TASK

Capstone Project - Consolidation

Visit our website

Introduction

WELCOME TO THE CONSOLIDATION CAPSTONE PROJECT!

In this capstone project, we are going to be consolidating all that you have learnt up to now. At this point, you are familiar with relational databases, application frameworks, version control systems, and containerisation technologies. Please feel free to review your past exercises to recap on what you've worked on.

Practical Task

- Download your Django Capstone Project from level 2.
- Add a .gitignore file to exclude any generated files such as binaries or virtual environments. Any generated output must be excluded from your Git repo.

  For more information about creating a .gitignore file, we recommend the following **reading.**

  We have included a .gitignore file in this task file with common files that should be added to a .gitignore file for a Django project.

  You can use this file in the root folder of your project and add to it if necessary. We suggest going through this file to see what types of files are usually excluded from a project when pushing it to a remote repository.

  All of the packages and binary files that are required by the Django project should be added again when somebody installs those packages using pip.
- Initialise a local Git repo for the project.
- If your Django project doesn't have a requirements.txt file, please create it before proceeding.

  The requirements.txt file is used when another developer wants to install the project on their machine. To prevent the tedious process of using the pip install command for every package that your project is

using one by one, they can simply use the following command instead:

```
pip install -r requirements.txt
```

All the packages listed in the requirements.txt file will be installed at once.

The requirements.txt file can be created manually, but there is a simple command to auto-generate this file. Navigate to your root directory with your virtual environment activated and run the following command:

```
python -m pip freeze -l > requirements.txt
```

(Please note that on some systems, you may have to use pip3 instead of pip in the terminal)

Note: The `-l` flag directly above refers to 'local'.

You may need to remove operating system-specific or unrelated items from an auto-generated requirements.txt file (e.g. pywin32) .

- Commit the project to the local repo.
- Branch from the master/main branch and name the new branch 'docs'
- In the 'docs' branch, add *docstrings* for a few of your functions, classes and modules/scripts. You can keep the documentation concise.
    - Commit the changes for each documented script one at a time before proceeding with the documentation of the next script. This ensures that you can rollback any changes you would have made should you need to.
- Generate user-friendly documentation using Sphinx and ensure you have the output stored in a docs folder in the project directory. For example:
    - hyperion/
        - blog/
        - **docs/**
        - hyperion/
        - polls/
        - ...

When setting up Sphinx for Django projects you will need to add the following to your **conf.py** file:

```
import os
import sys
import django
sys.path.insert(0, os.path.abspath('..'))
os.environ['DJANGO_SETTINGS_MODULE'] = 'Your_project_name.settings'
django.setup()
```

'`Your project name`' refers to the app name specified when you created a new app in your Django project (e.g. **`'blog.settings'`**)

Please do not exclude the generated documentation for this project because you want the audience of your repo to have easier access to your documentation. If you wish, you can investigate using pre-commit Git hooks to automate this, but note that this is not required.

- Commit all changes to the docs branch.
- Switch to the master/main branch.
- Branch from the master/main branch and name the new branch 'container'
- Add and commit a working Dockerfile to the container branch. Please ensure that your Django app works on a different computer.
- Switch back to the master/main branch.
- Merge the docs and container branches into the master/main branch.
- Add and commit a README.md file in your repo's parent directory. The file should outline the steps necessary to build and run your application with venv and Docker.
    - Ensure that you do not commit secrets such as passwords or access tokens to public repos. Your README.md file should inform the user how to acquire and add these themselves. (You can temporarily add this to the submission in a text file so that a reviewer will have quick access to this, it can be removed again

once the reviewer has marked the task as completed and no resubmission is required).
        - Remember: you use a .gitignore file to ensure that sensitive files are not tracked by Git. Include the .gitignore file provided or, if you wish, create a .gitignore file and include **the following code** text inside of it and place this file file at the root of your Django project.
    - If it doesn't already, please ensure that your repo includes a file named **requirements.txt** to automate the installation of the project's requirements.
    - Remember to exclude any venv or virtualenv files from your repo.
- Create a public remote repo on GitHub, GitLab, or any other similar Git host.
- Follow the instructions provided by your chosen Git host to push your local repo to the remote repo you've created.
- Upload a file named **capstone.txt** with a link to your public remote Git repo.

Rate us

Share your thoughts

HyperionDev strives to provide internationally excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.