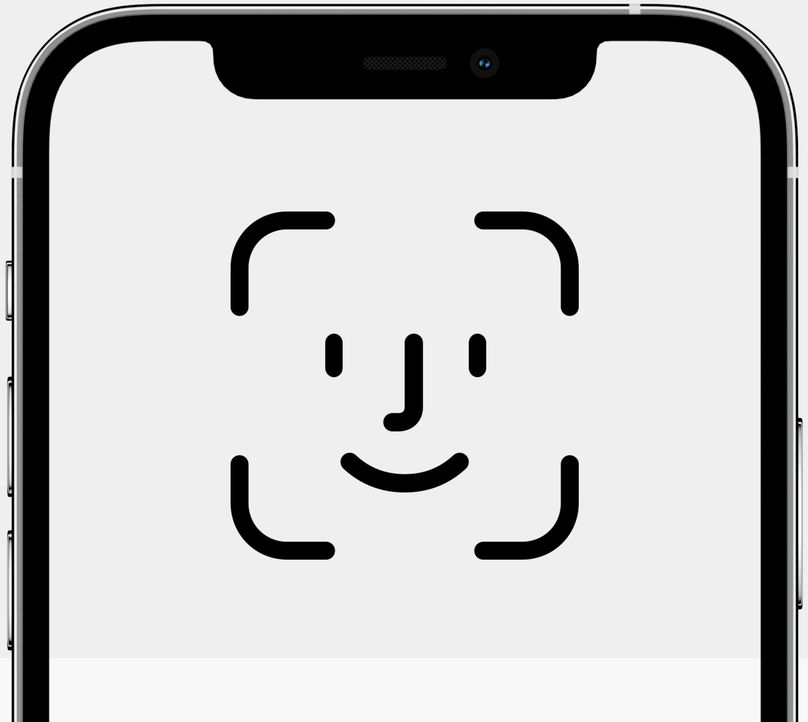
FACIAL RECOGNITION

USER AUTHENICATION



Team Members

6388093 Nitit Ngamphotchanamongkol
6388117 Siranut Akarawuthi
6388206 Tawan Meethong

ITCS461
Computer and
Communication Security

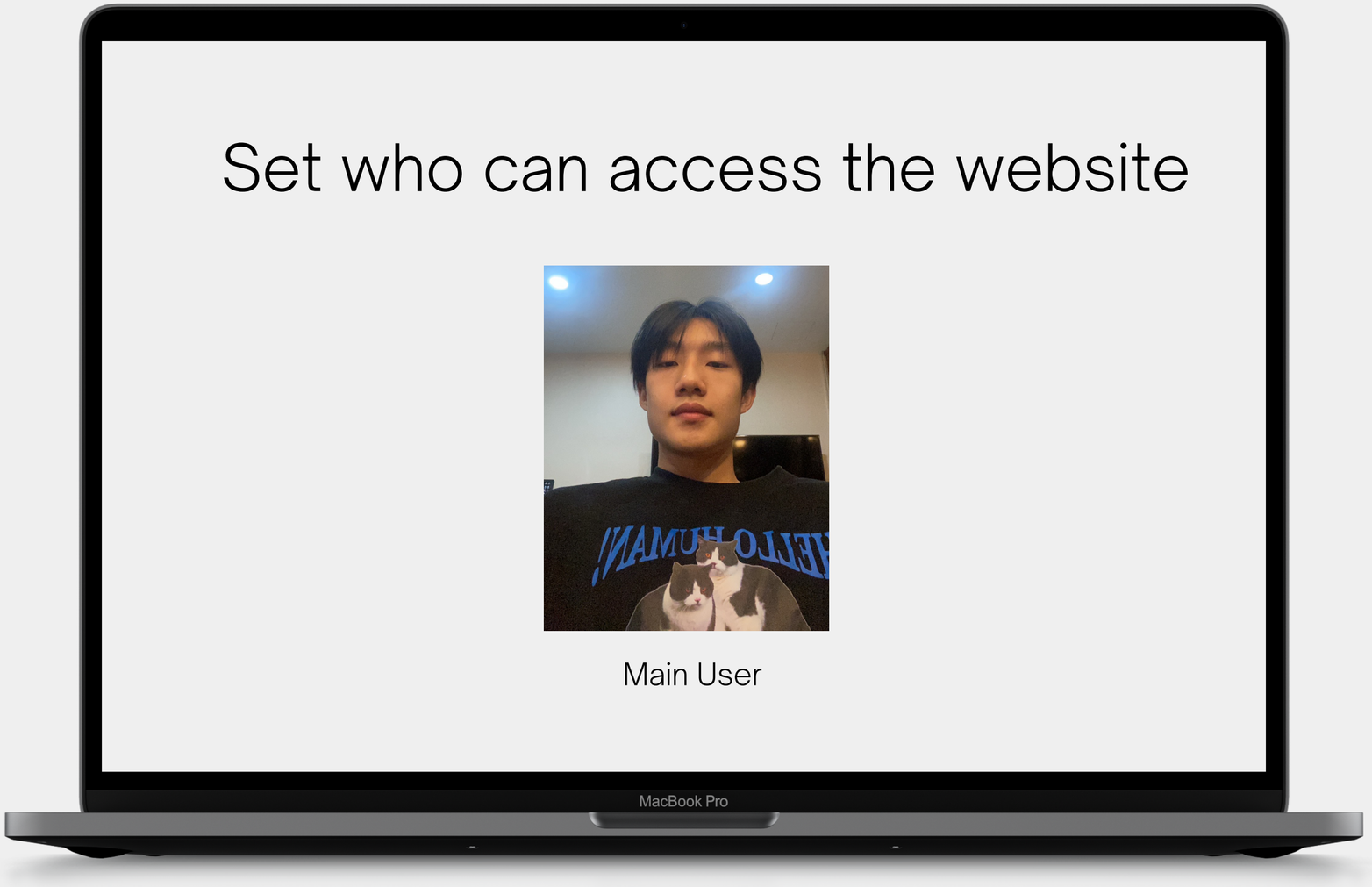
OUR CONCEPT

01

02

03

OUR CONCEPT



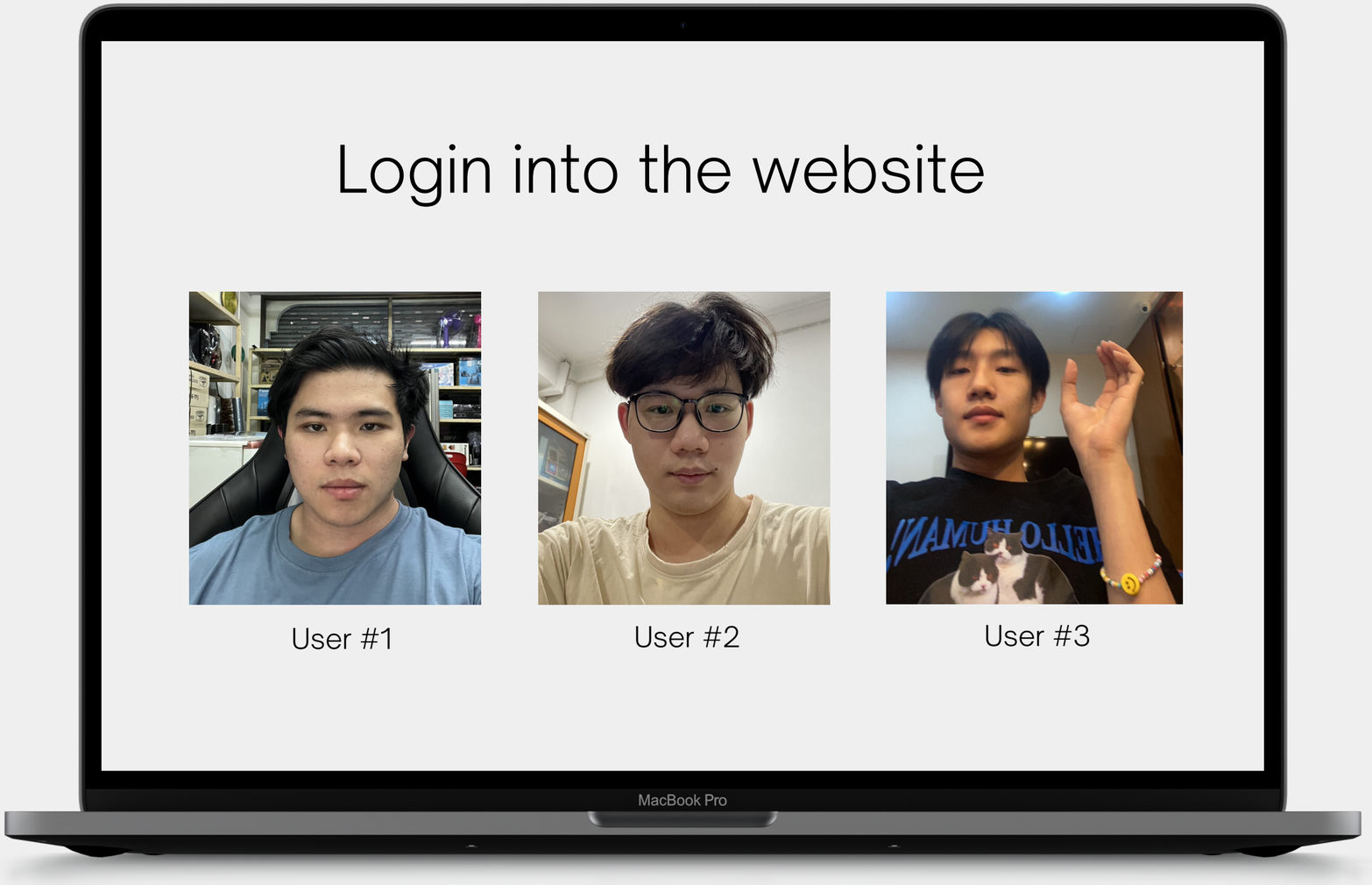
01

02

03

OUR CONCEPT

Main User



User #1

User #2

User #3

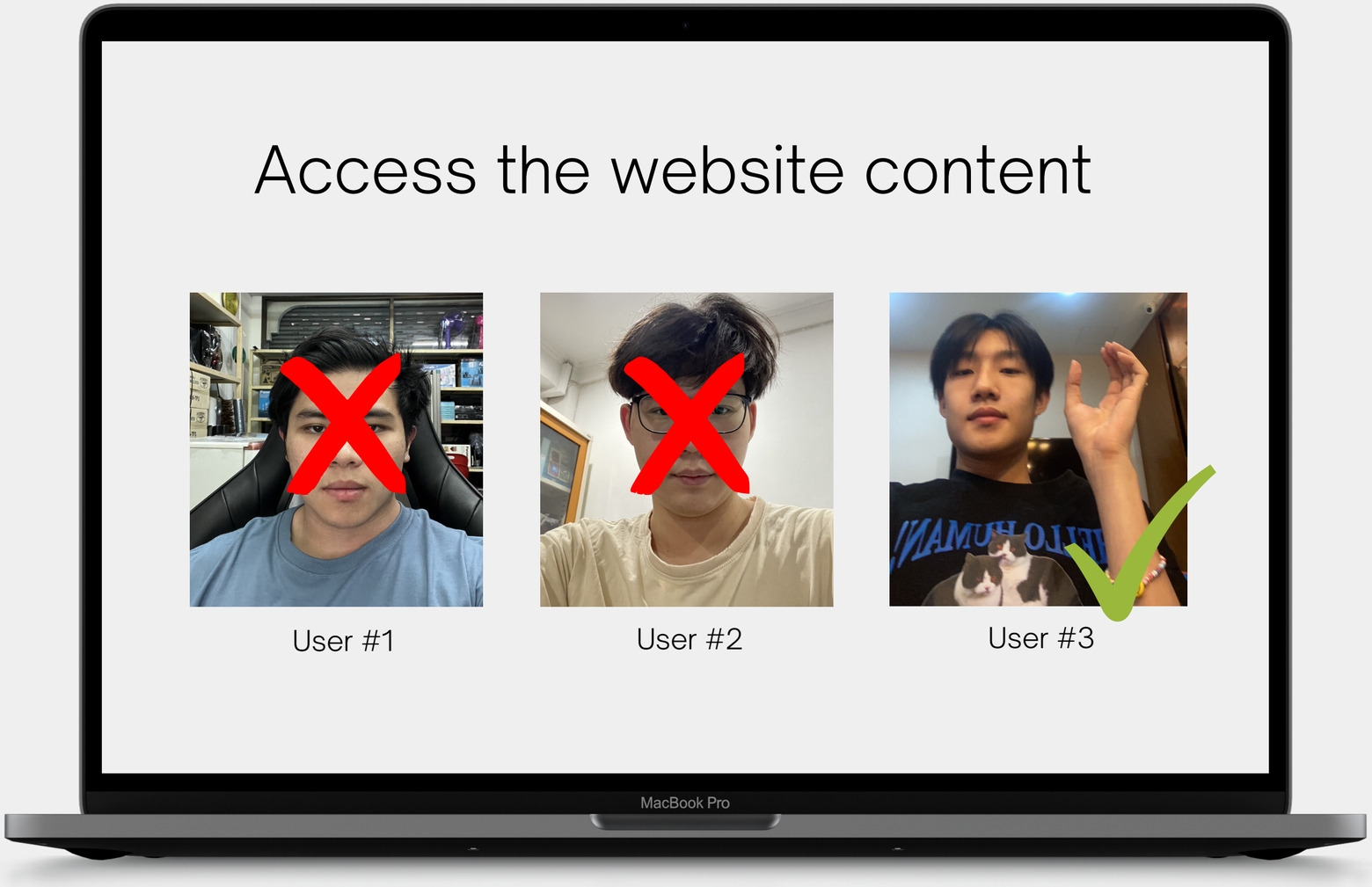
01

02

03

OUR CONCEPT

Main User



User #1

User #2

User #3

01

02

03

OUR CODE

```
import face_recognition
from flask import Flask, render_template, request, redirect, url_for

app = Flask(__name__)

