***A.)Introduction:***Generative models for emotional face generation aim to create realistic facial expressions that convey specific emotions, such as happiness, sadness, or anger, from neutral or synthetic inputs. Using techniques like **Generative Adversarial Networks (GANs)**, these models learn to map facial features to emotional expressions.

The underlying idea behind **GANs** is to set up a competitive game between two neural networks — **a generator and a discriminator** — where the generator creates synthetic data and the discriminator attempts to distinguish between real and fake data.

***B) Datasets Used:***

**1. CelebFaces Attributes (CelebA) Dataset (used for model: Test Code 1(CelebA Dataset))**

This dataset contains 202599 unique values (images) covering large pose variations, background clutter, diverse people, supported by a large quantity of images and rich annotations.

**2. RAF-DB Dataset (used for model: Test Code 2(RAF-DB DATASET))**

The Real-world Affective Faces Database (RAF-DB) is for recognizing emotion from facial expression. Contains 15000k facial images tagged with basic or compound expressions by 40 independent taggers. Images in this database are of great variability in subjects' age, gender and ethnicity, head poses, lighting conditions, occlusions (e.g. glasses, facial hair or self-occlusion), post-processing operations (e.g. various filters and special effects), etc.

**3.Flickr-Faces-HQ Dataset (FFHQ) (used for model: Test Code 3(Flickr-Faces-HQ Dataset ))**

The dataset consists of 52,000 high-quality PNG images at 512×512 resolution and contains considerable variation in terms of age, ethnicity and image background. It also has good coverage of accessories such as eyeglasses, sunglasses, hats, etc. The images were crawled from Flickr, thus inheriting all the biases of that website, and automatically aligned and cropped using dlib.

**4.CK+ dataset (used for model: TestCode4(CK+Dataset))**

Extended Cohn-Kanade dataset. This dataset contains 981 image files with emotions/expressions are defined as determined index below:

0 : Anger 1 : Disgust 2 : Fear 3 : Happiness 4 : Sadness 5 : Surprise 6 : Neutral

7 : Contempt

**Note: Currently working on…**

**Facial Expressions Training Data**

**Fer Affectnet Database - Processed for training a neural network**

**Description: Processed for a Neural Network**

AffectNet is a large database of faces labeled by "affects" (psychological term for facial expressions). In order to accommodate common memory constraints, the resolution was reduced down to 96x96. Meaning that all images are exactly 96x96 pixels.

No Monochromatic Images

Using Singular Value Decomposition, each image's Principal Component Analysis was calculated. The threshold for the "percentage of the first component (index 0) in the principal components" (in short the PFC%) was set to lower than 90%. This means that most if not all of the monochromatic images were filtered out.  
The PFC% was left inside the CSV file, should you prefer a lower PFC% threshold.

Link: <https://www.kaggle.com/datasets/noamsegal/affectnet-training-data>

**Planning on using Nvidia Cuda Toolkit and Nvidia cuDNN Deep Neural Network Library, for faster deep learning and Model Training using Tensorflow or Pytorch GPU libraries.**

*Nvidia Cuda Toolkit:* The NVIDIA® CUDA® Toolkit provides a development environment for creating high-performance, GPU-accelerated applications. With it, you can develop, optimize, and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms, and supercomputers. The toolkit includes GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler, and a runtime

Library.

*Nvidia cuDNN:* The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for [deep neural networks](https://developer.nvidia.com/deep-learning). cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, attention, matmul, pooling, and normalization.

