

# CS350 Milestone 5

## Design

Evan Duffield

# Design Pattern Catalog

- Keep requirements.txt updated

```
≡ requirements.txt X
Application > ≡ requirements.txt
1  black
2  django
3  django-crispy-forms
4  crispy-bootstrap4
5  asgiref
6  dj-database-url
7  gunicorn
8  psycopg2-binary
9  pytz
10 markdown
11 requests
12 pillow
13 sqlparse
14 whitenoise
15
```

# Design Pattern Catalog

- Separate local imports from Django modules

```
views_classes.py • views_functions.py
Application > helloapp > views_classes.py > ...
1  from django.shortcuts import render
2  from django.views import View
3  from .models import AppUser, Profile, User, Experience,
4  from django.contrib.auth.forms import UserCreationForm
5  from django.contrib.auth.mixins import LoginRequiredMixin
6  import helloapp.views_functions as vf
7  from django.urls import reverse_lazy
8  from django.shortcuts import get_object_or_404
9  from django.views.generic import CreateView, DeleteView,
10  from .forms import ProfilePictureForm
11
12  # User Views
13
```

```
views_classes.py • views_functions.py
Application > helloapp > views_classes.py > ...
1  from django.contrib.auth.forms import UserCreationForm
2  from django.contrib.auth.mixins import LoginRequiredMixin
3  from django.shortcuts import get_object_or_404, render
4  from django.urls import reverse_lazy
5  from django.views import View
6  from django.views.generic import CreateView, DeleteView, De
7
8  from .forms import ProfilePictureForm
9  from .models import AppUser, Profile, User, Experience, Edu
10  import helloapp.views_functions as vf
11  # User Views
```

# Design Pattern Catalog

- Maintain Django's Model-View-Template Design Pattern

```
class AppUser(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    accountCreationDate = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f'{self.user.username}'

    def get_absolute_url(self):
```

```
class UserAddView(CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy('login')
    template_name = 'registration/account_add.html'

class UserUpdateView(vf.IsUserRecordMixin, UpdateView):
    template_name = "user/edit.html"
    model = AppUser
    fields = '__all__'
    success_url = reverse_lazy('profile_list')
```

```
1  {% extends 'theme.html' %}
2
3  {% block title %}
4  Edit Profile
5  {% endblock title %}
6
7  {% block content %}
8  <html lang="en">
9
10 <head>
11     {% load static %}
12     <meta charset="UTF-8">
13     <meta name="viewport" content="width=device-width, initial-scale=1">
14     <title>BridgeOut - Edit Profile</title>
15     <link rel="stylesheet" href="{% static 'css/styles.css' %}">
16 </head>
17
```

# Refactoring

- Inline Variable

```
Application > helloapp > views_classes.py > ...  
from django.contrib.auth.forms import UserCreationForm  
from django.contrib.auth.mixins import LoginRequiredMixin  
from django.shortcuts import render  
  
from django.urls import reverse_lazy  
from django.shortcuts import get_object_or_404  
from django.views import View  
from django.views.generic import CreateView, DeleteView, Det
```

```
views_classes.py • views_functions.py  
Application > helloapp > views_classes.py > ...  
1 from django.contrib.auth.forms import UserCreationForm  
2 from django.contrib.auth.mixins import LoginRequiredMixin  
3 from django.shortcuts import get_object_or_404, render  
4 from django.urls import reverse_lazy  
5 from django.views import View  
6 from django.views.generic import CreateView, DeleteView, De  
7  
8 from .forms import ProfilePictureForm  
9 from .models import AppUser, Profile, User, Experience, Edu  
10 import helloapp.views_functions as vf  
11 # User Views
```

# Refactoring

- Inline Variable

```
def dispatch(self, request, *args, **kwargs):  
    user_test_result = self.get_test_func()  
    if not user_test_result:  
        return self.handle_no_permission()  
    return super().dispatch(request, *args, **kwargs)
```

```
def dispatch(self, request, *args, **kwargs):  
    if not self.get_test_func():  
        return self.handle_no_permission()  
    return super().dispatch(request, *args, **kwargs)
```

