Appendix A

Source code

A.1 Expressionmodel.py

```
import os
import sys

import numpy as np
import tensorflow as tf

from facialexpression.utils import *

EMOTIONS = ['angry', 'disgusted', 'fearful', 'happy', 'sad', '
    surprised', 'neutral']

def deepnn(x):
  x_image = tf.reshape(x, [-1, 48, 48, 1])
  # conv1
  W_conv1 = weight_variables([5, 5, 1, 64])
  b_conv1 = bias_variable([64])
  h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
  # pool1
  h_pool1 = maxpool(h_conv1)
  # norm1
  norm1 = tf.nn.lrn(h_pool1, 4, bias=1.0, alpha=0.001 / 9.0, beta
    =0.75)

  # conv2
  W_conv2 = weight_variables([3, 3, 64, 64])
  b_conv2 = bias_variable([64])
  h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
  norm2 = tf.nn.lrn(h_conv2, 4, bias=1.0, alpha=0.001 / 9.0, beta
    =0.75)
  h_pool2 = maxpool(norm2)

  # Fully connected layer
  W_fc1 = weight_variables([12 * 12 * 64, 384])
  b_fc1 = bias_variable([384])
  h_conv3_flat = tf.reshape(h_pool2, [-1, 12 * 12 * 64])
  h_fc1 = tf.nn.relu(tf.matmul(h_conv3_flat, W_fc1) + b_fc1)

  # Fully connected layer
```

```
  W_fc2 = weight_variables([384, 192])
  b_fc2 = bias_variable([192])
  h_fc2 = tf.matmul(h_fc1, W_fc2) + b_fc2

  # linear
  W_fc3 = weight_variables([192, 7])
  b_fc3 = bias_variable([7])
  y_conv = tf.add(tf.matmul(h_fc2, W_fc3), b_fc3)

  return y_conv


def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')


def maxpool(x):
  return tf.nn.max_pool(x, ksize=[1, 3, 3, 1],
                        strides=[1, 2, 2, 1], padding='SAME')


def weight_variables(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)


def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)


def train_model(train_data):
  fer2013 = input_data(train_data)
  max_train_steps = 30001

  x = tf.placeholder(tf.float32, [None, 2304])
  y_ = tf.placeholder(tf.float32, [None, 7])

  y_conv = deepnn(x)

  cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_conv)
   )
  train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
  correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_,
   1))
  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

  with tf.Session() as sess:
    saver = tf.train.Saver()
    sess.run(tf.global_variables_initializer())
    for step in range(max_train_steps):
      batch = fer2013.train.next_batch(50)
      if step % 100 == 0:
        train_accuracy = accuracy.eval(feed_dict={
          x: batch[0], y_: batch[1]})
        print('step %d, training accuracy %g' % (step, train_accuracy
   ))
```

```
      train_step.run(feed_dict={x: batch[0], y_: batch[1]})

      if step + 1 == max_train_steps:
        saver.save(sess, './models/emotion_model', global_step=step +
    1)
      if step % 1000 == 0:
        print('*Test accuracy %g' % accuracy.eval(feed_dict={
          x: fer2013.validation.images, y_: fer2013.validation.labels
   }))


def predict(image=[[0.1] * 2304]):
  x = tf.placeholder(tf.float32, [None, 2304])
  y_conv = deepnn(x)

  # init = tf.global_variables_initializer()
  saver = tf.train.Saver()
  probs = tf.nn.softmax(y_conv)
  y_ = tf.argmax(probs)

  with tf.Session() as sess:
    # assert os.path.exists('/tmp/models/emotion_model')
    ckpt = tf.train.get_checkpoint_state('./models')
    print(ckpt.model_checkpoint_path)
    if ckpt and ckpt.model_checkpoint_path:
      saver.restore(sess, ckpt.model_checkpoint_path)
      print('Restore ssss')
    return sess.run(probs, feed_dict={x: image})


def image_to_tensor(image):
  tensor = np.asarray(image).reshape(-1, 2304) * 1 / 255.0
  return tensor


def valid_model(modelPath, validFile):
  x = tf.placeholder(tf.float32, [None, 2304])
  y_conv = deepnn(x)
  probs = tf.nn.softmax(y_conv)

  saver = tf.train.Saver()
  ckpt = tf.train.get_checkpoint_state(modelPath)

  with tf.Session() as sess:
    print(ckpt.model_checkpoint_path)
    if ckpt and ckpt.model_checkpoint_path:
    saver.restore(sess, ckpt.model_checkpoint_path)
      print('Restore model sucsses!!')

    files = os.listdir(validFile)

    for file in files:
      if file.endswith('.jpg'):
        image_file = os.path.join(validFile, file)
        image = cv2.imread(image_file, cv2.IMREAD_GRAYSCALE)
        tensor = image_to_tensor(image)
        result = sess.run(probs, feed_dict={x: tensor})
        print(file, EMOTIONS[result.argmax()])
```