# Dummy data for demo purposes
users = [
    {'username': 'Nitit', 'image_path': 'D:/.Mahidol University/year 3/Security/Project/Data/Nitit_data.png'},
]

# Homepage
@app.route('/')
def index():
    message = ''
    return render_template('index.html', error=message)

# Dashboard page
@app.route('/dashboard')
def dashboard():
  return render_template('dashboard.html')
```

```
@app.route('/login', methods=['POST'])
def login():
  image_file = request.files['fileInput']

  for user in users:
    try:
      known_image = face_recognition.load_image_file(user['image_path'])
      known_encoding = face_recognition.face_encodings(known_image)[0]
      unknown_image = face_recognition.load_image_file(image_file)
      unknown_encoding = face_recognition.face_encodings(unknown_image)[0]
      face_distances = face_recognition.face_distance([known_encoding], unknown_encoding)
    except:
     message = 'We were unable to verify your identity based on the uploaded image. Please try again.'
     return render_template('index.html', error=message)

    print(face_distances[0])

    if face_distances[0] < 0.5:
      return redirect(url_for('dashboard'))

  message = 'We were unable to verify your identity based on the uploaded image. Please try again.'
  return render_template('index.html', error=message)
```

face_recognition

PACKAGE

»

```
known_image = face_recognition.load_image_file(user['image_path'])
known_encoding = face_recognition.face_encodings(known_image)[0]
unknown_image = face_recognition.load_image_file(image_file)
unknown_encoding = face_recognition.face_encodings(unknown_image)[0]
face_distances = face_recognition.face_distance([known_encoding], unknown_encoding)
```

face_recognition.api.load_image_file(file, mode='RGB')

Loads an image file (.jpg, .png, etc) into a numpy array
Parameters

- file – image file name or file object to load
- mode – format to convert the image to. Only 'RGB' (8-bit RGB, 3 channels) and 'L' (black and white) are supported.

