау | 0-5% |  |
| 5 | Қосымша ғылыми әдебиеттерді пайдалану | 0-5% |  |
| 6 | Орындалған тапсырманың бірегейлігі | 0-10% |  |
| 7 | Жұмысты қорғау | 0-20% |  |
|  | Барлығы | 0-100% | 100 |

Білім алушы аты-жөні

Булычов Руслан, Саттар Бекнұр

Білім беру бағдарламасы шифры мен атауы

6B06102 Computer Science

Оқытушы аты-жөні

Мукажанов Н. К.

Алматы 2023

**Мазмұны**

1. Кіріспе

2. Негізгі бөлім (есеп шығару)

* 2 студенттен тұратын шағын топтарға бірігу.
* Жоба мақсатын анықтау.
* Ақпарат жинау және пәндік саланы талдау.
* Веб сайт картасын жасау
* Сайтқа mockup әзірлеу
* Сайт функцияналы
* Ер диагрмма

3. Қорытынды

4. Пайдаланылған әдебиеттер тізімі.

**Кіріспе**

Тапсырмалар:

* 2 студенттен тұратын шағын топтарға бірігу.
* Жоба мақсатын анықтау.
* Ақпарат жинау және пәндік саланы талдау.
* Веб сайт картасын жасау
* Сайтқа mockup әзірлеу
* Веб сайт архитектурасын құру
* Сайттың барлық функционалын анықтау
* Деректер базасының реляциялық моделінің сұлбасын жасау және әр кесте бойынша қысқаша ақпарат беру.

**Негізгі бөлім (есеп шығару)**

**Тапсырма орындайтын:** Саттар Бекнұр, Булычов Руслан.

**Жоба тақырыбы:** Өнер галереясы үшін веб-сайтты әзірлеу.

**Жоба мақсаты:** Қолданушы немесе өнерге қызығушылығы бар адамдар сайтқа кіру арқылы белгілі адамдардың өнер туындыларын қарай алуға мүмкіндіктері болады. Жәнеде кез-келген адам сайтқа тіркелу арқылы өз туындыларын сайтқа сала алатын мүмкіншіліктері болады. Егер сайтта өз туындысын сатқысы келген жағдайда сатуға қою мүмкіншілігі болады.