A.2 Index.html

```
{% load  static%}
<!DOCTYPE HTML>
<html>

<head>
<title>feedback analyser</title>
<link rel="stylesheet" type="text/css" href="{% static 'style/style.
    css'%}" title="style" />
</head>

<body>
 <div id="main">
  <div id="header">
   <div id="logo">
    <div id="logo_text">
     <!-- class="logo_colour", allows you to change the colour of the
     text -->
     <h3>
      <center><a href="#"><font color="white" size="5">Feedback
    Detection</font></a></center>
     </h3>
     <br/><br/>
    </div>
   </div>
  </div>
  <div id="content_header"></div>
  <div id="site_content">
   <div id="content">

    <h1>Login Status : {{message}}</h1>

    <form name="form" action="/loginaction/">
     {% csrf_token %}
     <div class="form_settings">

      <p>
       <span>User Name :</span><input class="contact" type="text"
        name="username" value="" />
      </p>
      <p>
       <span>Password :</span><input class="contact" type="password"
        name="password" value="" />
      </p>
      <p style="padding-top: 15px">
       <span> </span><input class="submit" type="submit"
        name="contact_submitted" value="Login" />
      </p>
     </div>
    </form>

   </div>
  </div>
 </div>
</body>
</html>
```

A.3 Home.html

```
{% load  static%}
<!DOCTYPE HTML>
<html>

<head>

<link rel="stylesheet" type="text/css" href="{% static 'style/style.
    css'%}" title="style" />

    <style>
        #customers {
          font-family: Arial, Helvetica, sans-serif;
          border-collapse: collapse;
          width: 100%;
        }

        #customers td, #customers th {
          border: 1px solid #ddd;
          padding: 8px;
        }

        #customers tr:nth-child(even){background-color: #f2f2f2;}

        #customers tr:hover {background-color: #ddd;}

        #customers th {
          padding-top: 12px;
          padding-bottom: 12px;
          text-align: left;
          background-color: #4CAF50;
          color: white;
        }
    </style>

</head>

<body>
 <div id="main">
  <div id="header">
   <div id="logo">
    <div id="logo_text">

     <h3>
      <center><a href="#"><font color="white" size="5">Feedback
    Detection</font></a></center>
     </h3>
     <br/><br/>
    </div>
   </div>
            <div id="menubar">
    <ul id="menu">
     <li><a href="/start">Start</a></li>
                    <li><a href="/logout">Logout</a></li>
    </ul>
```

```
    </div>
   </div>
   <div id="content_header"></div>
   <div id="site_content">
    <div id="content">

     <h1>Login Status : {{message}}</h1>

     <table id="customers">
                    <tr>
                        <th>Starting Time</th>
                        <th>Ending Time</th>
                        <th>Positive Count</th>
                        <th>Negative Count</th>
                    </tr>
                      {% for feedback in feedbacks %}

                          <tr>
                              <td>{{ feedback.start_time }}</td>
                              <td>{{ feedback.end_time }}</td>
                              <td>{{ feedback.pcount }}</td>
                              <td>{{ feedback.ncount }}</td>
                          </tr>

                      {% endfor %}

                  </table>

    </div>
   </div>
  </div>
</body>
</html>
```

