TASK

Capstone Project – Consolidation

Visit our website

Introduction

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

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 capstone project, you will enhance a previous project by implementing version control, creating comprehensive documentation using Sphinx, and containerising the application for efficient deployment using Docker.

You will also need a requirements file. Recall this 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
```

Then, all the packages listed in the requirements.txt file will be installed at once. Please note that on some systems, you may have to use pip3 instead of pip in the terminal.

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 (the `-l` flag refers to 'local'.):

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

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

Instructions

A key focus of this project will be ensuring that your code is correct, well-formatted, and 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 that 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.

Take note:

A **.gitignore** file excludes 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**.

6. We have included a .gitignore file in this task folder 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.

Practical task

- Find your final Django Capstone Project on your local machine or in your course folders.
- Add a .gitignore file to your project folder.
    - Include the .gitignore file provided or, if you wish, create a .gitignore file that includes **the following code** text inside of it and place this file at the root of your Django project.
    - Remember to exclude any venv or virtualenv files from your repo.
- Initialise a local Git repo for the project.
- If your Django project doesn't have a requirements.txt file, please create it before proceeding.
- Commit the project to the local repo.
- Create a branch named '**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 roll back any changes you 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.

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 or the **Docker Playground**.
- 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 instruct the user on 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.
- Create a public remote repo on GitHub.
- Follow the instructions provided by your chosen GitHub 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.

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.