

**WYDZIAŁ  
ELEKTROTECHNIKI  
I INFORMATYKI**  
POLITECHNIKI RZESZOWSKIEJ

**Mateusz Wilkos, Patryk Walat**

Aplikacje internetowe

## **Dokumentacja projektu**

Koordynator projektu:  
mgr inż. Piotr Hadaj

Rzeszów, 2023



# Spis treści

|   |           |
|---|-----------|
| <b>1. Cel i zakres pracy .....</b>                            | <b>5</b>  |
| <b>1.1. Cel i zastosowanie aplikacji.....</b>                 | <b>5</b>  |
| <b>1.2. Model aplikacji.....</b>                              | <b>5</b>  |
| <b>2. Realizacja projektu .....</b>                           | <b>6</b>  |
| <b>2.1. Wykorzystane technologie.....</b>                     | <b>6</b>  |
| <b>2.1.1. Framework Django .....</b>                          | <b>6</b>  |
| <b>2.1.2. Framework Tailwind CSS.....</b>                     | <b>6</b>  |
| <b>2.1.3. PostgreSQL.....</b>                                 | <b>6</b>  |
| <b>2.1.4. Render .....</b>                                    | <b>6</b>  |
| <b>2.1.5. Retool.....</b>                                     | <b>6</b>  |
| <b>2.1.6. Chart.js.....</b>                                   | <b>6</b>  |
| <b>2.1.7. Git .....</b>                                       | <b>6</b>  |
| <b>2.1.8. Biblioteki języka python .....</b>                  | <b>7</b>  |
| <b>2.2. Struktura aplikacji w Django .....</b>                | <b>7</b>  |
| <b>2.2.1. Projekt Django .....</b>                            | <b>7</b>  |
| <b>2.2.2. Aplikacja Django.....</b>                           | <b>8</b>  |
| <b>2.2.3. Model.....</b>                                      | <b>8</b>  |
| <b>2.2.4. Widok .....</b>                                     | <b>8</b>  |
| <b>2.2.5. Szablon .....</b>                                   | <b>8</b>  |
| <b>2.2.6. Pliki konfiguracyjne routingu w aplikacji .....</b> | <b>9</b>  |
| <b>2.2.7. Pliki statyczne .....</b>                           | <b>9</b>  |
| <b>2.3. Aplikacja „main”.....</b>                             | <b>9</b>  |
| <b>2.4. Aplikacja „aimodels” .....</b>                        | <b>19</b> |
| <b>2.5. Aplikacja „yt_sentiment” .....</b>                    | <b>22</b> |
| <b>2.4. Zabezpieczenie aplikacji.....</b>                     | <b>27</b> |
| <b>3. Podsumowanie .....</b>                                  | <b>36</b> |
| <b>4. Literatura.....</b>                                     | <b>37</b> |



## **1. Cel i zakres pracy**

Celem niniejszego projektu jest zastosowanie zdobytej wiedzy dotyczącej tworzenia aplikacji internetowych w celu stworzenia aplikacji analizującej nacechowanie komentarzy umieszczonych pod filmem na serwisie Youtube. Aplikacja będzie jedną z usług znajdujących się na stronie fikcyjnej firmy oferującej rozwiązania oparte o sztuczną inteligencję.

### **1.1. Cel i zastosowanie aplikacji**

Celem aplikacji jest analiza sentymentu komentarzy znajdujących się pod filmem z serwisu Youtube, podanym przez użytkownika. Komentarz zaklasyfikowany może zostać jako nacechowany negatywnie, pozytywnie lub neutralnie. Na podstawie analizy wykonanej przez model przetwarzania języka naturalnego tworzony jest raport zawierający możliwą do pobrania ocenę dla każdego z komentarzy, informację o proporcjach nacechowania oraz treści komentarzy skrajnie pozytywnych i negatywnych.

### **1.2. Model aplikacji**

#### **1.2.1. Opis koncepcji**

Dla klientów aplikacji, kluczowe jest zrozumienie reakcji na materiały publikowane na platformie YouTube, monitorowanie opinii na temat kanałów i filmów oraz śledzenie ogólnego odbioru treści. Równie istotne jest dostarczenie czytelnych raportów i wizualizacji, aby łatwo przyswoić rezultaty analizy nacechowania komentarzy. Zrozumienie potrzeb klientów jest fundamentem dla naszych usług, aby sprostać oczekiwaniom i dostarczyć narzędzie wspierające podejmowanie decyzji biznesowych, m.in na podstawie analizy sentymentu komentarzy na YouTube.

#### **1.2.2. Opis logiki**

Pełna funkcjonalność platformy jest dostępna wyłącznie dla zarejestrowanych użytkowników. Udostępniając model przetwarzania języka naturalnego analizujący komentarze, staramy się zachęcić klientów do eksploracji pełnej gamy naszych usług, obejmującej różnorodne modele sztucznej inteligencji.

## **2. Realizacja projektu**

### **2.1. Wykorzystane technologie**

W projekcie wykorzystane zostały następujące technologie umożliwiające szybkie i bezpieczne stworzenie aplikacji webowej.

#### **2.1.1. Framework Django**

**Django** - framework do budowy aplikacji internetowych w języku Python, oferujący pełen zestaw narzędzi do szybkiego tworzenia bezpiecznych, skalowalnych i łatwo utrzymywanych aplikacji webowych.

#### **2.1.2. Framework Tailwind CSS**

**Tailwind** – framework do tworzenia stylów w projektach internetowych, umożliwiający tworzenie interfejsów użytkownika poprzez definiowanie stylów za pomocą klas CSS bezpośrednio w kodzie HTML.

#### **2.1.3. PostgreSQL**

**PostgreSQL** - jeden z najpopularniejszych otwartych systemów zarządzania relacyjnymi bazami danych.

#### **2.1.4. Render**

**Render** - platforma chmurowa oferująca hosting dla aplikacji internetowych oraz baz danych, umożliwiająca bezpośrednie skoncentrowanie się na rozwoju aplikacji, a nie na zarządzaniu infrastrukturą.

#### **2.1.5. Retool**

**Retool** - platforma internetowa pozwalająca na szybkie budowanie paneli administracyjnych dla baz danych.

#### **2.1.6. Chart.js**

**Chart.js** - biblioteka JavaScript typu open source do wizualizacji danych.

#### **2.1.7. Git**

**Git** - system kontroli wersji, umożliwiający skuteczne zarządzanie kodem źródłowym projektu, śledzenie zmian, współpracę zespołową oraz przywracanie wcześniejszych wersji kodu.

#### **2.1.8. Biblioteki języka python**

**Environ** – biblioteka umożliwiająca bezpieczne zarządzanie zmiennymi środowiskowymi w Pythonie.

**Pandas** – biblioteka wykorzystywana do manipulacji i analizy danych w języku.

**Transformers** – biblioteka udostępniająca gotowe modele przetwarzania języka naturalnego.

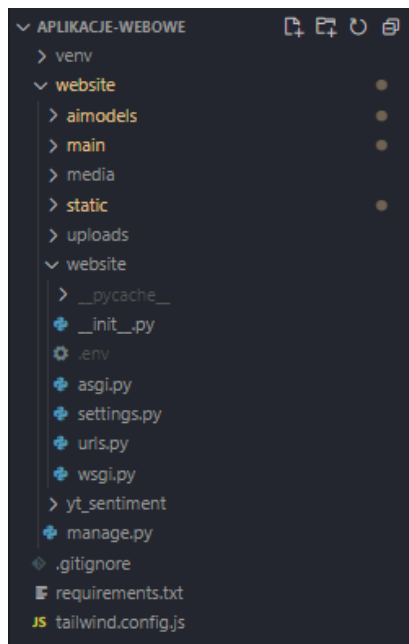
**Googleapiclient** – biblioteka umożliwiająca korzystanie z interfejsu API Google.

### **2.2. Struktura aplikacji w Django**

Struktura aplikacji webowej stworzonej w Django składa się z następujących komponentów.

#### **2.2.1. Projekt Django**

Projekt Django to najwyższy poziom hierarchii. Może zawierać wiele aplikacji Django. W projekcie znajduje się plik o nazwie settings.py, który zawiera ustawienia konfiguracyjne dla całego projektu.



Rysunek 1 Schemat główny projektu

### 2.2.2. Aplikacja Django

Aplikacja Django to samodzielna jednostka, która skupia się na jednym aspekcie funkcjonalności całego projektu. Wewnątrz aplikacji znajduje się wiele komponentów, takich jak modele, widoki, szablony, pliki statyczne, które są indywidualne dla każdej aplikacji. W naszym projekcie wyróżniamy trzy główne aplikacje: *main*, *aimodels*, *yt\_sentiment*.

### 2.2.3. Model

Modele w Django reprezentują strukturę bazy danych. Modele definiuje się w plikach *models.py* wewnątrz aplikacji, opisuje się nimi, jak mają być przechowywane dane.

### 2.2.4. Widok

Widoki odpowiadają za przetwarzanie żądań użytkownika i zwracanie odpowiedzi. Mogą być odpowiedzialne za przetwarzanie danych z modelu, renderowanie szablonów i obsługę logiki aplikacji. Widoki definiuje się w plikach *views.py* wewnątrz aplikacji.

### 2.2.5. Szablon

Szablony to pliki HTML, które w połączeniu z Django, pozwalają na dynamiczne generowanie zawartości strony. Przechowywane są w katalogu *templates* wewnątrz aplikacji.



Główne korzyści wynikające z korzystania z szablonów:

- Szablony są używane przez widoki do renderowania danych przed ich wysłaniem do klienta. Dane te są zazwyczaj przechowywane w kontekście i są dostępne w szablonie do wykorzystania.
- W szablonach można używać instrukcji warunkowych oraz pętli oraz innych konstrukcji sterujących znanych z języków programowania. Pozwala to na dynamiczne generowanie treści w zależności od warunków lub iteracyjne wyświetlanie komponentów.
- Django umożliwia korzystanie z mechanizmu dziedziczenia szablonów, co pozwala na zdefiniowanie ogólnego szablonu bazowego oraz specyficznych szablonów dziedziczących, które rozszerzają lub nadpisują fragmenty kodu z szablonu nadrzędnego.

#### **2.2.6. Pliki konfiguracyjne routingu w aplikacji**

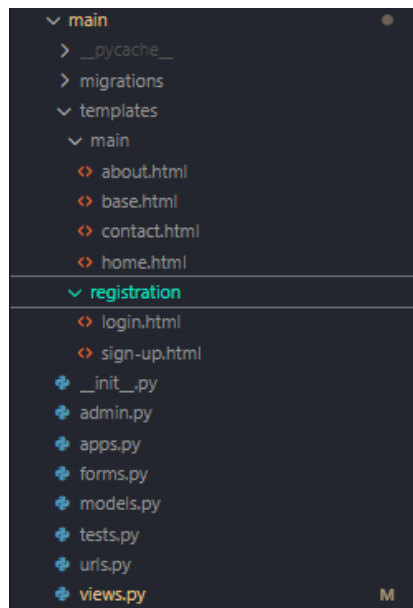
Są to pliki, w których określa się, jak Django ma mapować żądania HTTP na konkretne widoki. Konfigurację tras routingu definiuje się w plikach *urls.py* wewnątrz aplikacji.

#### **2.2.7. Pliki statyczne**

Pliki statyczne, takie jak arkusze stylów CSS, pliki JavaScript, obrazy, tła itp., są przechowywane w katalogu *static* wewnątrz aplikacji lub w głównym katalogu projektu.

### **2.3. Aplikacja „*main*”**

Aplikacja *main* obsługuje stronę domową, kontaktową, stronę, tzw. stronę *about*, oraz stronę logowania i rejestracji.