A.4 Results.html

```
{% load  static%}
<!DOCTYPE HTML>
<html>

<head>
<title>face expression</title>
<link rel="stylesheet" type="text/css" href="{% static 'style/style.
    css'%}" title="style" />

    <script>
window.onload = function () {

var chart = new CanvasJS.Chart("chartContainer", {
 animationEnabled: true,
 theme: "light2", // "light1", "light2", "dark1", "dark2"
```

```
 title:{
  text: "Feedback"
 },
 axisY: {
  title: "count"
 },
 data: [{
  type: "column",
  showInLegend: true,
  legendMarkerColor: "green",
  legendText: "Positive vs Negative",
  dataPoints: [

      { y: {{ ncount }}, label: "Negative" },
   { y: {{ pcount }}, label: "Positive" },
  ]
 }]
});
chart.render();
}
</script>
</head>

<body>
 <div id="main">
  <div id="header">
   <div id="logo">
    <div id="logo_text">
     <!-- class="logo_colour", allows you to change the colour of the text -->
     <h3>
      <center><a href="#"><font color="white" size="5">Feedback Detection</font></a></center>
     </h3>
     <br/><br/>
    </div>
   </div>
            <div id="menubar">
    <ul id="menu">
     <li><a href="/start">Start</a></li>
                    <li><a href="/logout">Logout</a></li>
    </ul>
   </div>
  </div>
  <div id="content_header"></div>
  <div id="site_content">
   <div id="content">

    <div id="chartContainer" style="height: 300px; width: 100%;"></div>
                <script src="https://cdn.canvasjs.com/canvasjs.min.js"></script>

   </div>
  </div>
 </div>
</body>
</html>
```

A.5 Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', '
    FacialExpressionFeedbackWeb.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and
    "
            "available on your PYTHONPATH environment variable? Did
    you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Appendix B

Screen shots

B.1 Login Page

The Django database login page streamlines user authentication by cross-referencing entered credentials with stored data. Leveraging Django's ORM capabilities, it seamlessly communicates with the database, ensuring robust authentication processes and secure storage of user information.

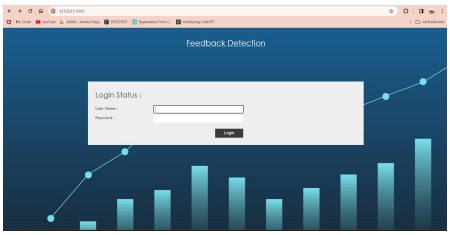

Figure B.1: Login Page

B.2 Login Details

In a Django database login details page, users can view and manage their account information securely. Integrated with Django's ORM, this page enables seamless interaction with the database, ensuring accurate retrieval and updating of user data.

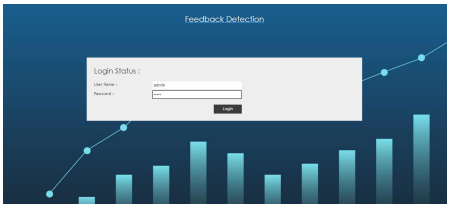

Figure B.2: Login Details

B.3 Admin Home Page

The Django database Admin Home Page provides administrators with an intuitive dashboard to oversee and manage site operations. Leveraging Django's ORM, it grants seamless access to database entities, facilitating efficient data management and system configuration.

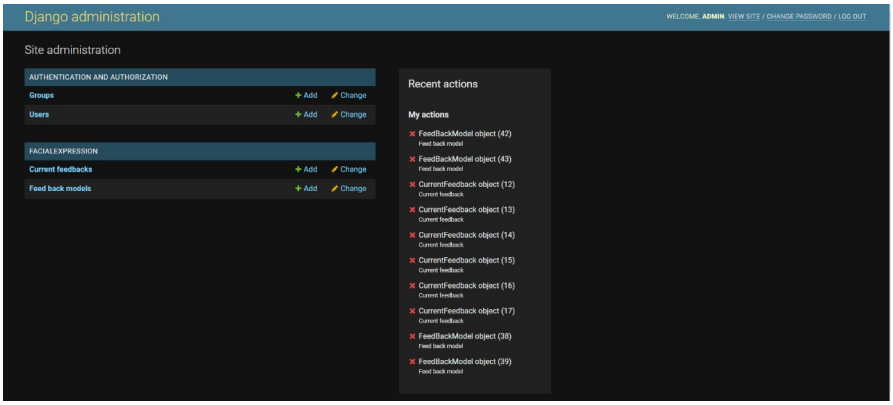

Figure B.3: Admin page

B.4 Facial Recognition

This module is responsible for detecting faces within images or video streams. It identifies the locations of faces, often utilizing techniques like Haar cascades or deep learning-based methods, and provides bounding boxes or facial landmarks for further analysis.