Returns image contents as numpy array

```
known_image = face_recognition.load_image_file(user['image_path'])
known_encoding = face_recognition.face_encodings(known_image)[0]
unknown_image = face_recognition.load_image_file(image_file)
unknown_encoding = face_recognition.face_encodings(unknown_image)[0]
face_distances = face_recognition.face_distance([known_encoding], unknown_encoding)
```

face_recognition.api.face_encodings(face_image, known_face_locations=None, num_jitters=1, model='small')
Given an image, return the 128-dimension face encoding for each face in the image.

Parameters
• face_image – The image that contains one or more faces
• known_face_locations – Optional - the bounding boxes of each face if you already know them.
• num_jitters – How many times to re-sample the face when calculating encoding. Higher is more accurate, but slower (i.e. 100 is 100x slower)
• model – Optional - which model to use. "large" or "small" (default) which only returns 5 points but is faster.

Returns A list of 128-dimensional face encodings (one for each face in the image)

```
known_image = face_recognition.load_image_file(user['image_path'])
known_encoding = face_recognition.face_encodings(known_image)[0]
unknown_image = face_recognition.load_image_file(image_file)
unknown_encoding = face_recognition.face_encodings(unknown_image)[0]
face_distances = face_recognition.face_distance([known_encoding], unknown_encoding)
```

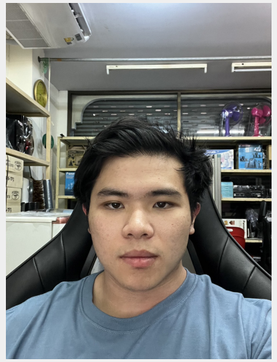
face_recognition.api.face_distance(face_encodings, face_to_compare)

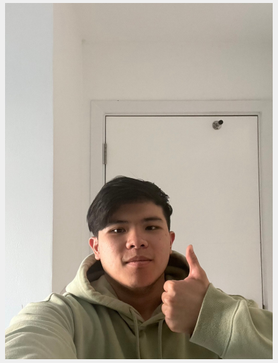
Given a list of face encodings, compare them to a known face encoding and get a euclidean distance for each comparison face. The distance tells you how similar the faces are.

Parameters
- face_encodings – List of face encodings to compare
- face_to_compare – A face encoding to compare against

Returns A numpy ndarray with the distance for each face in the same order as the ‘faces’ array