Rysunek 2 Katalog aplikacji „main”

Poniższy kod obsługuje trzy widoki dla różnych stron oraz formularz rejestracji użytkownika, który pozwala na utworzenie nowego konta i zalogowanie się użytkownika.

```

1  from django.shortcuts import render, redirect
2  from .forms import RegisterForm
3  from django.contrib.auth import login, logout, authenticate
4
5  *
6  def home(request):
7      return render(request, "main/home.html")
8
9
10 def contact(request):
11     return render(request, "main/contact.html")
12
13
14 def about(request):
15     return render(request, "main/about.html")
16
17     You, przedkioraj * fixed some issues with responsiveness
18 def sign_up(request):
19     if request.method == "POST":
20         form = RegisterForm(request.POST)
21         if form.is_valid():
22             user = form.save()
23             login(request, user)
24             return redirect("/home")
25     else:
26         form = RegisterForm()
27
28     return render(request, "registration/sign-up.html", {"form": form})
29

```

Rysunek 3 Plik views.py

Poniższy kod definiuje trasy URL, dla wszystkich widoków w projekcie.

```

1  from django.urls import path
2  from . import views
3  from django.conf import settings
4  from django.conf.urls.static import static
5  from django.contrib import admin
6  from yt_sentiment.views import yt_sentiment
7  from aimodels.views import all_models
8  from aimodels.views import aimodel
9  from yt_sentiment.views import yt_sentiment
10
11 urlpatterns = [
12     path("", views.home, name="home"),
13     path("contact", views.contact, name="contact"),
14     path("about", views.about, name="about"),
15     path("home/", views.home, name="home"),
16     path("sign-up/", views.sign_up, name="sing_up"),
17     path("all_models/", all_models, name="all_models"),
18     path("yt_sentiment/", yt_sentiment, name="yt_sentiment"),
19     path("all_models/<slug:slug>/", aimodel, name="aimodel"),
20 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
21

```

Rysunek 4 Plik urls.py

Kod w pliku *forms.py* tworzy formularz rejestracyjny dla użytkowników w aplikacji Django. Klasa dziedziczy po wbudowanym w Django formularzu `UserCreationForm`, który dostarcza podstawową funkcjonalność rejestracji użytkowników.

```

4  from captcha.fields import ReCaptchaField
5  from captcha.widgets import ReCaptchaV3
6
7
8  class RegisterForm(UserCreationForm):
9      email = forms.EmailField(required=True, max_length=50)
10     username = forms.CharField(required=True, max_length=50)
11     password1 = forms.CharField(
12         label="Password",
13         widget=forms.PasswordInput,
14         required=True,
15         max_length=100,
16     )
17     password2 = forms.CharField(
18         label="Password confirmation",
19         widget=forms.PasswordInput,
20         required=True,
21         max_length=100,
22     )
23     captcha = ReCaptchaField(widget=ReCaptchaV3)
24
25     class Meta:
26         model = User
27         fields = ["username", "email", "password1", "password2"]
28

```

Rysunek 5 Plik forms.py

Na poniższych zrzutach ekranu znajduje się kod bazowego szablonu `HTML base.html`, z którego korzystać będą szablony w pozostałych aplikacjach. Szablony będą dziedziczyły takie elementy jak pasek nawigacyjny, stopka oraz tło.

```

{% load static %}

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://cdn.tailwindcss.com"></script>
  <title>{% block title %}Aplikacje internetowe - Projekt{% endblock %}</title>
  <link rel="stylesheet" type="text/css" href="{% static '/css/main.css'%}">
</head>

```

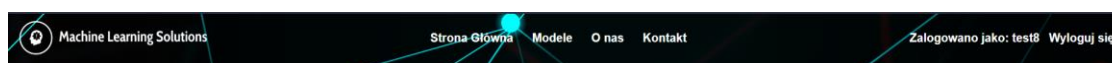
Rysunek 6 Plik base.html – sekcja head

```

<body class="bg-[url('{% static 'images/background-very-big.jpg' %})] bg-cover">
  <nav class="flex items-center justify-between px-5 py-3 text-xl">
    <div>
      <a href="/home"></a>
    </div>
    <ul class="hidden md:flex space-x-4 text-white font-semibold">
      <li><a href="/home" class="no-underline px-2">Strona Główna</a></li>
      <li><a href="/all_models" class="no-underline px-2">Modele</a></li>
      <li><a href="/about" class="no-underline px-2">O nas</a></li>
      <li><a href="/contact" class="no-underline px-2">Kontakt</a></li>
    </ul>
    <div class="md:hidden">
      <button id="burger-menu" class="text-white focus:outline-none">
        <svg class="w-6 h-6 fill="none" stroke="currentColor" viewBox="0 0 24 24"
          xmlns="http://www.w3.org/2000/svg">
          <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
            d="M4 6h16M4 12h16M4 18h16"></path>
        </svg>
      </button>
    </div>
    <ul class="hidden md:flex items-center space-x-4 font-semibold text-white">
      <!--ODWOLANIE SIE DO ADRESU URL-->
      <!--OBSLUZENIE PRZYCISKU WYLOGOWANIA-->
      {% if request.user.is_authenticated %}
      <span>Zalogowano jako: {{ user.username }}</span>
      <li><a href="{% url 'logout' %}">Wyloguj się</a></li>
      {% endif %}
      {% if not request.user.is_authenticated %}
      <li><a href="{% url 'login' %}">Zaloguj się</a></li>
      {% endif %}
    </ul>
  </nav>

```

Rysunek 7 Plik base.html – sekcja body – pasek nawigacyjny



Rysunek 8 Widok paska nawigacyjnego

Blok *content* może być nadpisywany przez szablony dziedziczące dzięki czemu możliwe jest wyświetlanie różnych treści przy zachowaniu elementów szablonu nadrzędnego.

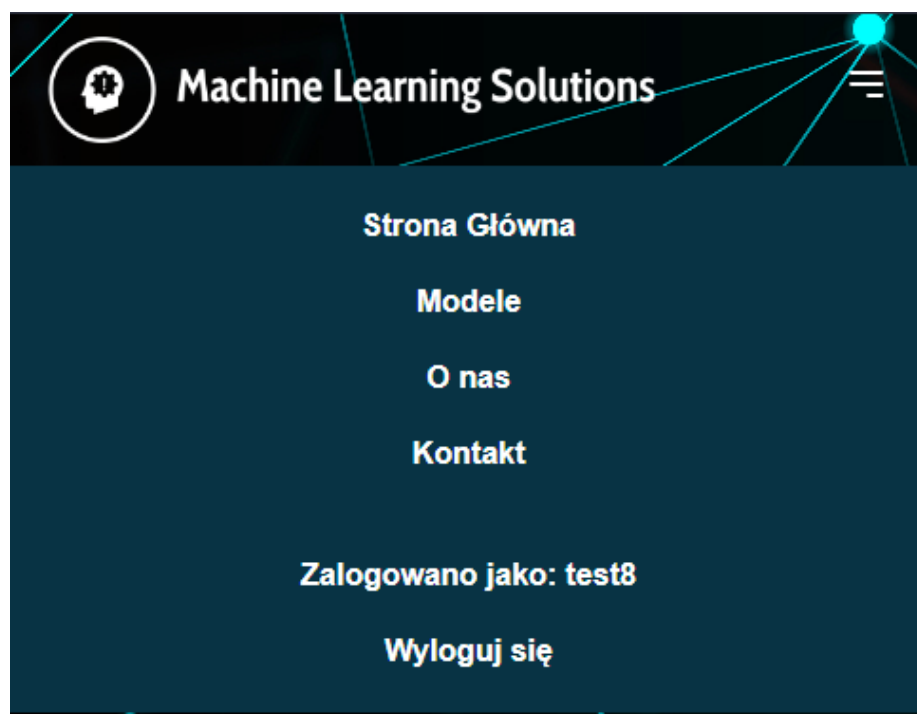
```

<div id="mobile-menu" class="md:hidden bg-cyan-950 text-white text-center">
  <ul class="py-3 font-semibold">
    <li><a href="/home" class="block px-4 py-2">Strona Główna</a></li>
    <li><a href="/all_models" class="block px-4 py-2">Modele</a></li>
    <li><a href="/about" class="block px-4 py-2">O nas</a></li>
    <li><a href="/contact" class="block px-4 py-2">Kontakt</a></li>
  </ul>
  <ul class="py-3 font-semibold">
    <!-- OBSŁUŻENIE PRZYCIŚKU WYLOGOWANIA -->
    <li><span class="block px-4 py-2">Zalogowano jako: {{ user.username }}</span></li>
    <li><a href="{% url 'logout' %}" class="block px-4 py-2">Wyloguj się</a></li>
    <li><a href="{% url 'login' %}" class="block px-4 py-2">Zaloguj się</a></li>
  </ul>
</div>

<div class="text-white min-h-screen items-center justify-center">
  {% block content %}
  {% endblock %}
</div>

```

Rysunek 9 Plik base.html – sekcja body – pasek nawigacyjny dla urządzeń mobilnych



Rysunek 10 Widok paska nawigacyjnego dla urządzeń mobilnych

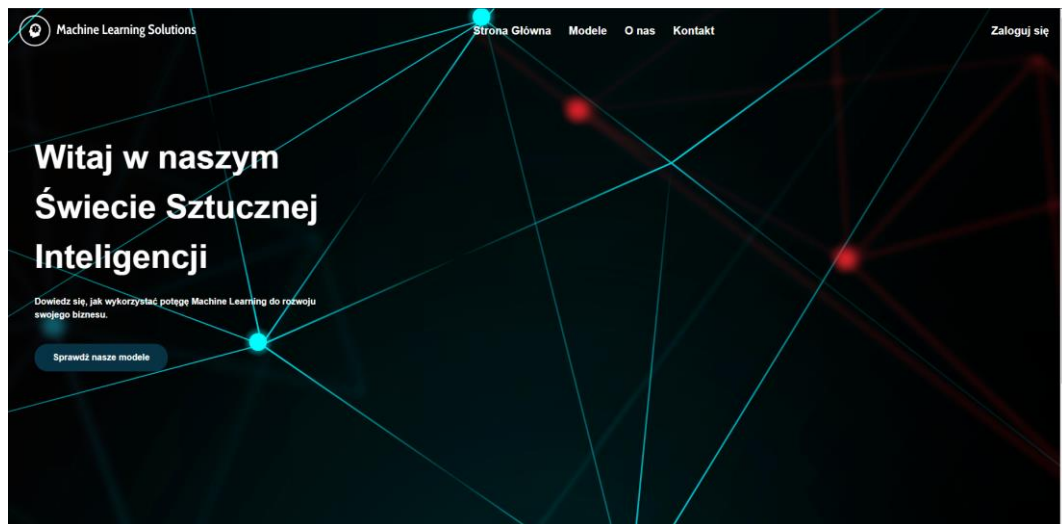
```

<script>
  document.getElementById('burger-menu').addEventListener('click', function () {
    document.getElementById('mobile-menu').classList.toggle('hidden');
  });
</script>

```

Rysunek 11 Plik base.html – skrypt wyświetlający menu nawigacyjne na urządzeniach mobilnych





