Timebox 1- Login

Oversigt

OpgaveNavn	Implementering af bruger login		
Implementering af krav	Delvis implementering af krav WEB-3		
Udført af	Jan	**Dato**	21-09-2021
Timebox	1	**Område**	Website

Contents

Introduktion

Denne opgave beskriver analyse, design og implementering af login for bruger på hjemmesiden.

Analyse

Django indeholder templates for både login og logout, samt python dekorater funktioner for tjek om en bruger er logget ind, for at kunne tilgå en side.

For at kunne bruge Djangos login templates, vil der skulle laves forskellige filer og ændringer i Django projektet:

1. Login page (template filer (HTML)).
    - For at viser noget på siden

2. Login page funktioner (View funktioner).
    - For at kunne omdirigere brugere, baseret på deres bruger typer
        - Admin
        - Office user
        - Field user

3. Ændringer til databasen ”user” tabel
    - For at kunne linke den sammen med Djangos templates for login og logout.

4. En *managers.py* fil.
    - for at kunne kreere egen definerede bruger typer / ændre de prædefinerede bruger typer.
    - Filen er er hvorledes sammenkædningen mellem Querys til databasen og Djangos templates hænger sammen.

5. *settings.py* skal ændres til at bruge en ”custom” login/logout.

Punkt 1 (Login page):

- Der skal være en baggrund, der indikerer Eurowind.
- Der skal være et username felt.
- Der skal være et password felt.
- Der skal være en knap for ”Login”.
- Der skal være en knap for ”Create user”.

Punkt 2 (Login funktioner i *views.py*):

- I hensyn til selve loginfunktionerne kan Djangos standard template bruges, under URL’en ”/accounts/login”.
- Omdirigeringen for brugere, kan gøres på deres userType felt fra Users tabellen i databasen, med 3 ”if statements”.
- For at bruge login siden som ”homepage”, kan der sættes at URL’en ”/”, automatisk omdirigerer til URL’en ”/accounts/login”.

Punkt 3 (Ændringer til databasens Users tabel i *forms.py*):

- Tabellen skal linkes sammen med Djangos bruger tabeller.
- ”Username” feltet for login skal sættes til at bruge Email, i stedet for brugernavn.
- ”Required” felter der skal udfyldes når en bruger bliver lavet en ny bruger skal sættes.

- Der skal sættes, at en ny bruger som standard er ”field user”, og ikke har admin rettigheder flagene som standard.
- Der skal sættes til at bruge en ”custom user manager”, til at håndtere den nye login ”måde”, gennem et ”CustomUserManager()” objekt.
- Email feltet i User tabellen skal sættes til at de altid skal være unikke.

Punkt 4 (*managers.py* fil):

- Der skal laves en custom managers klasse til ny bruger oprettelse med 2 funktioner inde i klassen
    - ”Create_user” funktionen, sætter hvilke database querys der skal laves ved en ”normal” bruger oprettelse (field user), og gemmer dem i databasen.
    - ”Create_superuser” funktionen, sætter hvilke database querys der skal laves ved en ”admin” bruger oprettelse (admin user), og gemmer dem i databasen, samt sætter nogle ”permission” flag som default.

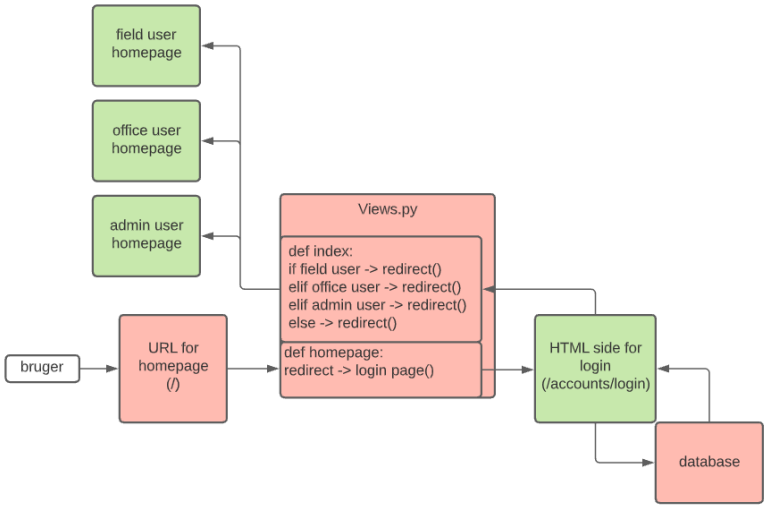
Punkt 5 (ændre *settings.py*):

- Indsæt LOGIN_REDIRECT_URL
- Indsæt LOGOU_REDIRECT_URL

Design

De indsatte diagrammer, vil kun vise hvad der aktivt er ændret af os, da der bruges Djangos template funktioner og filer, vil der være diverse mellemled, som der ikke er blevet ændret, og kører igennem Djangos standard opsætning.

