MUKHDA: Password and User ID Less Cross-Platform Authentication System

Aaryen Mehta 190495
Mohd Muzzammil 190503
Varenya Srivastava 190943

March 2023

1 Key Idea

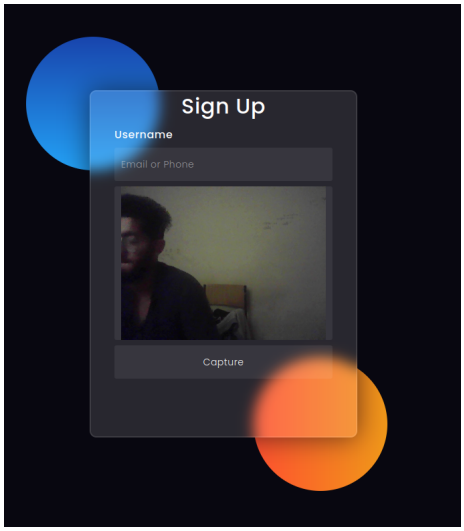
From the bucket of projects, we have chosen passwordless cross-platform authentication. As a step towards improving human-computer interaction, and to make things a bit easier on the user side, we have proposed a solution which does not involve using passwords when signing up to a website and also login does not require password. Also, the proposed solution uses a novel method of identifying user information in the database of a website, which is not based on hashes generated on signing up, called user ids.

2 Proposed Solution

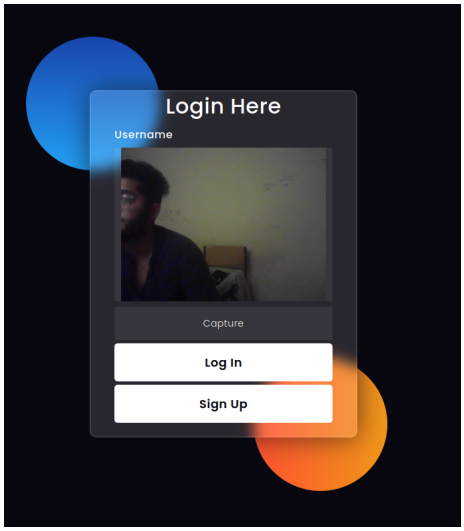
The proposed solution is as listed in the following points:

- **Product Requirement:** We planned on designing an authenticator app, which does not has a database of its own and only showing the face (or fingerprint) on both PC and smartphone would suffice to login.
- **Challenge:** Without storing the user information in the authenticator app, it would not be possible that even if both the faces match on PC and well as smartphone, which user trying to access the website.
- **Solution:** We proposed a new kind of generating user-ids for users that websites can adopt, where instead of using randomly generated user hashes as a user-id, when a user signs up, we store the face encodings (first we thought of storing images, but then moved to encodings because users might have privacy concerns regarding storing their image in the database) of the user in the website database. (Note that we are talking about the database of the website and the authenticator app in itself does not store any kind of data).
- **Implementation:** We have implemented the proposed application in a python application running on a Flask server. For face recognition, we are using python's face recognition library. We have used MIT App Inventor to create the android app to send image to the PC from smartphone.

- **Working details:** The app works as follows:
  - **Signup**: When a new user comes to the signup page, the page asks for a *username* and there is a image capture widget to capture the image of the user. The image is passed onto the backend where face encoding is extracted from the image and the encoding is saved, along with the username. Note that the image is **not** stored in the database.



  - **Login**: When a user tries to login, it asks the user for capturing the image, which is passed onto the backend. Note that user does not even need to key-in the username (which sometimes we tend to forget, again better human-compueter interaction), since we only need the face encodings.The images face encoding is extracted and an encoding matching algorithm runs to find the entry in the db whose encoding match with the given encoding. If none of the encoding match, try again option appears. Otherwise the website asks to open the smartphone authenticator app.



– **Two-Factor Authetication:** The authenticator app captures the image of the user and sends it to the server. Again, the face encodings are matched with the currently matched entry in the database. If the captured image matches, the user is logged in. Otherwise user is asked to try again.

