IMAGE BOOTCAMP design doc

1. **Authentication**

   Registration is done for two types of user – Student and Instructor. To keep authentication storage easy and easy code, this is integrated into one database. Further, login is integrated, so when logging in, the credentials can be checked from the same storage. Now, in case of login deadlock , it is resolved by checking a sub credential of student or instructor.
   Also, password hashing is not done in this as , it's all in development.

2. **Database**

   To keep matters simple here and to code faster , SQLite database is used. And to keep things simple, a simplistic approach is taken towards designing the database, also this approach can help prevent major intrusion as in SQL injection attacks.

3. **Image Storage**

   When Submitting the tasks by a student, the required info is stored, along with that, the filename of the image is also stored thus, image can be easy rendered in the template. Same goes in the case of Instructor, in the task db. The Images are stored in a backend folder having same quality. However, the in any case, the quality can obviously be changed in case of a higher file size given its type of extension and then the file be saved and stored.

4. **Assignment of task by instructor**

   The form here is kept totally in front end with viable options and keeping dynamic nature of the track type.

5. **Instructor Homepage**

   Shows the overall tasks defined the same user.

6. **Student Page**

   Student user can easily submit his/her task only once, and before submitting download the picture for editing. Student Can also see the overall tasks done by her and the overall tasks assigned to her.

7. **Admin**

   Shows admin level info regarding tasks allotted, tasks submitted and people associated in the web app, etc. Also, shows Graphs of different data.

8. **Grading**

   Instructor can grade a student based on his work / seeing his work. The Grading will be done out of 5 and the same grade will be shown on the Students Side.

(NOTE : Front end is not taken care of that ideally)

CODE STRUCTURE

1. Authentication is done using a flask extension named , Flask_login , again, after using this, user is again verified according to his role in the web app.
2. Templating – Default Template engine – JINJA templating system has been used.
3. Database – SQLite Database
4. Forms – Both types of forms, server side forms (executed in client side) and client side forms are there
5. Selective route system has also been implemented, which means if you are a student , you cant go to the instructor routes and vice versa.
6. OnClient Side . Jquery is also used for complex event triggers
7. For Admin Page, ChartsJs is used
8. For Overall client side, TailWindCSS framework is used.

Data Dumps

- As for Student tasks - Students can submit their tasks by clicking on a link ,which will dump their submit data onto an URL (for POST request) , which will first check their authenticity including user and files and then store the data.
- As For Instructor tasks, they can assign their tasks by clicking on a link ,which will dump their task data onto an URL (for POST request) , which will first check their authenticity including user and files and then store the data.

WARNING

For production like environment, first do a hashing of the passwords ,one can do so by hashlib or any cryptographic module, You can use salt based hashing or any pepper based hashing upto one's wish. *(Simple Tweaks in the code)*