UCSB
CS190B - F23

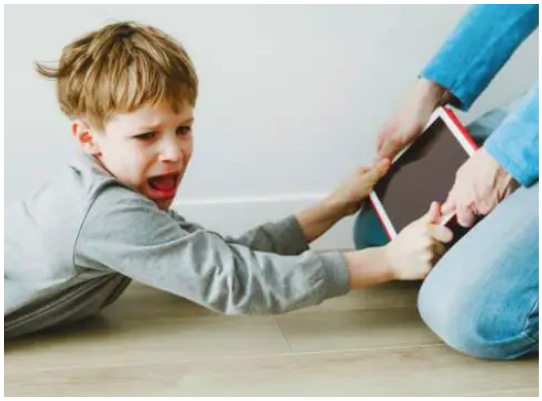
Final Project:
Facial Frenzy

By the expresso team

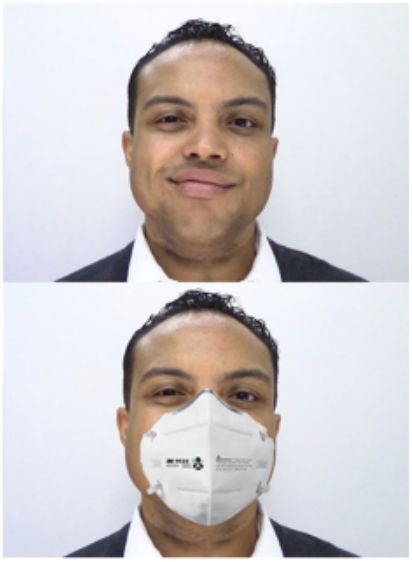
Team Members:

First and last names	Perm Number
Cappillen Lee	5554720
Gretchen Lam	5380514
Johnson Chan	5178223

1. Problem Overview



The growth of the digital age through social media, texting, and gaming has led to a large decline in face-to-face interactions – interactions that are essential to understanding and showing human emotions. This shift has had a negative impact on emotional intelligence and thus inhibits many people, more specifically, younger generations, from fostering meaningful human connections. A recent study from the Erikson Institute found that over 85% of parents grant their children access to over two hours of screen time per day. Many of these children spend their screen time watching non-enriching shows and videos and playing unsafe games in open-world communities. A study conducted by JAMA Pediatrics found that socioemotional development as well as many other skills are stunted by extraneous screen time at a young age. As children spend more time interacting with screens and less with people, their ability to read and express emotions becomes impaired. This increasing reliance on digital entertainment and decreasing emphasis on emotional engagement has not only affected children, but also older individuals such as teenagers and young adults, most notably after recent global events like the COVID-19 pandemic. Quarantine had led to prolonged periods of social isolation, and in turn, many people turned to technology-based interactions. People spent a majority of their day playing online games and/or scrolling endlessly through social media, an issue exacerbated by the inability to see each other face-to-face. Even in the time shortly after isolation, people were required to wear face masks, allowing them to easily hide their facial expressions in social settings. People were able to show happiness by simply raising their eyebrows, for instance, leading to an overall compromise on the perception of emotion. It is evident that the lack of face-to-face interactions is affecting people's social awareness and development. The fact that this problem is affecting younger generations is a major reason why it must be addressed.



2. Methods and Solution

The Solution

Facial Frenzy is designed to remedy the problems as mentioned above. By combining technology with the need for emotional and social development, the game is able to offer a unique, fun, and engaging solution for all ages. Unlike the digital content that is widely available today, which oftentimes focuses primarily on entertainment, Facial Frenzy places great emphasis on emotional engagement. By encouraging players to recognize and mimic a range of emotions, the game allows the conveyance of emotional expression to be a fun and immersive experience. In addition to enhancing emotional intelligence, Facial Frenzy also promotes healthy social interactions. The game features a live leaderboard with all players, and users can peruse each profile to see each other's most recent expressions, which can sometimes be very humorous. This aspect also helps all players stay socially engaged and connected with their peers. In conclusion, Facial Frenzy stands as a creative solution in the digital age. It aims to yield a more productive, beneficial, and educational screen time experience. It provides a platform that not only entertains its users, but also plays a crucial role in developing their emotional and social skills.