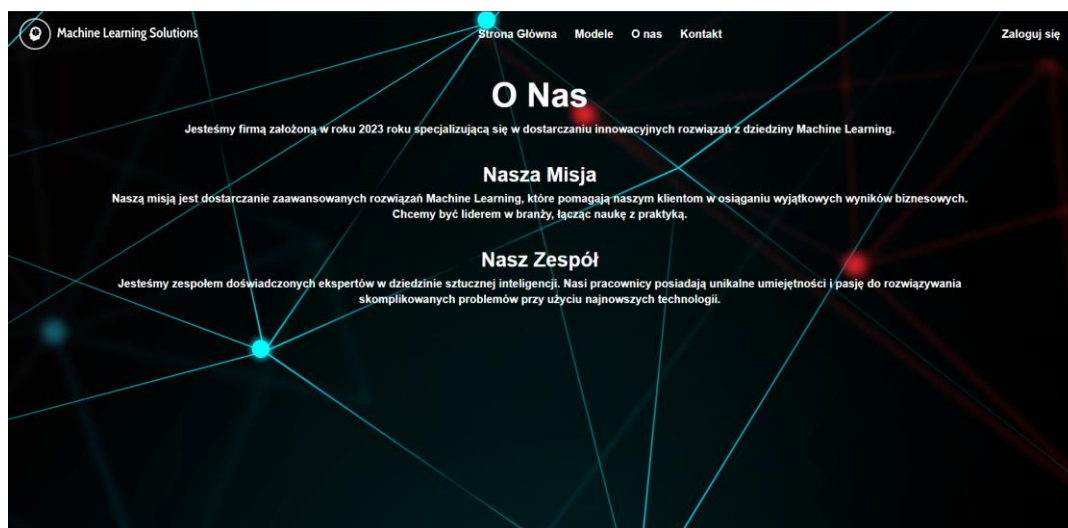
Rysunek 15 Widok strony głównej

```
{% extends 'main/base.html' %} You, 3 dni temu • fixed some issues with responsiveness

{% block title %}
O nas
{% endblock %}

{% block content %}
<div class="flex-center text-center mx-4 mt-10 sm:mx-20 md:mx-32 lg:mx-40">
<h1 class="text-4xl sm:text-6xl font-semibold mb-4">O Nas</h1>
<p class="text-sm sm:text-xl font-semibold leading-normal">
Jesteśmy firmą założoną w roku 2023 roku specjalizującą się w dostarczaniu innowacyjnych rozwiązań z dziedziny Machine Learning.
</p>
<br>
<br>
<h2 class="text-2xl sm:text-4xl font-semibold mb-2">Nasza Misja</h2>
<p class="text-sm sm:text-xl font-semibold leading-normal">
Naszą misją jest dostarczanie zaawansowanych rozwiązań Machine Learning, które pomagają naszym klientom w osiągnięciu wyjątkowych wyników biznesowych. Chcemy być liderem w branży, łącząc naukę z praktyką.
</p>
<br>
<br>
<h2 class="text-2xl sm:text-4xl font-semibold mb-2">Nasz Zespół</h2>
<p class="text-sm sm:text-xl font-semibold leading-normal">
Jesteśmy zespołem doświadczonych ekspertów w dziedzinie sztucznej inteligencji. Nasi pracownicy posiadają unikalne umiejętności i pasję do rozwiązywania skomplikowanych problemów przy użyciu najnowszych technologii.
</p>
</div>
{% endblock %}
```

Rysunek 16 Plik about.html – kod strony „O nas”



Rysunek 17 Widok strony „O nas”

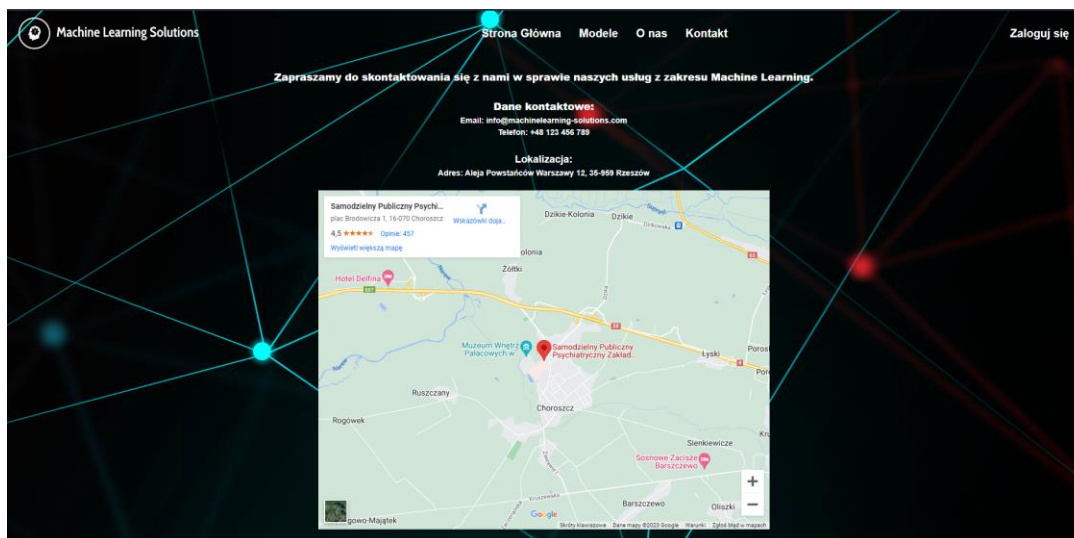
```

{% extends 'main/base.html' %}
{% block title %}
    Kontakt
{% endblock %}
{% block content %}


# <b>Zapraszamy do skontaktowania się z nami w sprawie naszych usług z zakresu Machine Learning.</b></h1> <b>Dane kontaktowe:</b></h2> Email: <a href="mailto:info@machinelearning-solutions.com">info@machinelearning-solutions.com</a></p> <style> .map-container { margin: 20px auto; max-width: 800px; } </style> <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d38283.03070243943!2d22.93937241335268!3d53.1516683!2m3!1f0!2f0!3f0!3m2!1!1024!2!768!4 </div>


```

Rysunek 18 Plik *contact.html* – kod strony kontaktowej



Rysunek 19 Widok strony kontaktowej

Poniższy kod znajdujący się w plikach *login.html* oraz *sign-up.html* obsługuje strony logowania i rejestracji z formularzami, obsługującymi błędy, które następnie są wyświetlane użytkownikowi.



```

{% block content %}
<div class="bg-cyan-950 flex flex-col md:flex-row rounded-2xl shadow-lg max-w-3xl p-5 items-center mt-10 m-auto">
  <div class="md:w-1/2 px-8 md:px:16 mb-4 md:mb-0">
    <h2 class="font-bold text-2xl text-white text-center">Zaloguj się</h2>

    <form method="post" class="flex flex-col gap-4">
      {% csrf_token %}
      <input class="p-2 mt-4 rounded-xl border text-black" type="text" name="username" placeholder="Nazwa użytkownika">

      <input class="p-2 rounded-xl border w-full text-black" type="password" name="password" placeholder="Hasło">

      {% if form.errors %}
        {% for field in form %}
          {% for error in field.errors %}
            <div class="p-2 bg-red-200 text-red-800 text-center rounded-xl">
              <p>{{ error|escape }}</p>
            </div>
          {% endfor %}
        {% endfor %}
        {% for error in form.non_field_errors %}
          <div class="p-2 bg-red-200 text-red-800 text-center rounded-xl">
            <p>{{ error|escape }}</p>
          </div>
        {% endfor %}
      {% endif %}

      <button class="bg-white rounded-xl text-cyan-950 py-2 hover:scale-105 duration-300">Zaloguj się</button>
    </form>

    <div class="mt-6 grid grid-cols-3 items-center text-white">
      <hr class="border-white col-span-1">
      <p class="text-center text-sm col-span-1">Lub</p>
      <hr class="border-white col-span-1">
    </div>

    <button class="bg-white border py-2 w-full rounded-xl mt-5 flex justify-center items-center text-sm hover:scale-105 duration-300 text-[#002074]">
      <svg class="w-3" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 48 48" width="25px">
        <path fill="#FFC107" d="M43.611,20.803H20.803V8.103c-1.649,4.657-6.08,8.103-11.309,8.103c-6.627,0-12.537-12-12.537-12c3.059,0,5.842,1.154,7.96
          <path fill="#FF3000" d="M6.306,14.691c-1.649,4.657-6.08,8.103-11.309,8.103c-6.627,0-12.537-12-12.537-12c3.059,0,5.842,1.154,7.961
          <path fill="#4CAF50" d="M24.446,16.667c-1.649,4.657-6.08,8.103-11.309,8.103c-6.627,0-12.537-12-12.537-12c3.059,0,5.842,1.154,7.961
          <path fill="#1976D2" d="M43.611,20.803H20.803V8.103c-1.649,4.657-6.08,8.103-11.309,8.103c-6.627,0-12.537-12-12.537-12c3.059,0,5.842,1.154,7.961
        </svg>
        Zaloguj się przez Google
      </button>

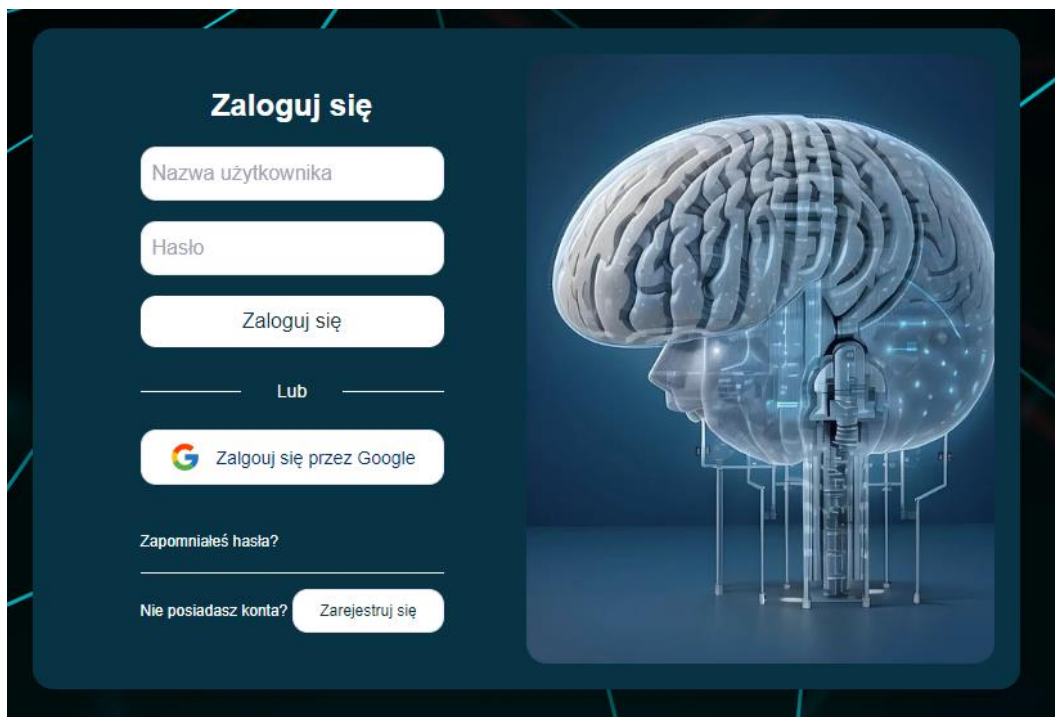
    <div class="mt-5 text-xs border-b border-white py-4 text-white">
      <a href="#">Zapomniałeś hasła</a>
    </div>

    <div class="mt-3 text-xs flex justify-between items-center text-white">
      <p>Nie posiadasz konta?</p>
      <a href="/sign-up">
        <button class="py-2 px-5 bg-white border rounded-xl hover:scale-110 duration-300 text-cyan-950">Zarejestruj się</button>
      </a>
    </div>

    <div class="md:w-1/2 md:block hidden">
      
    </div>
  </div>
{% endblock %}