*Бұл мақсаттар тапсырма орындау барысында өзгерістерге ұшырауы мүмкін!*

**Ақпарат жинау және пәндік саланы талдау.**

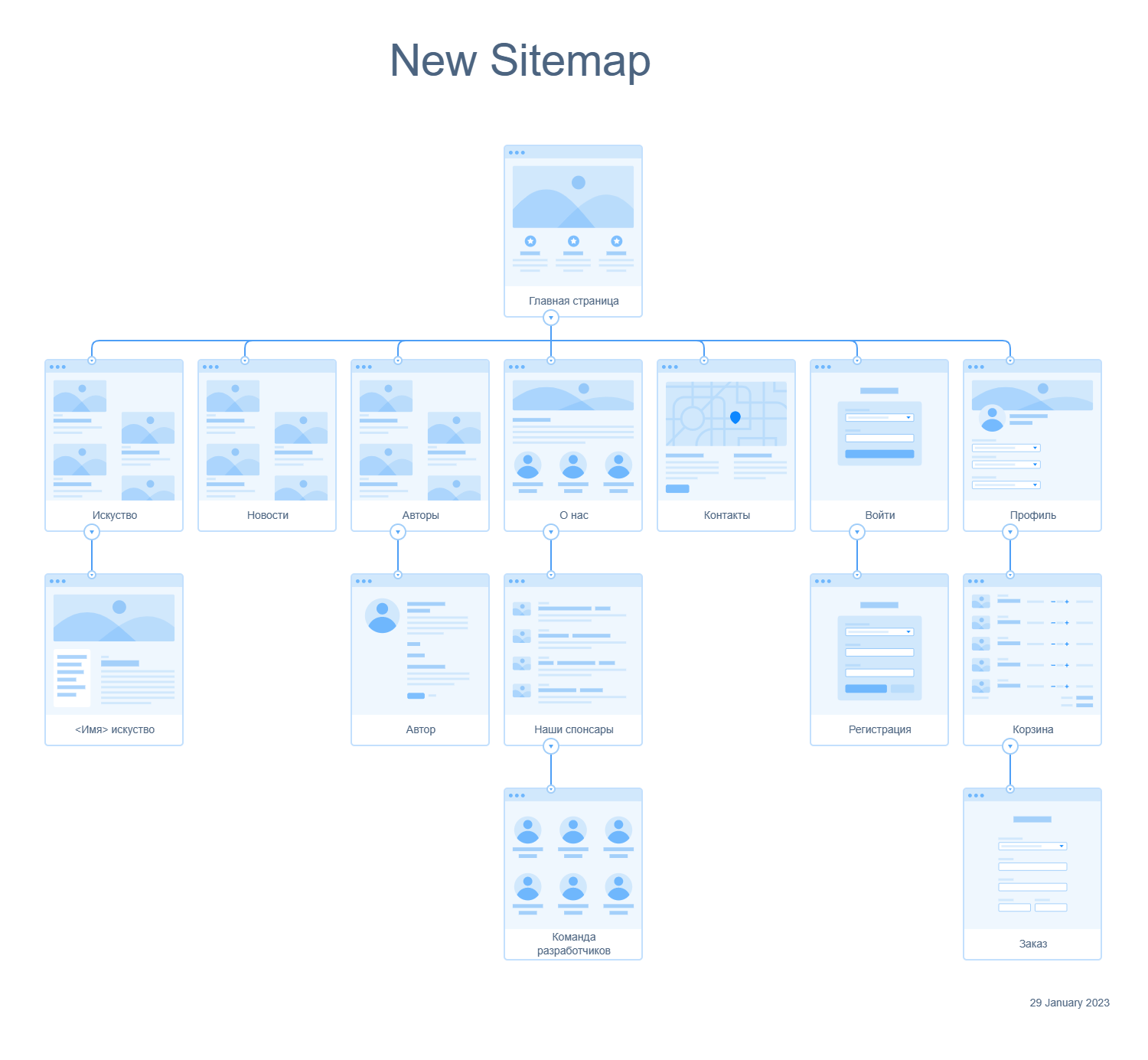
<https://focusgallery.kz/>

<https://www.gmirk.kz/kk/>

<http://artumar.kz/index.php>

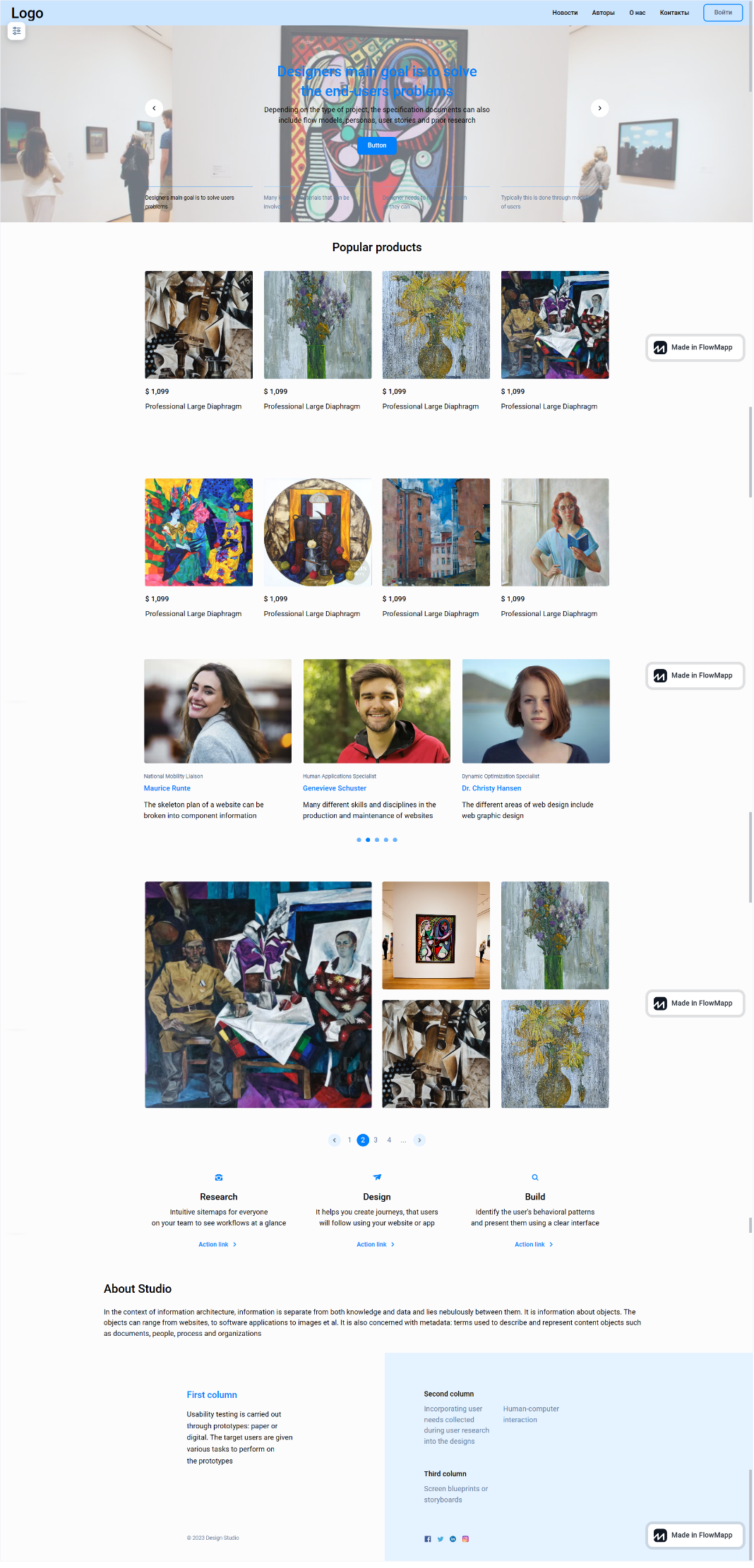
Біздің проектке жоғарыдағы сайттарды үлгі ретінде алып отырық. Бұл сайттарда кез келген өнер туындысы туралы ақпараттар және сол туындының авторлары туралы да жазылады. Кей туындыларды сатуға болатын мүмкіндік берсе болатын қылып қоятын мүмкіндік болады. Және әр туындыға өз ойларын(комментария) жаза алатын болады.

**Веб сайт картасын жасау**

****

**Сайтқа mockup әзірлеу**

<https://app.flowmapp.com/shared/wireframe-preview/9685c1dc-ad7f-4ec4-8544-e8850fa364ef>

