Building a Facial Recognition-Based Attendance System for Beginner in AI

GDSC AI Department
Speakers: Lai Tzi Syuen Suzanne, Jack Pang Chong

Date: 26th of January 2024
Time: 1:30pm - 4:30pm
Location: S-8-01

This document contains the guide to building the facial-recognition based attendance system from scratch, as well as an explanation of what each library in the project does and how you can utilize them for your other projects in the future!

1.0 The libraries and what they do

- cmake
    - cmake is a cross-platform build system used to manage the compilation and linking of software projects. It generates platform-specific build files (e.g., Makefiles) from a high-level configuration.

- dlib
    - Machine Learning: Dlib is a C++ toolkit containing various machine learning algorithms and tools for tasks like:
        - Face detection and recognition
        - Object tracking
        - Image processing
        - Data analysis

- opencv-python
    - OpenCV provides functions for image and video processing, object detection and recognition, feature extraction, and more.
        - You can read images by using this command:
            - **`image = cv2.imread("image.png")`**
        - Or even convert images to another color format:
            - **`gray_ver = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`**
        - And… chances are you'll be using cv2 to show images
            - **`cv2.imshow(image)`**
            - **`cv2.imshow(gray_ver)`**

- face-recognition
    - This library simplifies face recognition tasks using dlib's functionality.
        - Here, you can find faces using the library
            - `image = cv2.imread("group_photo.jpg")`
            - `face_locations = face_locations(image)`

- cvzone
    - cvzone extends OpenCV with additional tools for drawing shapes, creating interactive overlays, and working with webcams.

- firebase-admin
    - This library enables Python applications to interact with Firebase services like Realtime Database, Cloud Storage, and Authentication.
        - For instance, this is how you can access your real-time database
            - `ref = db.reference("users")`

- numpy
    - numpy is one of the most fundamental libraries in Python that you'll become very familiar with over time if you continue to build projects in the language
    - It is THE library used for numerical computing in Python, providing
        - Multidimensional arrays
        - Mathematical functions

        - Linear algebra operations
        - Random number generation
        - Integration with other scientific libraries

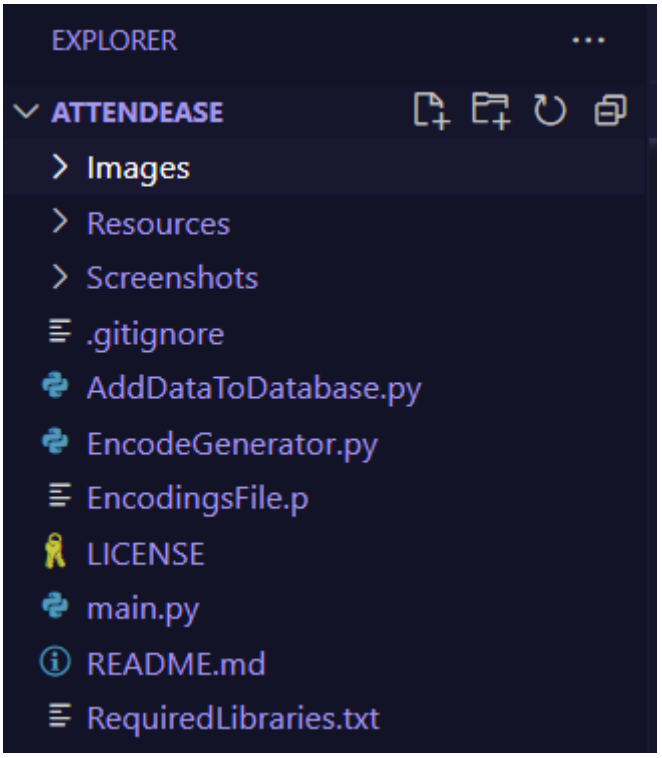
- pickle
    - pickle is a library that is used for the **serialization** of Python objects into byte streams for storage or transmission
        - One of the key commands in pickle is dump
        - **`pickle.dump(object, file)`**
            - The above command serializes the object and saves it into the specified file
        - Another key command is load
        - **`deserialized_date = pickle.load(file)`**
            - The above command reverses the effects of dump and retrieves data from a file and deserializes into a human-readable format

2.0 Step-by-step guide on the project

Hi guys! I'm assuming that you've come from the workshop I hosted on the 26th of January and are ready to claim your certifications, so I wish you luck and I hope this documentation is fun for you to read, and easy for you to follow :] For any questions, don't hesitate to reach me at LinkedIn/through the WhatsApp group so I can help troubleshoot your code!

Just remember that while you goal is to get that certificate, it's important to understand what you're doing as well. I'll be reachable if you have any questions at all + if you don't know what you're doing. We all start somewhere, and I'm constantly learning as well!

That being said, I tried to make this guide as enjoyable as possible, so good luck and let's get started 👍



- This is how my files look like inside my project folder. My project is called **AttendEase**
- Required files and directories:
    - Directories: Images, Resources, Screenshots
    - Files: AddDataToDatabase.py, EncodeGenerator.py, main.py
- You can ignore the rest

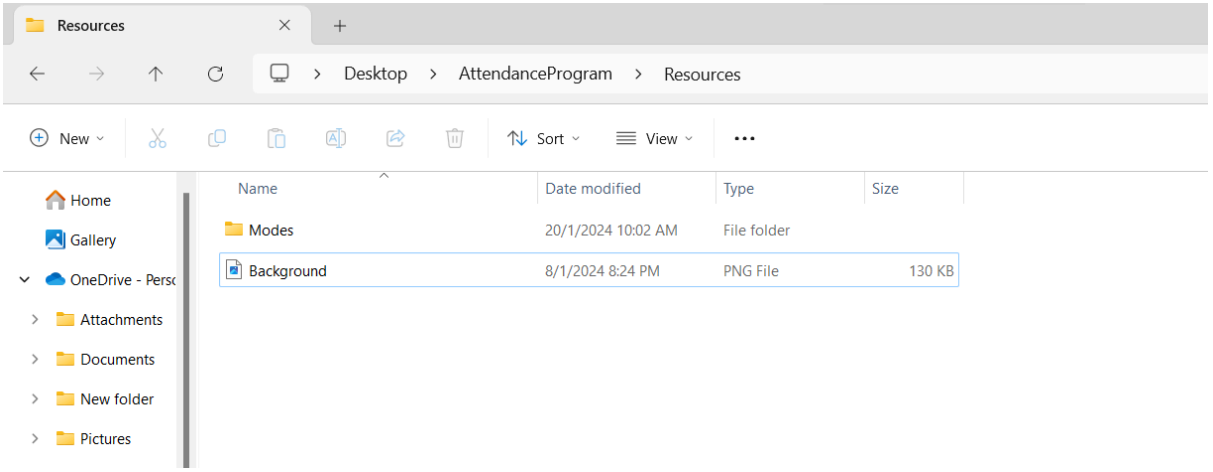
2.1 Webcam

main.py

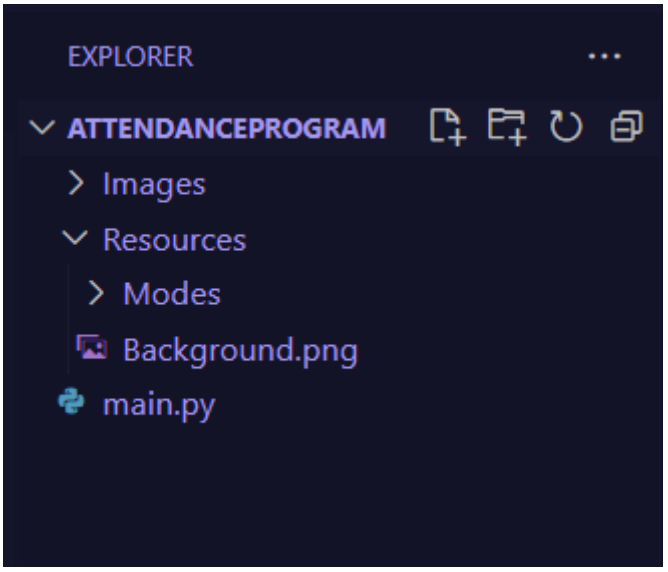
```python
#HIIII This is Suzanne
#This line essentially helps you import the openCV library, a very powerful library for image processing jobs
import cv2

#This line essentially creates a variable called "capture" that is connected to your device's connected camera
capture = cv2.VideoCapture(1)

#The first argument passed into set is to specify we are altering the WIDTH of the object. 1280 is how wide (in pixels) the window is
capture.set(3, 1280)

#The first argument passed into set is to specify we are altering the HEIGHT of the object. 720 is how tall (in pixels) the window is
capture.set(4, 720)

#A creation of an infinite loop is continues running forever unless directed otherwise
while True:
    #from capture (connected video camera), an "image" is taken
    success, image = capture.read()
    #the image is shown
    cv2.imshow("Attendance System", image)
    cv2.waitKey(1)
```

- The code shown above is written in the main document of the project, main.py
- Use capture = cv2.VideoCapture(1) if you are using an EXTERNAL WEBCAM
- User capture = cv2.VideoCapture(0) if you are using your BUILT-IN WEBCAM
- This is all the code you need to initialize and run your most basic of webcams

2.2 Importing graphics



- Using the resources zipped into the file I gave you, you can copy both the "Modes" folder and the Background.png into YOUR own Resources folder inside your Attendance System folder.
- The name of your parent folder **could be different** depending on what you named your project



- Upon pasting the Modes folder and Background.png into your "Resources" directory/folder, you should be able to see the changes reflected inside Visual Studio Code. Clicking on the drop down menu will allow you to see that Modes and Background.png are indeed in your project folder already.
- However, you still need to **import** it into your program.

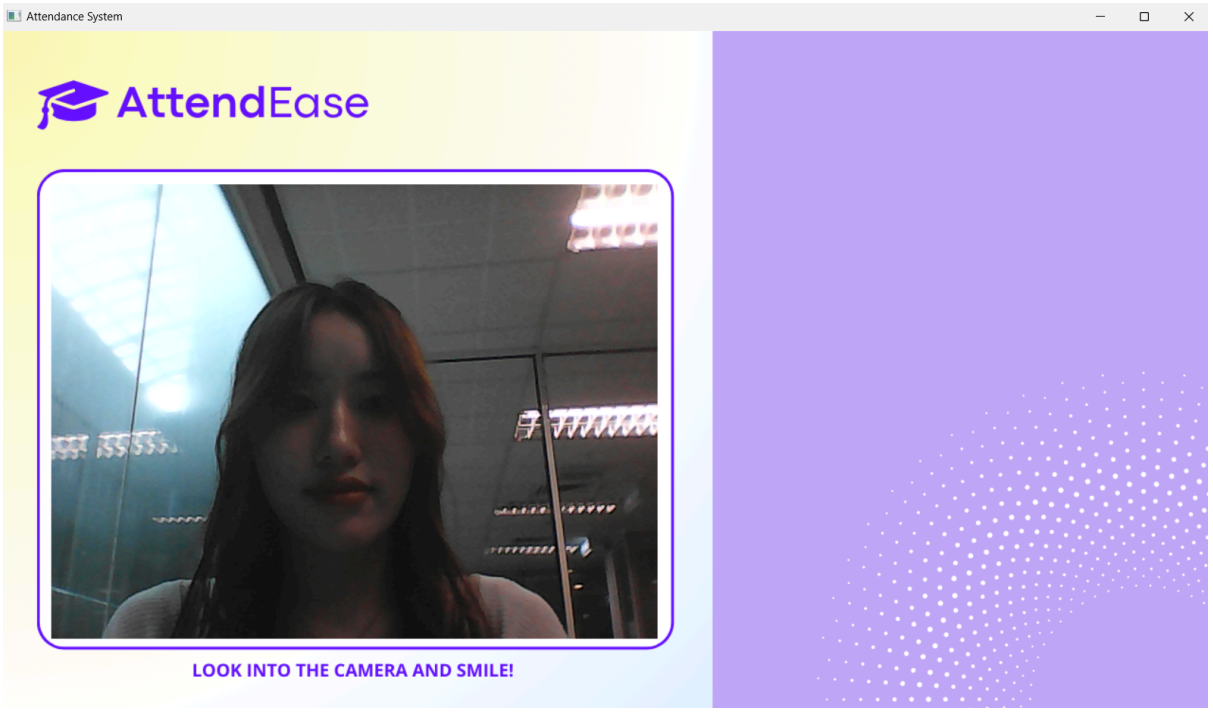
```python
#HIIII This is Suzanne
import cv2

capture = cv2.VideoCapture(1)
capture.set(3, 1280)
capture.set(4, 720)

#you are reading the "Background.png" file in your "Resources" directory and storing the image in a variable called backgroundImg
backgroundImg = cv2.imread('Resources/Background.png')

while True:
    success, image = capture.read()
    cv2.imshow("My Webcam", image)

    #using cv2 to ensure that your background image is shown in the window
    cv2.imshow("Attendance System", backgroundImg)

    cv2.waitKey(1)
```

- While still being in your main.py file, add the two lines below the purple comments. You are essentially importing the images in your resources directory into the python program.
- Next, you show the background image on your windows, which is what the imshow function is for.

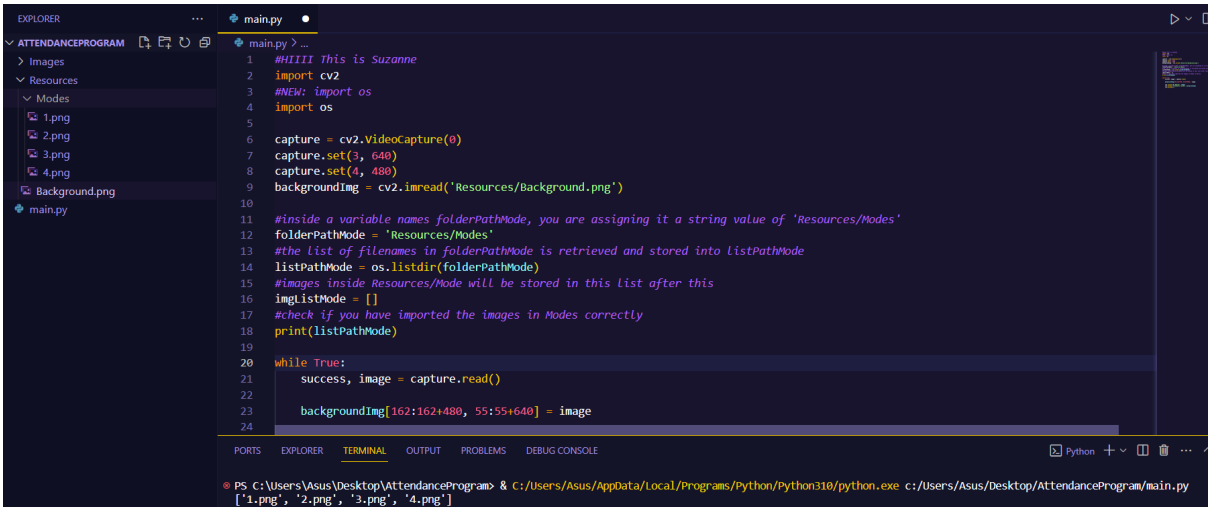
main.py

```python
#HIIII This is Suzanne
import cv2

capture = cv2.VideoCapture(0)
capture.set(3, 640) #change to 640 because we want to place the webcam IN our background image
capture.set(4, 480) #change to 480
backgroundImg = cv2.imread('Resources/Background.png')

while True:
    success, image = capture.read()

    #because we want to display the two components together, we must overlay the webcam feed over our image
    backgroundImg[162:162+480, 55:55+640] = image

    cv2.imshow("My Webcam", image)
    cv2.imshow("Attendance System", backgroundImg)
    cv2.waitKey(1)
```

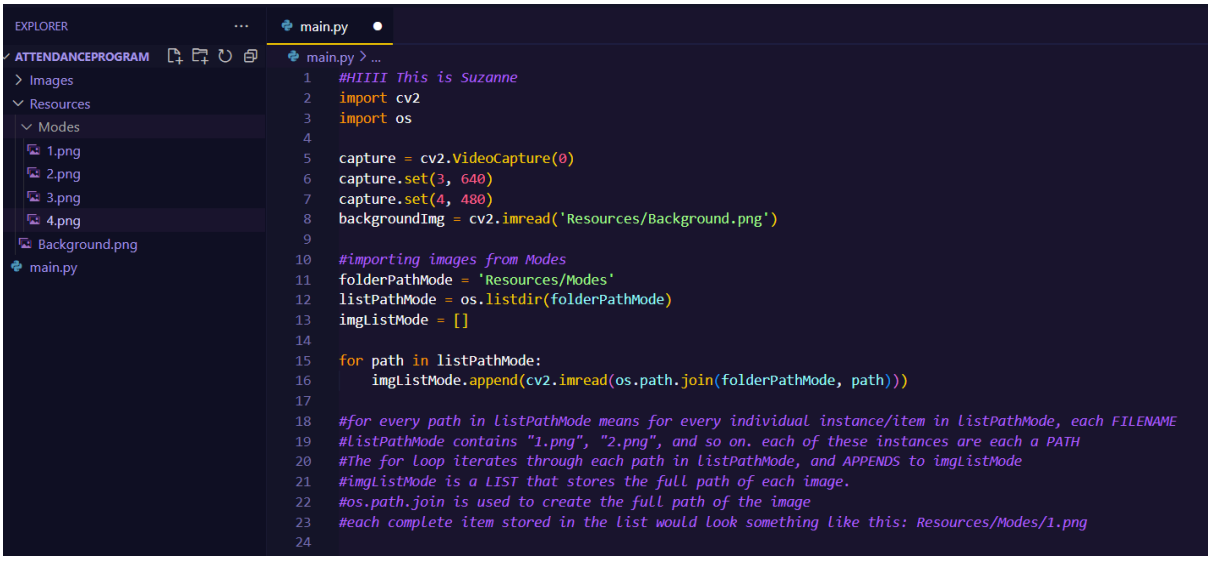
- To overlay your webcam over you background image, you will need to make the changes as shown in the code above.
    - Change the size of your webcam
    - Specify the start and end of your webcam's frame in both terms of height and width on the image



- If you're doing everything correctly, this should be what you see when you run the program now!



- Next, you want to ensure that the images inside “Modes” will be displayed.
- Import the “os” module
- Initialize a folderPathMode containing the path to your Modes folder and then store the list of filenames in listPathMode.
- To be sure that you’re doing things correctly, using “print(listPathMode)” to see if all your images within the actual Mode folder are being found by your program
- Compare the images listed on the left of the page (in Explorer) and what’s being printed in my TERMINAL on the bottom. They are the same! If you’ve reached this point, congratulations c:

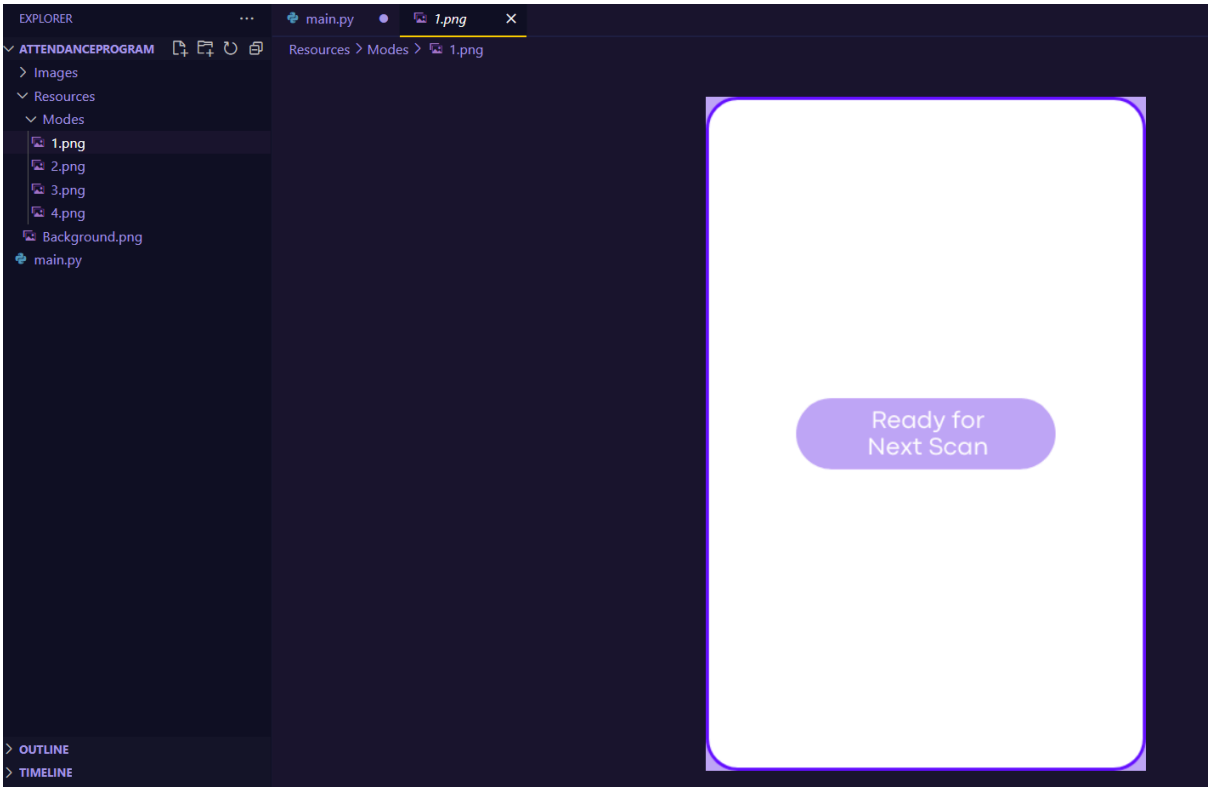


- Now that you know listPathMode has correctly found and stored all your images within the Modes directory, you will want to store each FULL path for the images into imgListMode

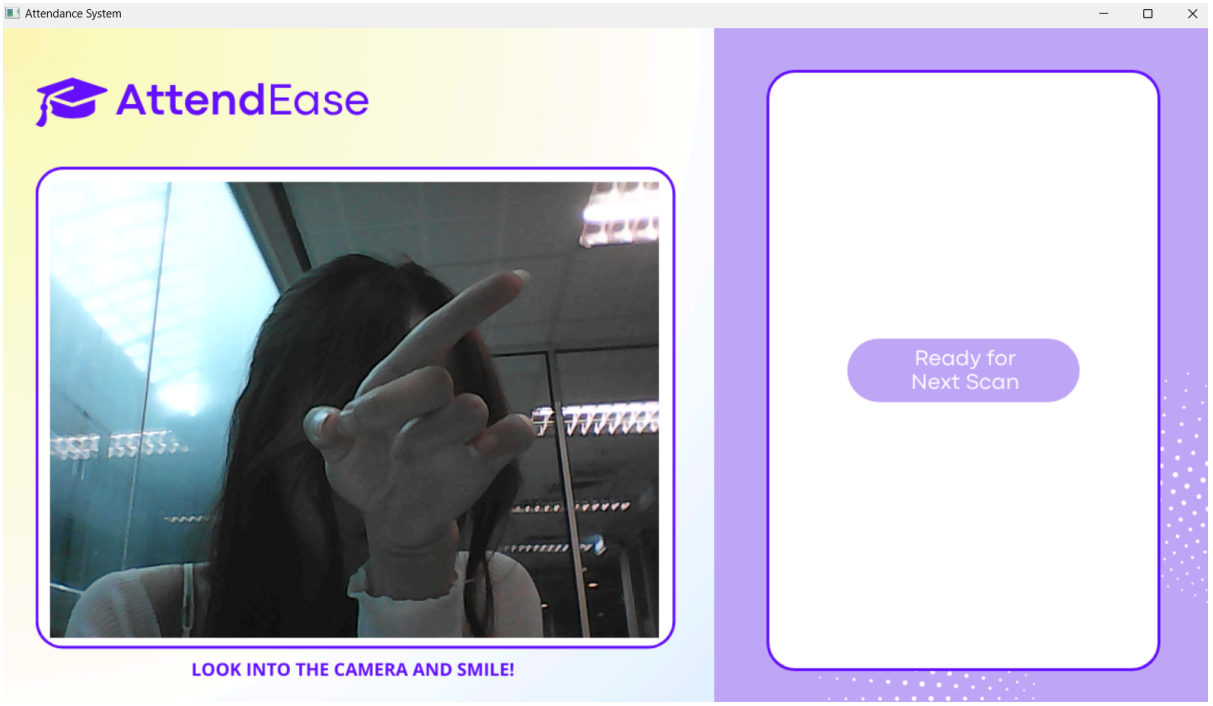
- A for loop is initialized to iterate through each item/path in listPathMode to join the full path together and consequently put it into imgListMode

```
18
19  while True:
20      success, image = capture.read()
21
22      backgroundImg[162:162+480, 55:55+640] = image
23      #overlay the FIRST element within imgListMode on top of the background
24      backgroundImg[44:44+633, 808:808+414] = imgListMode[0]
25
26
27      cv2.imshow("My Webcam", image)
28      cv2.imshow("Attendance System", backgroundImg)
29      cv2.waitKey(1)
30
31
```

- Overlay the first element of imgListMode on top of the background image



- Check which is the first element in imgListMode
- It’s the “Ready for Next Scan” image



- Yep! It’s the first element that gets overlaid to the right of your screen :]

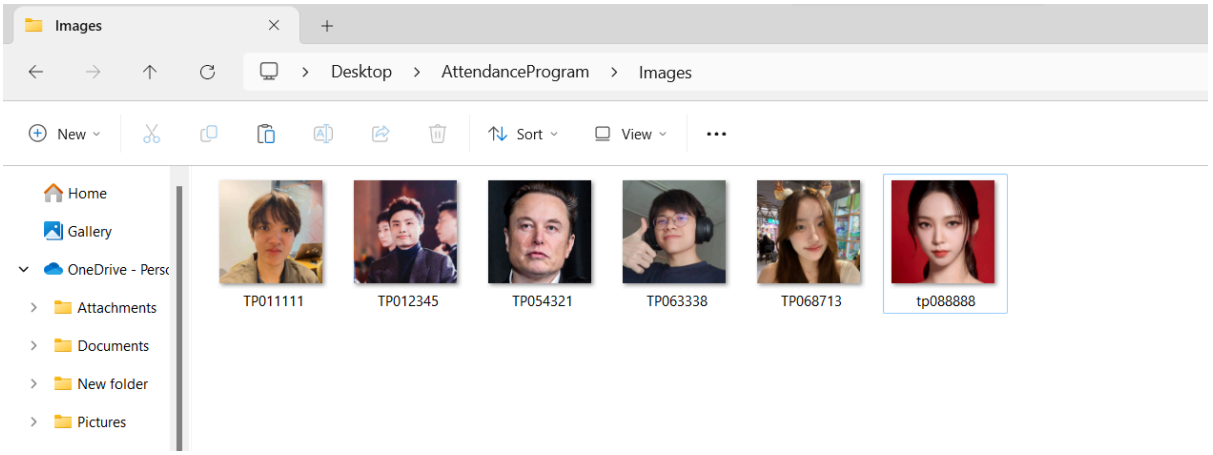
Two parts down! You’re learning fast ^^ 🎉

2.3 Facial encoding generator

The first thing you'd want to understand is probably what I wanted to know, too, when I first learned this: "What the hecc is a facial encoding generator? What do they even do? What is encoding?"

In the context of Machine Learning and AI, encodings are essentially **input data that has been transformed into a format that is more useful for the computers.** Encoders, in turn, capture the most essential features and patterns within the input data and discard what they deem is irrelevant to the process of "recognition"/ recognizing what the image is. Encodings are important because they are essential to SOOO many later processes in Machine Learning: deep processing, further analysis, and comparison with other pieces of data.

What we're going to be learning in this section is essentially how we can turn each face into a format that can be understood by machines. A format that stores all the unique features on each human face and therefore, helps the machine distinguish one person from the next.

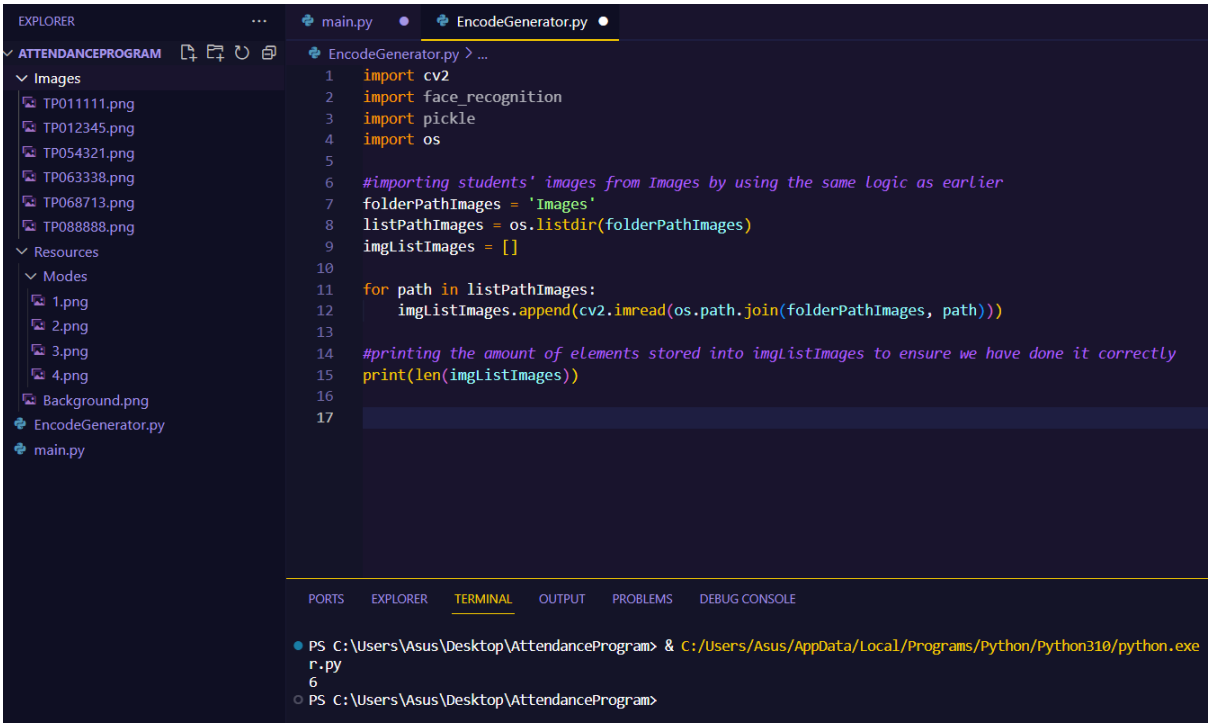


- In your project folder, within the Images directory/folder, you want to make sure that you have a few pictures inside.
- **These pictures must be 216x216**
- This is because there is only space for a square, 216x216 photo within the graphics that we imported earlier.
- We can use Python code to crop bigger photos, and it's very easy, but not when some pictures do not have the person's face centered in the middle, or when there are other people in the background. You'll have to implement another layer of face-recognition

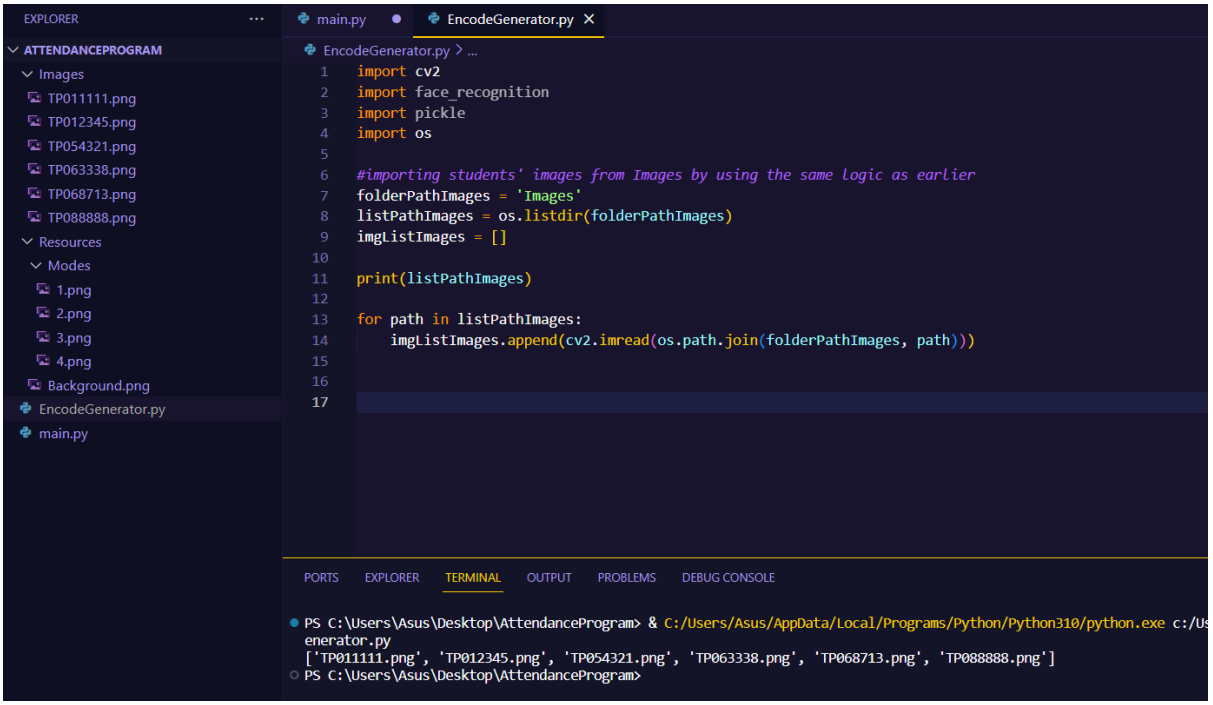
technology to ensure that the main face is identified within the picture, and manage to crop it out, and it'll take more time than we were allocated during the event

- Give each picture a unique name with similar formatting (because in real-life applications everything has to be uniform)

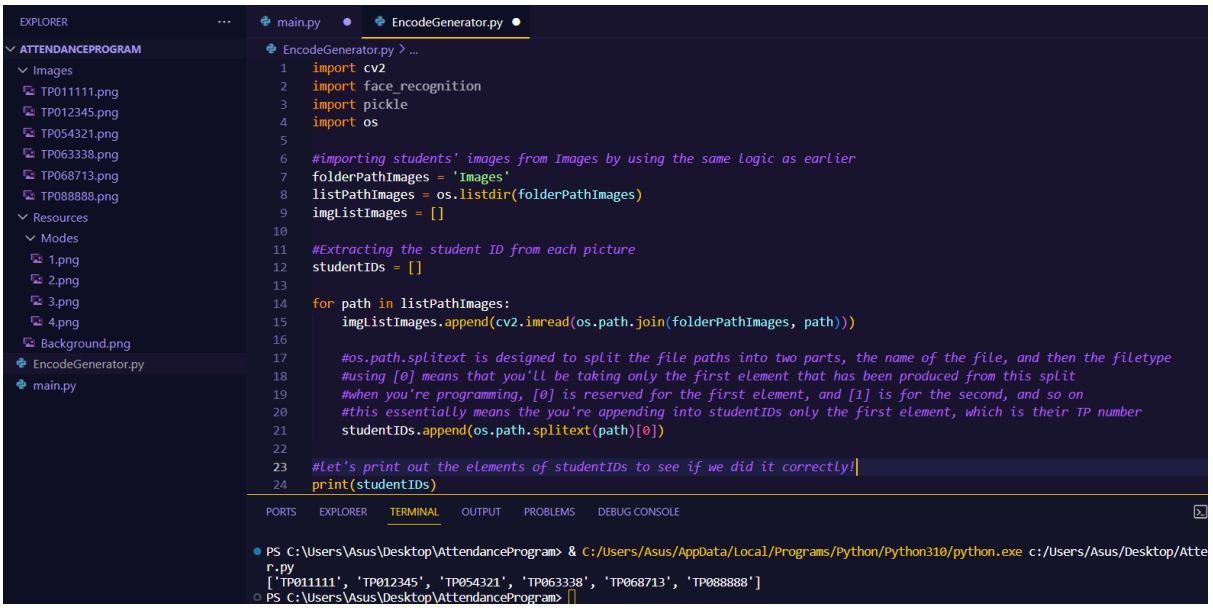
EncodeGenerator.py



- You can essentially just use the same code you used in main.py where you imported images from your Resources folder
- Remember to change the variable names so you don't get confused
- Make the variable names meaningful ya so you don't go (???) when you look at your program a few months down the line
- As you can see in the screenshot above, when I print out the len of the imgListImages list (len returns the length of the list, or simply put, how many items exist in a list), the length is 6, which is true when you count the number of images I have in the folder

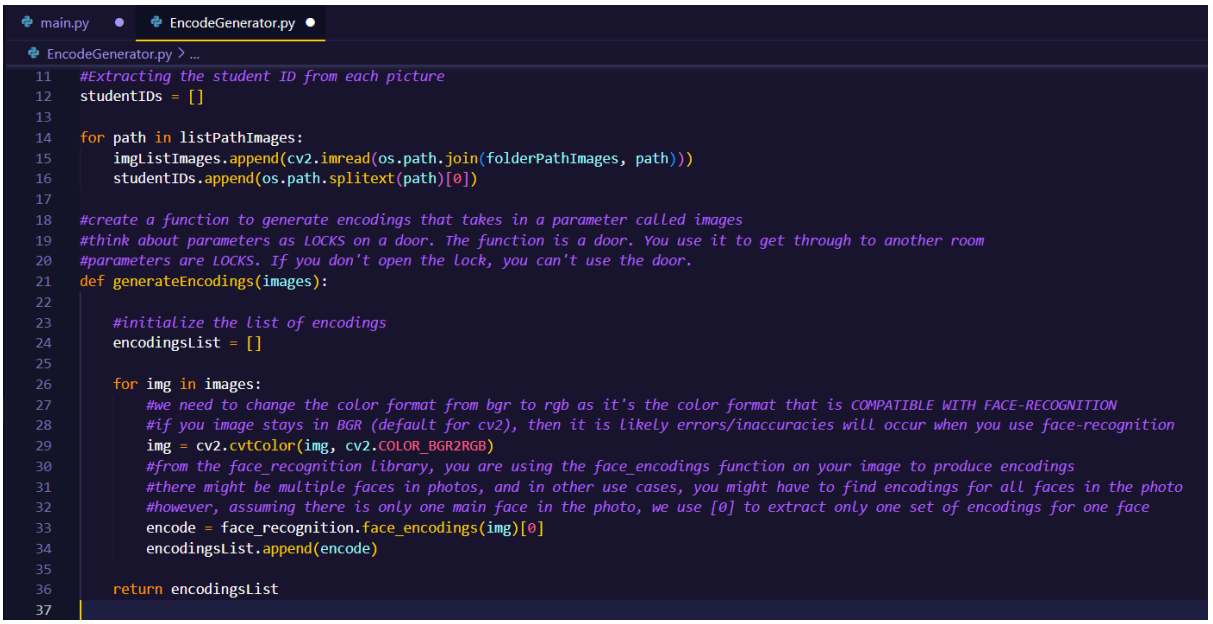


- Printing listPathImages (os.listdir takes in all the filenames in a path) will give you a list of names of the files from the folder
- In order to extract each student ID (without .png), you will need to split the string filenames



- The screenshot above shows you how you can extract the basename of the file without the file type (.png)
- The function os.path.splitext from the os module in Python essentially helps you split filenames into two parts/elements: the base name, and then the file type
- We are using [0] so that we can extract and append to our studentIDs lis ONLY the basename

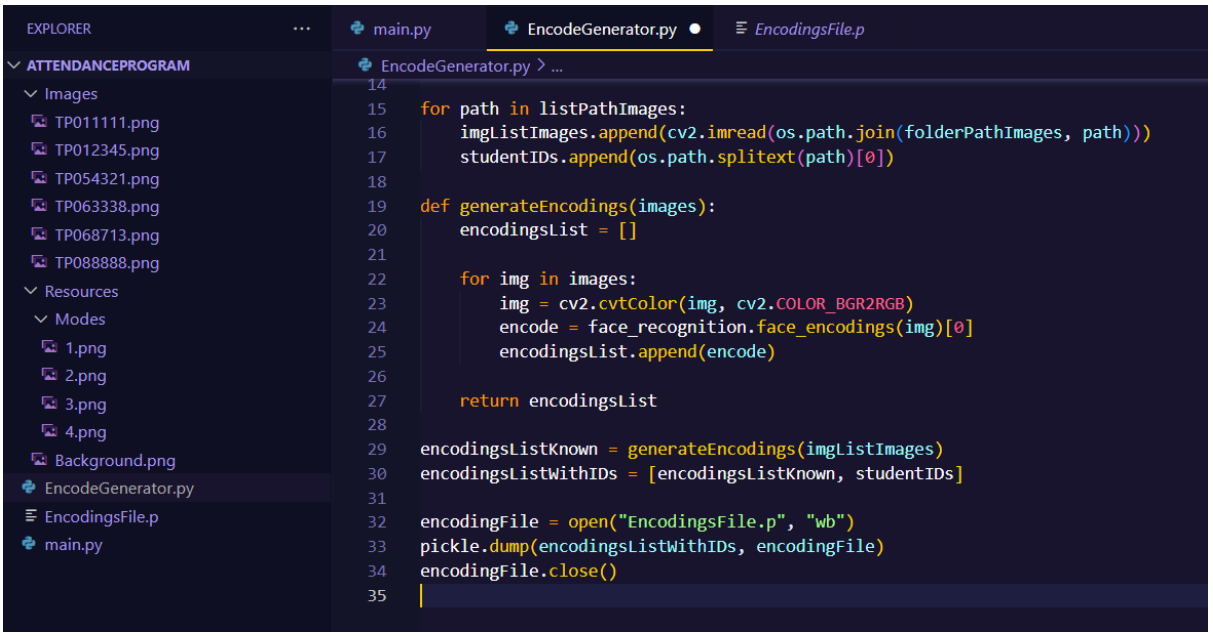
- By printing the elements out, we can see that we have done everything correctly and only the TP numbers are stored in studentIDs



- You can get rid of your print statement to declutter your code
- Now to generate the face encodings, you'll want to create a function to do that
- We create functions so that our code is reusable and we don't have to copy and paste every time we need to perform an action that might have to be performed more than once
- Our generateEncodings function takes in one parameter known as images (name of parameter doesn't matter besides giving the code readers an understanding of what the function needs to run)
- Intiailize a list called encodingsList
- By iterating through each individual element in images (the parameter, which is likely a list), you will be able to correct their color format and extract the encodings of the face in the picture. The list is appended with each encoding

```python
def generateEncodings(images):

    encodingsList = []

    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodingsList.append(encode)

    return encodingsList

encodingsListKnown = generateEncodings(imgListImages)
```

- Perform the function generateEncodings on our image list for all images inside the Images folder. In this scenario, imgListImages is known as an ARGUMENT, which is what you pass into a function when you are using it/calling it
- Think of arguments as keys in our door analogy
- Store all the encodings into a list called encodingsListKnown



- Initialize a list of encodings with the student IDs
- Open a new file called “EncodingsFile”
- .p is a pickle data file which is used to kept serialized data
- Pickle dump is a function used to serialize objects, flattening the data into a stream of bytes so that it’s easier to transmit over a network
- Serializing the data does NOT mean it’s encrypted. It’s still vulnerable to attacks and leaks. This is because storing the data

into a file using “dump” simply converts it into binary, which is still considered readable/can be converted into a readable form easily using pickle.load again

- Close the file for good practice
- Upon **running** the program, you will see the EncodingsFile.p file appear in Explore on the left side of the screen

AYO WHY YOU SO QUICK

Congratulations on making it this far! If you still have no idea what you’re doing, same ^^ you can ask me questions anytime tho 🙂🌹

2.4 Face-recognition library

In this part, you're going to be learning how to detect faces and compare them to our existing database of face encodings. The program will be able to tell if the person in the webcam is "known"! We'll need to become more familiar with what the face-recognition library does in Python, so let's dive into it!

main.py

main.py • EncodeGenerator.py

main.py > ...

```
1   #HIIII This is Suzanne
2   import cv2
3   import os
4   import pickle
5
6   capture = cv2.VideoCapture(0)
7   capture.set(3, 640)
8   capture.set(4, 480)
9   backgroundImg = cv2.imread('Resources/Background.png')
10
11  #importing images from Modes
12  folderPathMode = 'Resources/Modes'
13  listPathMode = os.listdir(folderPathMode)
14  imgListMode = []
15
16  for path in listPathMode:
17      imgListMode.append(cv2.imread(os.path.join(folderPathMode, path)))
18
19
20  #Let's import the encodings into our main file!
21  #First, open the file in read mode
22  encodingsFile = open("EncodingsFile.p", "rb")
23  #Next, "deserialize" the content in the file by using pickle.load
24  encodingsListWithIDs = pickle.load(encodingsFile)
```

- In main.py, you want to import the **pickle** library so you perform functions on your files
- In line 22, you can see that you have to open the EncodingsFile.p file (which contains our encodings list with student IDs) in read mode
- Like we mentioned earlier, you will be using pickle.load whenever you need to deserialize and read a serialized pickle file

```
19
20  #importing encodings into main.py
21  encodingsFile = open("EncodingsFile.p", "rb")
22  encodingsListWithIDs = pickle.load(encodingsFile)
23  encodingsFile.close()
24
25  encodingsListKnown, studentIDs = encodingsListWithIDs
26  print(studentIDs)
27
28
29  while True:
30      success, image = capture.read()
31
```

```
PORTS  EXPLORER  TERMINAL  OUTPUT  PROBLEMS  DEBUG CONSOLE

PS C:\Users\Asus\Desktop\AttendanceProgram> & C:/Users/Asus/AppData/Local/Programs/Python/Python3

['TP011111', 'TP012345', 'TP054321', 'TP063338', 'TP068713', 'TP088888']
```

- After you load the content of the file, you'll want to split each line into their corresponding lists
- From encodingsListWithIDs, you split it into two elements and get encodingsListKnown and studentIDs.
- Try printing studentIDs to see if you get the correct output! As you can see in my terminal, I've gotten it right

```
30  while True:
31      success, image = capture.read()
32
33      backgroundImg[162:162+480, 55:55+640] = image
34      #overlay the FIRST element within imgListMode on top of the background
35      backgroundImg[44:44+633, 808:808+414] = imgListMode[0]
36
37      #resize image because we will require a lot of computational power if we try to encode images that are large
38      smallImage = cv2.resize(image, (0,0), None, 0.25, 0.25)
39      smallImage = cv2.cvtColor(smallImage, cv2.COLOR_BGR2RGB)
40
41      #locating the face in the current frame
42      faceCurrentFrame = face_recognition.face_locations(smallImage)
43      #converting facial featuers in the current frame into encodings
44      encodeCurrentFrame = face_recognition.face_encodings(smallImage, faceCurrentFrame)
45
46
47      cv2.imshow("My Webcam", image)
48      cv2.imshow("Attendance System", backgroundImg)
49      cv2.waitKey(1)
50
```

- Import face-recognition library
- You will want to edit the while loop
- The first thing you need to do is resize the image (the webcam feed) because if we try to process the original image, it will be so large that the whole process will take a very long time and eat up a lot of computational resources
- Next, you want to ensure the color format is right for the facial recognition library we are using
- Find the location of the face in the current frame

- Encode the face in the current frame

```
41      #locating the face in the current frame
42      faceCurrentFrame = face_recognition.face_locations(smallImage)
43      #converting facial features in the current frame into encodings
44      encodeCurrentFrame = face_recognition.face_encodings(smallImage, faceCurrentFrame)
45
46
47      cv2.imshow("My Webcam", image)
48      cv2.imshow("Attendance System", backgroundImg)
49      cv2.waitKey(1)
50
51      for encodeFace, faceLocation in zip(encodeCurrentFrame, faceCurrentFrame):
52          matches = face_recognition.compare_faces(encodingsListKnown, encodeFace)
53          faceDistance = face_recognition.face_distance(encodingsListKnown, encodeFace)
54          print("Matches", matches)
55          print("Face Distance", faceDistance)
56
```

- We're using zip in this screenshot above as it's a built-in function in Python that allows you to take in multiple iterables (containers you can iterate through), and then allow you to iterate through these tuples
- Zip is often used to pair elements from different lists for simultaneous processing
- In this for loop, the face in the webcam is iterated through the list of known encodings to find a match
- The list of matches will be displayed as either True or False
- Face distance measures how similar your face is to a certain image/another face. The lower the score, the more similar it is

```
Matches [False, False, False, False, True, False]
Face Distance [0.64135338 0.64429182 0.85916578 0.73551777 0.51522842 0.62414138]
Matches [False, False, False, False, True, False]
Face Distance [0.62283253 0.65054975 0.89161003 0.70193612 0.52034132 0.61662894]
```

- As you can see, when I run my program, it says that I'm a match to image 5, which stores my own image

main.py | EncodeGenerator.py | EncodingsFile.p

main.py > ...

```python
#HIIII This is Suzanne
import cv2
import os
import pickle
import face_recognition
import numpy as np
import cvzone
```

- Make sure you have all of these libraries imported

```python
for encodeFace, faceLocation in zip(encodeCurrentFrame, faceCurrentFrame):
    matches = face_recognition.compare_faces(encodingsListKnown, encodeFace)
    faceDistance = face_recognition.face_distance(encodingsListKnown, encodeFace)
    print("Matches", matches)
    print("Face Distance", faceDistance)

    #for even cleaner results, you can calculate matchIndex by using the numpy library
    matchIndex = np.argmin(faceDistance)
    print("Match Index", matchIndex)
```

- We want our results to be clearer, so instead of [False, False, True, False], we'd ideally get a match index right away so we know which picture we're being matched to

```python

for encodeFace, faceLocation in zip(encodeCurrentFrame, faceCurrentFrame):
    matches = face_recognition.compare_faces(encodingsListKnown, encodeFace)
    faceDistance = face_recognition.face_distance(encodingsListKnown, encodeFace)
    matchIndex = np.argmin(faceDistance)

    if matches[matchIndex]:
        print("Registered Student Detected")
        print(studentIDs[matchIndex])
        break


```

PORTS EXPLORER TERMINAL OUTPUT PROBLEMS DEBUG CONSOLE

```
PS C:\Users\Asus\Desktop\AttendanceProgram> & C:/Users/Asus/AppData/Local/Programs/Python/Python310/python.exe c:
['TP011111', 'TP012345', 'TP054321', 'TP063338', 'TP068713', 'TP088888']
Registered Student Detected
TP068713
Registered Student Detected
TP068713
```

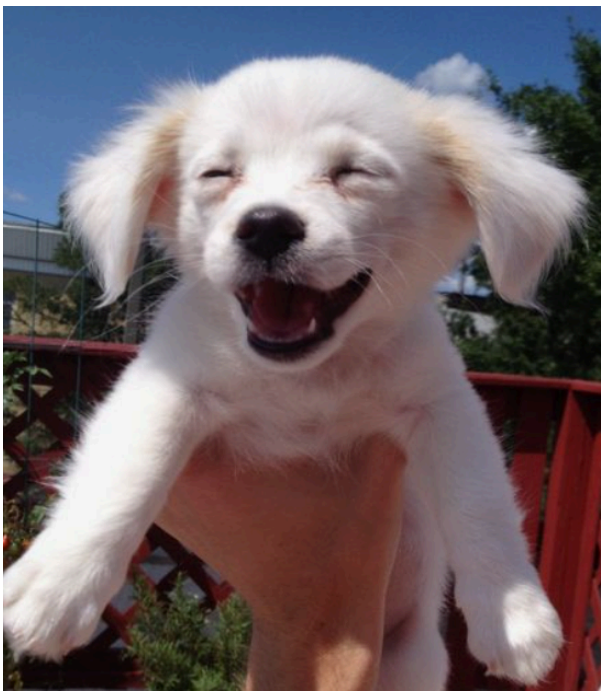
- Now that you know how each concept works, you can remove those print lines to change it to “Registered Student Detected” instead, as that’s how real-life attendance systems work!
- Additionally, you can use the matchIndex you’ve found to make it so that the program can print exactly the TP number of the face detected

```
        matchIndex = np.argmin(faceDistance)

        if matches[matchIndex]:
            #you can get rid of the print statements later
            print("Registered Student Detected")
            print(studentIDs[matchIndex])

            #let's get all corners of the face mapped from faceLocation!
            y1, x2, y2, x1 = faceLocation
            #resize it to 4x because we resized our webcam capture earlier to 0.25 its size
            y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
            #adjusting where the box's coordinaes are
            bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
            #draw the rectangle onto the backgroundImg image
            backgroundImg = cvzone.cornerRect(backgroundImg, bbox, rt = 0)

```

- This is what you can do if you want the fancy rectangle tracking your face around while you move!

```

while True:
    success, image = capture.read()

    backgroundImg[162:162+480, 55:55+640] = image
    #overlay the FIRST element within imgListMode on top of the background
    backgroundImg[44:44+633, 808:808+414] = imgListMode[0]

    #resize image because we will require a lot of computational power if we try to encode images that are large
    smallImage = cv2.resize(image, (0,0), None, 0.25, 0.25)
    smallImage = cv2.cvtColor(smallImage, cv2.COLOR_BGR2RGB)

    #locating the face in the current frame
    faceCurrentFrame = face_recognition.face_locations(smallImage)
    #converting facial features in the current frame into encodings
    encodeCurrentFrame = face_recognition.face_encodings(smallImage, faceCurrentFrame)

    for encodeFace, faceLocation in zip(encodeCurrentFrame, faceCurrentFrame):
        matches = face_recognition.compare_faces(encodingsListKnown, encodeFace)
        faceDistance = face_recognition.face_distance(encodingsListKnown, encodeFace)
        matchIndex = np.argmin(faceDistance)

        if matches[matchIndex]:
            #you can get rid of the print statements later
            print("Registered Student Detected")
            print(studentIDs[matchIndex])

            y1, x2, y2, x1 = faceLocation
            y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
            bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
            backgroundImg = cvzone.cornerRect(backgroundImg, bbox, rt = 0)


    cv2.imshow("Attendance System", backgroundImg)
    cv2.waitKey(1)

```

- This is how your whole code for the while loop in your main.py should look like!!

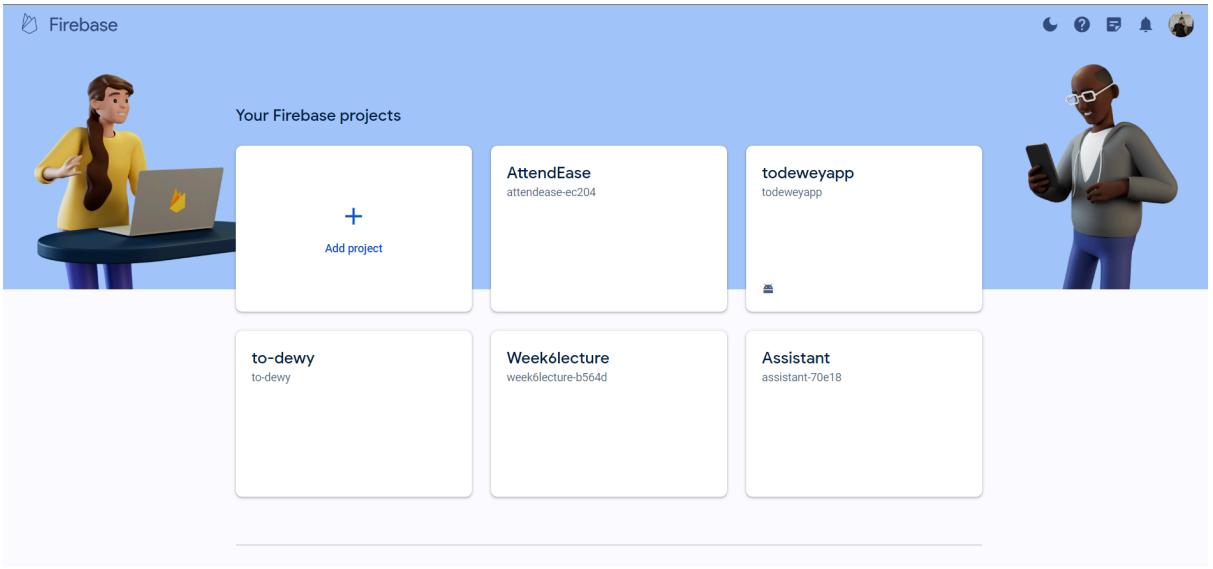


You when you realize you’re nearly at the finish line!

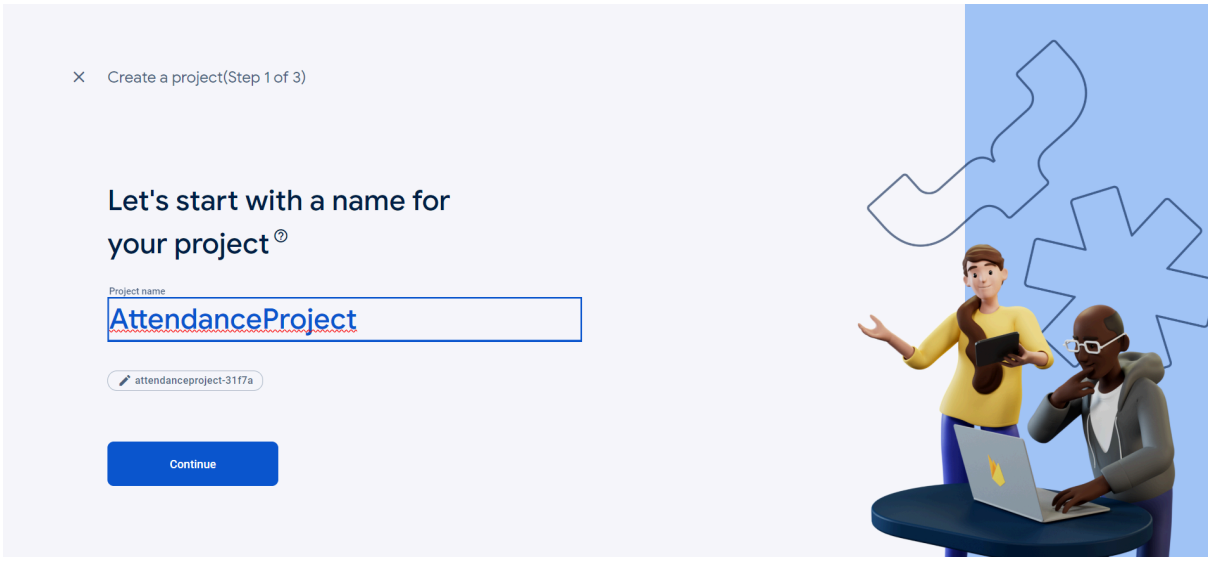
2.5 Firebase real-time database and storage connection

Let’s learn how to set up our database! Go to Firebase for this c:

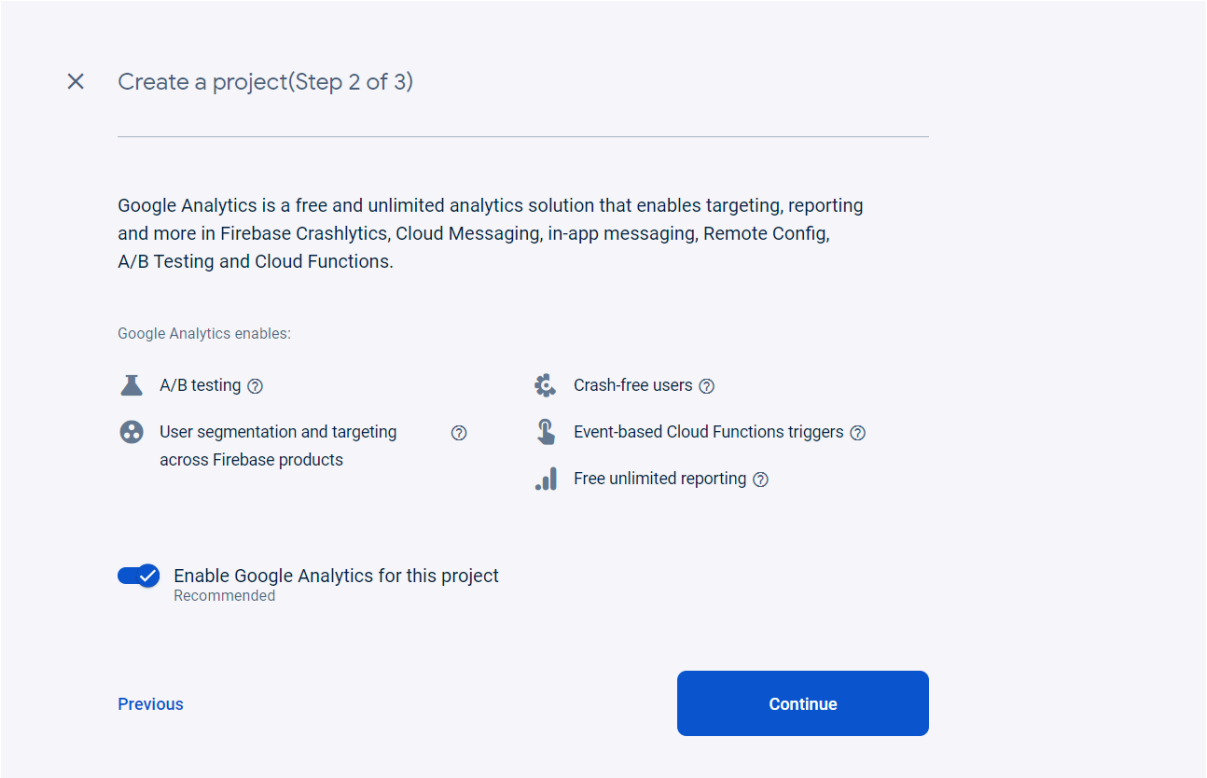
`Firebase console`



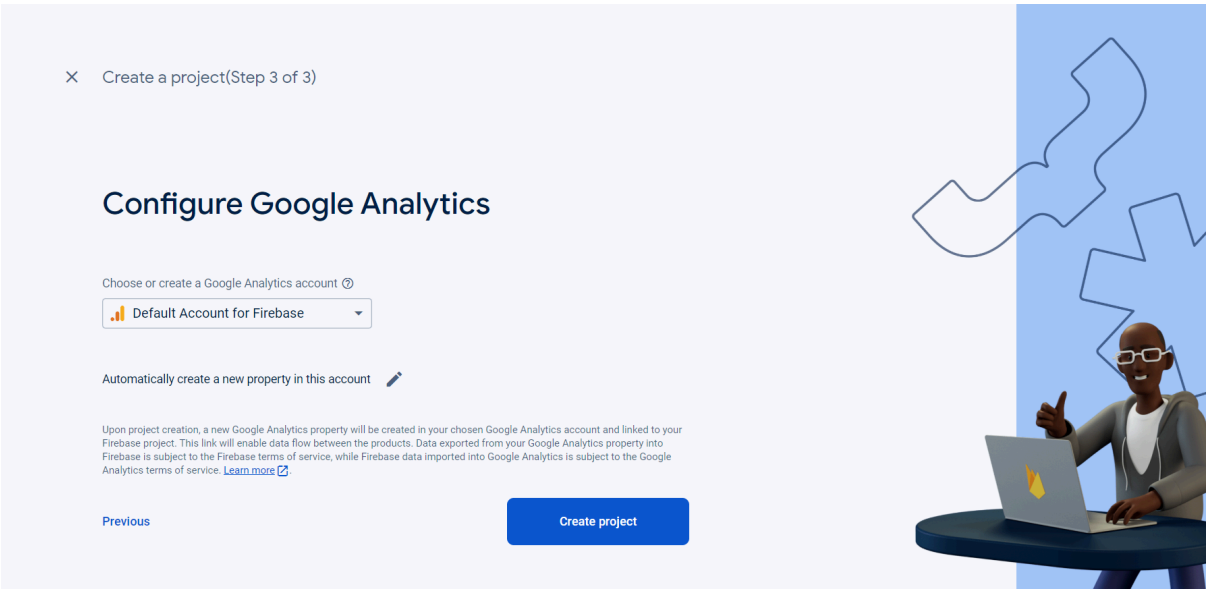
- Add project



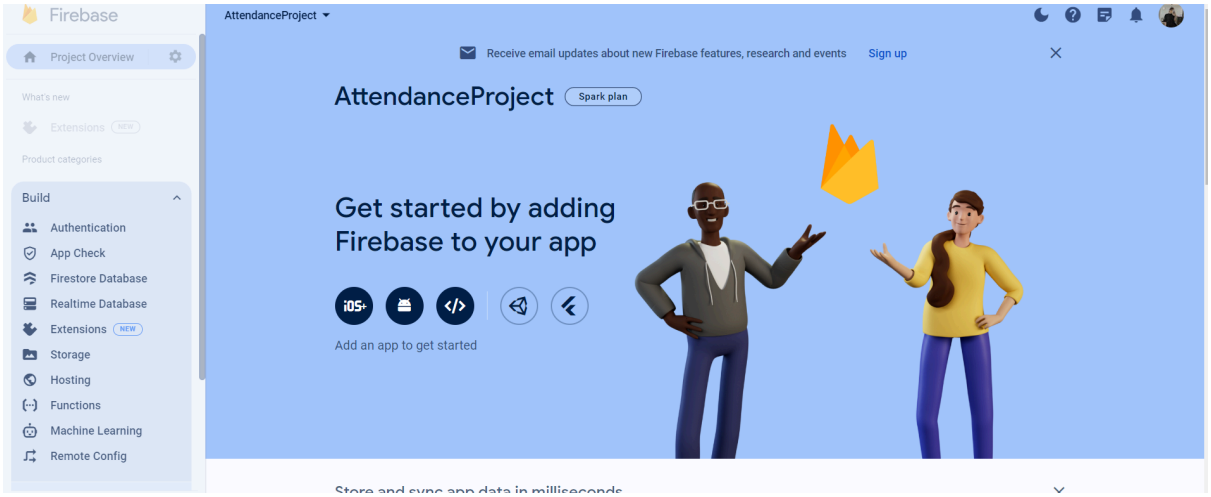
- Name your project whatever is appropriate
- Click on Continue



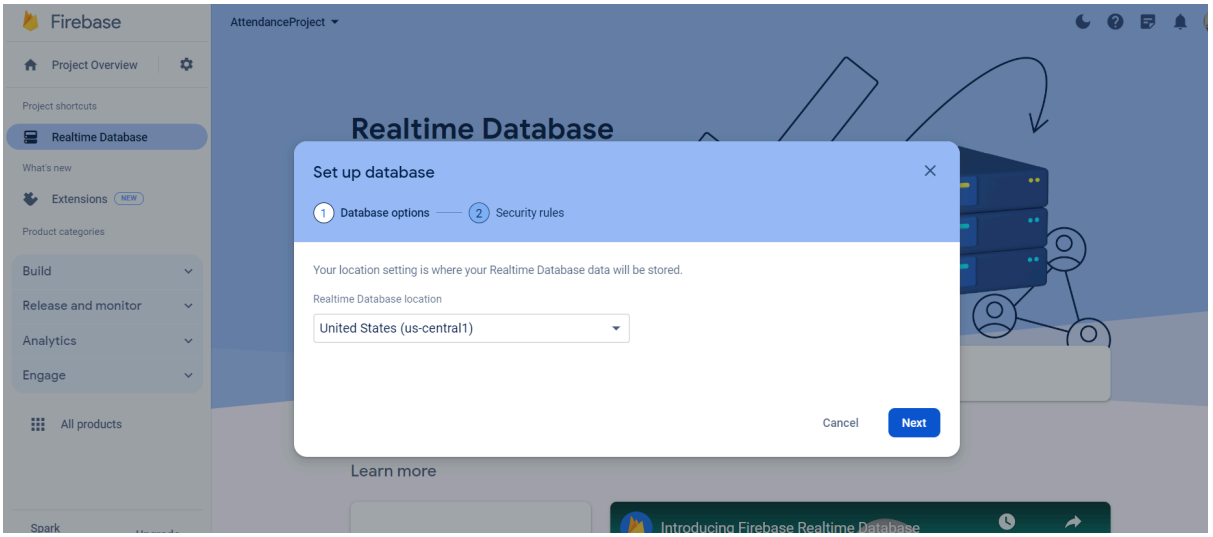
- Enable Google Analytics for the project and click Continue

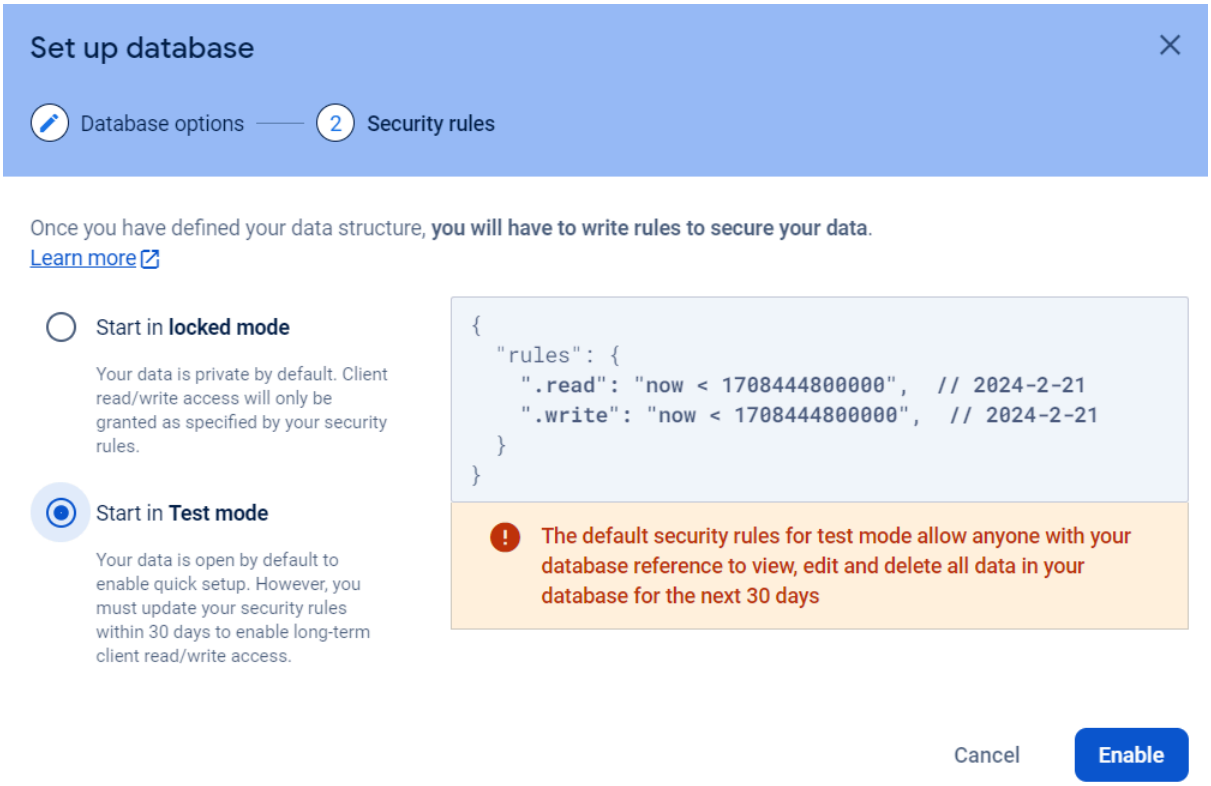


- Use your default account and create the project! Then you’re done with this part. Wait for the project to be created c:

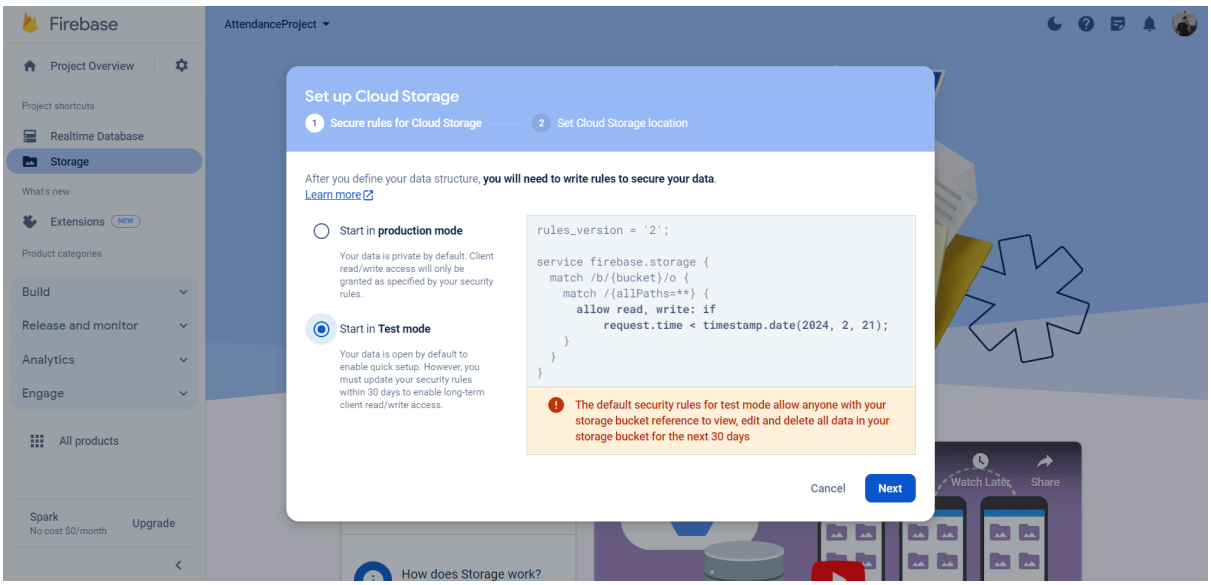


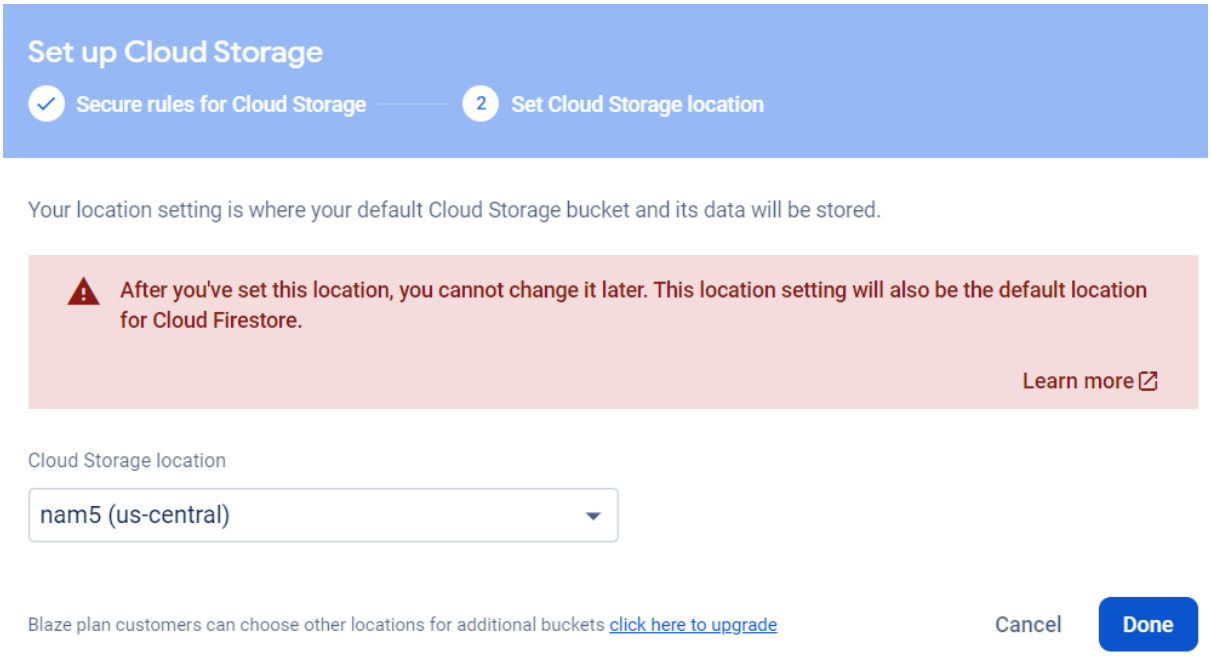
- From the left-side menu, click on Realtime Database



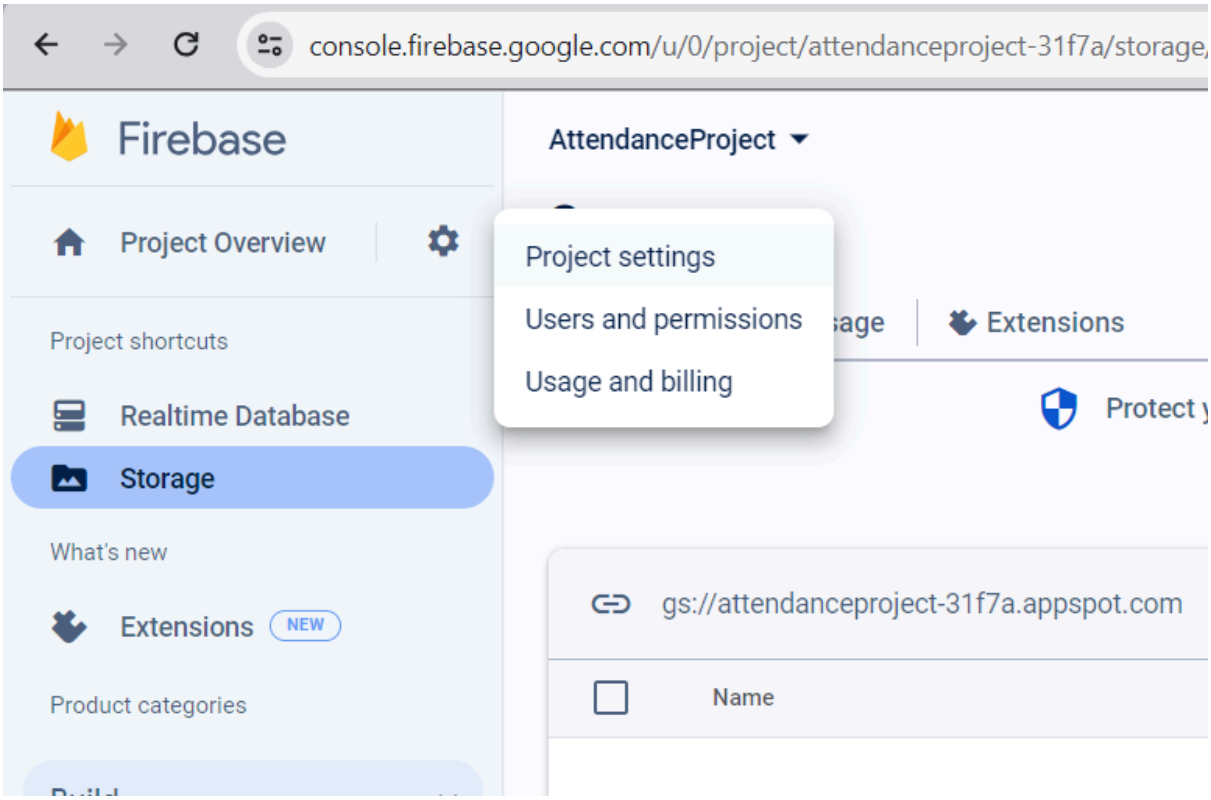


- Create a new Realtime database and choose the above options
- Click on Enable

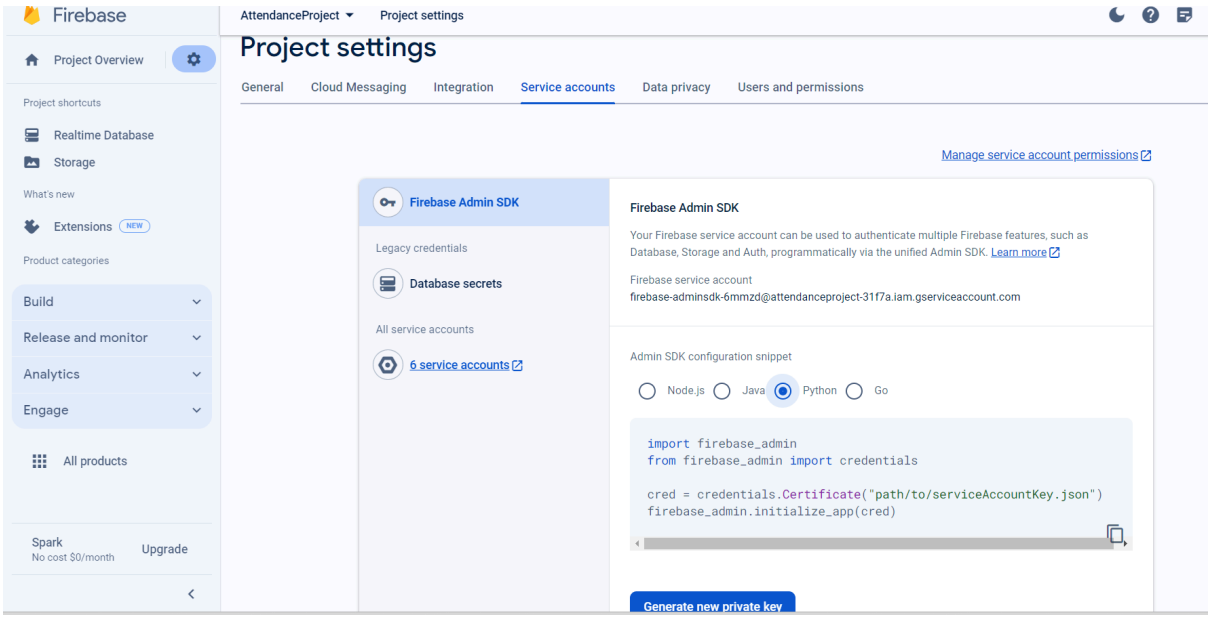




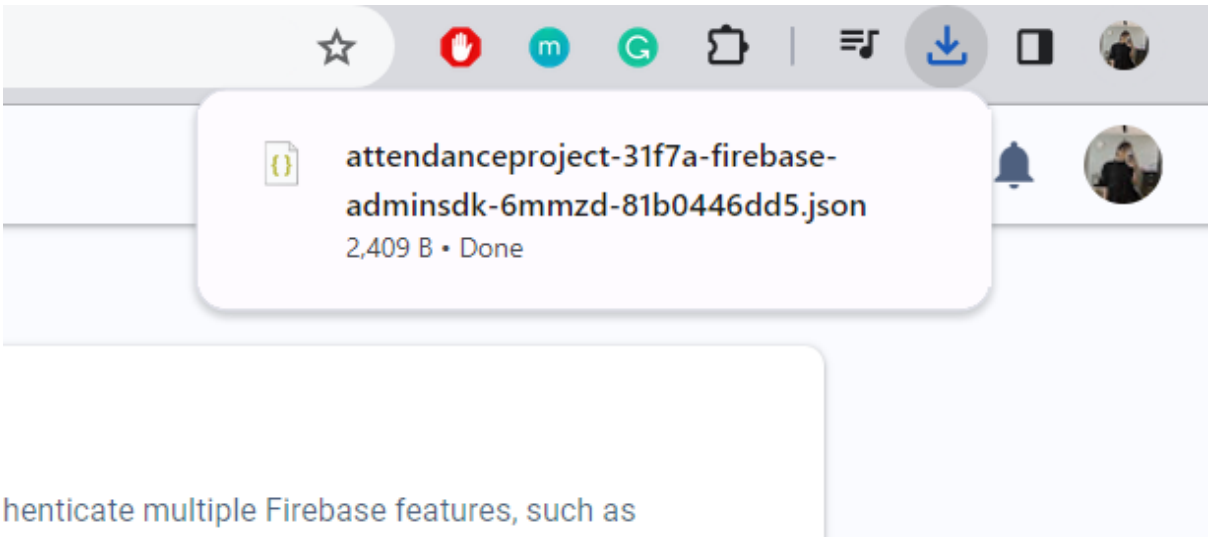
- Leave Realtime database temporarily so you can select Storage from the left-side menu. Start in Test Mode
- After selecting the location, click Done



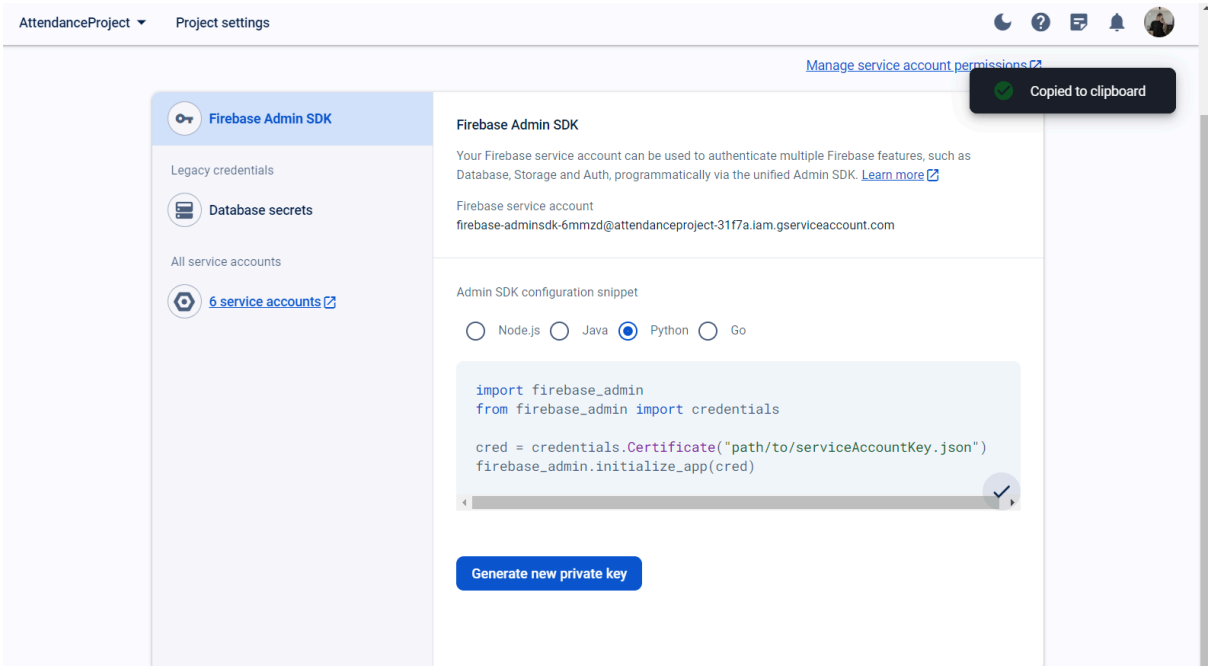
- After creating your Realtime database and your storage, click on settings (in your AttendanceProject/whatever you named your project) and click on Project Settings



- Click on Service accounts
- Click on Python
- Generate new private key

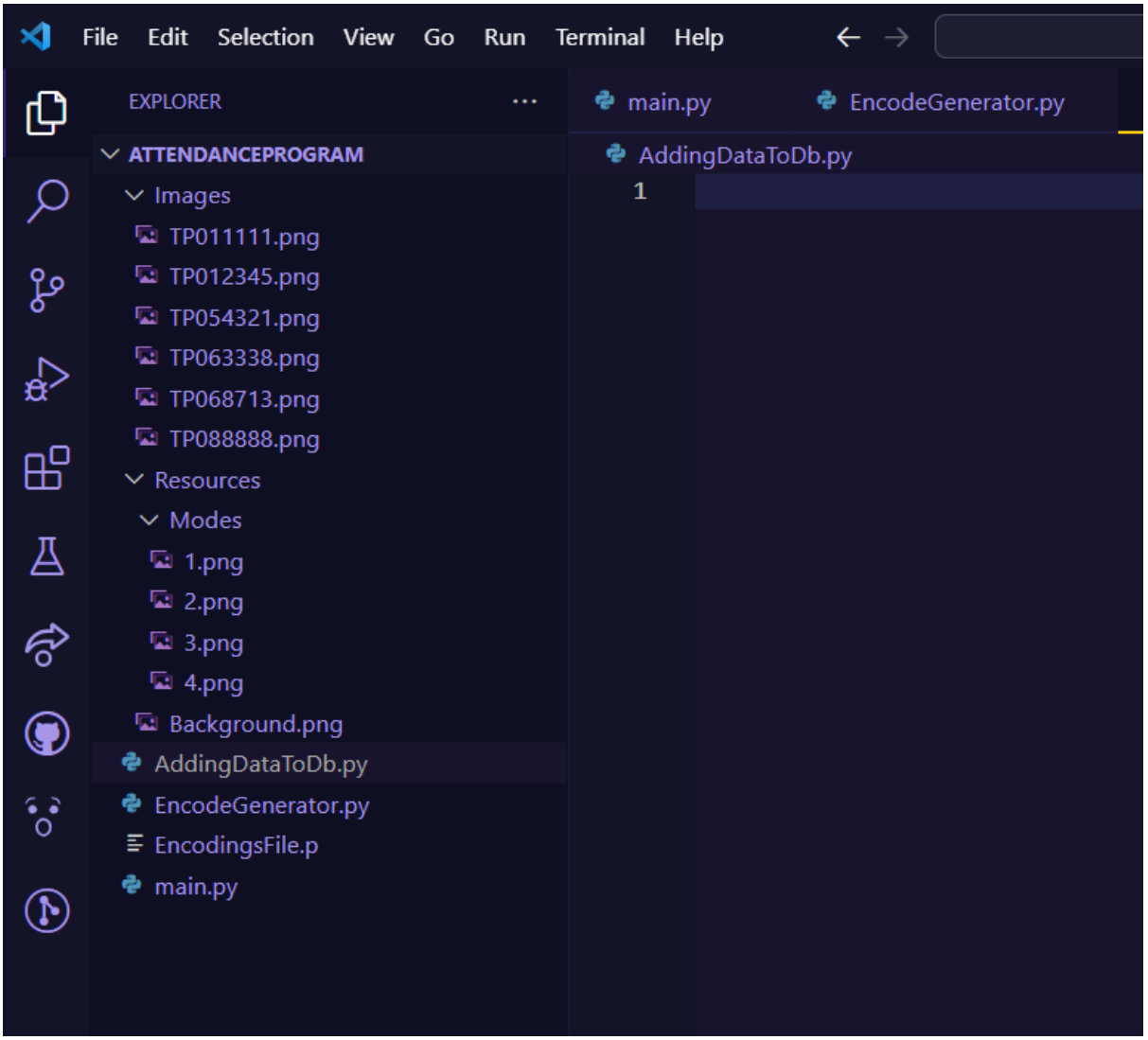


- Upon generating the private key successfully, a json file will automatically download

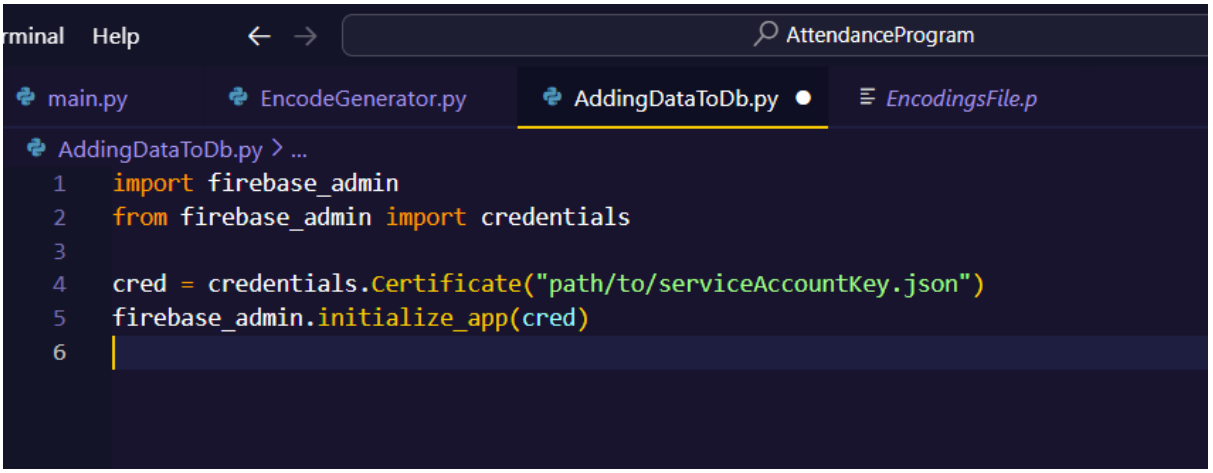


- Keep the downloaded file aside for now and go back to this page
- Copy the code

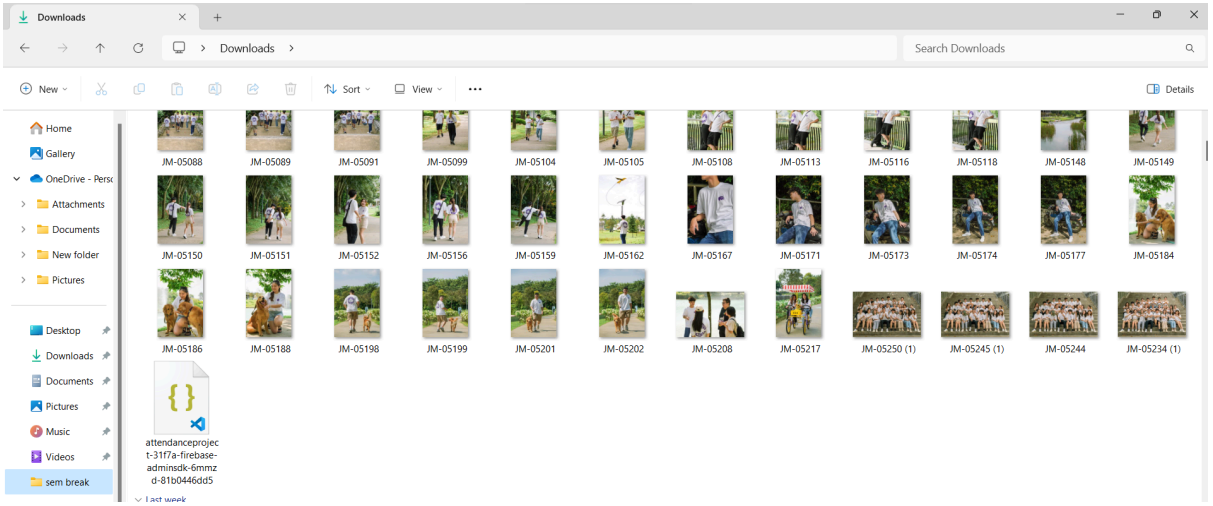
AddingDataToDb.py



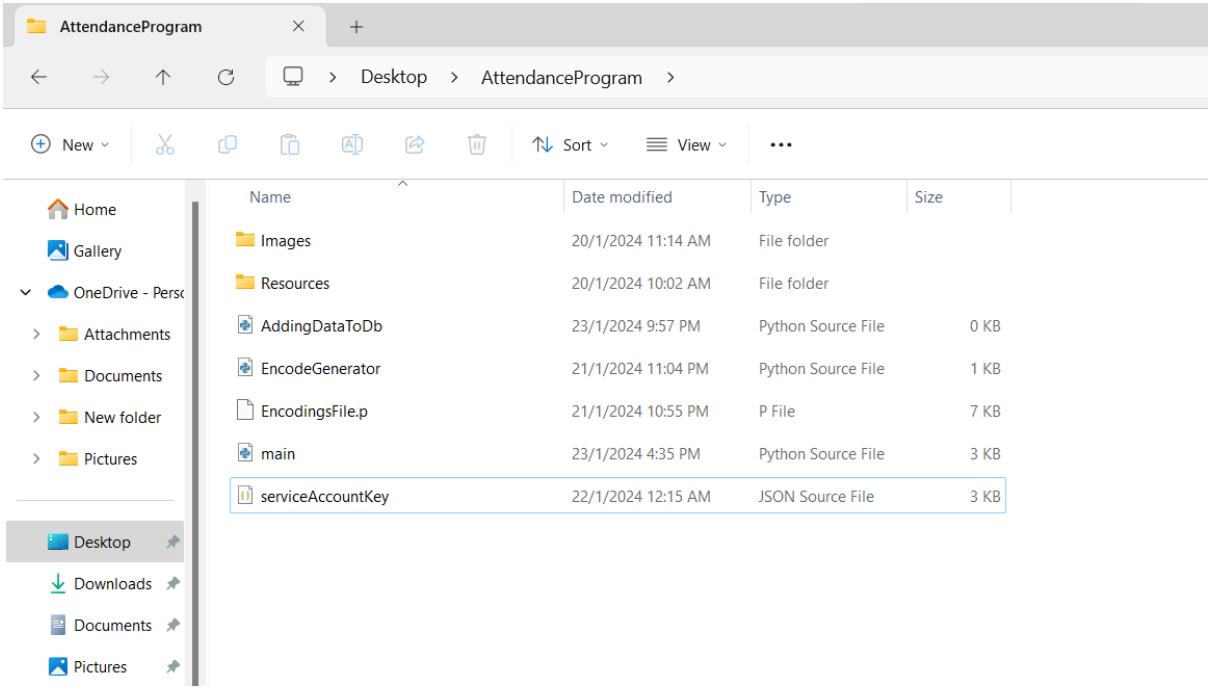
- Return to your IDE and create a new file called AddingDataToDb.py



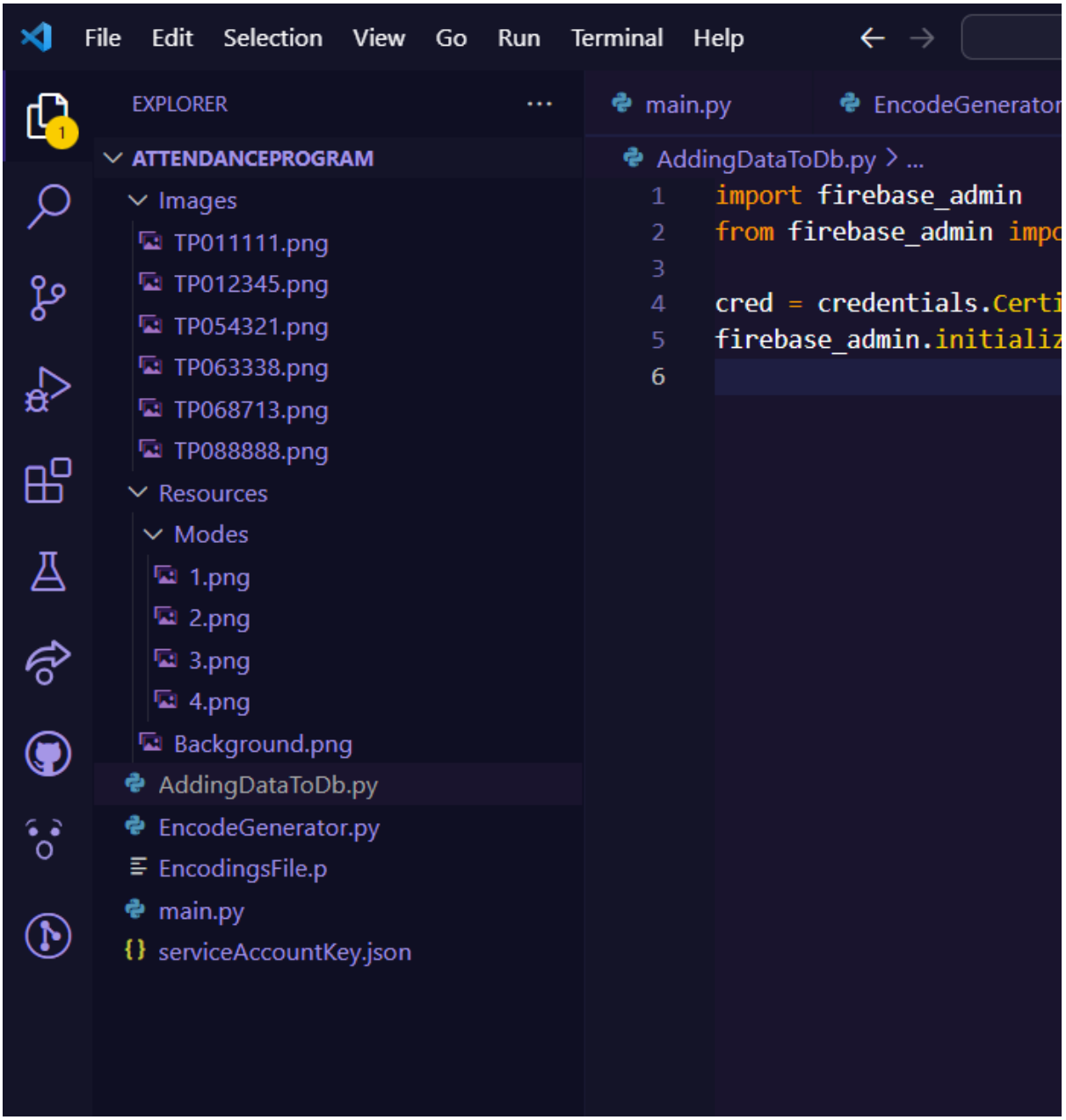
- Paste you copied code from Firebase here



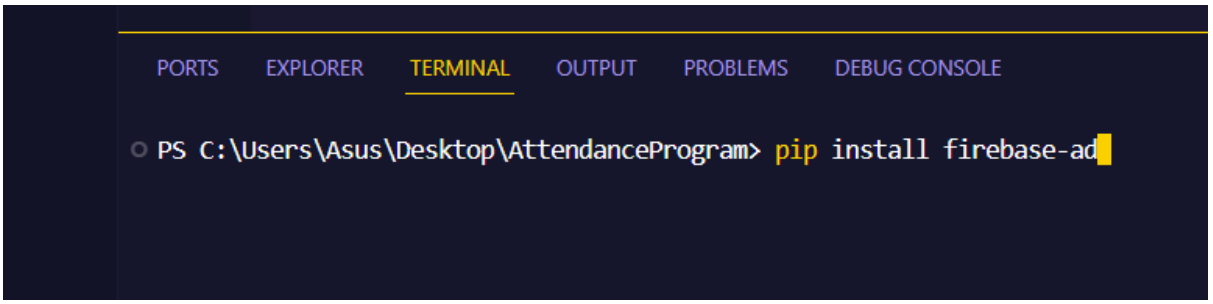
- Remember the file you downloaded earlier? Rename to serviceAccountKey.json