```

Rysunek 20 Plik login.html – kod panelu logowania



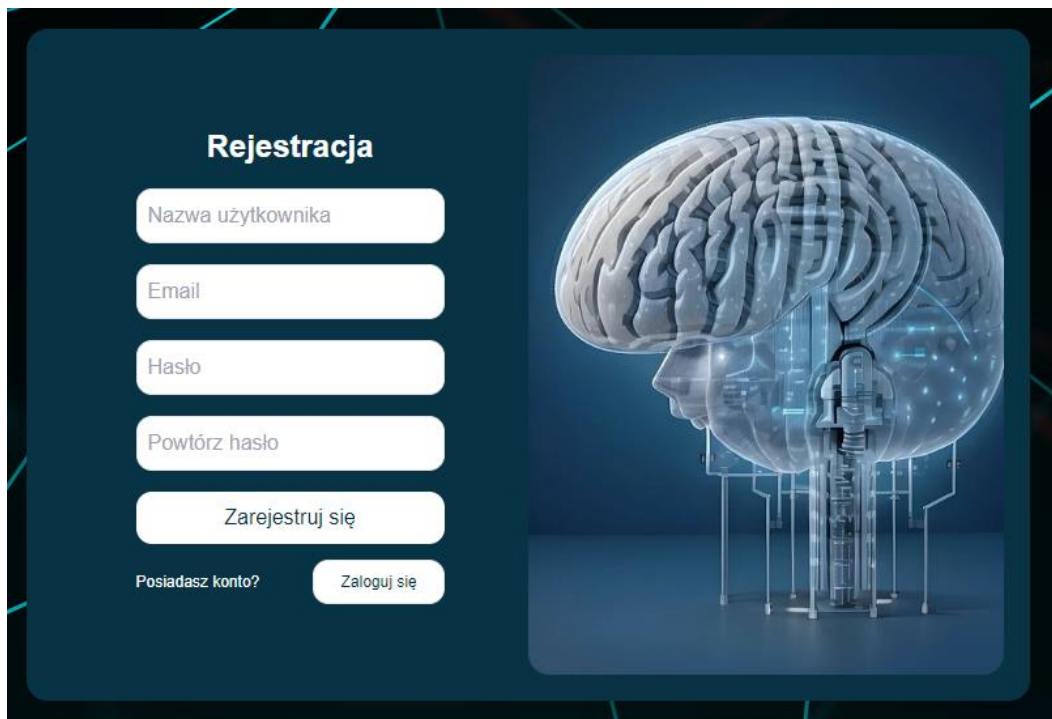
Rysunek 21 Widok panelu logowania

```

1  {% extends 'main/base.html' %}
2
3  {% block title %}
4      Rejestracja
5  {% endblock %}
6
7  {% block content %}
8      <div class="bg-cyan-950 flex flex-col md:flex-row rounded-2xl shadow-lg max-w-3xl p-5 items-center mt-10 m-auto">
9          <div class="md:w-1/2 px-8 md:px-16 mb-4 md:mb-0">
10             <h2 class="font-bold text-2xl text-white text-center">Rejestracja</h2>
11
12             <form method="post" class="flex flex-col gap-4">
13                 {% csrf_token %}
14                 <input class="p-2 mt-4 rounded-xl border text-black" type="text" name="username" placeholder="Nazwa użytkownika">
15
16                 <div class="relative">
17                     <input class="p-2 rounded-xl border w-full text-black" type="text" name="email" placeholder="Email">
18                 </div>
19
20                 <div class="relative">
21                     <input class="p-2 rounded-xl border w-full text-black" type="password" name="password1" placeholder="Hasło">
22                 </div>
23
24                 <div class="relative">
25                     <input class="p-2 rounded-xl border w-full text-black" type="password" name="password2" placeholder="Powtórz hasło">
26                 </div>
27
28                 {% if form.errors %}
29                     {% for field in form %}
30                         {% for error in field.errors %}
31                             <div class="p-2 bg-red-200 text-red-800 text-center rounded-xl">
32                                 <p>{{ error|escape }}</p>
33                             </div>
34                         {% endfor %}
35                     {% endfor %}
36                     {% for error in form.non_field_errors %}
37                         <div class="p-2 bg-red-200 text-red-800 text-center rounded-xl">
38                             <p>{{ error|escape }}</p>
39                         </div>
40                     {% endfor %}
41                 {% endif %}
42
43                 <button class="bg-white rounded-xl text-cyan-950 py-2 hover:scale-105 duration-300">Zarejestruj się</button>
44                 {{ form.captcha }}
45             </form>
46
47             <div class="mt-3 text-xs flex justify-between items-center text-white">
48                 <p>Posiadasz konto</p>
49                 <a href="/login">
50                     <button class="py-2 px-5 bg-white border rounded-xl hover:scale-110 duration-300 text-cyan-950">Zaloguj się</button>
51                 </a>
52             </div>
53         </div>
54
55         <div class="md:w-1/2 md:block hidden">
56             
57         </div>
58     </div>
59 {% endblock %}

```

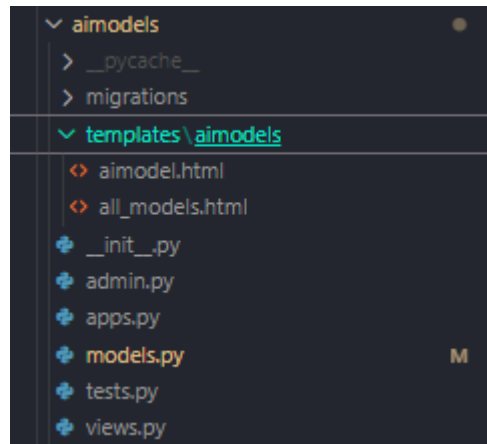
Rysunek 22 Plik sign-up.html – kod panelu rejestracji



Rysunek 23 Widok panelu rejestracji

## 2.4. Aplikacja „*aimodels*”

Aplikacja *aimodels* obsługuje stronę z modelami oferowanymi przez firmę oraz strony poszczególnych modeli.



Rysunek 24 Katalog aplikacji „*aimodels*”

W pliku *views.py* znajdują się dwa widoki. Widok *all\_models* pozwala na filtrowanie modeli według kategorii i wyszukiwania, prezentując wyniki na stronie, natomiast widok *aimodel* wyświetla szczegóły konkretnego modelu sztucznej inteligencji.

```
from django.shortcuts import render
from django.db.models import Q
from aimodels.models import AiModel, Category

# Create your views here.

You, przedwczoraj • fixed more issues with responsiveness

def all_models(request):
    categories = Category.objects.all()
    aimodels = AiModel.objects.all()

    active_category = request.GET.get("category", "")

    if active_category:
        aimodels = aimodels.filter(category__slug=active_category)

    query = request.GET.get("query", "")

    if query:
        aimodels = aimodels.filter(
            Q(name__icontains=query) | (Q(description__icontains=query))
        )

    context = {
        "categories": categories,
        "aimodels": aimodels,
        "active_category": active_category,
    }
    return render(request, "aimodels/all_models.html", context)

def aimodel(request, slug):
    aimodel = AiModel.objects.get(slug=slug)

    return render(request, "aimodels/aimodel.html", {"aimodel": aimodel})
```

Rysunek 25 Plik *views.py*

```

from django.db import models
from PIL import Image
from io import BytesIO
from django.core.files import File

# Create your models here.

You, przedwczoraj | 2 authors (PatrikW7 and others)
class Category(models.Model):
    name = models.CharField(max_length=255)
    slug = models.SlugField()

    You, przedwczoraj | 2 authors (You and others)
    class Meta:
        ordering = ("name",)

    def __str__(self):
        return self.name

You, przedwczoraj | 2 authors (PatrikW7 and others)
class AiModel(models.Model):
    category = models.ForeignKey(
        Category, related_name="aimodels", on_delete=models.CASCADE
    )
    name = models.CharField(max_length=255)
    slug = models.SlugField()
    price = models.FloatField()
    description = models.TextField()
    image = models.ImageField(upload_to="uploads/", blank=True, null=True)
    thumbnail = models.ImageField(upload_to="uploads/", blank=True, null=True)

    You, przedwczoraj | 2 authors (You and others)
    class Meta:
        ordering = ("name",)

    def make_thumbnail(self, image, size=(300, 300)):
        img = Image.open(image)
        img.convert("RGB")
        img.thumbnail(size)

        thumb_io = BytesIO()
        img.save(thumb_io, "JPEG", quality=85)

        thumbnail = File(thumb_io, name=image.name)
        return thumbnail

    def get_thumbnail(self):
        if self.thumbnail:
            return self.thumbnail.url
        else:
            if self.image:
                self.thumbnail = self.make_thumbnail(self.image)
                self.save()

            return self.thumbnail.url
        else:
            return "https://via.placeholder.com/240x240x.jpg"

    def __str__(self):

```

Rysunek 26 Plik models.py

```

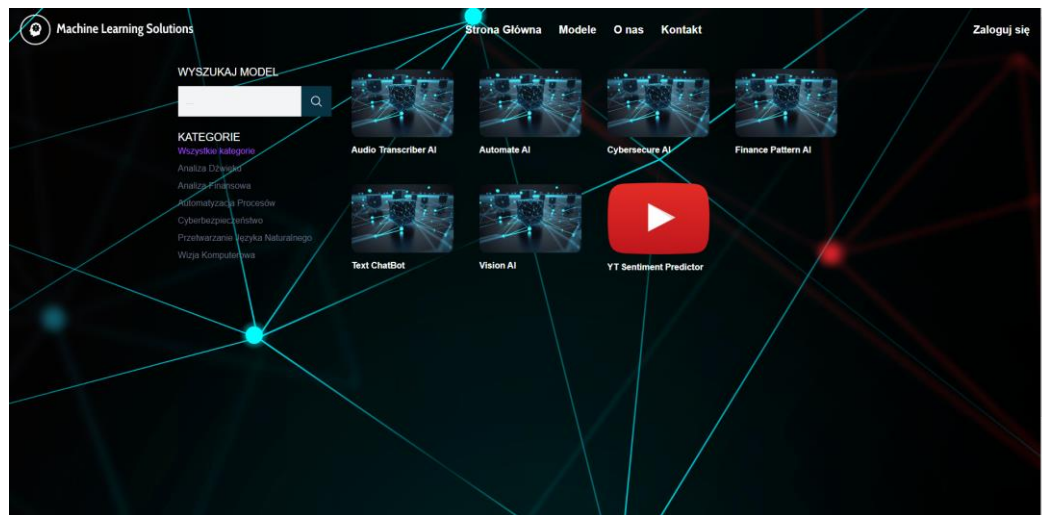
{% extends 'main/base.html' %}
{% block title %}
    Modele
{% endblock %}
{% block content %}

<div class="max-w-7xl mx-auto flex flex-wrap items-start py-6 px-6 xl:px-0">
    <div class="filters w-full lg:w-1/5">
        <h3 class="mb-3 text-xl uppercase">Wyszukaj model</h3>

        <form method="get" action=".">
            <div class="flex">
                <input type="text" name="query" class="p-4 bg-gray-100 border-0 text-black placeholder="...">
                <button class="p-4 bg-gray-900 border-0">
                    <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor" class="w-6 h-6">
                        <path stroke-linecap="round" stroke-linejoin="round" d="M21 21 5.197 5.197M8 8 0 16" />
                    </svg>
                </button>
            </div>
        </form>
        <h3 class="mt-6 text-xl uppercase">Kategorie</h3>
        <div class="space-y-2">
            <li><a href="{% url 'all_models' %}" class="{% if not active_category %} text-purple-500 {% else %} text-gray-500 {% endif %}">Wszystkie kategorie</a></li>
            {% for category in categories %}
                <li><a href="{% url 'all_models' %}?category={{category.slug}}" class="{% if category.slug == active_category %} text-purple-500 {% else %} text-gray-500 {% endif %}">{{category.name}}</a></li>
            {% endfor %}
        </div>
    </div>
    <div class="aimodels w-full lg:w-4/5 -mt-4 flex items-center flex-wrap px-18">
        {% for aimodel in aimodels %}
            <div class="w-full md:w-1/3 xl:w-1/4 p-6">
                <a href="{% url 'aimodel' %}?slug={{aimodel.slug}}">
                    
                </a>
                <div class="pt-3 flex items-center justify-between font-roboto">
                    <a href="{% url 'aimodel' %}?slug={{aimodel.slug}}">{{aimodel.name}}</a>
                </div>
            </div>
        {% endfor %}
    </div>
</div>
{% endblock %}