TEST RESULTS

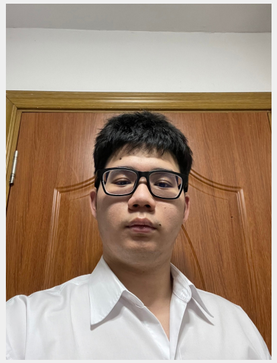


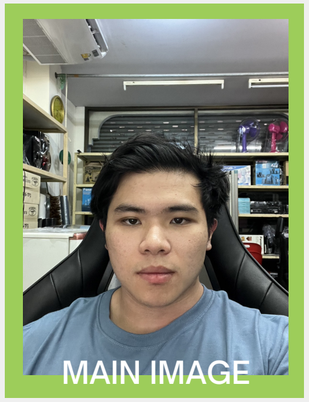


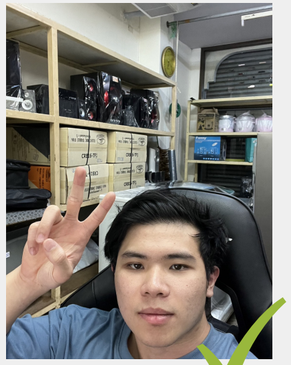







0.19861786


0.48672777

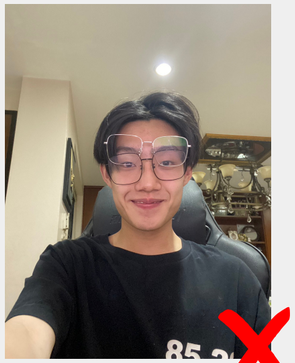

0.33977759


0.34989035


0.40692876


0.63185175

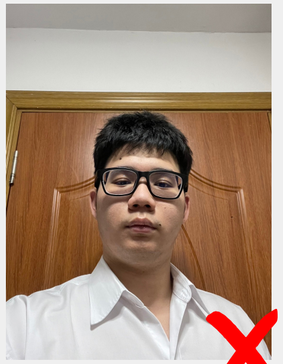

0.62124515

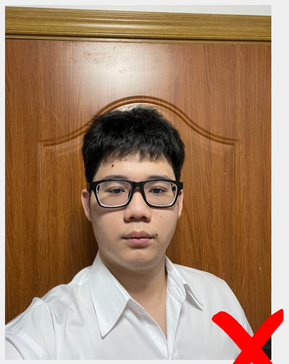

0.65604795


0.56942761

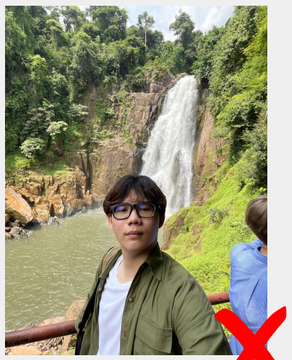

0.57852109


0.6644986


0.50995695


0.52782246

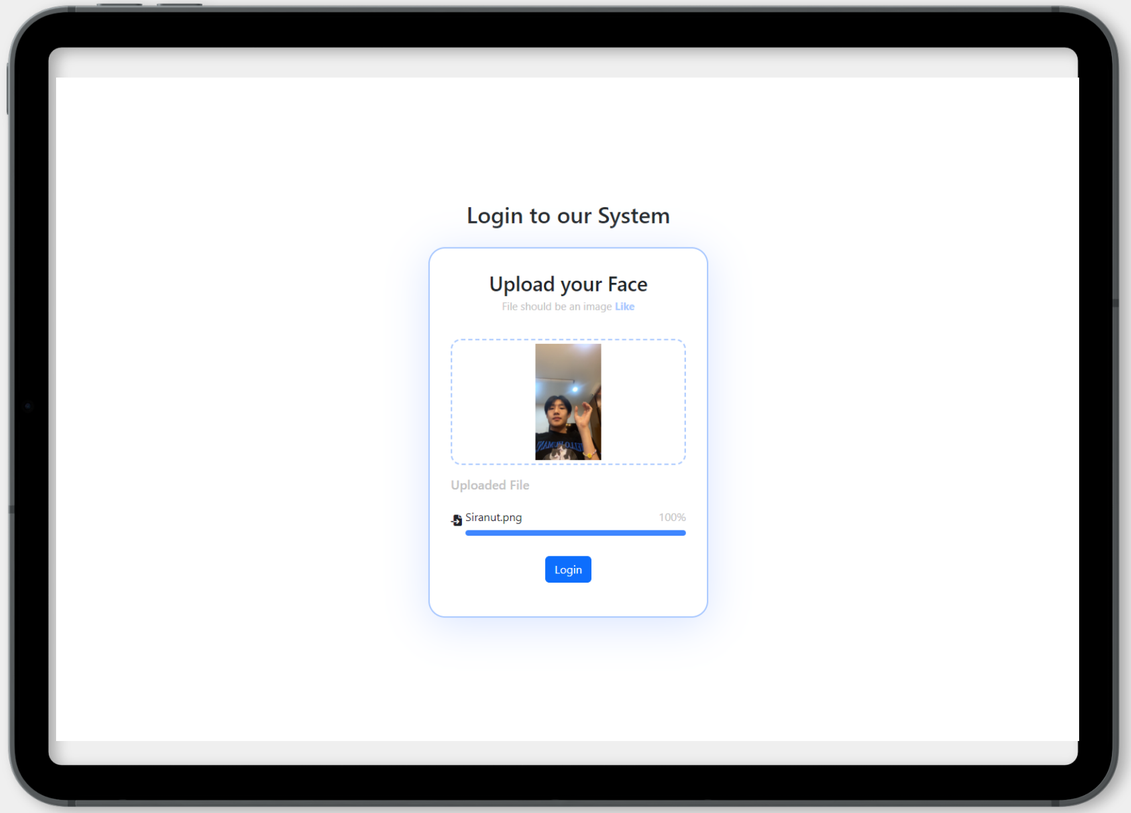

0.52416827


0.52756898


0.57681206


0.52480254

LIVE DEMO


Login to our System
Upload your Face
File should be an image Like
Uploaded File
Siranut.png
100%
Login