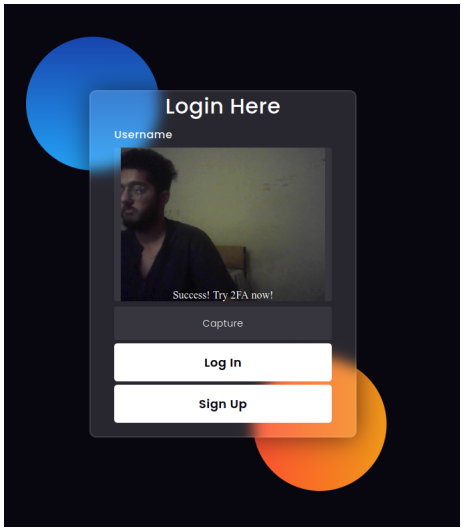


Figure 1: Two Factor Authentication

Figure 2: 2FA App on Android

- **Link to the working demo for the app:** https://github.com/AaryenMehta/web-app-cgs402/blob/main/app-demo.mp4

Complications: We have found the following issues with this idea:

- **Latency**: Since we are running an encoding matching algorithm for each user present in the database for logins, this can become really slow when we have a large number of users.
- **Image Privacy**: One issue that users might have is to have privacy issues if their image is stored in the website. This can be tackled by storing only the face encodings in the website database.

3 Further Improvements

The below points lists the scopes of improvement for this project:

- The current method uses 2D face-recognition system, which is prone to get cracked by placing images instead of real face. This can be replaced by 3D contour-based face-recognition system, which would be more robust.

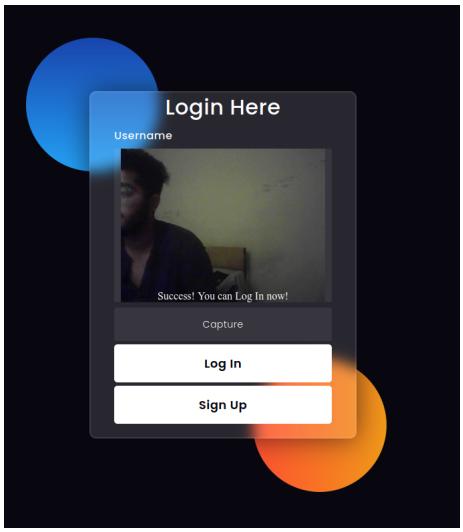


Figure 3: Page on Successful 2FA

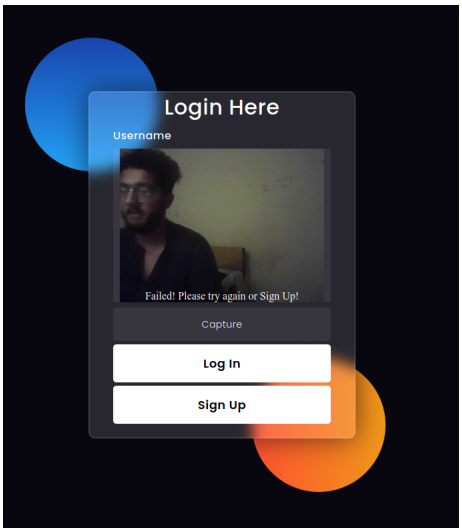


Figure 4: Page on Error

- We can think on ideas how to improve upon latency issues which can come in picture when the app scales.
- Right now, we only use facial data for login and signup. This facility can be extended to fingerprint based system, where we can similarly store fingerprint encodings for the user.

References

[1] Link to our public repo: `https://github.com/AaryenMehta/web-app-cgs402`

[2] Face Recognition Module: `https://pypi.org/project/face-recognition/`

[3] OpenCV: `https://opencv.org/`

[4] Frontend: `https://codepen.io/fghty/pen/PojKNEG`

[5] Flask: `https://flask.palletsprojects.com/en/2.2.x/`

[6] MIT App Inventor: `http://ai2.appinventor.mit.edu/`

[7] Login & Signup Boilerplate: `https://github.com/bhuvansingla/flask-login`

[8] Image Capture Boilerplate: `https://towardsdatascience.com/camera-app-with-flask-and-opencv-bd147f6c0eec`