```

Rysunek 27 Plik all\_models.html



Rysunek 28 Widok strony z modelami

```
{% extends 'main/base.html' %}
{% block title %}
    Modely
{% endblock %}
{% block content %}

<div class="max-w-6xl mx-auto flex flex-wrap py-6 px-6 xl:px-0">

    <div class="lg:flex w-full mb-6 lg:mb-0">
        <div class="lg:w-3/5 order-last lg:order-first">
            
        </div>

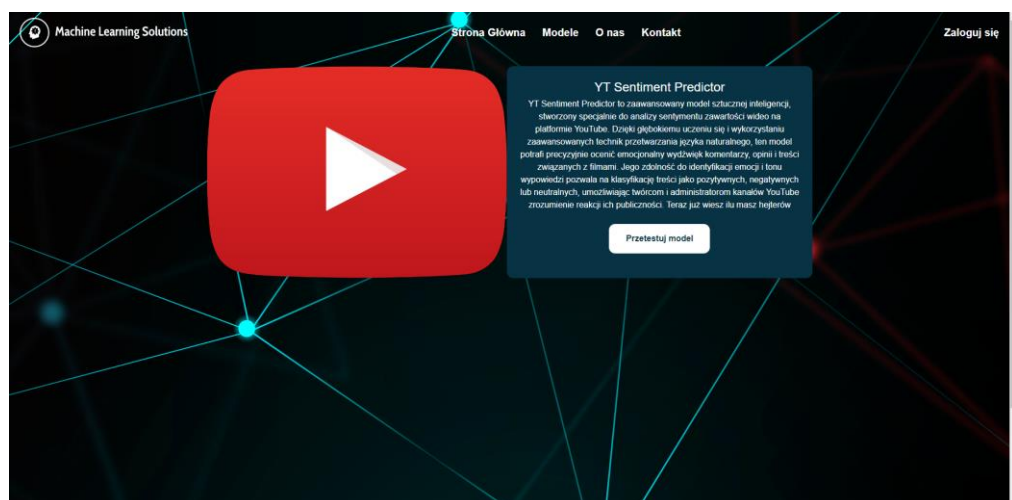
        <div class="lg:w-2/5 lg:p-6 bg-cyan-950 text-center rounded-xl order-first lg:order-last">
            <h1 class="text-2xl">{{ aimodel.name }}</h1>
            <p class="mt-1 text-white-700">{{ aimodel.description }}</p>

            {% if aimodel.name == 'YT Sentiment Predictor' %}
                <a href="/yt_sentiment" class="mt-6 inline-block px-8 py-4 rounded-xl bg-white text-cyan-950 hover:bg-yellow-100">
                    <b>Przetestuj model</b>
                </a>
            {% else %}
                <a href="#" class="mt-6 inline-block px-8 py-4 rounded-xl bg-white text-cyan-950 hover:bg-yellow-100">
                    <b>Przetestuj model</b>
                </a>
            {% endif %}
        </div>
    </div>

</div>

{% endblock %}
```

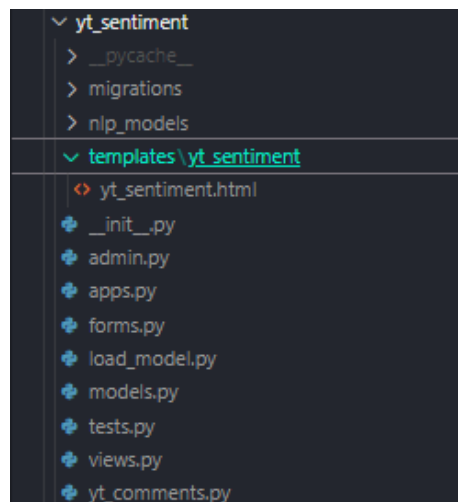
Rysunek 29 Plik aimodel.html



Rysunek 30 Widok przykładowej strony modelu

## 2.5. Aplikacja „yt\_sentiment”

Aplikacja `yt_sentiment` z wykorzystaniem modelu sztucznej inteligencji przeprowadza analizę sentymentu komentarzy znajdujących się pod filmem z serwisu Youtube, który poda użytkownik. Wyniki przekazywane są do szablonu `yt_sentiment.html`.



Rysunek 31 Katalog aplikacji „yt\_sentiment”

Kod w pliku `yt_comments.py`, umożliwia pobieranie komentarzy z filmu na platformie YouTube przy użyciu YouTube Data API.

```
import googleapiclient.discovery
import googleapiclient.errors
import os
import re

def get_comments(youtube, **kwargs):
    comments = []
    results = youtube.commentThreads().list(**kwargs).execute()

    while results:
        for item in results["items"]:
            comment = item["snippet"]["topLevelComment"]["snippet"]["textDisplay"]
            comments.append(comment)

            if "nextPageToken" in results:
                kwargs["pageToken"] = results["nextPageToken"]
                results = youtube.commentThreads().list(**kwargs).execute()
            else:
                break

    return comments

def get_video_comments(url, api_key):
    os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"

    video_id = re.search(r"(?<=)[^&]+", url)[0]
    youtube = googleapiclient.discovery.build("youtube", "v3", developerKey=api_key)

    comments = get_comments(
        youtube, part="snippet", videoId=video_id, textFormat="plainText"
    )
    return comments

def main(video_id, api_key):
    return get_video_comments(video_id, api_key)
```

Rysunek 32 Plik `yt_comments.py`

Funkcja `load_model`, umożliwia załadowanie wytrenowanego wcześniej modelu przetwarzania języka naturalnego.

```
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification

def load_model(path):
    tokenizer = AutoTokenizer.from_pretrained(path)
    model = TFAutoModelForSequenceClassification.from_pretrained(path)

    return model, tokenizer
```

Rysunek 33 Plik `load_model.py`

Poniższy kod z pliku `views.py`, definiuje widok Django, który obsługuje analizę sentymentu komentarzy z filmu na YouTube. Widok zabezpieczony jest przed dostępem niezalogowanych użytkowników.

```
from django.shortcuts import render
from pathlib import Path
from .yt_comments import get_video_comments
from .load_model import load_model
from transformers import pipeline
import environ
import pandas as pd
from django.contrib.auth.decorators import login_required
from urllib.error import HTTPError
from googleapiclient.errors import HttpError
import os.path

env = environ.Env()
environ.Env.read_env()

GOOGLE_API_KEY = env("GOOGLE_API_KEY")
base_path = Path(__file__).parent
model_path = Path(base_path, "nlp_models/twitter-roberta-base-sentiment")
model, tokenizer = load_model(path=model_path)
tokenizer_kwargs = {"padding": True, "truncation": True, "max_length": 512}

nlp = pipeline(
    "sentiment-analysis", model=model, tokenizer=tokenizer, **tokenizer_kwargs
)

result = {}

part_PROJECT_PATH = os.path.abspath(os.path.dirname(__name__))
full_PROJECT_PATH = part_PROJECT_PATH + r"static\file.csv"
```

Rysunek 34 Plik `views.py` – załadowanie modelu przetwarzania języka naturalnego

```

@login_required(login_url="/login")
def yt_sentiment(request):
    if request.method == "POST":
        try:
            text = request.POST.get("text")
            comments = get_video_comments(text, GOOGLE_API_KEY)

            if not comments:
                result = {"available": "Brak komentarzy"}

                return render(
                    request, "yt_sentiment/yt_sentiment.html", {"sentiment": result}
                )
            else:
                analysed_comments = [nlp(comment, model)[0] for comment in comments]
                df_result = pd.DataFrame(analysed_comments)

                sentiment_percentage = df_result["label"].value_counts().values / len(
                    df_result
                )
                sentiment_percentage = [round(num, 2) for num in sentiment_percentage]
                num_comments = len(df_result)
                percents = df_result["label"].value_counts()
                lab = df_result["label"]
                percent_help = {}

                for label, value in zip(percents.index, percents.values):
                    percent_help[label] = value

                labels = list(percent_help.keys())
                values = list(percent_help.values())

                index = []
                [index.append(i + 1) for i in range(len(comments))]
                df_comments = pd.DataFrame(
                    {
                        "Komentarz": comments,
                        "Wynik analizy": lab,
                    }
                )

            except HttpError:
                result = {"available": "Brak komentarzy"}

                return render(
                    request, "yt_sentiment/yt_sentiment.html", {"sentiment": result}
                )

            try:
                most_pos_com_id = df_result[df_result["label"] == "Pozytywny"][
                    "score"
                ].idxmax()
                most_pos_com = comments[most_pos_com_id]
            except ValueError:
                most_pos_com = "Nie masz żadnych pozytywnych komentarzy 😞"

            try:
                most_neg_com_id = df_result[df_result["label"] == "Negatywny"][
                    "score"
                ].idxmax()
                most_neg_com = comments[most_neg_com_id]
            except ValueError:
                most_neg_com = "Nie masz żadnych negatywnych komentarzy! 😞"

            df_comments.to_csv(full_PROJECT_PATH)

            result = {
                "num_comments": num_comments,
                "most_pos_com": most_pos_com,
                "most_neg_com": most_neg_com,
                "percents": percent_help.items,
                "labels": labels,
                "values": values,
            }

            return render(
                request,
                "yt_sentiment/yt_sentiment.html",
                {"sentiment": result, "df": df_comments},
            )
        else:
            return render(request, "yt_sentiment/yt_sentiment.html")

```

Rysunek 35 Plik views.py – funkcja yt\_sentiment



Na poniższych zrzutach ekranu znajduje się kod z pliku *yt\_sentiment.html*. Ten kod definiuje szablon Django do wyświetlania wyników analizy sentymentu komentarzy na Youtube. Strona zawiera formularz, w którym użytkownik może wprowadzić URL filmu. Po analizie wyświetlane są wyniki na podstawie danych z widoku *yt\_sentiment*.

```
<form method="post">
  {% csrf_token %}
  <input type="text" name="text" class="text-black rounded-xl border py-3 px-3 md:py-4 md:px-4 lg:py-5 lg:px-5">
  <button type="submit" class="hover:bg-transparent hover:border-cyan-950 hover:text-white duration-300 hover:
</form>
```

Rysunek 36 Plik *yt\_sentiment.html* – formularz

```
<div class="flex-center text-center pt-10 text-2xl">
  {% if sentiment.num_comments %}
    <h1>Liczba komentarzy: {{ sentiment.num_comments }}</h1>
    <ul>
      {% for kategoria, procent in sentiment.percents %}
        {% if kategoria == "Pozytywny" %}
          <li><span style="color: #06D6A0;">{{ kategoria }}</span> - {{ procent }}</li>

        {% elif kategoria == "Negatywny" %}
          <li><span style="color: #EF476F;">{{ kategoria }}</span> - {{ procent }}</li>

        {% else %}
          <li><span style="color: #7d8597;">{{ kategoria }}</span> - {{ procent }}</li>
        {% endif %}
      {% endfor %}
    </ul>

    PátrykW7, w zeszłym miesiącu • Value Error no comments - Exception
    <h1>Najbardziej pozytywny komentarz: "{{ sentiment.most_pos_com }}"</h1>
    <h1>Najbardziej negatywny komentarz: "{{ sentiment.most_neg_com }}"</h1>

  {% else %}
    <h1> {{ sentiment.available }}</h1>
  {% endif %}

  <style type="text/css">
    .chart-container {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 400px;
    }

    .chartBox {
      width: 80%;
      max-width: 650px;
    }
  </style>

  <div class="chart-container">
    <canvas id="myChart" class="chartBox"></canvas>
  </div>
```