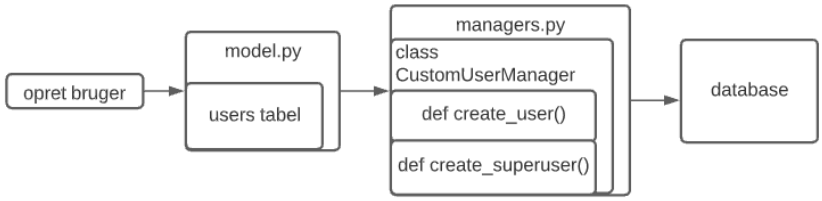
Ved login vil en bruger kun kunne se de grønne felter på Figur 1, hvor de røde felter er hvad der vil køre i backenden. Diagrammet viser hvordan nogle af de forskellige punkter under analysen hænger sammen.



Figur 1 login

Diagramet på Figur 2 viser hvordan nogle af de nævnt filer under analysen bliver brugt.

Ved bruger oprettelse vil modellen blive brugt, for derefter at hive CustomUserManageren ind, og så gemme det i databasen.



Figur 2 opret bruger

Implementering

Dette afsnit vil bruge de samme punkter som listet under analysen.

Punkt 1 (Login page):

- Der skal være en baggrund, der indikerer Eurowind.

Som der kan ses på Figur 3, bliver der hentet en baggrund fra en ”static” mappe, som er sat til at fylde hele baggrunden af hjemme siden

```
<head>
    <link rel="stylesheet" href="{% static 'website/style.css' %}">
    <style>
        body {
            background-image: url('{% static "website/EWE-baggrundsbillede-website.png" %}');
            background-repeat: no-repeat;
            background-attachment: fixed;
            background-size: 100% 100%;
        }
    </style>
    <meta charset="UTF-8">
    <title>login</title>
</head>
```

Figur 3 html baggrund

- Der skal være et username felt.

Som der kan ses på Figur 4, bliver hentet Djangos form for bruger login, hvor input feltet vil være formens username, som er blevet ændret til at skulle være en email.

```
<div class="form-group">

    <label for="{{ form.username.id_for_label }}"
           style="...">Email:</label>
    <br>
    {{ form.username }}
    {{ form.username.errors }}

</div>
```

Figur 4 html user name

- Der skal være et password felt.

Som der kan ses på Figur 5, bliver hentet Djangos form for bruger login, hvor input feltet vil være formens password.

```
<div class="form-group">

    <label for="{{ form.password.id_for_label }}"
    style="...">Password:</label>
    <br>
    {{ form.password }}
    {{ form.password.errors }}
</div>
```

Figur 5 html password

- Der skal være en knap for ”Login”.
- Der skal være en knap for ”Create user”.

Som der kan ses på Figur 6, er der lavet knapper til både ”login” og ”create user”.

```
<div>
    <br>
<button type="submit" style="..." >Log in</button>
</div>
<br>
<br>
<br>
<div>
<button onClick="document.location.href = '../../create_user'" style="...">Create user</button>
</div>
```

Figur 6 html knapper

Punkt 2 (Login funktioner i *views.py*):

Som set udefra Figur 8, vil funktionen fra Figur 7, blive brugt hvis der bliver gået ind på URL’en: ’ ’, som vil omdirigere brugeren til URL’en: ’/accounts/login/’, hvorefter funktion på Figur 9, vil videre omdirigere en, baseret på brugerens ”userType”.

```
def homepage(request):
    return redirect('/accounts/login')
```

Figur 7 homepage redirect

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),
    path('login_redirect/', views.index),
    path('create_user/', include('create_user.urls')),
    path('field_user_home/', include('field_user.urls')),
    path('', views.homepage),
    path('office_user_home/', include('office_user.urls')),
    #path('rest_api/', include('rest_api.urls')),
]
```

Figur 8 URL's

```
@login_required
def index(request):
    usertype = request.user.userType

    if usertype == 'Field user':
        return redirect('/field_user_home')
    elif usertype == 'Office user':
        return redirect('/office_user_home')
    elif usertype == 'Admin':
        return redirect('/admin')
    else:
        return redirect('')
```

Figur 9 omdirigering

Punkt 3 (Ændringer til databasens Users tabel i *forms.py*):

Klassen til at lave Users tabellen, vil ikke længere hente fra Djangos ”models.Model”, men bruge Djangos AbstractBaseUser og PermissionsMixin modeller, som set på Figur 10:

- Brug af AbstractBaseUser modellen, fortæller Django at der skal bruges en custom bruger model, i stedet for Djangos egen model.

- Brug af PermissionsMixin, gør det muligt at bruge Djangos permission flag, som fx is_superuser der fortæller at bruger er admin bruger, samt is_staff der fortæller at brugeren har tilladelse til at bruge admin siden.

```
"For Users:"
class Users(AbstractBaseUser, PermissionsMixin):
```

Figur 10 User model imports

Feltet for email er sat til at skulle være unik, som set på Figur 11.

```
email = models.EmailField(_('email adress'), unique=True)
```

Figur 11 email felt

De forskellige flag for ”staff” og ”admin” er som standard sat til ”False”, som set på Figur 12, samt er der sat et felt, for at kunne se hvornår en bruger er oprettet.

