Hyperiondev

Task

Capstone Project – News Application

Visit our website

Introduction

You have already explored user login and authentication within the Django framework, as well as customising users and managing them through user groups and permissions. Additionally, you have gained experience in utilising Django's built-in email system for sending messages, integrating third-party APIs, and writing unit tests for a RESTful API. These competencies will be assessed in this capstone project.

DEVELOPER PORTFOLIO

Developers who have the edge are those who find ways to apply their newfound skills from the get-go. As you may know, a **developer portfolio** (a collection of online creations that you have made) allows you to demonstrate your skills rather than just telling people about them. It's a way of bringing your CV to life and introducing yourself to the world. As you learn more skills and put these into practice, each project that you complete will become more efficient and eye-catching.

These capstone projects give you the means to create projects for your very own developer portfolio, allowing you to walk away from this course not only with a certificate but, more importantly, with a head start to your tech career!

THE TASK AT HAND

In this project, you will be tasked with the development of a news application using Django. This news application is expected to provide readers with access to news articles authored by independent journalists, who may be affiliated with multiple curated publications under the oversight of editors. Subscribers will have the ability to receive newsletters and access articles published either by the publications themselves or by individual journalists. To clarify, a subscription can pertain to either a publication or an individual journalist.

Furthermore, whenever an editor approves a news article submission, it will be disseminated to subscribers via email and shared on a designated X (formerly Twitter) account. This project will also necessitate the application of skills acquired from previous tasks to design and implement a RESTful API and create an effective user login and authentication system.

This project provides you with the opportunity to acquire valuable experience in tasks, such as formulating roles and delineating distinctions between roles and groups, as well as composing unit tests for the assessment of a third-party RESTful API.

Take note

Functional requirements refer to the activities that are required of the program. **Non-functional requirements** refer to characteristics of the system as a whole, including its constraints, but excluding the activities covered in the functional requirements.

Instructions

A key focus of this project will be ensuring that your code is correct, well-formatted, readable, and that it adheres to the **PEP 8 style guide**. In this regard, **make sure that you do the following** (and double-check before submitting your work to avoid losing marks unnecessarily!):

1. Identify and remove all syntax, runtime, and logical errors from your code.
2. Make sure that your code is readable. To ensure this, add comments to your code, use descriptive variable names, and make good use of whitespace and indentation.
3. Make sure that your code is modular. Create functions to perform specific units of work.
4. How you choose to write code to create the solution to the specified problem is up to you. However, make sure you write your code as efficiently as possible.
5. Use defensive coding to validate user input and make provisions for errors that may occur using exception-handling techniques.

Practical task

In this task, you will be building a news application. This news application will allow readers to view articles published by publishers and independent journalists.

You will have to design your application before you implement it. Read the instructions and identify the functional and non-functional requirements. Ensure that you cater for UI/UX by planning the front end of your application. Make your submission as comprehensive as possible, and make sure that you have normalised the tables in your database before implementing your design.

Follow these steps:

1. Start a new Django project using `django-admin startproject project_name`.
2. Create a new Django application using `python manage.py startapp appname`.
3. Ensure that your application contains the following models.

Article:

- Should indicate whether articles have been approved by the editor or not.

Publisher:

- The publisher can have multiple editors and journalists.

Custom user:

- Users should be assigned a role and be added to groups. The group that the user is assigned to should be based on their role.
  - Ensure that you assign each group the necessary permissions (see below).
- The following are roles that should be created and the relevant permissions that their groups should have.
  - Reader
    - Can only view articles and newsletters.
  - Editor
    - Can view, update, and delete articles and newsletters.
  - Journalist
    - Can create, view, update, and delete articles and newsletters.

- The custom user model should contain custom fields for the following:
    - Fields for users who have the Reader role:
        - The user's subscriptions to publishers.
        - The user's subscriptions to journalists.
    - Fields or users who have the Journalist role:
        - Articles that the user has published independently.
        - Newsletters that the user has published independently.
    - If a user has a Journalist role, the program should assign the fields for the reader a value of 'None', and vice versa.

4. Create templates, set up views, and route paths that will allow users with an Editor role or assigned to the Editor group to review and approve articles.
    - Ensure that you have appropriately implemented access control within your system.
    - Once an article has been approved for publishing by an editor, you are required to implement the design techniques that you have learned in previous practical tasks to modify the program using one of the two approaches listed below:
    - 1 – Using signals
        - Make use of Django signals to send the article via email to all users who have subscribed to either the publisher or the journalist of the article;
        - Make use of Django signals, X's HTTP API, and Python's 'request' module to post the article to an X account; or
    - 2 – Not Using Signals
        - Alternatively, you may avoid signals and implement the above logic in the view that handles the logic, which allows editors to approve an article.

- Design and implement a RESTful API that will enable the retrieval of articles for publishers and journalists that a third-party API client could have subscribed to. This includes the following subtasks:
    - Create serialisers to serialise/deserialise Django models into JSON/XML formats for API interaction.
    - Define views to handle the API requests.
    - Configure URLs and map views to endpoints.
- Use your knowledge of unit testing and working with Django's REST framework to design and implement unit tests to test the third-party RESTful API. You may use Postman to test your endpoints. Feel free to share screenshots of your Postman tests. Ensure that you receive articles based on the subscriptions of the API client.
- Lastly, you will need to migrate your database to MariaDB.

Please ensure your Capstone Project submission is comprehensive, considering that this project will determine your grade for this section of work, and will also be uploaded to your GitHub portfolio and available to potential employers to showcase your expertise. Present yourself as a skilled software engineer by putting forth your best effort in this endeavour.

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.