# Refactoring

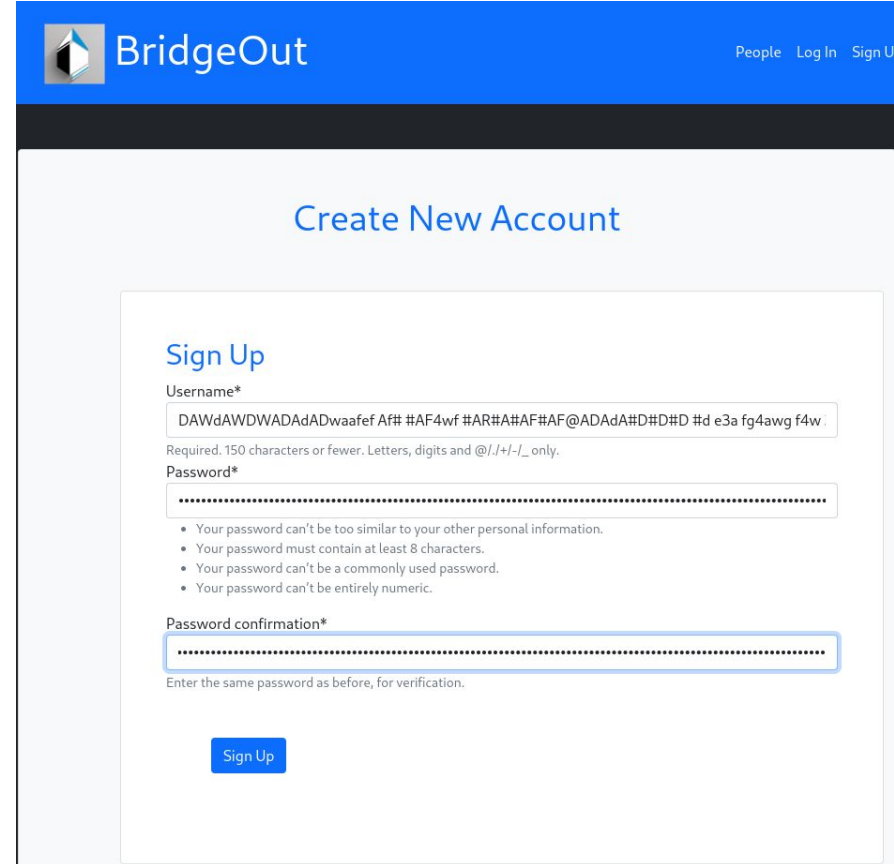
- Rename Method

```
class IsUserProfileFromExpMixin(AccessMixin):  
    def test_func(self):  
        experience = get_object_or_404(Experience, pk =  
        return self.request.user == experience.profile.a
```

```
class IsUserProfileFromExperienceMixin(AccessMixin):  
    def test_func(self):  
        experience = get_object_or_404(Experience, pk =  
        return self.request.user == experience.profile.a  
  
    def get_test_func(self):
```

# Help with Implementation

- Tested the Application's major features.
- Tested field limits.



The screenshot shows the BridgeOut website's sign-up page. The header is blue with the BridgeOut logo and navigation links for 'People', 'Log In', and 'Sign Up'. The main heading is 'Create New Account'. Below it, the 'Sign Up' section contains a 'Username\*' field with a long alphanumeric string, a 'Password\*' field with a masked password, and a 'Password confirmation\*' field with a masked password. A list of password requirements is provided between the password fields. A 'Sign Up' button is at the bottom.

BridgeOut

People Log In Sign Up

## Create New Account

### Sign Up

Username\*

DAWdAWDWADAdADwaafef Af# #AF4wf #AR#A#AF#AF@ADAdA#D#D#D #d e3a fg4awg f4w.

Required: 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

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.

Sign Up



# Help with Implementation

- Helped secure the server for production.
  - Changed DEBUG to False in the settings.py to hide version data of Django from attackers.
  - Made sure HTTP/HTTPS were the only open ports on the web server.

```
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False

ALLOWED_HOSTS = [
    '*',
]

CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

```
Nmap scan report for 162.159.140.98
Host is up (0.0041s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Cloudflare http proxy
443/tcp   open  ssl/https    cloudflare
8080/tcp  open  http         Cloudflare http proxy
8443/tcp  open  ssl/https-alt cloudflare

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.38 seconds
user1@bricked: ~$
```

# AI Prompts

- What are some design patterns that can help organize a large Django project?
- What are some examples of design patterns?
- How would you refactor this method?
- What is the standard for storing local modules in a django project?
- What could this method be renamed to make it more maintainable?
- Give me some strings to test input limits on my website.

## Next Steps

- Finish Milestone 6

Thanks for Watching!