Digital Badges

A badge is a digital piece of evidence with which a student demonstrates that they have mastered certain skills or knowledge. Badges are a great instrument to make acquired knowledge and skills visible and 'portable'.

Detailed Requirement

1. Design a page to upload the badge details. The page should contain the following fields associated with a badge. The following fields have to be stored in the database on clicking Save.

   Badge name: <string input>
   Badge description: <string input>
   Upload badge: <upload option for PNG>
   Eligible students: <string input of email ids in a comma separated list>
   Save: <button>

2. Once the upload is done, redirect to a separate page to list all the badges. This page must allow CRUD operations.
3. Have a global API endpoint to authorize the badge based on request origin. Use API parameters to verify if the student is eligible for the badge. If the user is eligible, return the PNG with status code 200 return a 403 with a relevant message.

   API: http://localhost:8082/badge/verify?name=<badge_name>&email=<email_id>

Technology to use

- Python3.7.9/Flask for the backend.
- SQL for the database. You may use SQLite.

Important things to keep in mind

Do not use a separate framework for the front-end.

- Simple, separate HTML pages using Jinja is enough. Do not write any complicated javascript code. This website doesn't require it. Some simple autofill / onclick / other event triggers are okay.
- No need to optimize javascript and DOM loading.
- Keep CSS separate if you have any. This is very important as we will not be evaluating the CSS files. Keep them separate so that its easier to evaluate the rest of the code.

Keep the uploading functionality separate.
You can store the PNG on your local filesystem or cloud somewhere.

Do not use Apache or NGINx, or any webserver.

- You should run your app using a flask run app or a similar command.
- If your application needs any extra commands to be run while initializing, add a shell script called run.sh, which does that.
- The shell script should run on Ubuntu 18.04 LTS.
- You may ship a Python script (run.py) instead of a shell script if you want to make it cross platform. This will not fetch you extra marks.
- You may have an install.sh which installs the app.
- You may ship a Python script (install.py) to make it cross platform. This will not fetch you extra marks.

Sending over the code

- Create a public GitHub repository and share the link with us.
  - Commit a readme with instructions to deploy/run the code and basic documentation about the code if required.
  - Commit your code regularly to see the workflow, with sensible commit messages.
  - A single commit of the entire code at the end will be rejected.
  - Even if you have just two sensible commits, it will be considered.
  - One extra fake commit to satisfy this criterion does not count.

- Give proper instructions on how to make this app run on my machine.
- You can make the following assumptions
  - We know how to run a pip install -r requirements.txt and run a flask app
  - We use Ubuntu 18.04 LTS.
  - Any other instructions should be copy-pasteable from your readme and works on Ubuntu 18.04 LTS.
  - Keep the instructions as simple and concise as possible.