****

-Сайттың барлық функцияларын анықтау

*Сайт пайдаланушыларына:*

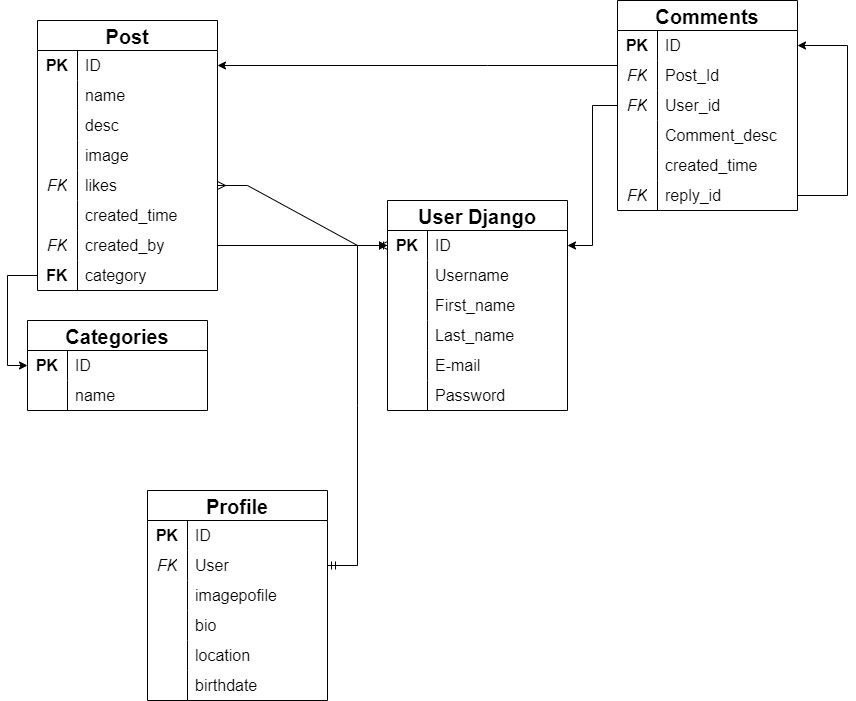
* *Меню*
* *Комментарий қалдыру*
* *Лүпіл қою*
* *Пост жүктеу*
* *Пост өзгерту*
* *Пост жою*
* *Ұқсас посттарды ұсыну*
* *Санаттағы посттар санын көрсету*
* *Сурет жүктеу*
* *Сайтқа тіркелу*

*Сайт администраторына:*

* *Пайдаланушының қызметінің бәрі*
* *Пайдаланушыларды басқару*

-Деректер базасының реляциялық моделінің сұлбасын жасау және әр кесте бойынша қысқаша ақпарат беру.

Ер диаграмма:



Кестелер:

Post

|  |  |  |  |
| --- | --- | --- | --- |
| PK | ID | Integer | Продукт id |
|  | name | Varchar | Продукт аты |
|  | Desc | Varchar | Описания |
|  | Image | Varchar | Продукт суреті |
| FK | likes | Integer | Лүпіл саны |
|  | Create\_time | Date | Енгізілген уақыты |
| FK | Create\_by | Interger | Енгізген қолданушы |
| FK | category | Integer | Санаты |

Categories

|  |  |  |  |
| --- | --- | --- | --- |
| PK | ID | Integer | Санат id |
|  | name | Varchar | Санат аттары |

User Django

|  |  |  |  |
| --- | --- | --- | --- |
| PK | ID | Integer | Қолданушы id |
|  | Username | Varchar | Қолданушы никі |
|  | First\_name | Varchar | Фамилия |
|  | Last\_name | Varchar | Имя |
|  | E-mail | Varchar | Ғаламтор поштасы |
|  | Password | Varchar | Құпия сөзі |

Profile

|  |  |  |  |
| --- | --- | --- | --- |
| PK | ID | Integer | Профиль id |
| FK | User | Varchar | Профиль аты |
|  | Imageprofile | Varchar | Профиль суреті |
|  | Bio | Varchar | Жынысы |
|  | Location | Varchar | Мекен жайы |
|  | birthdate | Date | Туылған күні |

Comments

|  |  |  |  |
| --- | --- | --- | --- |
| PK | ID | Integer | Комментарии Id |
| FK | Post\_id | Integer | Продукт id |
| FK | User\_id | Integer | Колданушы id |
|  | Comment\_desc | Varchar | Комментарии |
|  | Create\_time | Date | Құрылған уақыт |
|  | Reply\_id | Integer | Жауап id |

**Негізгі кодтар мен скриндар:**

**Models.py**

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

class Category(models.Model):  
 name = models.CharField(max\_length=255)  
  
 class Meta:  
 ordering = ("name",)  
 verbose\_name\_plural = 'Категории'  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class Item(models.Model):  
 category = models.ForeignKey(Category, related\_name='items', on\_delete=models.CASCADE)  
 name = models.CharField(max\_length=255)  
 description = models.TextField(blank=True, null=True)  
 image = models.ImageField(upload\_to='item\_images', blank=True, null=True)  
 likes = models.ManyToManyField(User, related\_name='likes', blank=True)  
 created\_by = models.ForeignKey(User, related\_name='items', on\_delete=models.CASCADE)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
  
 class Meta:  
 ordering = ("name",)  
 verbose\_name\_plural = 'Посты'  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 def total\_likes(self):  
 return self.likes.count()  
  
  
class Comment(models.Model):  
 item = models.ForeignKey(Item, on\_delete=models.PROTECT)  
 user = models.ForeignKey(User, on\_delete=models.PROTECT)  
 reply = models.ForeignKey('Comment', on\_delete=models.PROTECT, null=True, related\_name='replies')  
 content = models.TextField()  
 comment\_time = models.DateTimeField(auto\_now\_add=True)  
  
 class Meta:  
 verbose\_name\_plural = 'Комментарии'  
  
 def \_\_str\_\_(self):  
 return '{}-{}'.format(self.item.name, str(self.user.username))

class Profile(models.Model):  
 user = models.OneToOneField(User, on\_delete=models.CASCADE)  
 imageUser = models.ImageField(upload\_to='main\_images', blank=True, null=True)  
 bio = models.TextField(max\_length=500, blank=True, null=True)  
 location = models.CharField(max\_length=30, blank=True, null=True)  
 birth\_date = models.DateField(null=True, blank=True)  
  
 class Meta:  
 verbose\_name\_plural = 'Профиль'