***C) Steps involved in training this GAN Model:***

Steps for Training the GAN Model

1. **Dataset Preparation**:  
   * Download and inspect the RAF-DB dataset.
   * Preprocess images: resize to 64×6464 \times 64, normalize pixel values to [−1,1][-1, 1].
   * Map images to their labels using train\_labels.csv.
2. **Define GAN Model**:  
   * **Generator**: Up samples random noise to generate realistic images using Dense, Conv2DTranspose, and BatchNormalization.
   * **Discriminator**: Distinguishes real from fake images using Conv2D layers and outputs a probability.
3. **Prepare for Training**:  
   * Batch the dataset using tf.data.Dataset.
   * Define loss functions:
     1. **Generator Loss**: Measures how well fake images fool the Discriminator.
     2. **Discriminator Loss**: Measures the ability to distinguish real from fake.
   * Initialize optimizers like Adam with a learning rate of 1×10−41 \times 10^{-4}.
4. **Training Loop**:  
   * For each epoch:
     1. Generate fake images from random noise.
     2. Compute losses for both Generator and Discriminator.
     3. Backpropagate and update weights using optimizers.

for epoch in range(EPOCHS):

for image\_batch in dataset:

gen\_loss, disc\_loss = train\_step(image\_batch)

print(f"Epoch {epoch + 1}, Gen Loss: {gen\_loss:.4f}, Disc Loss: {disc\_loss:.4f}")

5. **Evaluation**:

Generate and visualize sample images to evaluate quality:  
 noise = tf.random.normal([1, 100])

plt.imshow(((generator(noise, training=False)[0] + 1) / 2.0))

1. **User Inference**:  
   * Preprocess a user-uploaded image, pass it through the Generator, and display the generated smiling face alongside the original.

