Project-3: Automated Attendance Marking Using Computer Vision

Team members: Nolfin Ilya, Aimat Kulmakhan, Olzhas Yergali

Introduction

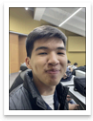
The objective of this project is to automate attendance marking for lecture halls by analyzing a single photo or video of a classroom. The system will identify individual students in the image, cross-reference them with a predefined list, and mark them as "attended" in a simple web-based interface.

Objectives

1. Collect and preprocess images of at least 100 students for training and testing.✅
2. Design and train a computer vision model for student identification.✅
3. Develop a simple web interface to display a list of students, where attendance will be marked automatically. ✅
4. Deploy the application to demonstrate real-time attendance marking using classroom photos or videos. ✅

Part 1: Data Collection and Preparation

1. Collect labeled images of at least 100 students from classroom photos or videos.


Galymzhan Zhanabay.jpg


Gaziz Yermekuly.jpg

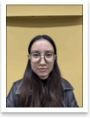

Gulzhan Kurmanov


Gulzhan Yesenov


Iliyas Nartay.jpg


Ilya Nolfin.jpg

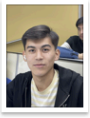

Inkar Khairatkyzy.jpg


Korkem Maxutbek.jpg

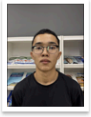

Lashyn Sholbay.jpg


Madina Mukashov


Madina Urazalin


Madiyar Tolen.jpg


Maksym


Miras

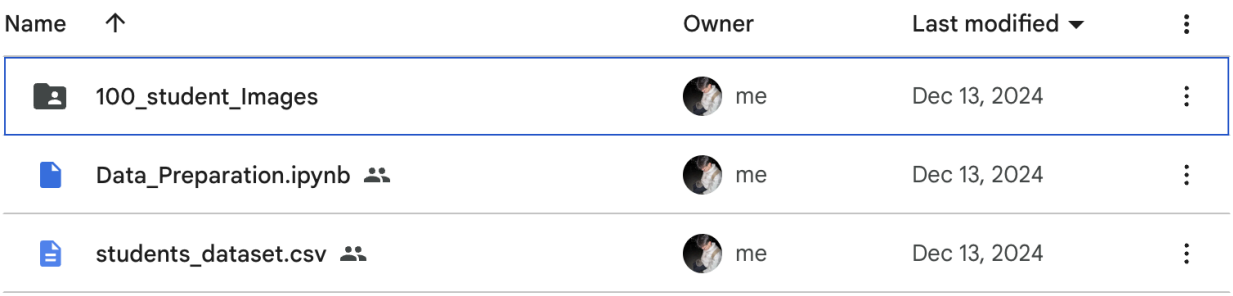

Mukhtar Agai


Nuradil Kanat.jpg


Nurdaulet

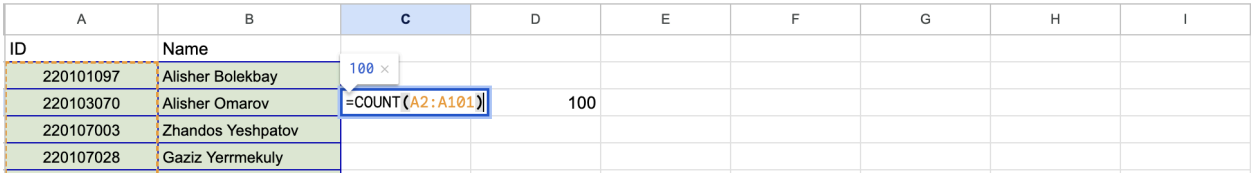

Nuri Sapak.jpg

2.Preprocess the data, including resizing, normalization, and labeling.
Labeled dataset and a Jupyter Notebook showcasing preprocessing steps yo



u can find here:
https://drive.google.com/drive/folders/1n5bYdG362a7_5jZazXrulQbAgVxepvpo?usp=sharing

There is a students_dataset.csv that contain 100 students with ID and Names.



There is a piece of DataPrepation.ipynb code. Full code you can see on drive.

