# Workshop: Petstagram

This document contains the fifth and final part of the Petstagram Workshop. First, we will add a functionality to send each newly registered user a **greeting email**. Then, we will add **pagination on the profile details page**. Next, we will **test the app**. And finally, we will **deploy using Docker and Amazon**.

The full project description can be found in the [**Workshop Description Document**](http://svn.softuni.org/admin/svn/python-web/Sept-2022/Python-Web-Basics/06-Workshop-Part-1/06-Workshop-Description.docx).

## Workshop - Part 5.1

### Send "Registration greetings" Email

When **new user is registered** on the site, we want them to receive an **automatic greetings email**. We have the **template in this workshop resources**. It is an email template and we should **only change the username** (of the receiver) on it:



To implement the functionality we can **use a signal**. Let us add a **signals.py** file in our **accounts** app: Graphical user interface

Description automatically generated

We want to **invoke the signal just after the profile is saved**. The signal checks if the **user is created**, and if so - **creates the email with the given template and sends it to them**:

from django.core.mail import send\_mail  
from django.db.models.signals import post\_save  
from django.dispatch import receiver  
from django.template.loader import render\_to\_string  
from django.utils.html import strip\_tags  
  
from petstagram import settings  
from petstagram.accounts.models import PetstagramUser  
  
  
@receiver(signal=post\_save, sender=PetstagramUser)  
def send\_greeting\_email(sender, instance, created, \*\*kwargs):  
 if created:  
 subject = "Registration greetings"  
 html\_message = render\_to\_string('email-greeting.html', {'profile': instance})  
 plain\_message = strip\_tags(html\_message)  
 to = instance.email  
 send\_mail(  
 subject=subject,  
 message=plain\_message,  
 from\_email=settings.EMAIL\_HOST\_USER,  
 recipient\_list=[to], html\_message=html\_message  
 )

In the **setting.py** file, we should add the **email credentials**. In this workshop, we will be using **Gmail** **account**. To use it we need to set that **the email uses TLS**, the host is "**smtp.gmail.com**", the Gmail port SMTP port is **587**, the email host user is **our email address**, the password is an **automatically generated password by Gmail** when we add a "third-party apps with account access": Text, application

Description automatically generated

### Add Pagination

The next thing we want to add to our app is a pagination on the profile details page. We want to **show only 2 images per page**. Open the **account/views.py** file and we can **add the pagination** to our context: Graphical user interface, text, application

Description automatically generated

Now, let us open the **profile-page-page.html** template and **add the pagination code**:

*...*

*<!-- Start Last Uploaded Photos Section -->*<div class="pet-photos">  
  
 *<!-- Link to Last Uploaded Pet Photo -->* **{% for photo in page\_obj.object\_list %}**  
 <a href="{% url 'photo-details' photo.id %}">  
 *<!-- Pet Photo -->* <img src="{{ photo.photo.url }}"  
 alt="pet img">  
 </a>  
 **{% endfor %}**

**<div class="pagination">  
 <span class="step-links">  
 {% if page\_obj.has\_previous %}  
 <a href="?page=1">&laquo; first</a>  
 <a href="?page={{ page\_obj.previous\_page\_number }}">previous</a>  
 {% endif %}  
  
 <span class="current">  
 Page {{ page\_obj.number }} of {{ page\_obj.paginator.num\_pages }}.  
 </span>  
  
 {% if page\_obj.has\_next %}  
 <a href="?page={{ page\_obj.next\_page\_number }}">next</a>  
 <a href="?page={{ page\_obj.paginator.num\_pages }}">last &raquo;</a>  
 {% endif %}  
 </span>  
 </div>**</div>  
  
*<!-- End Last Uploaded Photos Section -->*

*...*

## Workshop - Part 5.2

### Testing

Test the forms, models and views in the **petstagram/photos** app (but not any libraries or functionality provided as part of Python or Django).

## Workshop - Part 5.3

### Deploy

Deploy the project using AWS and Docker.