**Forms.py**

from django import forms  
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm  
from django.contrib.auth.models import User  
from .models import \*  
  
  
class LoginForm(AuthenticationForm):  
 username = forms.CharField(widget=forms.TextInput(attrs={  
 'placeholder': 'Your username',  
 'class': 'w-full py-4 px-6 rounded-xl'  
 }))  
 password = forms.CharField(widget=forms.PasswordInput(attrs={  
 'placeholder': 'Your password',  
 'class': 'w-full py-4 px-6 rounded-xl'  
 }))  
  
  
class SignupForm(UserCreationForm):  
 class Meta:  
 model = User  
 fields = ('username', 'email', 'password1', 'password2')  
  
 username = forms.CharField(widget=forms.TextInput(attrs={  
 'placeholder': 'Your username',  
 'class': 'w-full py-4 px-6 rounded-xl'  
 }))  
 email = forms.CharField(widget=forms.EmailInput(attrs={  
 'placeholder': 'Your email address',  
 'class': 'w-full py-4 px-6 rounded-xl'  
 }))  
 password1 = forms.CharField(widget=forms.PasswordInput(attrs={  
 'placeholder': 'Your password',  
 'class': 'w-full py-4 px-6 rounded-xl'  
 }))  
 password2 = forms.CharField(widget=forms.PasswordInput(attrs={  
 'placeholder': 'Repeat password',  
 'class': 'w-full py-4 px-6 rounded-xl'  
 }))  
  
  
class UserForm(forms.ModelForm):  
 class Meta:  
 model = User  
 fields = ('username', 'first\_name', 'last\_name', 'email')  
  
  
class ProfileForm(forms.ModelForm):  
 class Meta:  
 model = Profile  
 fields = ('imageUser', 'bio', 'location', 'birth\_date')

from django import forms  
from .models import Item, Comment  
  
INPUT\_CLASSES = 'w-full py-4 px-6 rounded-xl border'  
  
  
class NewItemForm(forms.ModelForm):  
 class Meta:  
 model = Item  
 fields = ('category', 'name', 'description', 'image')  
 widgets = {  
 'category': forms.Select(attrs={  
 'class': INPUT\_CLASSES  
 }),  
 'name': forms.TextInput(attrs={  
 'class': INPUT\_CLASSES  
 }),  
 'description': forms.Textarea(attrs={  
 'class': INPUT\_CLASSES  
 }),  
 'image': forms.FileInput(attrs={  
 'class': INPUT\_CLASSES  
 }),  
 }  
  
  
class EditItemForm(forms.ModelForm):  
 class Meta:  
 model = Item  
 fields = ('name', 'description', 'image',)  
 widgets = {  
 'name': forms.TextInput(attrs={  
 'class': INPUT\_CLASSES  
 }),  
 'description': forms.Textarea(attrs={  
 'class': INPUT\_CLASSES  
 }),  
 'image': forms.FileInput(attrs={  
 'class': INPUT\_CLASSES  
 }),  
 }  
  
  
class CommentForm(forms.ModelForm):  
 class Meta:  
 model = Comment  
 fields = ('content',)  
 widgets = {  
 'content': forms.TextInput(attrs={  
 'class': INPUT\_CLASSES  
 })  
 }

**Urls.py**

from django.conf import settings  
from django.conf.urls.static import static  
  
from django.contrib import admin  
from django.urls import path, include  
  
  
urlpatterns = [  
 path('', include('main.urls')),  
 path('items/', include('item.urls')),  
 path('dashboard/', include('dashboard.urls')),  
 path('admin/', admin.site.urls),  
] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

from django.contrib.auth import views as auth\_views  
from django.urls import path  
from . import views  
from .forms import LoginForm  
  
app\_name = 'main'  
  
urlpatterns = [  
 path('', views.index, name='index'),  
 path('contact/', views.contact, name='contact'),  
 path('signup/', views.signup, name='signup'),  
 path('avtors/', views.avtors, name='avtors'),  
 path('login/', auth\_views.LoginView.as\_view(template\_name='main/login.html', authentication\_form=LoginForm), name='login'),  
 path('logout/', auth\_views.LogoutView.as\_view(template\_name='main/index.html'), name='logout')  
]

from django.urls import path  
from . import views  
  
app\_name = 'item'  
  
urlpatterns = [  
 path('new/', views.new, name='new'),  
 path('<int:pk>/', views.detail, name='detail'),  
 path('<int:pk>/delete/', views.delete, name='delete'),  
 path('<int:pk>/edit/', views.edit, name='edit'),  
 path(r'like/$', views.like\_item, name='like\_item')  
]

from django.urls import path  
from . import views  
  
app\_name = 'dashboard'  
  
urlpatterns = [  
 path('', views.index, name='index')  
]