Rysunek 37 Plik *yt\_sentiment.html* – raport analizy komentarzy

```
<div class="flex-center text-center pt-10 text-2xl">
  {% if sentiment.num_comments %}
    <h1>Liczba komentarzy: {{ sentiment.num_comments }}</h1>
    <ul>
      {% for kategoria, procent in sentiment.percents %}
        {% if kategoria == "Pozytywny" %}
          <li><span style="color: #06D6A0;">{{ kategoria }}</span> - {{ procent }}</li>

        {% elif kategoria == "Negatywny" %}
          <li><span style="color: #EF476F;">{{ kategoria }}</span> - {{ procent }}</li>

        {% else %}
          <li><span style="color: #7d8597;">{{ kategoria }}</span> - {{ procent }}</li>
        {% endif %}
      {% endfor %}
    </ul>

    PátrykW7, w zeszłym miesiącu • Value Error no comments - Exception
    <h1>Najbardziej pozytywny komentarz: "{{ sentiment.most_pos_com }}"</h1>
    <h1>Najbardziej negatywny komentarz: "{{ sentiment.most_neg_com }}"</h1>

  {% else %}
    <h1> {{ sentiment.available }}</h1>
  {% endif %}
```

Rysunek 38 Plik *yt\_sentiment.html* – raport analizy komentarzy

```

<style type="text/css">
  .chart-container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 400px;
  }

  .chartBox {
    width: 80%;
    max-width: 650px;
  }
</style>

<div class="chart-container">
  <canvas id="myChart" class="chartBox"></canvas>
</div>

<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<div id="plotly-chart"></div>

<script>
  {{ plot_html|safe }}
</script>

<script>
  const ctx = document.getElementById('myChart');

  const labels = {{ sentiment.labels|safe }};
  const values = {{ sentiment.values|safe }};

  const backgroundColors = labels.map((label) => {
    if (label === 'Pozytywny') {
      return '#86D6A8';
    } else if (label === 'Negatywny') {
      return '#EF476F';
    } else if (label === 'Neutralny') {
      return '#7d8597';
    } else {
      return 'rgb(128, 128, 128)';
    }
  });

  const data = {
    labels: labels,
    datasets: [{
      data: values,
      maintainAspectRatio: false,
      backgroundColor: backgroundColors,
      hoverOffset: 4
    }]
  };

  const config = {
    type: 'doughnut',
    data: data,
  };

  new Chart(ctx, config);
</script>

<br>
<br>

```

Rysunek 39 Plik *yt\_sentiment.html* – wykres kołowy

```

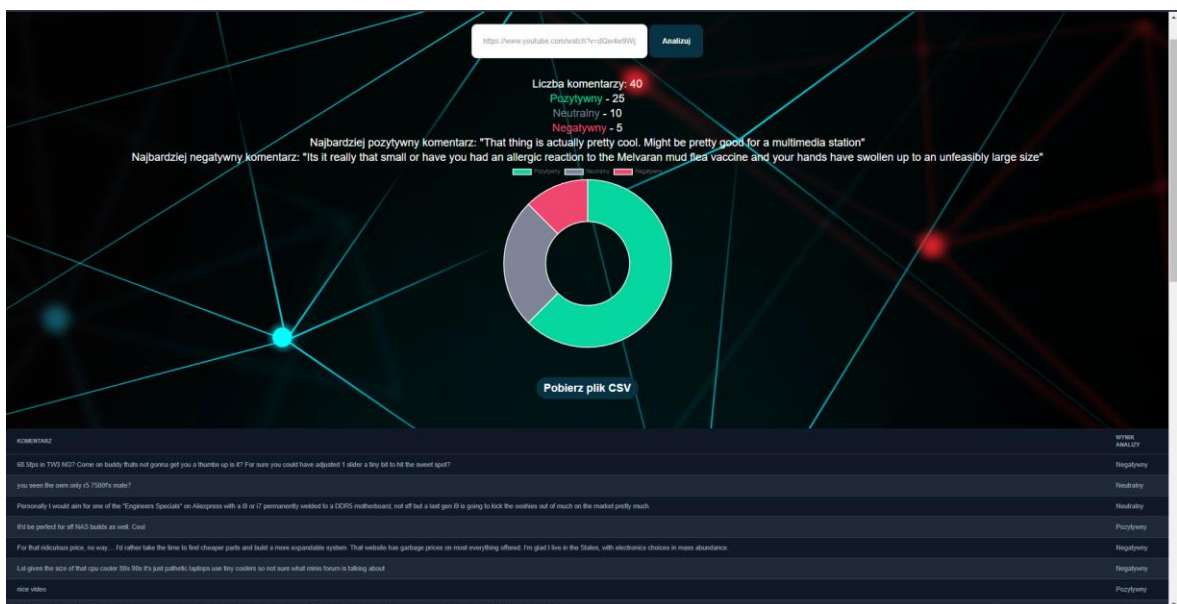
<div>
  {% if sentiment.num_comments %}
    {% load static %}
    <a href="{% static 'file.csv' %}" download="file.csv">
      <button type="button" class="{% bg-cyan-950 hover:bg-transparent text-white font-bold py-2 px-4 rounded-full %}">Pobierz plik CSV</button>
    </a>
  {% endif %}
</div>

<br>
<br>

<div class="relative overflow-x-auto shadow-md sm:rounded-lg">
  <table class="w-full text-sm text-left rtl:text-right text-gray-500 dark:text-gray-400">
    <thead class="text-xs text-gray-700 uppercase bg-gray-50 dark:bg-gray-700 dark:text-gray-400">
      <tr class="{% odd:bg-white even:dark:bg-gray-900 even:bg-gray-50 even:dark:bg-gray-800 border-b dark:border-gray-700 %}">
        {% for column in df.columns %}
          <th scope="col" class="px-6 py-3">{{ column }}</th>
        {% endfor %}
      </tr>
    </thead>
    <tbody>
      {% for index, row in df.iterrows %}
        <tr class="{% odd:bg-white even:dark:bg-gray-900 even:bg-gray-50 even:dark:bg-gray-800 border-b dark:border-gray-700 %}">
          {% for value in row %}
            <td scope="col" class="px-6 py-3">{{ value }}</td>
          {% endfor %}
        </tr>
      {% endfor %}
    </tbody>
  </table>
</div>

```

Rysunek 40 Plik yt\_sentiment.html – tabela komentarzy



Rysunek 41 Wynik analizy komentarzy

## 2.6. Zabezpieczenie aplikacji

### 2.6.1. Zabezpieczenia przed atakami XSS

Django chroni aplikacje przed atakami XSS poprzez automatyczne escapowanie danych wyświetlanych w szablonach. Framework domyślnie zabezpiecza wyjście danych, co uniemożliwia wykonywanie złośliwego kodu JavaScript w przeglądarce. Dodatkowo, Django umożliwia konfigurację nagłówków HTTP, takich jak Content Security Policy (CSP), co pozwala na precyzyjną kontrolę nad źródłami zasobów.

### 2.6.2. Zabezpieczenia przed atakami CSRF

Django wyposażone jest w mechanizmy obrony przed atakami CSRF (Cross-Site Request Forgery) poprzez wykorzystanie unikalnego tokenu CSRF. Automatycznie

dodawany jest do formularzy w szablonach, a przy każdym żądaniu sprawdzany jest zgodność tokena. Ten mechanizm uniemożliwia atakującemu wykorzystanie autoryzacji użytkownika do wykonywania niepożądanych działań na ich koncie.

```
<form method="post">
  {% csrf_token %}
  <input type="text" name="text" class="text-black rounded-xl border py-3 px-3 md
  <button type="submit" class="hover:bg-transparent hover:border-cyan-950 hover:t
</form>
```

`{% csrf_token %}` jest tagiem umieszczanym w formularzach szablonów HTML, który generuje i wstawia unikalny token CSRF do formularza HTML.

### 2.6.3. Zabezpieczenia przed atakami SQL-Injection

Django zapewnia wbudowane zabezpieczenia przeciwko atakom SQL Injection poprzez użycie mechanizmu ORM (Object-Relational Mapping). Ten mechanizm separuje warstwę bazodanową od logiki aplikacji, automatycznie parametryzując zapytania SQL. Dzięki temu, dane przekazywane do baz danych są automatycznie escapowane i nie są interpretowane jako kod SQL, eliminując potencjalne luki, które mogłyby być wykorzystane do ataków.

Automatyczne escapowanie oznacza, że Django samodzielnie przekształca dane tak, aby unieważnić jakiegokolwiek specjalne znaki SQL, które mogłyby być wprowadzone przez użytkownika. Na przykład, jeśli użytkownik wprowadzi ciąg znaków zawierający znaki specjalne SQL, Django automatycznie zabezpieczy te znaki, eliminując ryzyko wstrzyknięcia kodu SQL.

### 2.6.4. Uwierzelnianie modeli użytkowników

Django obsługuje uwierzelnianie poprzez wbudowany system kont użytkowników. Proces uwierzelniania jest zazwyczaj inicjowany przez formularz logowania, który przekazuje dane uwierzelniające, takie jak nazwa użytkownika i hasło. Mechanizm uwierzelniania weryfikuje te dane zapisane w bazie danych, sprawdzając ich poprawność. W przypadku zgodności, Django tworzy sesję użytkownika, co pozwala mu utrzymać zalogowanie między różnymi stronami aplikacji.

### 2.6.4. Mechanizm sesji użytkownika

Mechanizm sesji użytkownika w Django elementem, który umożliwia przechowywanie informacji o stanie zalogowania między różnymi żądaniami HTTP. Po uwierzelnieniu, Django tworzy sesję użytkownika, która jest przechowywana po stronie serwera. Sesje te są identyfikowane przez unikatowy identyfikator sesji, który jest przekazywany za pomocą pliku cookie lub parametru URL.

### 2.6.5. Zapewnienie bezpieczeństwa zmiennym środowiskowym

W celu zapewnienia bezpieczeństwa zmiennym środowiskowym wykorzystana została biblioteka Environ, która pomaga w zarządzaniu zmiennymi środowiskowymi. Environ umożliwia bezpieczną obsługę zmiennych środowiskowych w aplikacji. Dzięki niej możliwe jest kontrolowanie i ochrona poufnych informacji, takich jak klucze do zewnętrznych interfejsów programistycznych API, dane uwierzytelniające do chmurowej bazy danych, zapewniając jednocześnie bezpieczeństwo i kontrolę nad konfiguracją.

```
.env x settings.py
website > website > .env
1 RECAPTCHA_PUBLIC_KEY=6LeA-NQoAAAAAD6CaaEB92gVhYozjctvSth2DNwy
2 RECAPTCHA_PRIVATE_KEY=6LeA-NQoAAAAADy_sgNP_F7_0BLcc3-MbupPMZYr
3
4 DATABASE_URL=postgres://ai_database_user:nFhR0q3nuwGPauShmiyZ3jFBCItgXWk0@dpg-ckrn5a01hnes73fusoc0-a.frankfurt-postgres
5
6 GOOGLE_API_KEY=AIzaSyDdoevpmj4XGD2YaquRlvx101kDKH94ji4
```

```
env = environ.Env()
environ.Env.read_env()

GOOGLE_API_KEY = env("GOOGLE_API_KEY")
```

Ważnym krokiem w przypadku korzystania z systemu kontroli wersji Git jest zaprzestanie śledzenia pliku zawierającego zmienne środowiskowe w celu uniemożliwienia wycieku zmiennych do zdalnego publicznego repozytorium. Umieszczenie nazwy pliku `.env` w pliku `.gitignore`, zapobiega śledzeniu wrażliwego pliku przez system Git.

```
# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/
```

## 2.6.7. Wdrożenie wymagań dotyczących długości haseł użytkowników

Utworzona została polityka haseł użytkowników, która wymaga od nowych kont hasła spełniającego określone wymogi długości i złożoności. Poprzez ustalenie minimalnej długości hasła, zapewniono, że nowo tworzone konta użytkowników będą posiadały silne, trudne do złamania hasła, jest to podejście chroniące szczególnie przed atakami typu brute force.

```

class RegisterForm(UserCreationForm):
    email = forms.EmailField(required=True, max_length=50)
    username = forms.CharField(required=True, max_length=50)
    password1 = forms.CharField(
        label="Password",
        widget=forms.PasswordInput,
        required=True,
        max_length=100,
        min_length=8,
    )
    password2 = forms.CharField(
        label="Password confirmation",
        widget=forms.PasswordInput,
        required=True,
        max_length=100,
        min_length=8,
    )
    captcha = ReCaptchaField(widget=ReCaptchaV3)

```

W tym przypadku ustawiliśmy parametr `min_length`, tak aby minimalna liczba znaków jakie musi wprowadzić użytkownik do poprawnego utworzenia konta wynosiła 8. Klasa `RegisterForm` dziedziczy po `UserCreationForm`, co pozwala na rozszerzenie standardowego formularza rejestracji. Definiuje ona pola: `email`, `username`, `password1`, `password2`.

```

PeWeX47, 2 months ago | 1 author (PeWeX47)
8 class RegisterForm(UserCreationForm):
9     email = forms.EmailField(required=True, max_length=50)
10    username = forms.CharField(required=True, max_length=50)
11    password1 = forms.CharField(
12        label="Password",
13        widget=forms.PasswordInput,
14        required=True,
15        max_length=100,
16    )
17    password2 = forms.CharField(
18        label="Password confirmation",
19        widget=forms.PasswordInput,
20        required=True,
21        max_length=100,
22    )

```

Pola posiadają ograniczenia maksymalnej długości znaków, blokując możliwość przekazania do bazy danych ogromnych zbiorów tekstu, co mogłoby zaważyć na jej wydajności.

