Exercise 2.8: Deploying a Django Project

Learning Goals

- Enhance user experience and look and feel of your web application using CSS and JS
- Deploy your Django web application on a web server
- Curate project deliverables for your portfolio

Reflection Questions

1. Explain how you can use CSS and JavaScript in your Django web application.

   Using custom.css to have the scrollbar always display to prevent layout changes when it appears/disappears.
   Using Bootstrap for the whole UI
   Importing a simple JS script so the directions are displayed as intended (Using markdownsX/markdownify)

2. In your own words, explain the steps you'd need to take to deploy your Django web application.

   1. <base_dir>/<project_folder>/prod/prod.py: `wsgi_app = "recipe_project.wsgi:application"`
   2. <project_folder>/settings/prod.py:

```
from .base import *

# Debug to false
env = environ.Env(DEBUG=(bool, False))
environ.Env.read_env(str(BASE_DIR / ".env"))

# Read secret key and debug prom .env file
SECRET_KEY = env.str("SECRET_KEY")
DEBUG = env.bool("DEBUG")

# insert hosted apps url later
ALLOWED_HOSTS = ["*"]
CSRF_TRUSTED_ORIGINS = ["*"]
# Adding necessary middleware “whitenoise”
MIDDLEWARE.insert(1, "whitenoise.middleware.WhiteNoiseMiddleware")

# Static URLS
STATIC_ROOT = str(BASE_DIR / "staticfiles")
STATICFILES_DIRS = (str(BASE_DIR / "static"),)
STATICSTORAGE = "whitenoise.storage.CompressedManifestStaticFilesStorage"

# Connect to database remotely
import dj_database_url
DATABASE_URL = env.str("DATABASE_URL")
DATABASES = {
"default": dj_database_url.config(default=DATABASE_URL),
}
```

3. Updating makefile and running 'make prod-collectstatic' to collect as static files in one folder

```
prod-start:
  python manage.py runserver --settings=recipe_project.settings.prod
prod-install:
  pip install -r requirements/prod.txt
prod-migrate:
  python manage.py migrate --settings=recipe_project.settings.prod

prod-mm:
  python manage.py makemigrations --settings=recipe_project.settings.prod
prod-superuser:
  python manage.py createsuperuser --settings=recipe_project.settings.prod
prod-gunicorn:
  gunicorn --env DJANGO_SETTINGS_MODULE=recipe_project.settings.prod --bind 0.0.0.0:8000 -c recipe_project/prod/prod.py
--log-file -
prod-collectstatic:
  python manage.py collectstatic --settings=recipe_project.settings.prod
```

4. On railways, create a postgres DB, copy it's public URL and put in the local .env file as DATABASE_URL.
   Also have the SECRET_KEY inside the .env file (have .env in .gitignore to not upload it to GH)
5. Run migrations to create tables in our new postgres DB.
6. Create web-server on railway from the GH repo and add variables: SECRET_KEY (copy from .env file) and DATABASE_URL (not the public one) to connect the server to the DB.
7. Create and copy domain inside the web-servers networking settings. Port: 8000
8. Paste the URL in side ALLOWED_HOSTS and CSRF_TRUSTED_ORIGINS in prod.py.
   Put 'https://' in front of the URL for CSRF...
9. Push the changes to the GH repo

3. You've now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
   a. What went well during this Achievement?

      Everything except for the deployment process, which always seems to be a bit of a wild ride.

   b. What's something you're proud of?

      I am proud of:
      - sticking to my commitment regarding calls and deliverables
      - all I've learned
      - the app I've built
      - the time frame in which I was able to achieve this