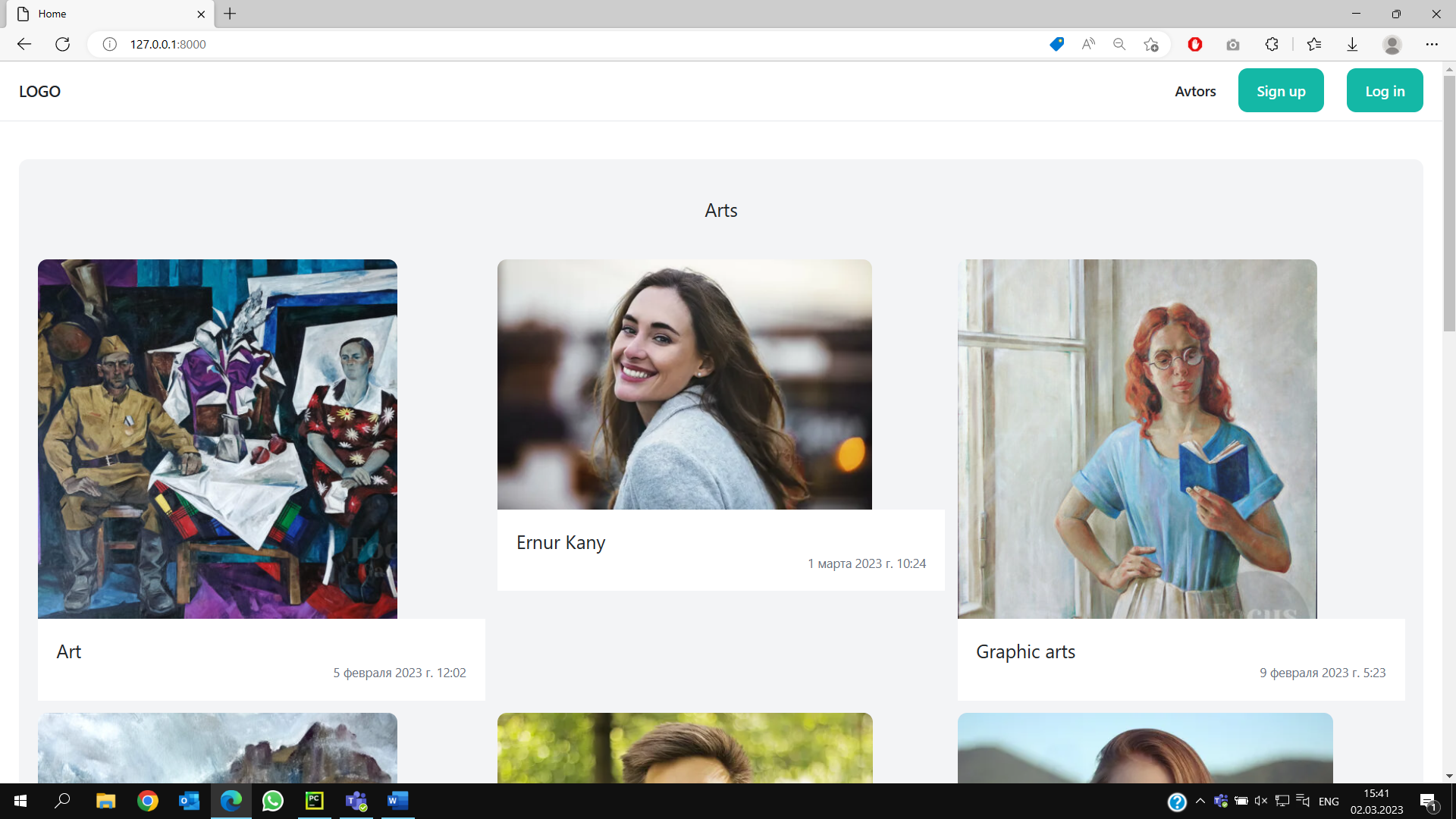
**Views.py**

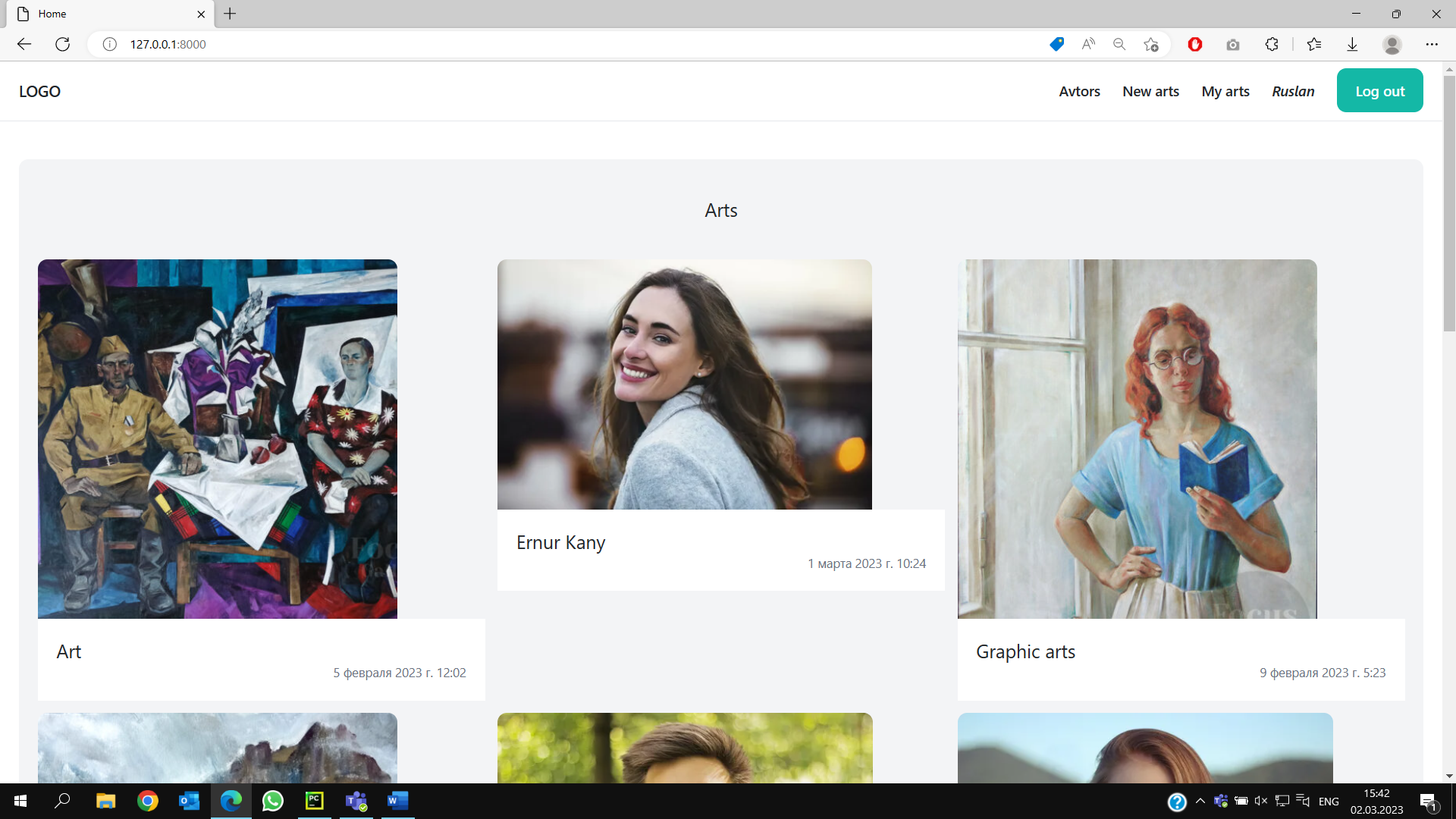
from django.contrib.auth.decorators import login\_required  
from django.contrib.auth.models import User  
from django.db import transaction  
from django.shortcuts import render, redirect  
from item.models import \*  
from .models import \*  
from .forms import SignupForm, UserForm, ProfileForm  
  
