Excellent! You’re almost ready for deployment. Here’s a practical step-by-step checklist to run your Flask cybersecurity awareness app and package it for installation, **with credential masking and secure distribution**:

**1. Run Locally**

* Ensure your environment is set up (Python, pip, virtualenv).
* Install requirements:

bash

pip install -r requirements.txt

* Set environment variables for secrets (never hard-code credentials!):

bash

export FLASK\_APP=run.py

export SECRET\_KEY=your-production-secret

export DATABASE\_URL=your-db-url

export MAIL\_USERNAME=your-mail-user

export MAIL\_PASSWORD=your-mail-pass

# etc.

* Run the app:

bash

flask db upgrade

flask run

**2. Credential Masking & Secure Configuration**

* **Use .env files:** Store secrets in a .env file and load them using [python-dotenv](https://pypi.org/project/python-dotenv/).
* **Never commit .env to Git**—add it to your .gitignore.

**Example .env:**

Code

SECRET\_KEY=super-secret-key

DATABASE\_URL=postgresql://user:pass@host:5432/db

MAIL\_USERNAME=admin@example.com

MAIL\_PASSWORD=your-email-pass

**Load in config.py:**

Python

from dotenv import load\_dotenv

import os

load\_dotenv()

SECRET\_KEY = os.getenv('SECRET\_KEY')

SQLALCHEMY\_DATABASE\_URI = os.getenv('DATABASE\_URL')

MAIL\_USERNAME = os.getenv('MAIL\_USERNAME')

MAIL\_PASSWORD = os.getenv('MAIL\_PASSWORD')

**3. Packaging for Installation**

**a. Create a Setup Script**

Add a setup.py or, better, a pyproject.toml (if using modern standards) for pip installation.

**Example setup.py:**

Python

from setuptools import setup, find\_packages

setup(

name='cybersec-awareness-app',

version='1.0.0',

packages=find\_packages(),

include\_package\_data=True,

install\_requires=[

'Flask', 'python-dotenv', 'Flask-Login', 'Flask-Mail', 'Flask-SQLAlchemy', # and others

],

entry\_points={

'console\_scripts': [

'cybersec-app=run:main',

],

},

)

**b. Create an Install Script**

Add a install.sh or install.bat that:

* Creates a virtualenv
* Installs dependencies
* Prompts for credentials (writes to .env)
* Runs initial DB migrations

**Example install.sh:**

bash

#!/bin/bash

python3 -m venv venv

source venv/bin/activate

pip install -r requirements.txt

echo "SECRET\_KEY=$(openssl rand -hex 24)" > .env

read -p "Database URL: " dburl

echo "DATABASE\_URL=$dburl" >> .env

read -p "Mail Username: " mailuser

echo "MAIL\_USERNAME=$mailuser" >> .env

read -s -p "Mail Password: " mailpass

echo -e "\nMAIL\_PASSWORD=$mailpass" >> .env

flask db upgrade

echo "Setup complete. To start, run: source venv/bin/activate && flask run"

**4. Distribute as a ZIP or Tarball**

* Exclude all .env, \_\_pycache\_\_, and other sensitive files.
* Include your install.sh, requirements.txt, and app files.

**5. Documentation**

Prepare a README.md with:

* Installation steps
* How to set environment variables
* How to run and test

**6. Advanced: Masking and Security**

* Consider using a tool like [python-dotenv-vault](https://github.com/dotenv-org/dotenv-vault) for environment secret management.
* Use [PyInstaller](https://www.pyinstaller.org/" \t "_blank) or [shiv](https://shiv.readthedocs.io/en/latest/) to create a binary—note: not 100% secure, but can obscure code for non-developers.

**7. Final Checklist**

* All credentials loaded from environment (never hard-coded)
* .env in .gitignore
* All dependencies in requirements.txt
* Setup/install script prompts user for secrets
* Documentation is clear for admins/users

**If you want, I can provide:**

* A sample .env
* A sample requirements.txt
* A sample setup.py or pyproject.toml
* A full install.sh script

**Let me know if you want any of these files generated!**

Top of Form

Bottom of Form