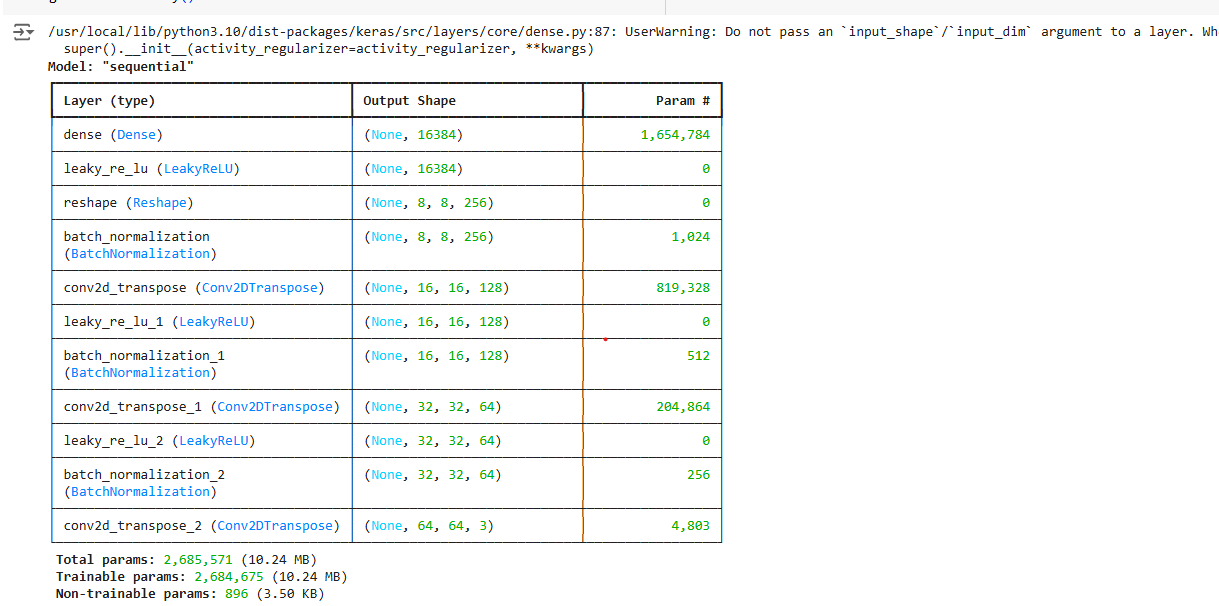
***D. Coding Section:***

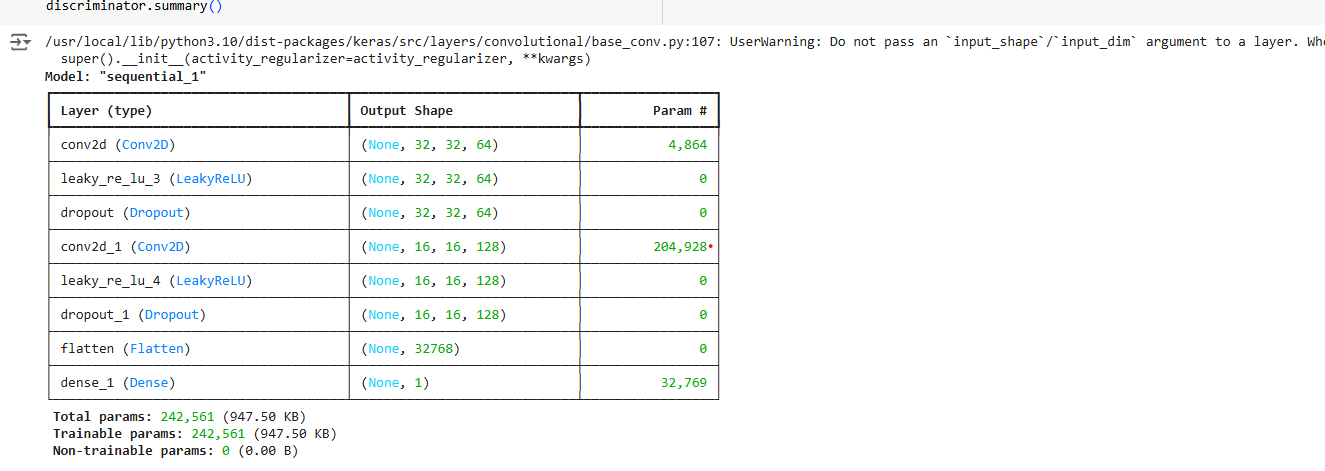
Three Test codes have been implemented through google collab as link follows:

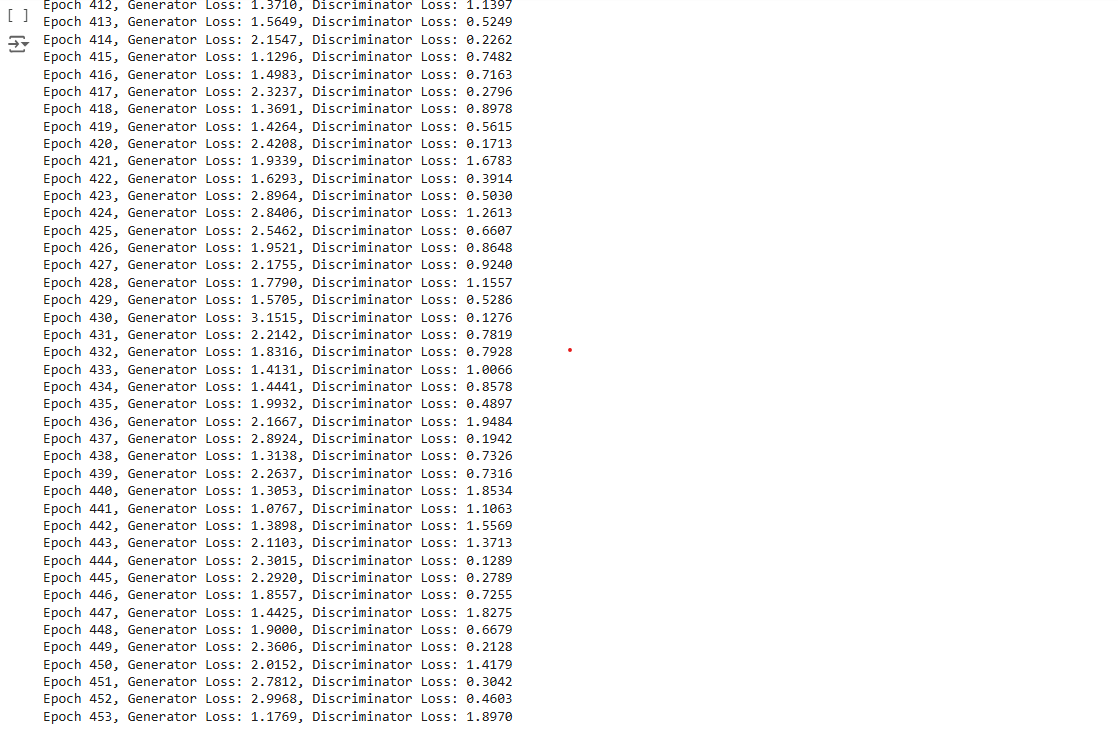
* <https://colab.research.google.com/drive/1WMwPHyHvFeWUH6OoiaVpdB1bVh7DvbHa?usp=sharing>
* <https://colab.research.google.com/drive/1pgGQEOtgAyxKRr_vnjgxL2iRkEO8P9Oh?usp=sharing>
* <https://colab.research.google.com/drive/17sBCzPpPZkbyqW3yA-aZumF8bBMquXtQ?usp=sharing>