  
def index(request):  
 items = Item.objects.filter()[0:9]  
 categories = Category.objects.all()  
 context = {  
 'categories': categories,  
 'items': items,  
 }  
 return render(request, 'main/index.html', context)  
  
  
def contact(request):  
 return render(request, 'main/contact.html')  
  
  
def signup(request):  
 if request.method == 'POST':  
 form = SignupForm(request.POST)  
  
 if form.is\_valid():  
 form.save()  
 return redirect('/login/')  
 else:  
 form = SignupForm()  
 return render(request, 'main/signup.html', {'form': form})  
  
  
from django.http import HttpResponseRedirect  
from django.contrib.auth.decorators import login\_required  
from django.shortcuts import render, get\_object\_or\_404, redirect  
from django.urls import reverse  
  
from .forms import \*  
from .models import \*  
  
  
def detail(request, pk):  
 global comment\_form  
 item = get\_object\_or\_404(Item, pk=pk)  
 comments = Comment.objects.filter(item=item).order\_by('id')  
 related\_items = Item.objects.filter(category=item.category).exclude(pk=pk)[0:3]  
  
# Проверка лайка  
 is\_liked = False  
 if item.likes.filter(id=request.user.id).exists():  
 is\_liked = True  
  
 if request.method == 'POST':  
 comment\_form = CommentForm(request.POST or None)  
 if comment\_form.is\_valid():  
 content = request.POST.get('content')  
 comment = Comment.objects.create(item=item, user=request.user, content=content)  
 comment.save()  
 return redirect('item:detail', pk=item.id)  
 else:  
 comment\_form = CommentForm()  
  
 context = {  
 'item': item,  
 'related\_items': related\_items,  
 'is\_liked': is\_liked,  
 'total\_likes': item.total\_likes(),  
 'comments': comments,  
 'comment\_form': comment\_form,  
 }  
 return render(request, 'item/detail.html', context)  
  
  
@login\_required  
def new(request):  
 if request.method == 'POST':  
 form = NewItemForm(request.POST, request.FILES)  
  
 if form.is\_valid():  
 item = form.save(commit=False)  
 item.created\_by = request.user  
 item.save()  
  
 return redirect('item:detail', pk=item.id)  
  
 else:  
 form = NewItemForm()  
 return render(request, 'item/form.html', {'form': form, 'title': 'New item'})  
  
  
def edit(request, pk):  
 item = get\_object\_or\_404(Item, pk=pk, created\_by=request.user)  
 if request.method == 'POST':  
 form = EditItemForm(request.POST, request.FILES, instance=item)  
  
 if form.is\_valid():  
 form.save()  
  
 return redirect('item:detail', pk=item.id)  
  
 else:  
 form = EditItemForm(instance=item)  
 return render(request, 'item/form.html', {'form': form, 'title': 'Edit item'})  
  
  
def delete(request, pk):  
 item = get\_object\_or\_404(Item, pk=pk, created\_by=request.user)  
 item.delete()  
 return redirect('dashboard:index')  
  
  
def like\_item(request):  
 item = get\_object\_or\_404(Item, id=request.POST.get('item\_id'))  
 is\_liked = False  
 if item.likes.filter(id=request.user.id).exists():  
 item.likes.remove(request.user)  
 is\_liked = False  
 else:  
 item.likes.add(request.user)  
 is\_liked = True  
 return redirect('item:detail', pk=item.id)