Dodatkowo dane nieprawidłowo wprowadzone do formularza generują komunikaty o błędach.

### Rejestracja

test3


@www@gmail.com

...

...

Zarejestruj się

Posiadasz konto? [Zaloguj się](#)



### Rejestracja

Nazwa użytkownika

Email

Hasło

Powtórz hasło

A user with that username already exists.


Enter a valid email address.

Ensure this value has at least 8 characters (it has 3).

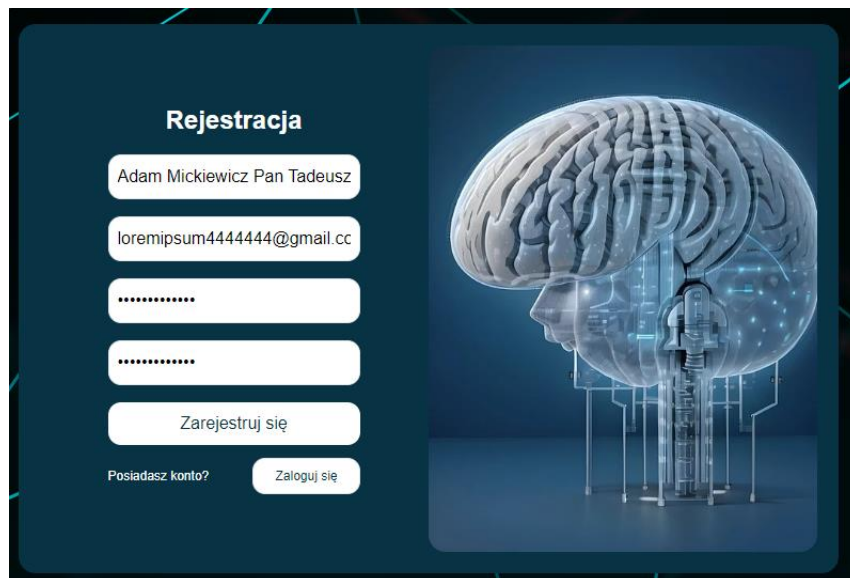
Ensure this value has at least 8 characters (it has 3).

Zarejestruj się

Posiadasz konto? [Zaloguj się](#)







**Rejestracja**

Adam Mickiewicz Pan Tadeusz

loremipsum4444444@gmail.cc

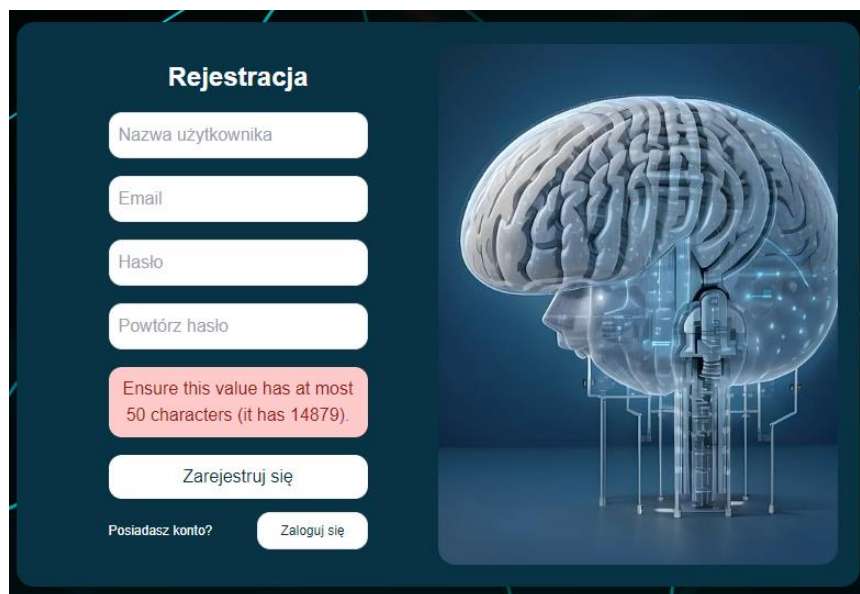
.....

.....

Zarejestruj się

Posiadasz konto?

The illustration shows a human brain in profile, rendered in a translucent blue style. Inside the brain, there is a complex network of glowing blue lines and nodes, resembling a neural network or a futuristic circuitry. The brain is set against a dark blue background with some faint, glowing lines.



**Rejestracja**

Nazwa użytkownika

Email

Hasło

Powtórz hasło

Ensure this value has at most 50 characters (it has 14879).

Zarejestruj się

Posiadasz konto?

The illustration is identical to the one in the first image, showing a translucent blue brain with internal glowing circuitry.

### 2.6.8. Implementacja reCaptcha v3

W projekcie zastosowany został mechanizm reCaptcha v3 jako warstwę zabezpieczeń przed fałszywym ruchem sieciowym. reCaptcha v3 pozwala na automatyczną weryfikację interakcji użytkowników z naszą aplikacją. Ta forma Captcha działa w tle, nie wymaga bezpośredniej interakcji użytkownika z żadnymi testami, co minimalizuje zakłócenia w doświadczeniu użytkownika, jednocześnie zwiększając poziom ochrony przed botami i niepożądanymi działaniami.



```

8  class RegisterForm(UserCreationForm):
9      email = forms.EmailField(required=True, max_length=50)
10     username = forms.CharField(required=True, max_length=50)
11     password1 = forms.CharField(
12         label="Password",
13         widget=forms.PasswordInput,
14         required=True,
15         max_length=100,
16     )
17     password2 = forms.CharField(
18         label="Password confirmation",
19         widget=forms.PasswordInput,
20         required=True,
21         max_length=100,
22     )
23     captcha = ReCaptchaField(widget=ReCaptchaV3)
24

```

Na powyższym zrzucie ekranu widoczne jest tworzenie specjalnego pola formularza `ReCaptchaField`, które wykorzystuje interfejs `ReCaptchaV3` do weryfikacji, czy użytkownik jest człowiekiem, co stanowi dodatkową warstwę zabezpieczeń w aplikacji, zwłaszcza przy formularzach, które wymagają ochrony przed spamem lub działaniami automatycznymi.

#### 2.6.9. Ograniczenie dostępu do wybranych podstron dla niezarejestrowanych użytkowników

Zastosowane zostały ograniczenia dotyczące dostępu do określonego obszaru strony. Podstrona z aplikacją analizującą nacechowanie komentarzy jest niedostępna dla użytkowników, którzy nie są zarejestrowani i uwierzytelnieni. Ta strategia zapewnia ochronę aplikacji przed nieautoryzowanymi wywołaniami modelu sztucznej inteligencji, zwiększając tym samym funkcjonalność dla zarejestrowanych i uwierzytelnionych użytkowników. Dekorator, `@login_required`, pozwala na ograniczenie dostępu do widoku `yt_sentiment` dla zalogowanych użytkowników.

```

33  @login_required(login_url="/login")
34  def yt_sentiment(request):
35      if request.method == "POST":
36          try:
37              text = request.POST.get("text")
38              comments = get_video_comments(text, GOOGLE_API_KEY)
39

```

#### 2.6.10. Zabezpieczenie panelu administratora

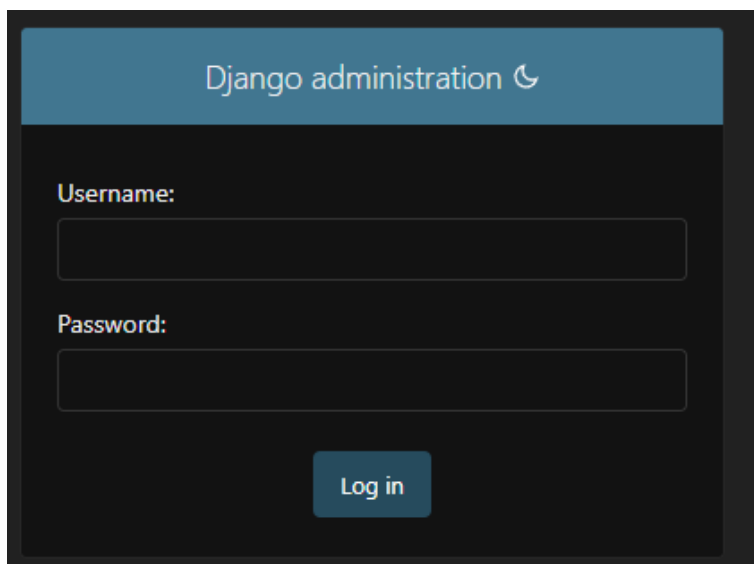
W celu zabezpieczenia panelu administratora, zmieniona została domyślna ścieżka prowadząca do formularza logowania dla użytkowników administracyjnych oraz ustawione zostało silne hasło, którego złamanie jest trudne metodami brute force.

```
path("admin/", admin.site.urls),
```

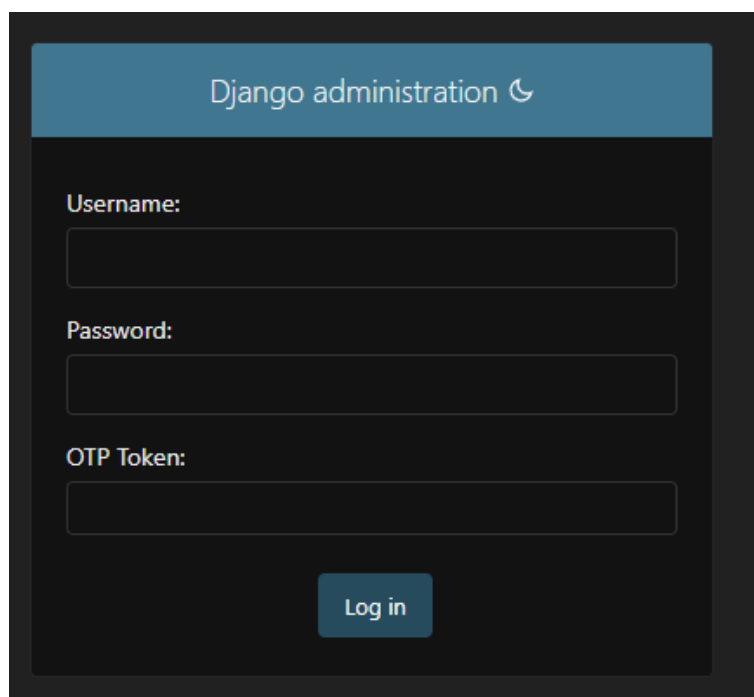
```
path("konstantynopol/", admin.site.urls),
```

Dostęp do panelu administracyjnego zabezpieczony został dodatkowo z wykorzystaniem uwierzytelniania wieloskładnikowego, a dokładnie hasła jednorazowego, generowanego z

wykorzystaniem aplikacji Google Authenticator. Uwierzytelnianie wieloskładnikowe wprowadza dodatkową warstwę zabezpieczeń, wymagając od użytkowników potwierdzenia swojej tożsamości za pomocą czegoś więcej niż tylko tradycyjnego hasła. W tym przypadku, generowane jednorazowe hasła z aplikacji Google Authenticator stanowią dodatkowy element bezpieczeństwa, eliminując ryzyko ataków opartych na przechwyceniu stałego hasła. To podejście podnosi poziom bezpieczeństwa, sprawiając, że dostęp do panelu administracyjnego staje się bardziej odporny na próby nieautoryzowanego dostępu.