**Code Snapshots:**

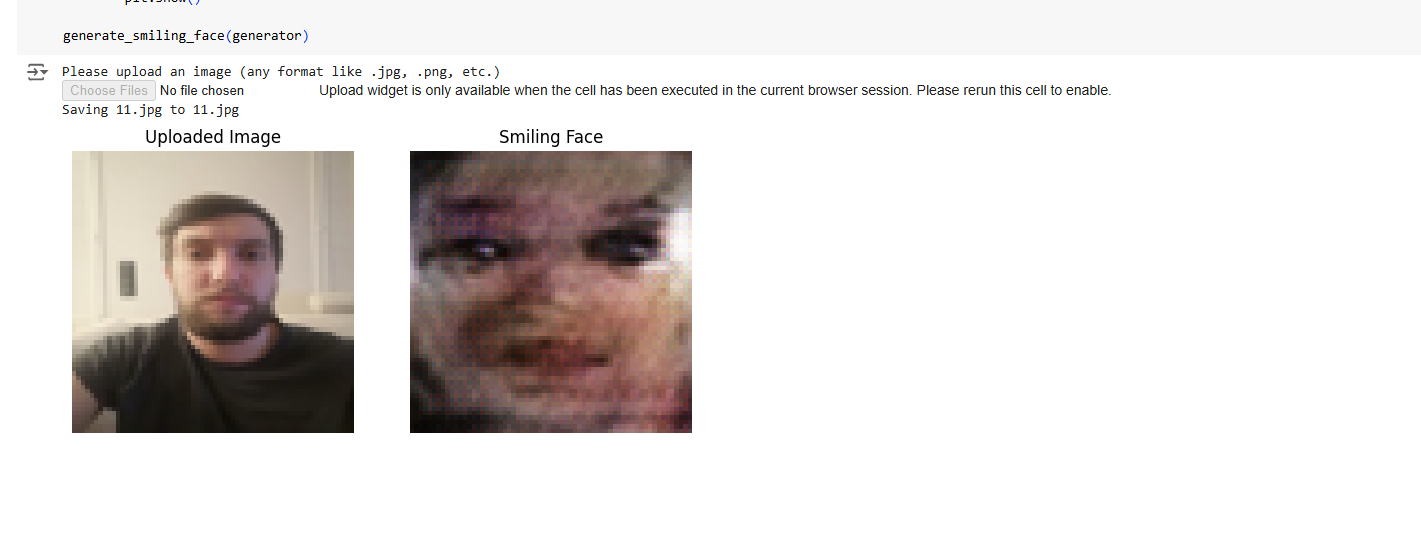
****





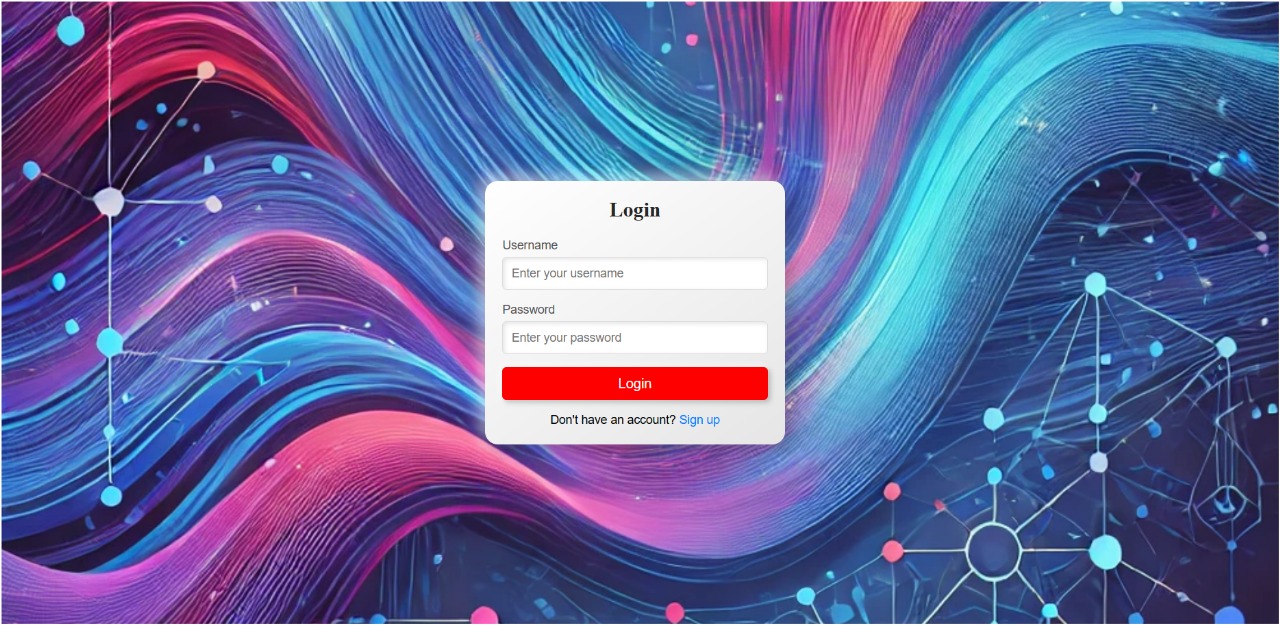


**EXPECTED OUTPUTS:**

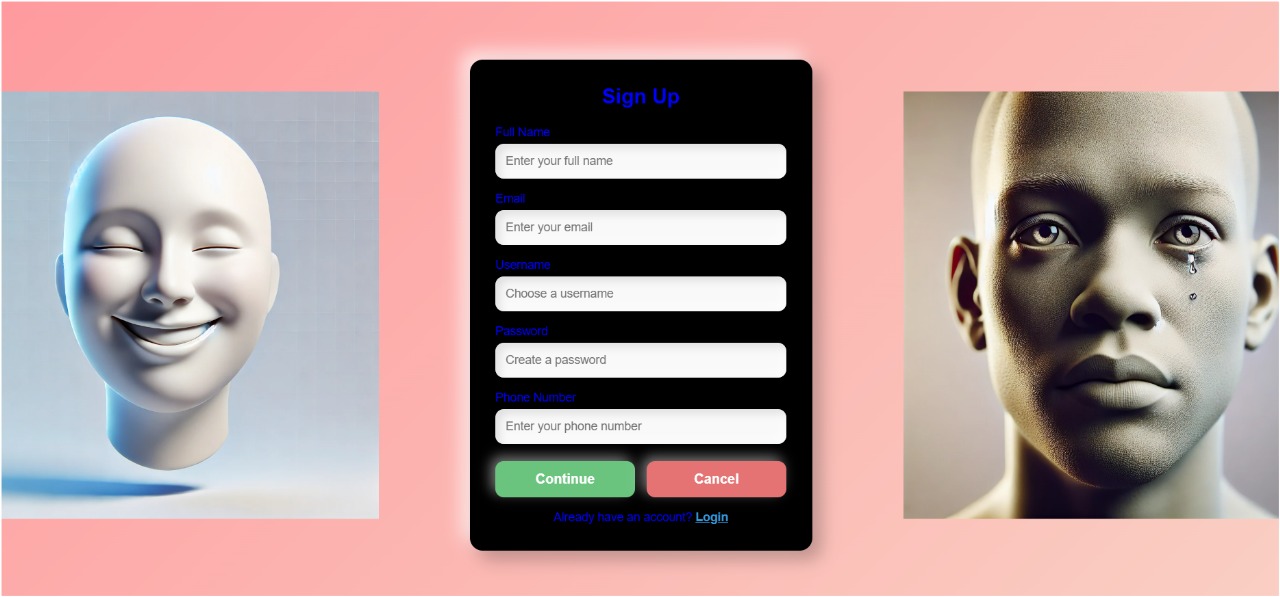


***E) Website Design(Front-End):***

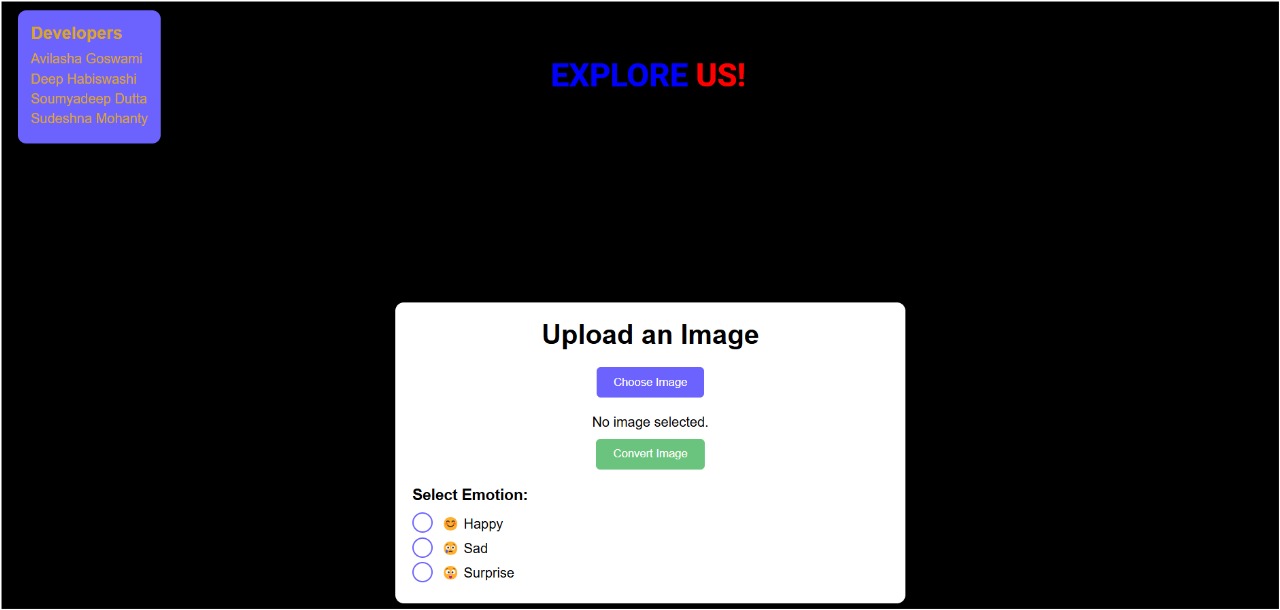
**Login Page-**



**Sign-Up Page:**

****

**Prompt Page:**

****

**F)BACKEND(DEPLOYMENT)**

We are thinking of deploying using Python-Flask as the backend. No work has been started on the backend yet, we are still training the models required.