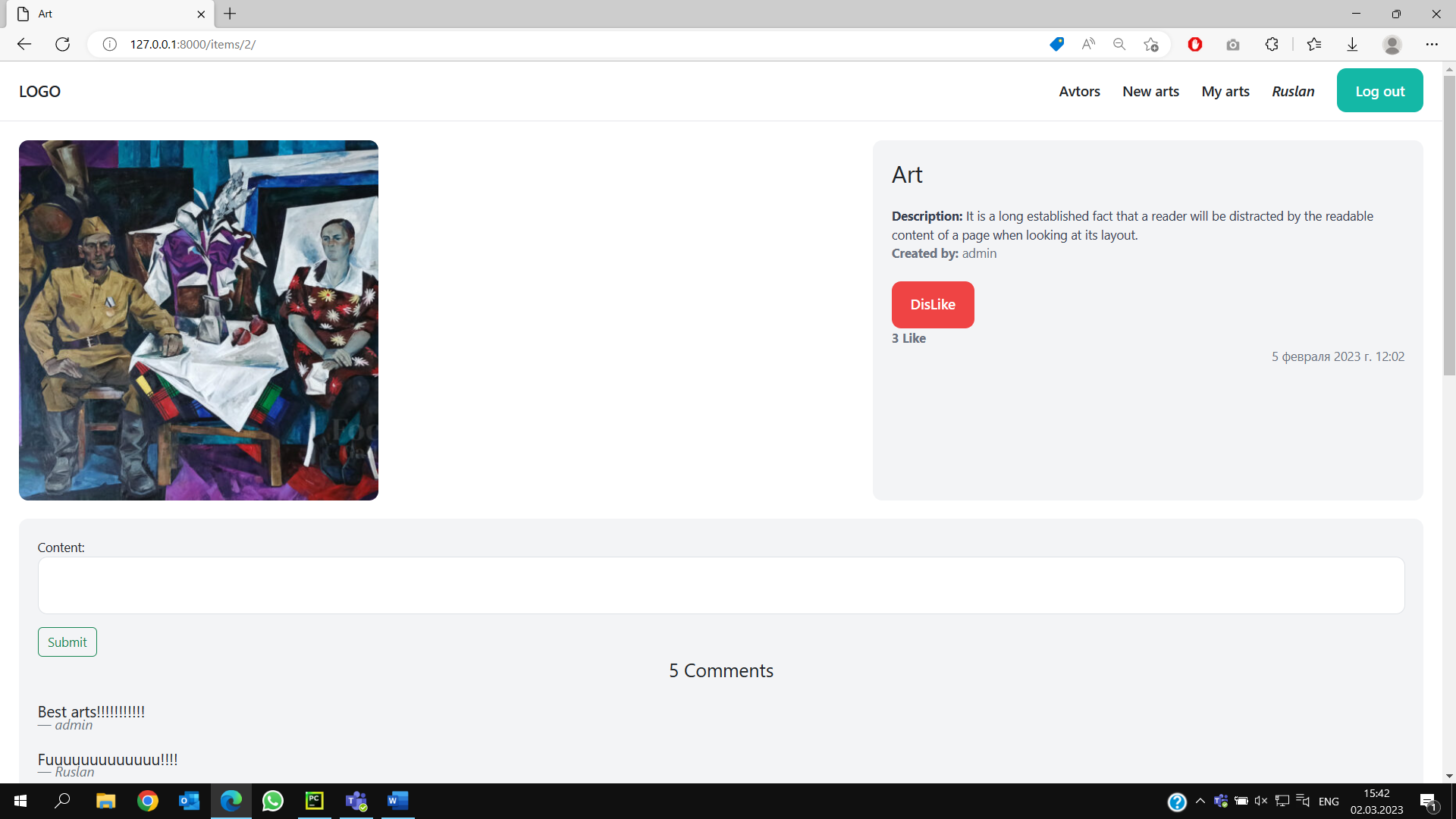
from django.contrib.auth.decorators import login\_required  
from django.shortcuts import render  
from item.models import Item  
  
  
@login\_required  
def index(request):  
 items = Item.objects.filter(created\_by=request.user)  
 return render(request, 'dashboard/index.html', {'items': items})