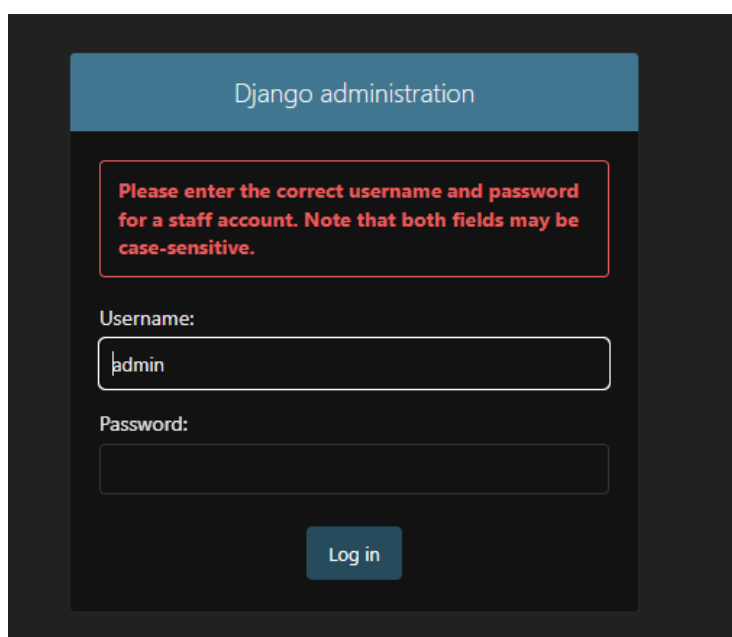
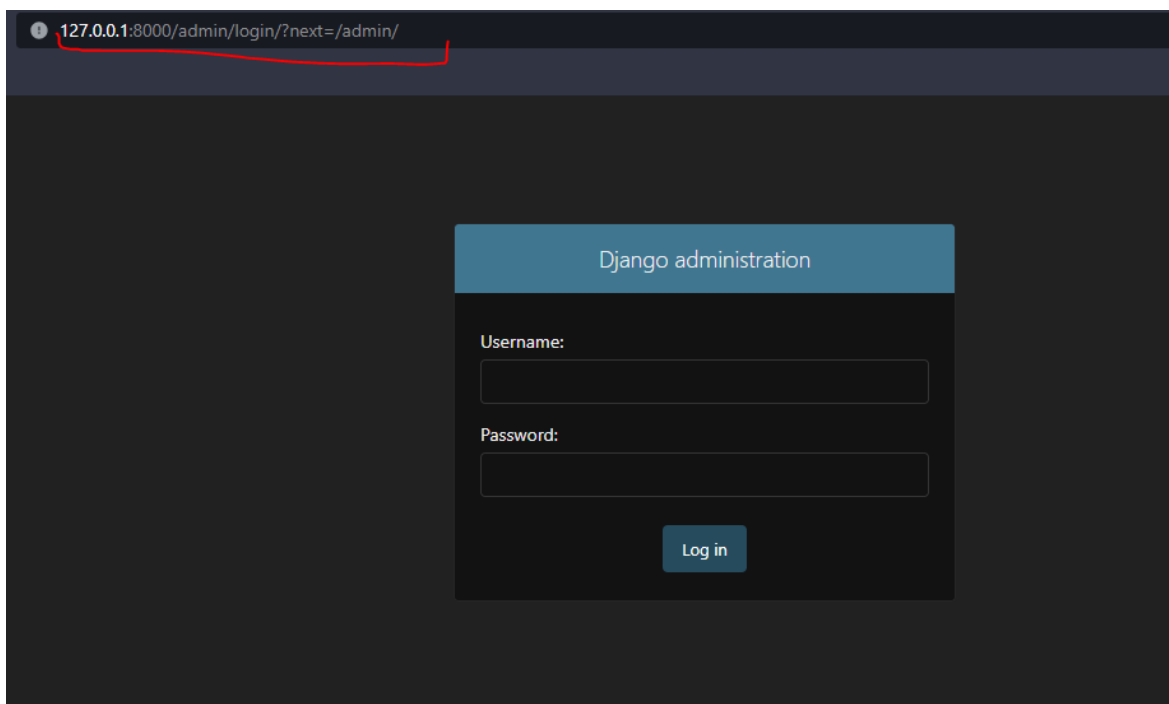
The image shows the Django administration login page. It has a dark blue header with the text "Django administration" and a small icon. Below the header, there are two input fields: "Username:" and "Password:". Below these fields is a blue button labeled "Log in".

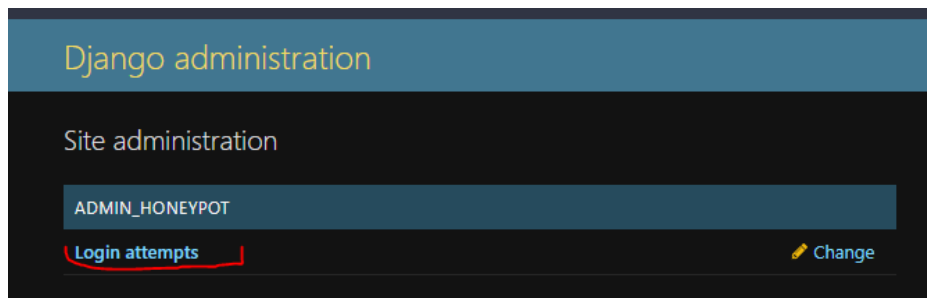


The image shows the Django administration login page with an additional field for OTP Token. It has a dark blue header with the text "Django administration" and a small icon. Below the header, there are three input fields: "Username:", "Password:", and "OTP Token:". Below these fields is a blue button labeled "Log in".



Ostatnim krokiem w zabezpieczeniu formularza logowania do panelu administracyjnego jest utworzenie tzw. honeypot. Jest to fikcyjny panel logowania umieszczony pod domyślnym adresem logowania w Django, który już wcześniej został przez nas zmieniony. Fikcyjny formularz zachowuje się w sposób zbliżony do funkcjonalnego, jednakże nie umożliwia zalogowania się nawet w przypadku podania prawdziwych danych autoryzujących. Dodatkowo honeypot gromadzi logi z prób zalogowania. Logi widoczne są z panelu administratora.





Select login attempt to change

Search

| USERNAME | IP ADDRESS | SESSION                          | TIMESTAMP                 | URL                        |
|----------|------------|----------------------------------|---------------------------|----------------------------|
| sdasd    | 127.0.0.1  | None                             | Dec. 18, 2023, 9:27 a.m.  | /admin/login/?next=/admin/ |
| sds      | 127.0.0.1  | None                             | Dec. 18, 2023, 10:07 a.m. | /admin/login/?next=/admin/ |
| 123      | 127.0.0.1  | lg14ojwkv8f9arjlr27izpt4ni2503y  | Dec. 18, 2023, 10:08 a.m. | /admin/login/?next=/admin/ |
| dfdsf    | 127.0.0.1  | lg14ojwkv8f9arjlr27izpt4ni2503y  | Dec. 18, 2023, 10:12 a.m. | /admin/login/?next=/admin/ |
| admin    | 127.0.0.1  | None                             | Dec. 18, 2023, 10:13 a.m. | /admin/login/?next=/admin/ |
| r2r23    | 127.0.0.1  | btlg6e6p2bty7djl518xzi7dll5r7lvh | Dec. 18, 2023, 11:07 a.m. | /admin/login/?next=/admin/ |
| sadasd   | 127.0.0.1  | None                             | Dec. 18, 2023, 4:39 p.m.  | /admin/login/?next=/admin/ |
| admin    | 127.0.0.1  | None                             | Dec. 19, 2023, 12:24 a.m. | /admin/login/?next=/admin/ |

Change login attempt

**admin**

Path: /admin/login/?next=/admin/

Username: admin

Ip address: 127.0.0.1

Session key: -

User-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36

SAVE Save and continue editing

### 3. Podsumowanie

Podczas realizacji projektu z przedmiotu Aplikacje Internetowe dążyliśmy do stworzenia interaktywnej i responsywnej aplikacji internetowej, skupiając się na świadczeniu usług opartych o modele sztucznej inteligencji dla potrzeb biznesowych. Naszym głównym celem było wykorzystanie zaawansowanych technologii, takich jak Django i Tailwind, aby opracować innowacyjną i responsywną platformę. Jednym z kluczowych elementów była implementacja modelu YT\_Sentiment, umożliwiającego analizę nacechowania komentarzy filmików na platformie YouTube. Ta funkcjonalność stanowiła istotną część naszego projektu, pozwalając użytkownikom na ocenę emocjonalnego wydźwięku komentarzy związanych z filmami. Wdrożony model uzupełnia podejście biznesowe do analizy oraz lepszego zrozumienia, jak treści na platformie

YouTube są odbierane. Do przeprowadzenia analizy wystarczy skopiować i wkleić link do filmu w dedykowane pole.

W naszym projekcie korzystaliśmy z bazy danych PostgreSQL, w której znajdują się encje przechowujące informacje o zarejestrowanych użytkownikach (nazwy, email'e, oraz zaszyfrowane hasła), bazę hostowaliśmy na platformie Render. Dodatkowo, wykorzystaliśmy połączenie z interfejsem API YouTube, umożliwiającym pobieranie danych z platformy w celu dostarczenia ich do naszego modelu. Najpierw, wykorzystując API YouTube, pobieraliśmy dane dotyczące komentarzy filmów, po pobraniu danych z YouTube przeprowadzaliśmy proces tokenizacji, czyli podziału tekstu na mniejsze jednostki zwane tokenami, na podstawie których przeprowadzaliśmy analizę nacechowania komentarzy.

Wyniki analizy są prezentowane w formie wykresu, który obrazuje procentowy rozkład komentarzy według ich nacechowania (pozytywne, negatywne, neutralne). Dodatkowo, udostępniamy tabelę zawierającą szczegółowe wyniki analizy. Istnieje również możliwość pobrania tych danych w formacie pliku .csv po naciśnięciu dedykowanego przycisku znajdującego się na stronie internetowej, dzięki czemu użytkownicy mają szeroki wachlarz opcji prezentacji wyników analizy.

## 4. Literatura

Dokumentacja Django - <https://docs.djangoproject.com/en/5.0/>

Dokumentacja Tailwind CSS - <https://tailwindcss.com/docs>

Dokumentacja Chart.js - <https://www.chartjs.org/docs/latest/>

<https://extension.umaine.edu/plugged-in/technology-marketing-communications/web/tips-for-web-managers/embed-map/>

<https://www.geeksforgeeks.org/csrf-token-in-django/>

<https://www.geeksforgeeks.org/how-to-add-google-recaptcha-to-django-forms/>

<https://docs.djangoproject.com/en/5.0/topics/security/>

[https://medium.com/@ITservices\\_expert/django-security-best-practices-fortifying-your-web-application-da18fa368bf9](https://medium.com/@ITservices_expert/django-security-best-practices-fortifying-your-web-application-da18fa368bf9)

<https://www.w3schools.com/django/>

<https://www.cloudwithdjango.com/multi-factor-authentication-mfa-for-your-django-admin-page/>