Facial Recognition

In order to create an intuitive and safe login/signup feature, we wanted to utilize facial recognition. Our model used Python libraries such as face_recognition for facial recognition utilities as well as OpenCV for video capture, GUI elements, and saving images. The process begins with OpenCV capturing real-time video data from the webcam. As users present their faces to the camera, we call methods from the face_recogntion library to detect and encode facial features, creating a unique facial signature for each individual. These facial encodings are then compared against a stored directory of known faces to determine if the user is already recognized and registered. In addition, when a face is detected, the program calculates a confidence score using the face_distance method, which compares the present face to a known face encoding and gets a euclidean distance for that comparison. This confidence value is essential for ensuring accuracy in identification. For unknown faces / new users, the program prompts them to capture a screenshot of their face, which is then encoded and added to the directory, effectively creating a new account for them. As a result, this feature allows for a secure, quick, and seamless user identification process. To enhance the accuracy and reliability of this process, we also incorporated a waiting mechanism. Once a potential match in faces is found, the program continuously checks if the same face is recognized in subsequent frames for 3 seconds. Only after these 3 seconds are users able to log in and play. This waiting period significantly helps to decrease false positives, thereby furthering the security of the entire feature. We also use OpenCV's GUI methods to provide real-time feedback to the user, as shown by how the program displays prompts and recognition results on the screen. Overall, these technologies are incorporated together to build a reliable authentication experience.

Emotion Recognition

The emotional recognition game was built based off of a public model called ‘fer2013’ and utilizes haarcascade for facial recognition. We use cv2 to open the camera and take video inputs into the system and form emotional analysis on the individual frame. From the inputs, we modify it until it matches specified qualifications for analysis, which includes converting the frames to gray and also applying various offsets for better analysis. All of this combined creates an emotional recognition program that analyzes the player’s emotion on each frame and updates the emotion text shown in the window. The emotional analysis from the live video feed dictates the difficulty and dynamics of the game, creating a unique and personalized gaming experience for each player. As the player progresses through the game, they are given a randomly generated emotion that never repeats twice. The harder emotions such as fear and disgust gives the player higher points and easier emotion gives lower points to the players. To make the gaming experience even more immersive, we incorporated real-time feedback mechanisms. The emotional analysis results are displayed as text in the game window. Beyond the gaming aspect, the emotional recognition system also serves as a valuable tool for self-awareness and emotional regulation.

Backend/API

The game, once installed on a local machine, will communicate with a Flask web application hosted on our Raspberry Pi. This integration with the backend will add a social and competitive aspect to our game. Players can scour the leaderboard page that’s rendered by the Flask web app. In this page the player can view the lifetime scores of other players. They can also inspect the pictures of other players by clicking on the usernames on the leaderboard. Once in the player profile page, people can see the face expressions of players made from prior games. This social aspect of the game will encourage people to work on their emotions and practice more to compete against top players. The web app also stores a player’s game sessions using a Postgres database and Firebase File Storage. Once a game is completed, the game session is uploaded to the Flask web app. The Flask web app will update the player’s lifetime score and upload the player’s images taken from the game. Saving player game sessions and their pictures will help players observe their improvement and progress. The backends provide a feature that complements the game and allows players to better engage with the game to improve their emotional development.

3. Results and Findings

Facial Recognition

The facial recognition feature of our project has demonstrated promising results, as shown by its efficiency and accuracy when identifying users. There have also been some interesting discoveries and key learnings, one of which revolved around tradeoffs. In order to make the model more accurate, we fine tuned the hyperparameters of the program. For instance, we upped the face match threshold (which determines how strict the model is when deciding whether a face in the current frame matches a known face in the faces directory) and labeled any face with a confidence value of less than 97% as “Unknown”. While these changes did improve the accuracy in identifying people correctly, the model became a lot more finicky. If a recognized person tilted their head at an angle (e.g. looking downward, to the left/right, etc.), the confidence value would drop significantly, and they would be unrecognizable by the model. For the sake of our project, however, we assumed that players would look straight into the webcam, which eliminated this low confidence issue. However, it was interesting to note the tradeoff between higher accuracy and flexibility in facial recognition. Overall, this discovery emphasized the need for a more versatile and reliable recognition algorithm, preferably one that can handle a wider range of facial orientations. Another valuable insight we learned came from the positive user feedback concerning the facial recognition login feature. We received positive comments about the way in which players could easily sign up and login without clicking a button (aside from entering their name at account creation). This finding suggested a growing interest in biometric security and pointed toward a fondness of convenience, similar to how many people enjoy using fingerprint identification or facial recognition to access their mobile devices.