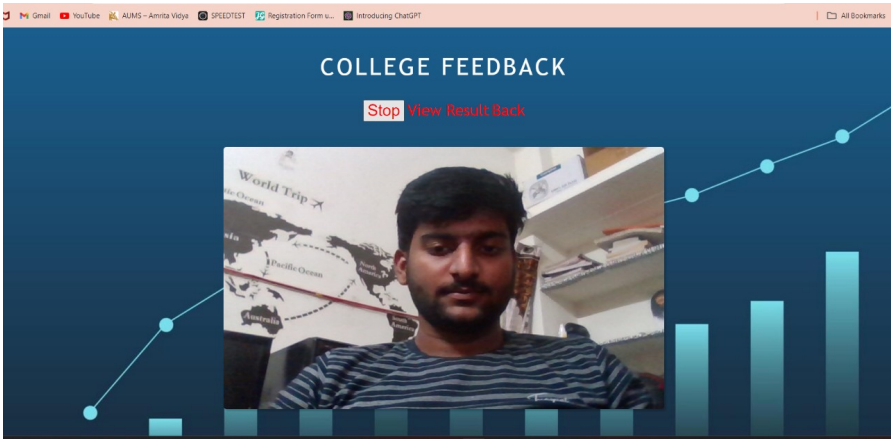

Figure B.4: Facial detection

B.5 Audience Feedback

feedback or recommendations based on the analyzed facial ex- pressions and audience reactions. It provides visualizations, summaries, or actionable insights to presenters or performers to help them improve their communication and en- gagement with the audience.

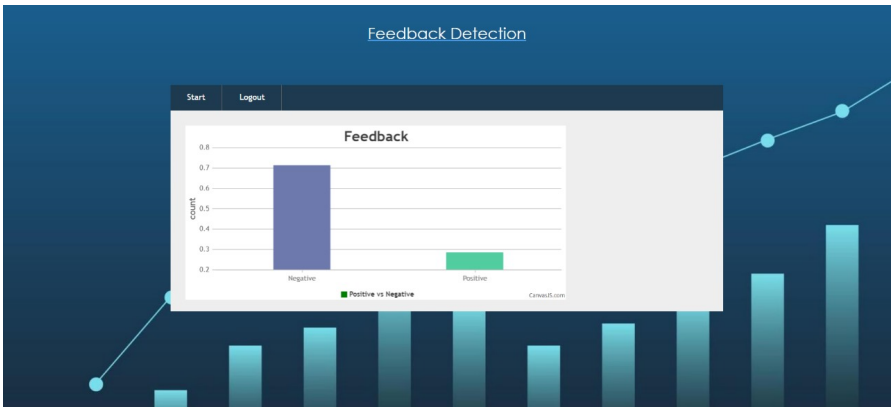

Figure B.5: Audience feedback

Appendix C

Data sets used in the project

C.1 Datasets used

FER2013 (Facial Expression Recognition 2013 Dataset): A dataset consisting of facial images collected from the internet, labeled with seven basic emotions.Size contains over 35,000 images categorized into seven emotion classes.Frequently used for training and benchmarking facial emotion recognition algorithms due to its large size and diverse emotion labels.These datasets provide valuable resources for training and evaluating facial emotion recognition models, enabling researchers and practitioners to develop more accurate and robust systems for analyzing audience emotions and providing real-time feedback.

Name	**Username**	**Phone**	**Password**	**Date**
sai charan	038	9948427070	12345	March 2, 2024
gokul	181	9347644887	56789012	February 21, 2024
yashwanth	235	8712569603	123456	March 10, 2024
Narasimha	458	8341521254	234567	March 15, 2024

Table 6.1: User Data