Django Backend Code Challenge

This challenge consists in creating an email template system using only Django models and the Django admin application. This application needs to generate dynamic emails based on the user associated with the email.

Required Behavior

The application needs at least 2 Django models to work. In addition, you should use the Django auth application to manage users and permissions.

Email Templates:

- Django CRUD System to manage email templates. The required information to store in the template is:
  - Subject
  - Content
  - Attachments (Not Required)

Email:

- Django CRUD System to manage email. The required information to store in the email is:
  - Related User
  - Related Template
  - String for "from email"
  - Subject
  - Content
  - Attachments (Not Required)
- It is not necessary to send an email, just create them.
- Every time a new email is created, it should save the user email in the field "from_email" of the email instance.
- Whenever a new email is created, it should render the Email Template Content into the actual email content.
- Whenever a new email is created, it should generate the Email Template Subject into the real email subject.
- The context to use when rendering the email should contain the following data: (Hint: see Template and Context classes from Django templates app)
  - user: User instance
  - timestamp: A timestamp when the email was created
  - inverted_name: "last_name, first_name" of the user
- (Not Required) Every time a new email is created, it should copy the email template attachments and relate them to the newly created email.

Constraints

- Don't create HTML templates. Use only the Django admin interface and Django models.
- Don't send the email.
- The content of the message should have a max capacity of 4095 characters
- The subjects should have a max capacity of 1023 characters
- Only superusers should be able to create email templates.
- Staff users should not be able to create email templates, but they should be able to create an email using a template.

Bonus (Not required):

- Use raw_id_fields for the Django admin forms.
- Also, save the generated context as a JSON string in the email model.
- Staff users can see templates but not edit them.
- Create an AdminTabularInline of Emails, and use it in the Email Template Admin class to show previously created emails that use the template.
- Attachments are not required but are desired. Use another model for these.

Points to evaluate

- File architecture
- Models Composition
- Django Admin Compositions
- User Permissions
- Email Generation

Delivery

There is no time constraint for the challenge, please send the challenge code in a personal GitHub repository with instructions to run it. ***BIG PLUS*** *if you also deliver the challenge deployed in a free hosting environment using docker. (Not required)*

Application Test

Create an email template with the following data:

> **Subject:** *{{ user.get_full_name }}, welcome to the new Python Course!*
>
> **Content:**
> *This email is intended for {{ user.inverted_name }},*
>
> *We are pleased to let you know that you have been selected to participate in our new Python Course. Please send us your information and documents to complete the registration process to continue with the course. In addition, we have attached the course syllabus to this email.*
>
> *You have seven days from now ({{ timestamp }}) to complete the registration.*
>
> *Greetings from Python School!*
> *ATTACHMENTS RELATED TO TEMPLATE*

Expected result:

Create an email using that template for a user named “Juan Pérez”.

> **Subject:** *Juan Pérez, welcome to the new Python Course!*
>
> **Content:**
> *This email is intended for Pérez, Juan.*
>
> *We are pleased to let you know that you have been selected to participate in our new Python Course. Please send us your information and documents to complete the registration process to continue with the course. In addition, we have attached the course syllabus to this email.*
>
> *You have seven days from now (2022-01-01 00:00:00) to complete the registration.*
>
> *Greetings from Python School!*
> *ATTACHMENTS RELATED TO EMAIL*

The content of the timestamp would be dynamic based on the creation date of the email.