Import libraries

```
[ ]  import tensorflow as tf
     import numpy as np
     import cv2
     import os
     import matplotlib.pyplot as plt
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.preprocessing import image
```

Convert Images to JPG and save it

```
import os
import shutil
import pyheif
from PIL import Image
import matplotlib.pyplot as plt
import zipfile

def convert_all_to_jpg(base_dir):
    for root, _, files in os.walk(base_dir):
        for file in files:
            file_path = os.path.join(root, file)
```

Stages of Data Preparation:

1. Unzip folder with images
2. Convert Images from HEIC to JPG format
3. Crop Images (to have only faces on image)
    a. Resizing images to 224x224 pixels
    b. Using DeepFace to Crop only face.
4. Label Images by Student names

Label Images by Name of Student

```
[ ]  import os
     import shutil

     source_dir = "/Users/nolfinil/Downloads/All faces for ML Final"
     new_dir = "/Users/nolfinil/Downloads/newTrainData"
```

Part 1 is Finished

Part 2: Model Development and Training

1.Train a computer vision model (e.g., a CNN or a transformer-based model) to recognize students' faces from images.

Importing Libraries

```
[ ]  from PIL import Image, ImageDraw
     import face_recognition
     from matplotlib import pyplot as plt
     import matplotlib.image as mpimg
     import face_recognition
     import pandas as pd
```

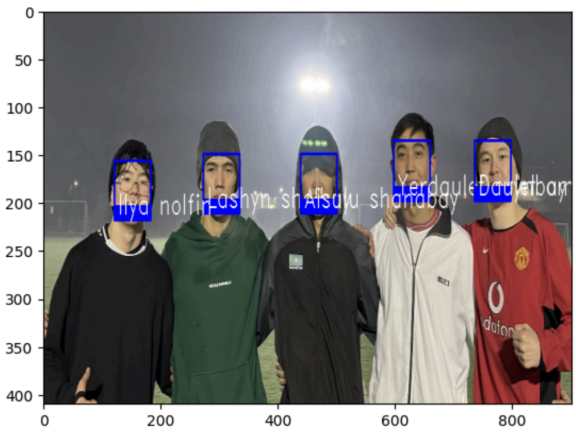
First Training For our model

```
[ ]  import cv2
     import numpy as np
     import os

     path = "/Users/nolfinil/Downloads/newTrainData"

     known_names = []
     known_name_encodings = []
     attendance = {}
     recognized_names = []
     images = os.listdir(path)
```

```
Name of Students:  ['Nuri sapak', 'Zhanna urazalin', 'Serik kazhibek',
Yerdaulet orynbay: Present
Lashyn sholbay: Present
Aisulu shanabay: Present
Daulet omarov: Present
Ilya nolfin: Present
<matplotlib.image.AxesImage at 0x124dd4c50>
```



2. Save model

second training and saving model

```
import pickle
import face_recognition
import os
import cv2
import numpy as np

TRAINED_DATA_PATH = "/Users/nol

def load_saved_training_data():
    if os.path.exists(TRAINED_D
        with open(TRAINED_DATA_
            known_names, known_
        # print("Loaded saved t
        return known_names, kno
    else:
        raise FileNotFoundError
```

```
plt.imshow(processed_image, aspect='auto')
```

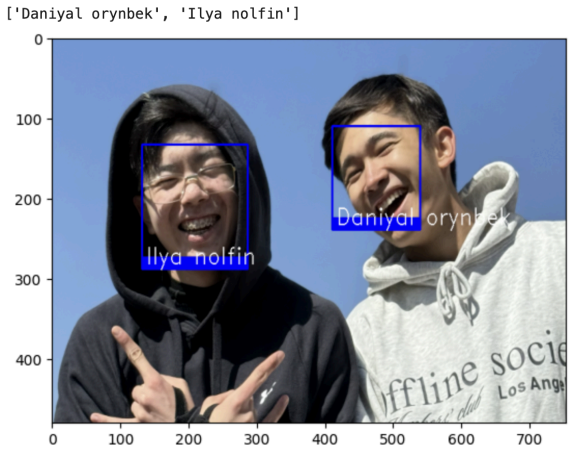
```
<matplotlib.image.AxesImage at 0x1212a1490>
```

3.Validate the model using appropriate metrics like accuracy and precision.

testing and validation

```
test_path = "/Users/nolfinil/Downloads/test_dataset"
images = os.listdir(test_path)
all_names_2d = []
all_names = []
for _ in images:
    image_path = test_path + "/" + _
    if _ == '.DS_Store':
        continue
    with open(image_path, "rb") as test_image_file:
        attendance_data, processed_image, recognized_names = predict_children(test_image_file)
    print(recognized_names)
    all_names_2d.append(recognized_names)
    all_names.extend(recognized_names)
    plt.imshow(processed_image, aspect='auto')
    plt.show()
```

4.Results Images



5.Accuracy

checking accuracy

```python
if len(all_names) != len(y_true):
    min_length = min(len(all_names), len(y_true))
    all_names = all_names[:min_length]
    y_true = y_true[:min_length]

# matches
matches = sum(1 for true, pred in zip(y_true, all_names) if true == pred)

accuracy = matches / len(y_true)

print(f"Accuracy: {accuracy:.2%}")
```

```
Accuracy: 80.53%
```

6.Model Evaluation

calculating scores

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

accuracy = accuracy_score(y_true, all_names)
precision = precision_score(y_true, all_names, average='macro', zero_division=1)
recall = recall_score(y_true, all_names, average='macro', zero_division=1)
f1_macro = f1_score(y_true, all_names, average='macro', zero_division=1)
f1_micro = f1_score(y_true, all_names, average='micro', zero_division=1)

#results
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 (macro) Score: {f1_macro:.2f}")
print(f"F1 (micro) Score: {f1_micro:.2f}")
```

```
Accuracy: 0.81
Precision: 0.63
Recall: 0.97
F1 (macro) Score: 0.63
F1 (micro) Score: 0.81
```

Full Code you can see on drive.

What is the Face Recognition Library?

Face Recognition

Recognize and manipulate faces from Python or from the command line with
the world's simplest face recognition library.
Built using dlib's state-of-the-art face recognition
built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

Labeled Faces in the Wild Home

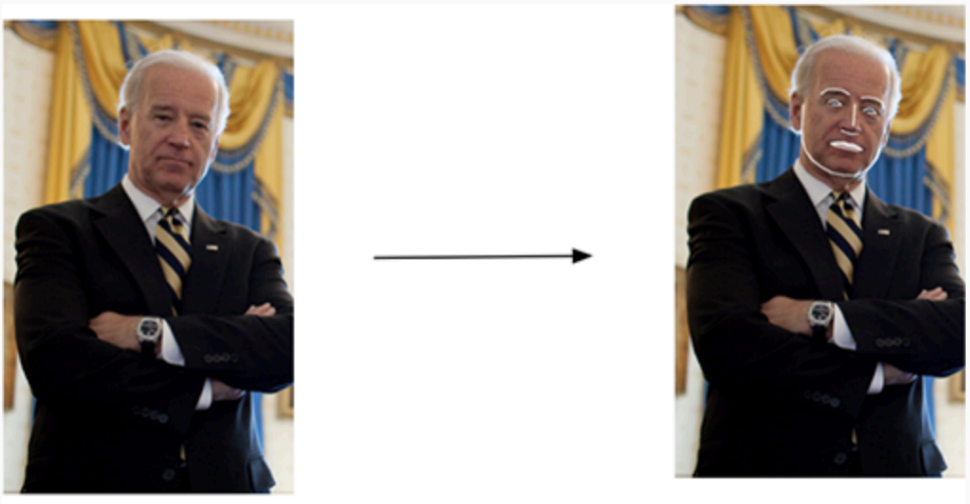
Information:

- 13233 images
- 5749 people
- 1680 people with two or more images

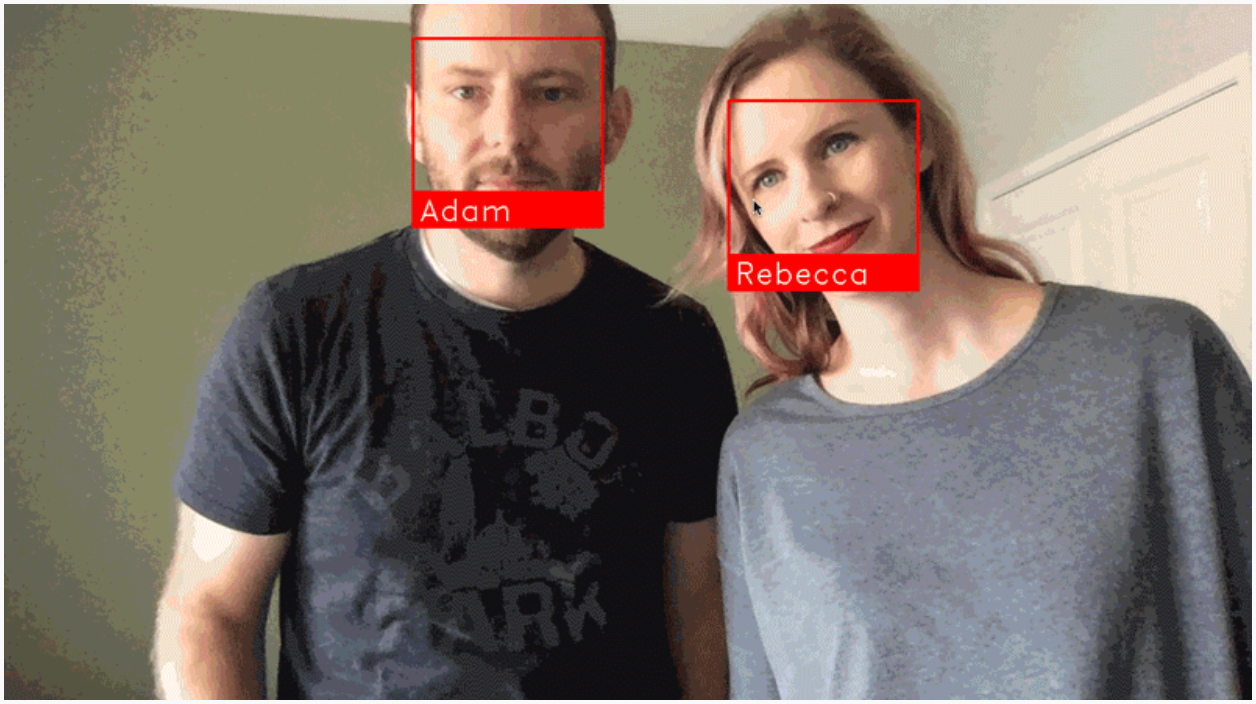
```
face_locations = face_recognition.face_locations(image)
```

Find and manipulate facial features in pictures

Get the locations and outlines of each person's eyes, nose, mouth and chin.



Input Output




```
face_locations = face_recognition.face_locations(image)

face_encodings = face_recognition.face_encodings(image,
face_locations)

for (top, right, bottom, left), face_encoding in
zip(face_locations, face_encodings):

    matches =
    face_recognition.compare_faces(known_name_encodings,
    face_encoding)

    name = "" face_distances =
    face_recognition.face_distance(known_name_encodings,
    face_encoding) best_match = np.argmin(face_distances)
```

Our algorithm compare face locations and face encodings of all students that model has trained with face locations and face encodings of tested photos.

Conclusion for part 2:

In summary, our automated face recognition attendance checking system achieved notable results. Utilizing a face recognition library, we obtained an accuracy of 81%, precision of 63%, recall of 97%, and F1 scores of 63% (macro) and 81% (micro). These metrics demonstrate the system's effectiveness in capturing high recall values, ensuring most of the faces in the dataset are identified. However, precision and F1 scores indicate room for improvement in reducing false positives. Moving forward, further enhancements and training on diverse datasets will be crucial for improving overall performance.

Part 2 is Finished

Part 3: Attendance System Development

1. Build a web application where the system can:
    - ■ Upload a photo or video of the classroom. ✅
    - ■ Identify students present in the image and mark them as "attended." ✅
    - ■ Display the updated attendance list on the webpage.✅
2. Extra Credit (Optional):
    1. Integrate real-time video analysis for live attendance marking. ✅

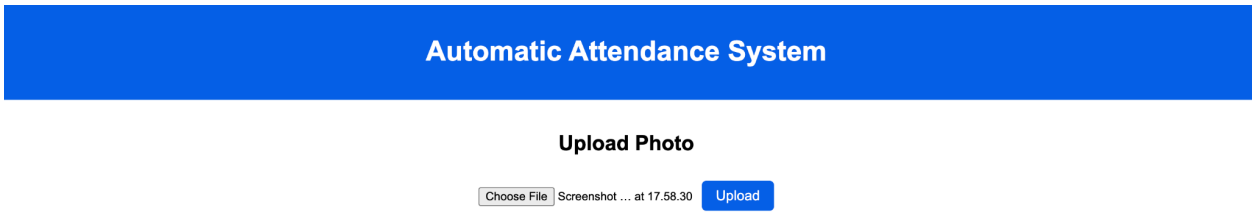
    2. Add security features to verify the authenticity of attendance.
    3. Improve performance using advanced face recognition libraries like FaceNet or DeepFace. We used face recognition library. ✅

3.Functional web application with a demo.

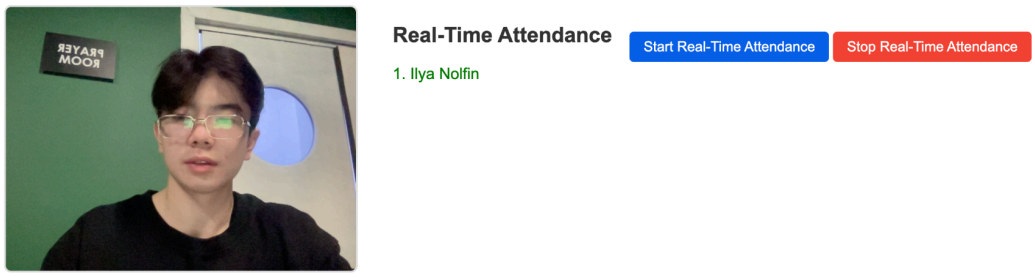
4.Python scripts for integration and deployment.
    Full Code you can see on drive.

Demo Site to show abilities:

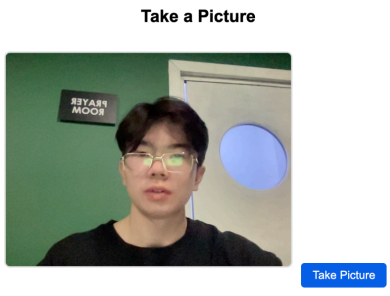
1) Upload photo function



2) Real time attendance

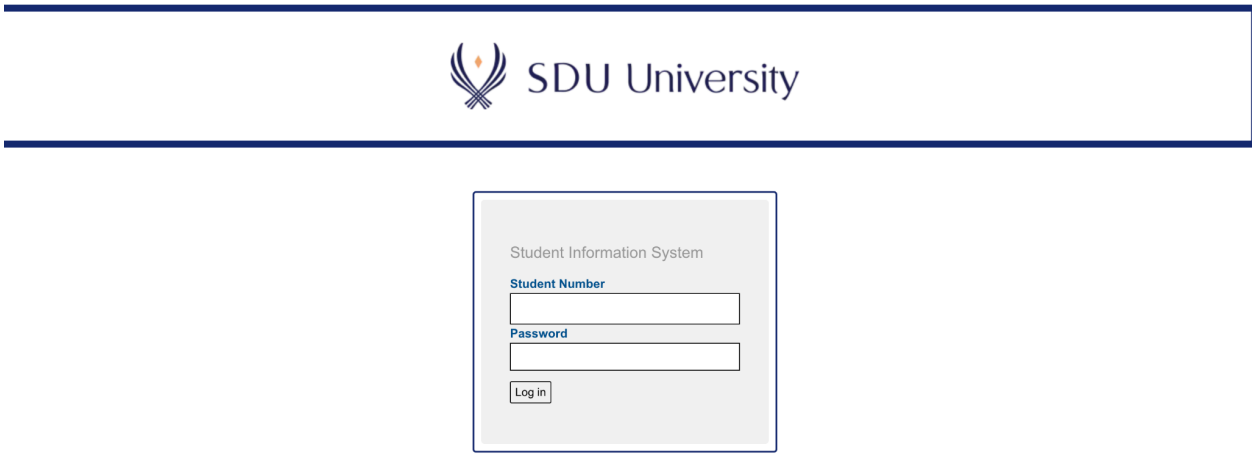


3) Take a picture

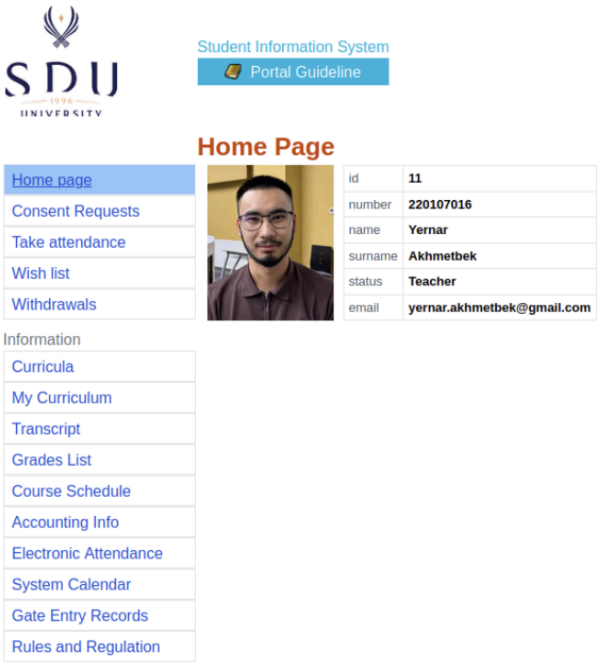


Official site:

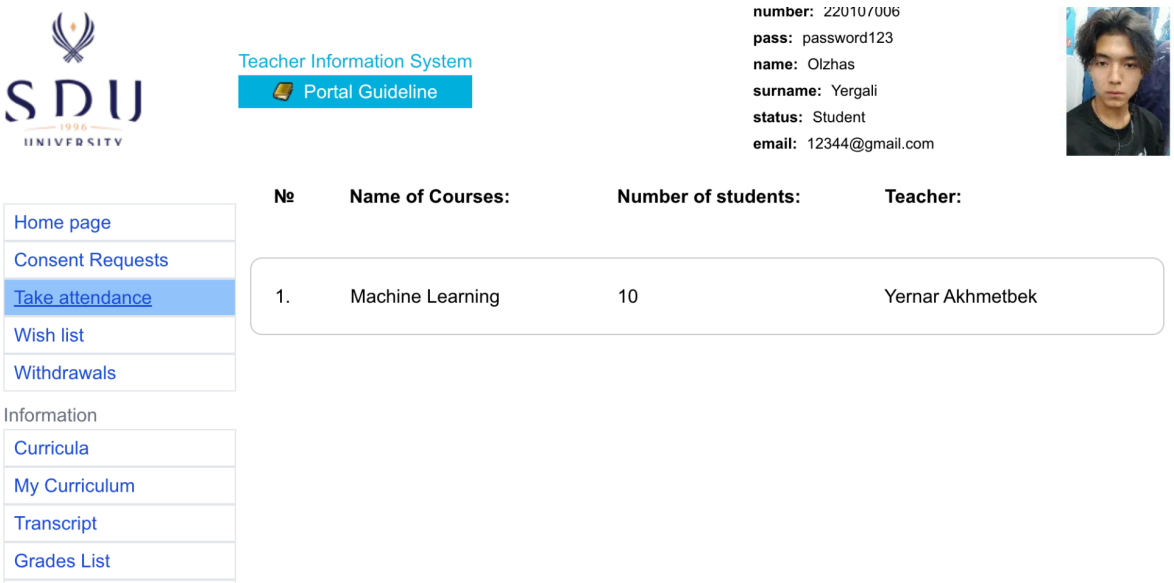
1) Login page



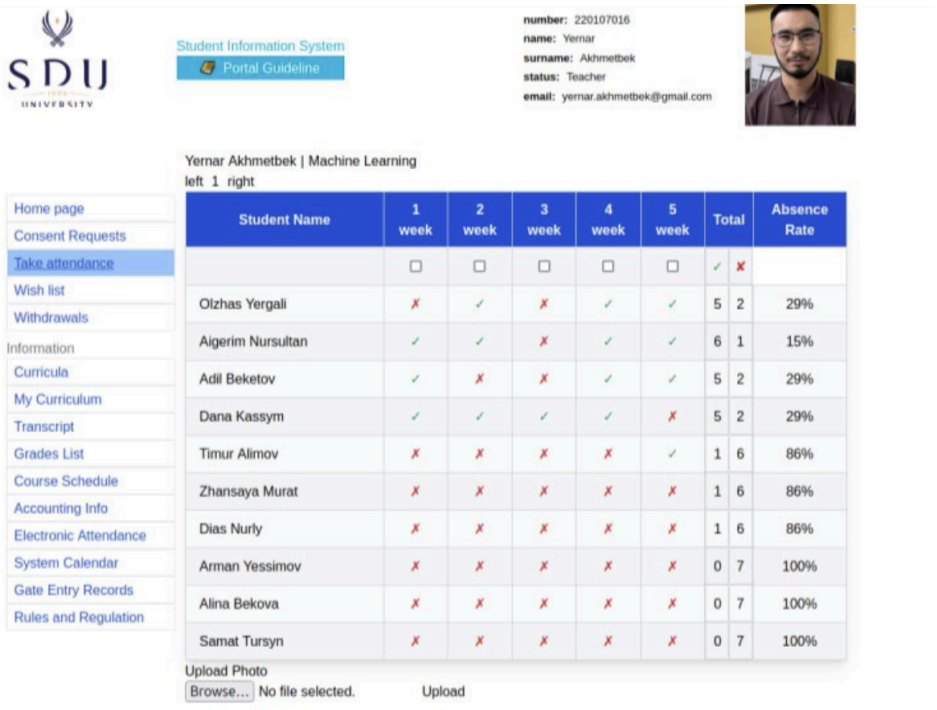
2) Home page



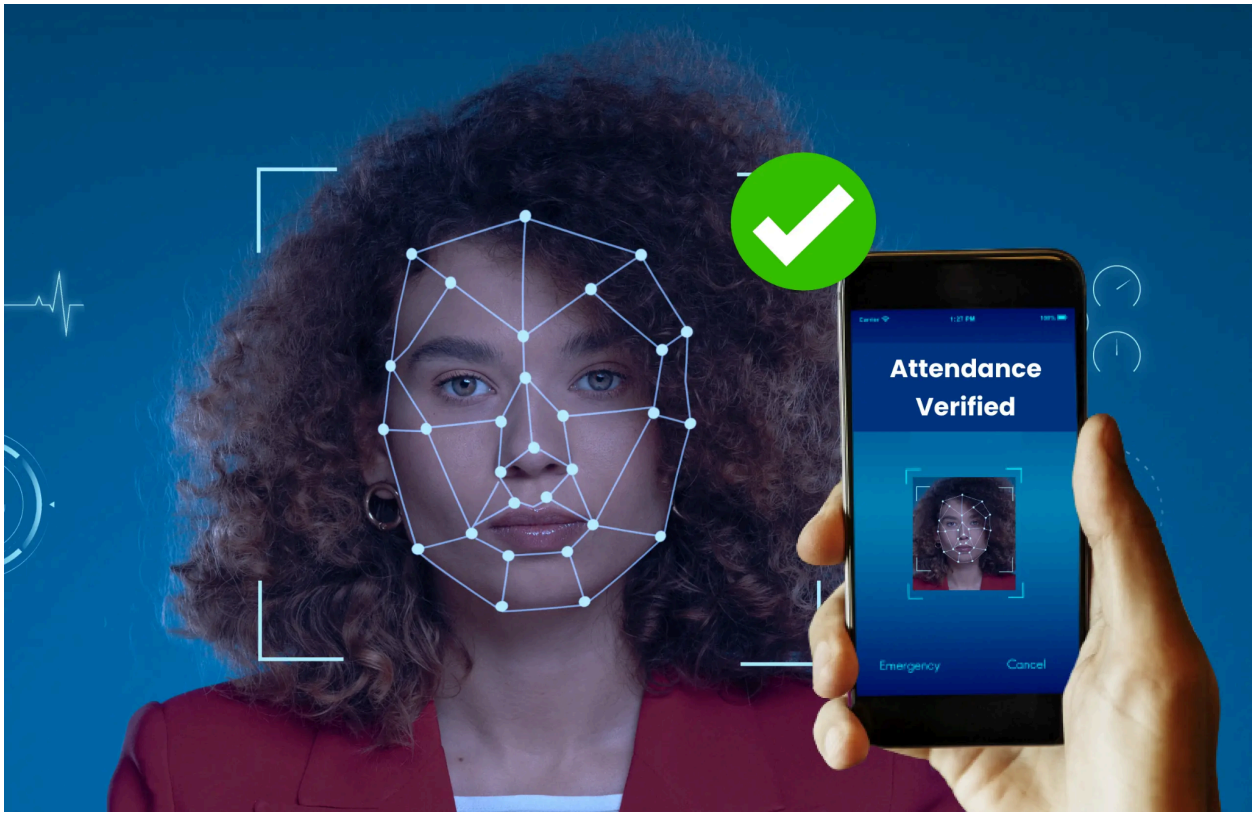
3) Take attendance



4) Taking attendance



Part 3 is finished!!!



Conclusion

Summarizing our results, In this project, we successfully developed an automated attendance system using computer vision to identify and track students' attendance in lecture halls based on classroom images or videos. The system leverages face recognition technology to accurately identify students, marking their attendance with minimal manual intervention.

Project is Finished

Thanks for your attention!