```
is_staff = models.BooleanField(default=False)
is_superuser = models.BooleanField(default=False)
date_joined = models.DateTimeField(default=timezone.now)
```

Figur 12 flag og date_joined

Brugernavn’s feltet, samt hvilke felter der skal udfyldes ved bruger oprettelse er blevet specificeret, som set på Figur 13.

```
#set email to be the login "name"
USERNAME_FIELD = 'email'
REQUIRED_FIELDS = ['name', 'phoneNumber', 'company']
```

Figur 13 username + required fields

For at bruge den nye custom manager, bliver der gemt en instans af manageren, som set på Figur 14.

```
objects = CustomUserManager()
```

Figur 14 CustomUserManager kald

Punkt 4 (*managers.py* fil):
Som set på Figur 15, bliver email’en og de ”required” felter fra Figur 13, sat ind som bruger oplysninger, for til sidst at blive gemt i databasen, vedhjælp af ”user.save()” funktionen.

```
def create_user(self, email, password, phoneNumber, company, name, **extra_fields):
    """
    create and save users with the given email and password.

    :param Email:
    :param password:
    :param extra_fields:
    :return:
    """

    email = self.normalize_email(email)
    user = self.model(email=email,
                      name=name,
                      phoneNumber=phoneNumber,
                      company=company,
                      **extra_fields)
    user.set_password(password)
    user.save()
    return user
```

Figur 15 create_user managers.py

På Figur 16, kan der ses at flagene fra Figur 12, sættes til at være "True", da det der oprettes her en en admin burger, samt bliver der tjekket om de 2 nødvendige flag er blevet sat til "True", og hvis dette ikke er tilfældet, vil der blive givet en fejl.

```python
def create_superuser(self, email, password, **extra_fields):
    """
    create and save a SuperUser with the given email and password.

    :param self:
    :param email:
    :param password:
    :param extra_fields:
    :return:
    """
    extra_fields.setdefault('is_staff', True)
    extra_fields.setdefault('is_superuser', True)

    if extra_fields.get('is_staff') is not True:
        raise ValueError(_('Superuser must have is_staff=True'))
    if extra_fields.get('is_superuser') is not True:
        raise ValueError(_('Superuser must have is_superuser=True'))
    return self.create_user(email, password, **extra_fields)
```

Figur 16 create_superuser managers.py

Punkt 5 (ændre *settings.py*)

I *settings.py*, vil der skulle sættes, hvilken URL det nye custom login skal omdirigere hen til, og hvor Djangos logout template skal dirigere en hen, som kan ses på Figur 17.

```python
LOGIN_REDIRECT_URL = '/login_redirect/'
LOGOUT_REDIRECT_URL = '/'
```

Figur 17 settings.py

Verifikation

For at verificere at de nye ændringer er indført og virker vil der blive testet:

- Der kan logges ind på den nye loginpage.
- Ved login skal der bruges email, i stedet for brugernavn.
- Der blive omdirigeret hen til de korrekte steder.
    - Field user login -> field user homepage.
    - Office user login -> office user homepage.
    - Admin user login -> admin user homepage.
    - Create user button -> create user homepage.
    - Homepage -> login page.
- Ved oprettelse af en ny bruger, skal de satte nødvendige felter udfyldes.
- I tilfælde af at der tilgås en side hvor en bruger skal være logget ind, uden at være logget ind, vil der blive dirigeret hen til login siden.
- Når der bliver logget ud, bliver der omdirigeret til login siden.

Tabel 1: Tests til verifikation af opgave

Test	Test Steps	Pre-requisites	Pass-betingelser	Resultat
Homepage til login page	1. Gå til hjemmesidens homepage	• Serveren er oppe at køre	• Der bliver omdirigeret til loginpagen	Bestået
Login bruger omdirigering + log ud	1. Login på en office user 2. Log ud 3. Login på en field user 4. Log ud 5. Login på en admin user 6. Log ud	• Serveren er oppe at køre • Homepages for de forskellige brugere er lavet • Der er oprettet forskellige bruger typer	Der bliver dirigeret hen til de korrekte steder: • Field user login -> field user homepage • Office user login -> office user homepage • Admin user login -> admin user homepage • Log ud -> login page	Bestået
Bruger oprettelse	1. Login som admin 2. gå ind på users tabellen 3. tryk på ”add users” 4. tryk på save 5. udfyld felter 6. tryk på save	• Serveren er oppe at køre • Der er lavet en admin bruger	• Uden udfyldning skal der gives en ”error” melding ved de felter der skal udfyldes. • Med udfyldning vil der blive oprette en ny bruger.	Bestået
Login krævet	1. gå ind på en bruger homepage, uden at være logget ind.	• Serveren er oppe at køre • Der er logget ud som bruger	• Der bliver dirigeret hen til login page.	Bestået

Testresultat

Alle testene er passeret succesfuldt.

Konklusion

Der er mange forskellige dele der blevet hivet fat i, for at få funktionaliteterne i dette dokument til at virke, og der er kun blevet undersøgt de filer der skal aktivt ændres i, hvilket er forstået til et niveau, hvor bruger modellen kan ændres, samt oprette et brugerdefineret login.