Emotion Recognition

The results obtained from the emotional recognition game show a good outcome, indicating robust player engagement. The game's straightforward format lets the player comprehend the game easily and allows for users of varying levels of familiarity to enjoy the game. Notably, the game accurately discerns the player's emotion at the conclusion of the time limit, promptly updating scores within the gameplay. The integration of API calls ensures the seamless transmission of resulting images and scores to the database, leading the player to foster a sense of accomplishment and healthy competition among players. To further enhance the user experience, the next steps would be to implement improvements in the system for displaying in-game text, with a focus on optimizing readability and overall visual appeal. In terms of technical refinements, attention should be directed towards strengthening error handling mechanisms, particularly in the areas of face detection and emotion recognition processes. This optimization will contribute to the overall reliability of the game, ensuring a smooth and frustration-free user experience.

Backend/API

The results from the Flask web app are promising. The backend is able to register new players into the Postgres database and return player data when requested. Another result from the web app is that it will be able to store images for a particular player. This is accomplished by designing POST routes in the Flask web app that accepts uploads of image files. In addition to uploading files, the web app also supports downloading the images that belong to a particular player. This allows players to view their prior game sessions by visiting a dedicated profile page for the player. Another result is that the web app displays the top players in the game through a rendered html page.
One of the findings for this project is that, when uploading images to a Flask app, the POST request can only contain the headers and the file content. This was an interesting find because file uploads had never been covered extensively in the course. To make a file upload POST request, the request content-type headers need to be specified to 'multipart/form-data' and the content of the request can only contain the binaries of the file. Another great find from this project is that the Raspberry Pi works well for hosting small scale web applications. It was able to receive uploaded images and store them in the Firebase storage service. In addition to that it was able to fetch those images from Firebase so that it could be rendered to the user. It's important to note that the images were compressed and scaled down so that it would require less bandwidth to transfer the images. The project has no content delivery network service, thus it was important that the media being transmitted over the internet was small.

4. Challenges

Facial Recognition

Our facial recognition model proved to be quite difficult when it came to accuracy. Since facial recognition acted as our main login feature, we wanted it to be highly accurate as to ensure users would be logged in as themselves. The initial model seemed to frequently use encoded faces rather than simply labeling unrecognized users as “Unknown”. In the beginning stages, most peers that we tested our model on (who should have been “Unknown”) were labeled as either Cappillen, Johnson, or Gretchen since we had already uploaded images of ourselves. After upping the face match threshold and using the face confidence value to better determine unrecognized users, our model slightly improved. However, there were instances where unrecognized users would rapidly switch between labels (e.g. switching between Johnson, Cappillen, and Unknown). In order to remedy this, we modified the code to add a feature where users were labeled as “Unknown” unless the model recognized a consistent face for 3 seconds. After that 3 seconds, the user would be able to play the game, thereby logging them in. We still conducted tests on this model: removing ourselves from the known faces, having our peers use their faces, etc. In the end, this proved to be quite successful, and we have not run into issues with the model or logic.

Emotion Recognition

The emotional recognition model encountered a challenge related to the accuracy of detecting emotions. Initial tests and models used were notably imprecise, frequently categorizing most facial expressions made by the player as either "happy" or "neutral," while being insensitive to other nuanced emotions. This presented an issue as we aimed for the game's emotion recognition to be highly accurate, ensuring maximum enjoyment for players. To address this, we revamped the emotion_reg.py file, enhancing its accuracy and organizing it into a utils folder for improved readability. Additionally, we identified a superior model and optimized its weights, along with refining the input video format to boost accuracy. The outcome of these efforts resulted in the creation of a more refined and precise emotion recognition model seamlessly integrated into the game. After rigorous testing on the updated code, we successfully achieved accurate recognition for all available emotions. This comprehensive improvement culminated in the development of an emotionally immersive gaming experience, marking a successful transformation of the initial challenges into a functional and engaging emotional game.

Backend / API

There were several challenges when developing the backend. Dropbox was the initial file storage service that had been used for the Flask web app. However, there were issues in refreshing the access token using the secret keys. Documentation on Dropbox’s Python SDK was not in detail and didn’t provide enough specifications on how to use the functions to update the access token. Hours were spent debugging the Dropbox SDK until it had been decided to migrate to Firebase. Using the Firebase SDK worked much

better than Dropbox. However, more issues came up when uploading image files to the Flask web app. There were minor issues in the POST request contents that made it hard to debug the issue. However, after many attempts it was discovered that solely uploading the binary object of the image file was the solution. The Flask web app accepted the parameters of the HTTP POST request and received the image binaries.