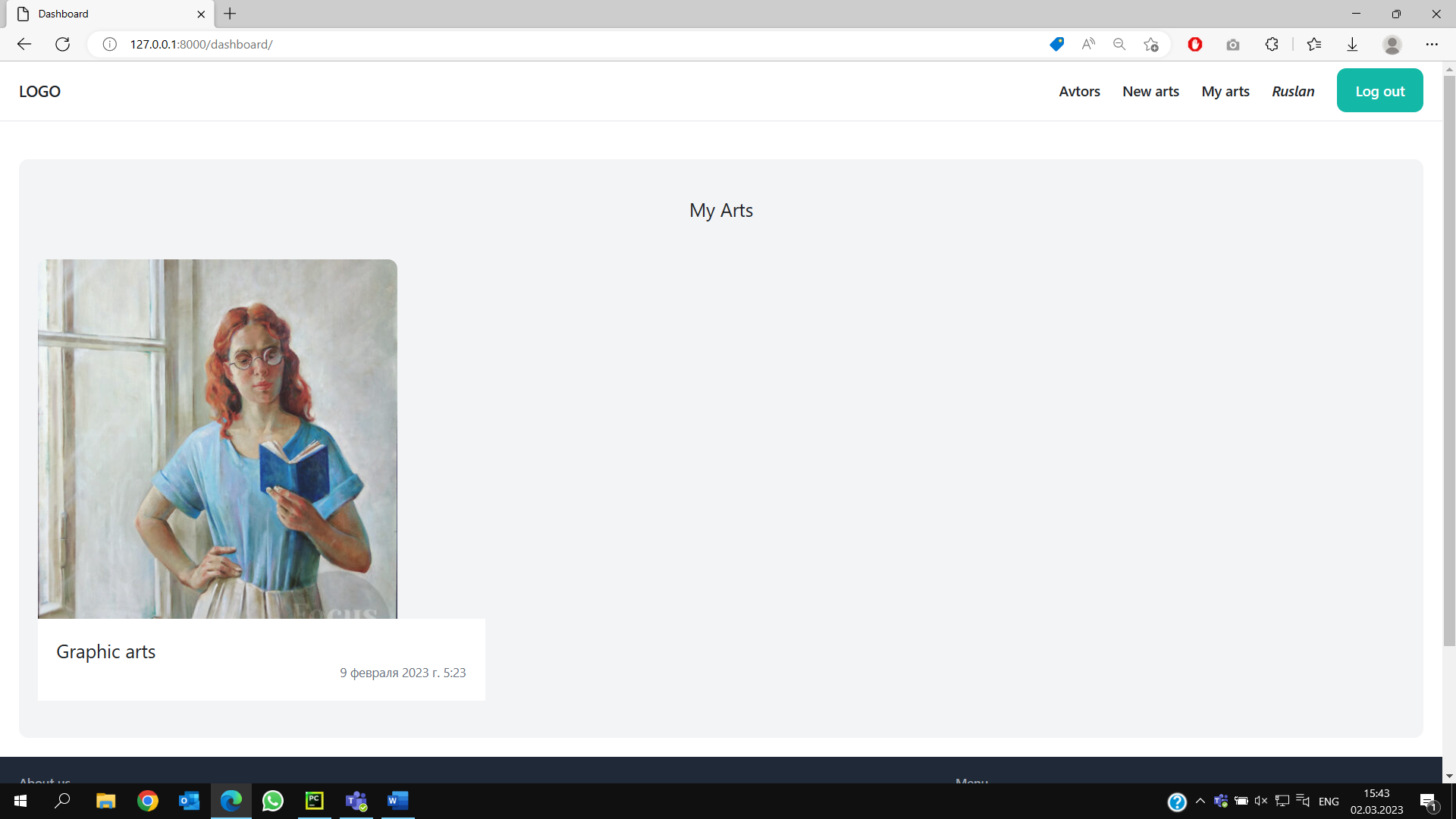
**Admin.py**

from django.contrib import admin  
from .models import \*  
  
  
class CategoryAdmin(admin.ModelAdmin):  
 list\_display = ('id', 'name')  
 list\_display\_links = ('id', 'name')  
 search\_fields = ('id', 'name')  
  
  
admin.site.register(Category, CategoryAdmin)  
  
  
class ItemAdmin(admin.ModelAdmin):  
 list\_display = ('id', 'name', 'created\_by', 'created\_at')  
 list\_display\_links = ('id', 'name')  
 search\_fields = ('id', 'name')  
  
  
admin.site.register(Item, ItemAdmin)  
  
  
class CommentAdmin(admin.ModelAdmin):  
 list\_display = ('id', 'content')  
 list\_display\_links = ('id', 'content')  
 search\_fields = ('id', 'content')  
  
  
admin.site.register(Comment, CommentAdmin)







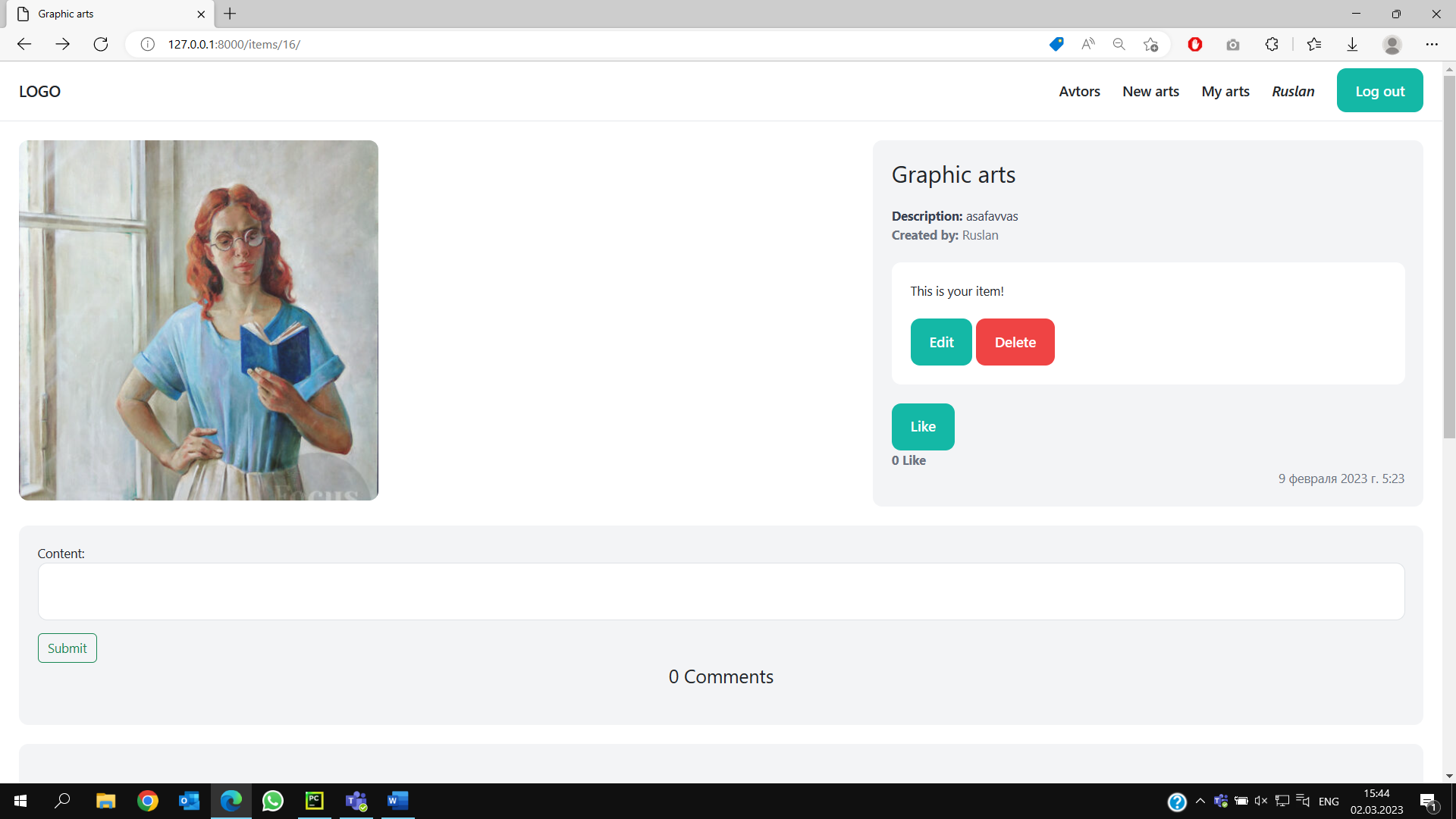


Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



AdminDjango

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание