**VYTAUTAS MAGNUS UNIVERSITY**
FACULTY OF INFORMATICS
DEPARTMENT OF APPLIED INFORMATICS

# INTERNET TECHNOLOGY (INF3002_EN)

**General Documentation**

**Student:**Dmytro Vodianytskyi IF2100075

**Lab assistant:** Vytautas Barzdaitis

Kaunas, 2023

# Contents

# 1 Users app

The users app is responsible for managing user authentication and user profile functionality for the website. It includes various files like migrations, static, templates, admin.py, apps.py, forms.py, models.py, urls.py, and views.py.

## 1.1 /migrations/

This directory stores the database schema changes made over time through Django's migrate command.

## 1.2 /static/

This directory contains static files, such as CSS, JavaScript, images and so on that are used by the application. In this app, users/static/users directory contains CSS files for user interface, such as google_btn.css, style_register.css, style_ind.css, style_login_sing.css, and style_logout.css.

## 1.3 /templates/

This directory contains HTML templates for the app. It is organized by app name and stores all the HTML templates used by the app. In this app, users/templates/users directory contains HTML templates for user authentication and user profile functionality, such as account.html, logged_out.html, login.html, register.html, register_tampl.html, and singup.html.

## 1.4 /admin.py

This file is responsible for registering models in Django admin panel. In this app, admin.py file registers User model.

## 1.5 /apps.py

This file stores the app configuration. In this app, apps.py defines the UsersConfig class, which is responsible for configuring the app name and default auto field.

## 1.6 /models.py

```
from django.db import models

# Create your models here.
from django.contrib.auth.models import AbstractUser
from django.core.validators import validate_slug
from django.utils.translation import gettext_lazy as _
```

Necessary imports

```
class User(AbstractUser):
username = models.CharField(
    verbose_name="Username",
    max_length=150,
    unique=True,
    validators=[validate_slug],
    error_messages={
        'unique': _("A user with that username already exists."),
    },
)

email = models.EmailField(verbose_name='E-mail', blank=True,
                            unique=True, null=True)
email_checked = models.DateTimeField(blank=True, null=True)

def __str__(self):
    #return '%s (%s)' % (self.get_full_name(), self.email)
    return '%s (%s)' % (self.username, self.email)
```

This file defines the data models used by the app. In this app, `models.py` defines the `User` model that extends Django's built-in `AbstractUser` class. The model contains fields like `username`, `email`, `email_checked` and `__str__` method.

## 1.7 /forms.py

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

from .models import User
```

Necessary imports

```
class LoginForm(AuthenticationForm):
    pass

class SignupForm(UserCreationForm):
    class Meta:
        model = User
        fields = ("username", "email")
        field_classes = {'username': UsernameField}
```

This file contains Django forms used for the app. In this app, `forms.py` defines the `LoginForm` and `SignupForm` classes that extend Django's built-in `AuthenticationForm` and `UserCreationForm` classes respectively. These forms are used to handle user login and registration functionality.

## 1.8 /urls.py

```
from django.urls import path, include
from .views import UserLoginView, SignupView, UserLogoutView

from . import views
```

Necessary imports

```
app_name = 'users'

urlpatterns = [
    path('',views.index, name= 'index'),
    path('account/', views.account, name= 'account'),
    path(r'login/', UserLoginView.as_view(), name= 'login'),
    path(r'logout/', UserLogoutView.as_view(), name= 'logout'),
    path(r'singup/', SignupView.as_view(), name= 'singup'),
    path('social-auth/', include('social_django.urls', namespace='social'))
]
```

This file contains the app's URL patterns. In this app, `urls.py` defines the URL patterns for views like `index`, `account`, `login`, `logout`, `singup`, and `social-auth`.

## 1.9 /views.py

```
from django.contrib.auth.views import LoginView, LogoutView
from django.shortcuts import render

from django.contrib.auth import login
from django.views.generic import FormView
from django.http import HttpResponseRedirect
from django.urls import reverse

from .forms import LoginForm, SignupForm
from .models import User

from polls.models import Vote, Question, Choice
```

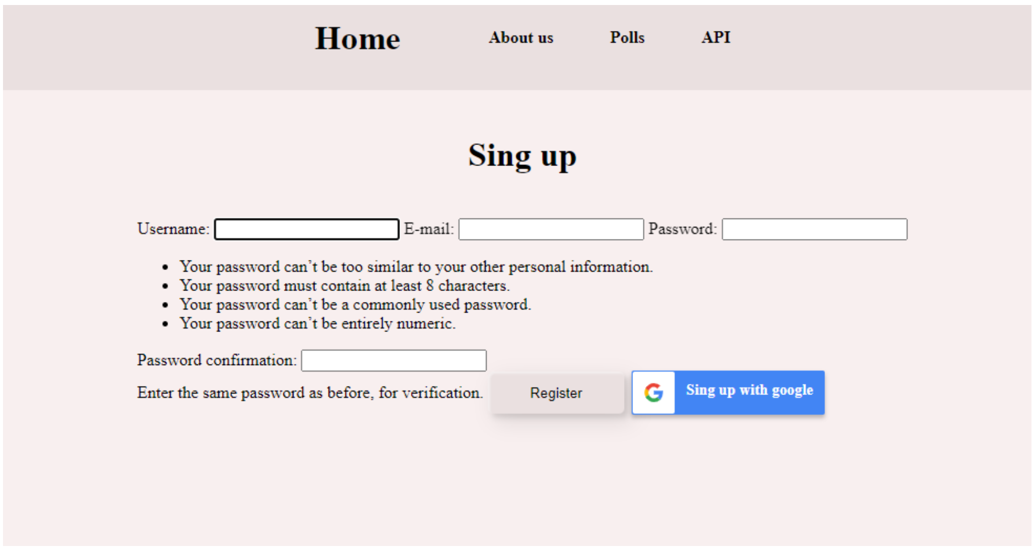
Necessary imports

```
class SignupView(FormView):
template_name = 'users/singup.html'
form_class = SignupForm

def form_valid(self, form):
    username = form.cleaned_data['username']
    email = form.cleaned_data['email']
    raw_password = form.cleaned_data['password1']
    user = User.objects.create_user(username, email, raw_password)
    login(self.request, user)

    return HttpResponseRedirect(reverse('main_page:index'))
```

`SignupView`: This view is responsible for rendering the signup form and creating a new user when the form is submitted. It inherits from `FormView` and uses `SignupForm` as its form class. When the form is valid, the method `form_valid` is called, which creates a new user with the information provided by the user in the form. It then logs in the user using the `login` method provided by Django's `auth` module and redirects the user to the index page. The template used by this view is `users/signup.html`.



Sing up

Username: E-mail: Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

Figure 1: Screenshot of the sign up page

```
class UserLoginView(LoginView):
    form_class = LoginForm
    template_name = 'users/login.html'
```

`UserLoginView`: This view is responsible for rendering the login form and logging in the user when the form is submitted. It inherits from `LoginView` and uses `LoginForm` as its form class. The template used by this view is `users/login.html`.

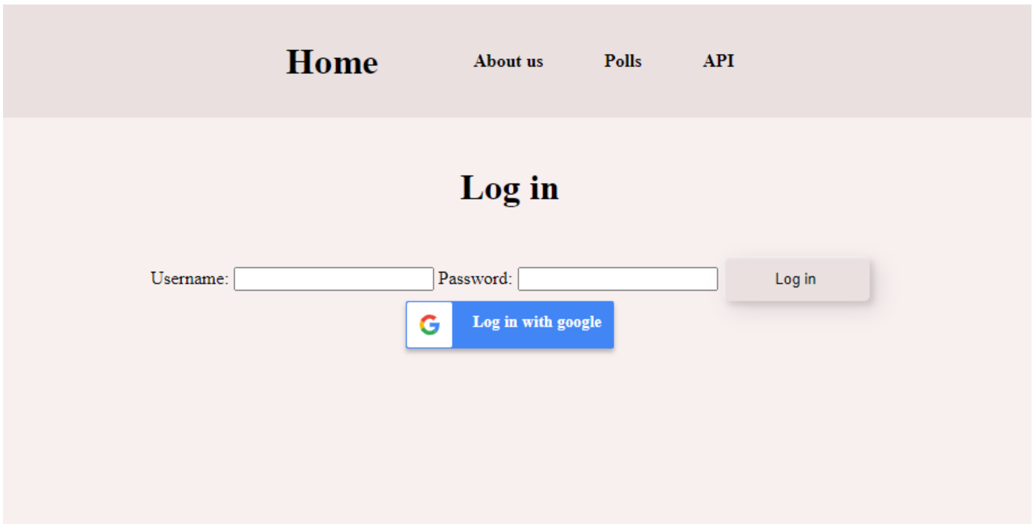


Figure 2: Screenshot of the log in page

```
class UserLogoutView(LogoutView):
    template_name = 'users/logged_out.html'
```

`UserLogoutView`: This view is responsible for logging out the user and rendering the logged-out page. It inherits from `LogoutView`. The template used by this view is `users/logged_out.html`.

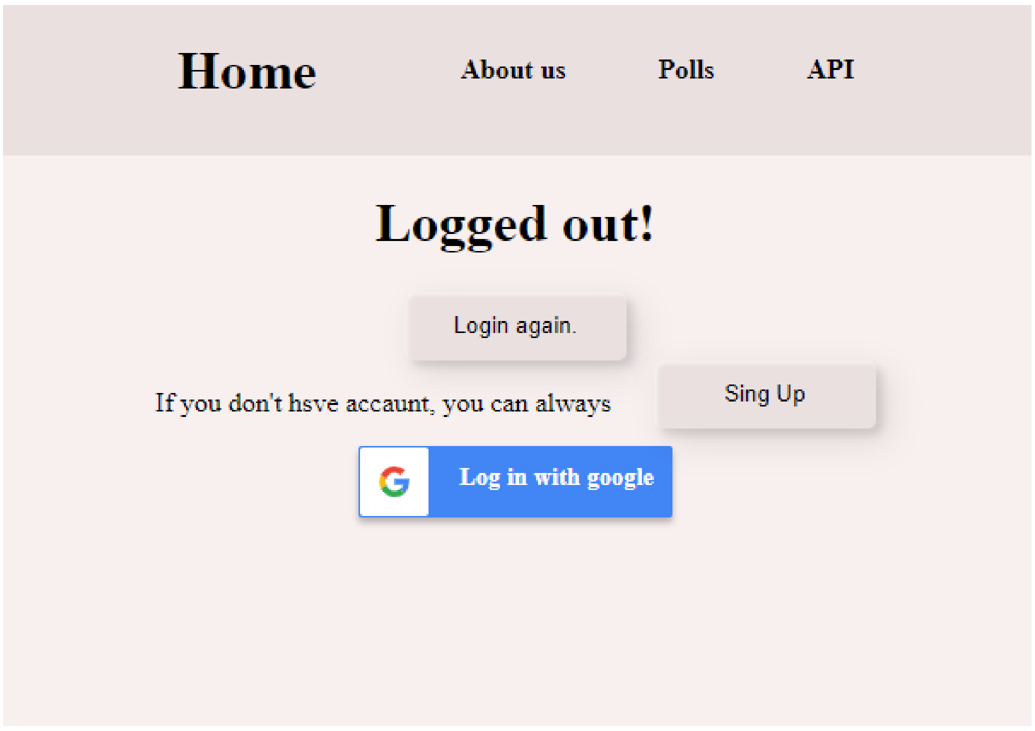


Figure 3: Screenshot of the log out page

```
    def account(request):
        vote = Vote.objects.filter(user= request.user)
        return render(request, 'users/account.html', {'vote' : vote})
```

`account`: This view is responsible for rendering the user account page. It first retrieves all the votes that the user has cast by querying the `Vote` model using the current user as a filter. It then renders the `users/account.html` template, passing the retrieved votes to the template as a context variable.