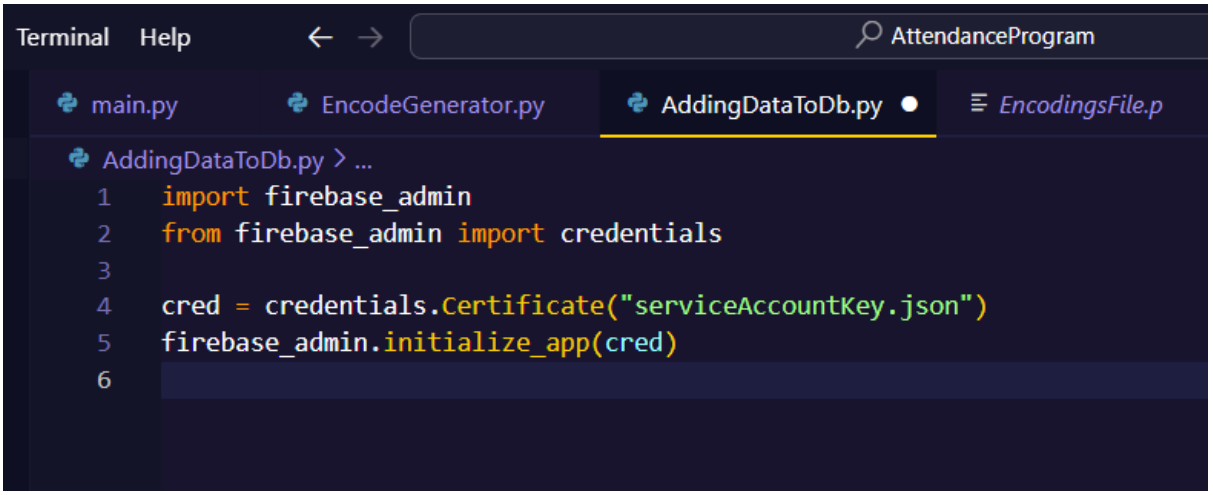
- Move the serviceAccountKey.json file to your project folder



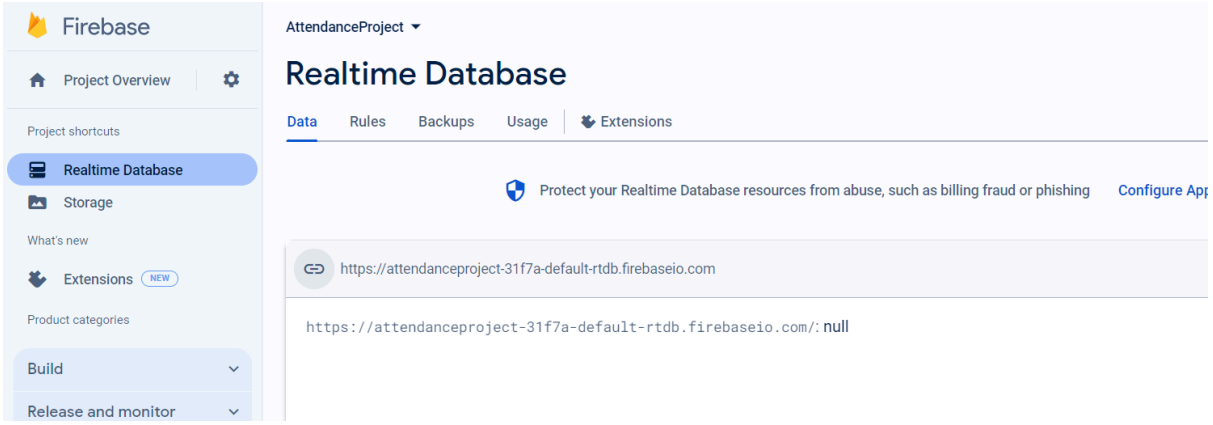
- You should be able to see the file inside your Explorer in Visual Studio Code now



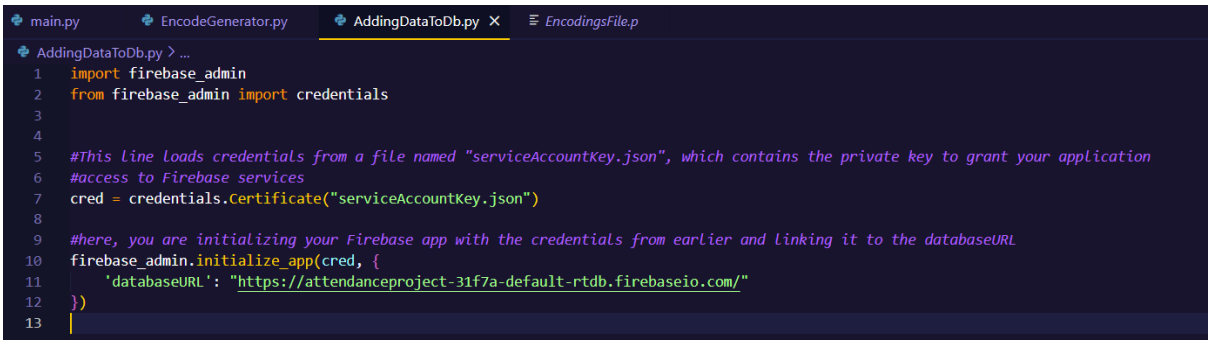
- In order to be able to import firebase_admin in your program, you'll want to pip install firebase-ad in your terminal first.



- Remove the "path/to" section within the credentials as the serviceAccountKey file is directly inside the main project folder



- Go back to your Firebase tab and copy the link inside your REALTIME DATABASE. This is your database URL

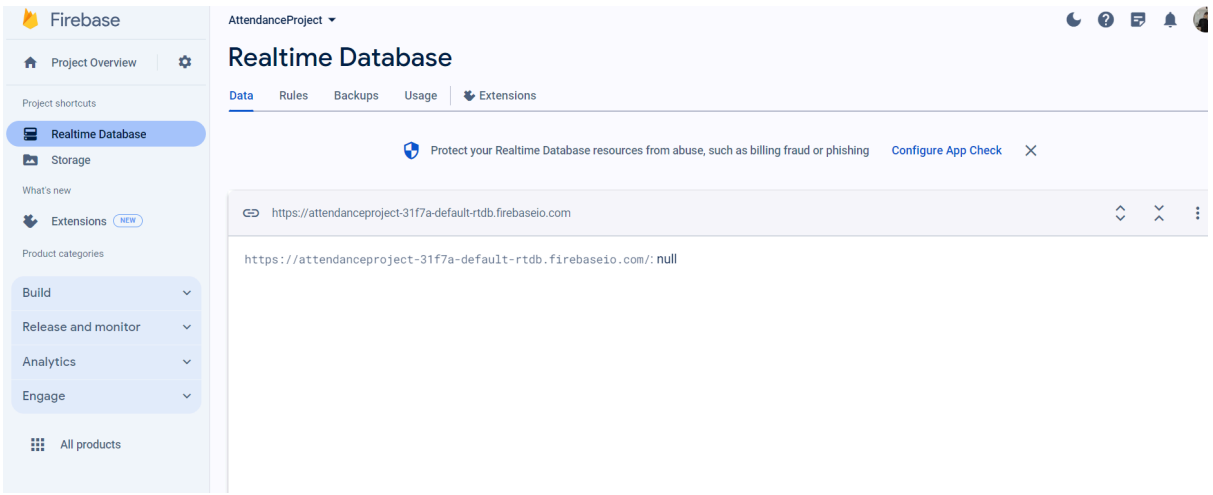


- In AddingDataToDb, you should have written these few lines to initialize and link your system to the Firebase realtime database that you have created earlier

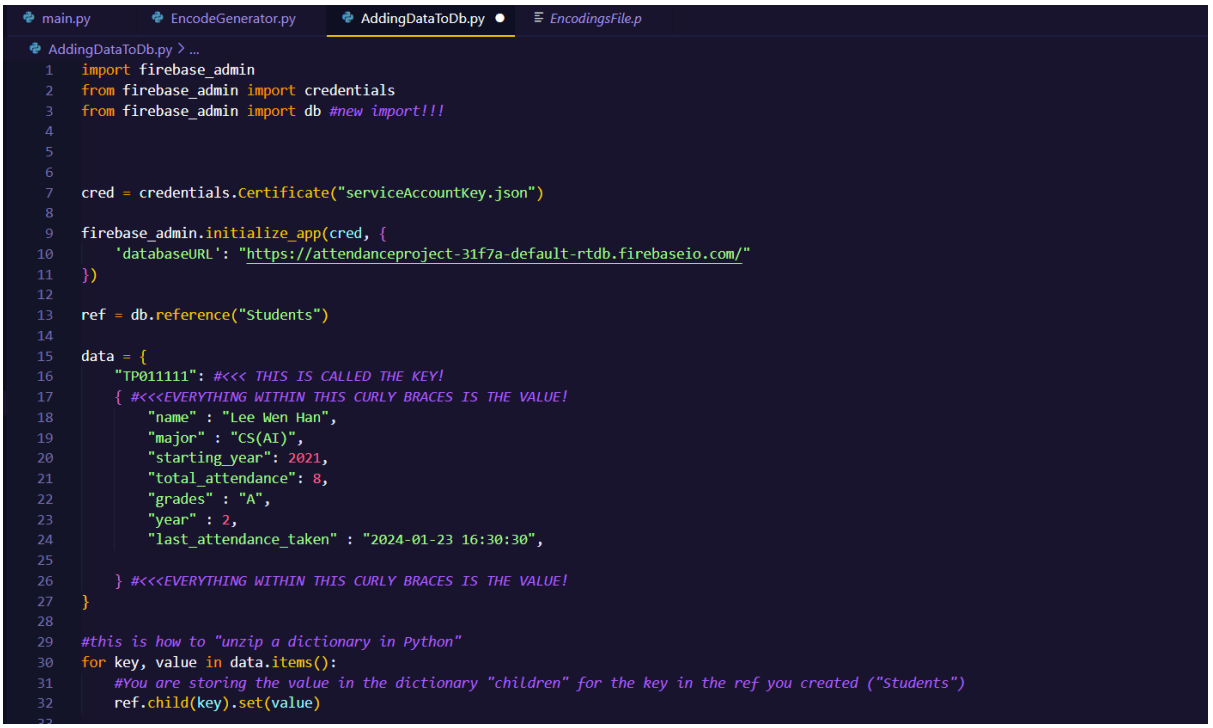
AddingDataToDb.py > ...

```python
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db #new import!!!



cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred, {
    'databaseURL': "https://attendanceproject-31f7a-default-rtdb.firebaseio.com/"
})

ref = db.reference("Students")

data = {
    "TP011111":
    {
        "name" : "Lee Wen Han",
        "major" : "CS(AI)",
        "starting_year": 2021,
        "total_attendance": 8,
        "grades" : "A",
        "year" : 2,
        "last_attendance_taken" : "2024-01-23 16:30:30",

    }
}

```

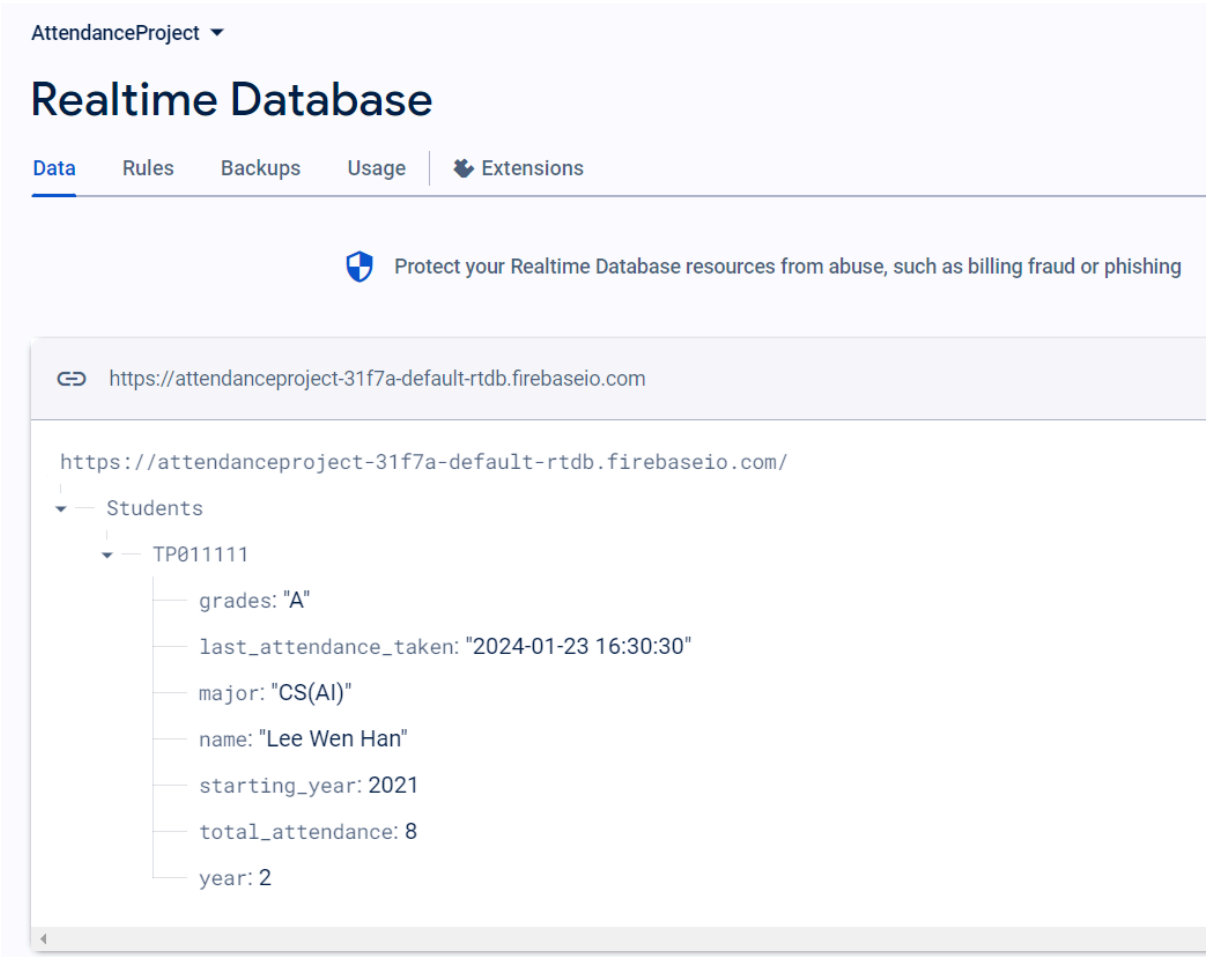
- Firstly, you wanna import db from firebase_admin
- Next, you'll want to create a reference called "Students" which tells your program: hey, I want to identify/reference this node in my database! Once you have your reference, you'll be able to perform actions on the data stored in the "Students" node of your database
- From line 15 to 26, you are essentially defining a Python dictionary called "data"
- This dictionary currently contains information about a single student named LEE WEN HAN
- The UNIQUE IDENTIFIER for this student is what precedes his details, which is his tp number



- Check your realtime database now!
- There should be completely nothing inside



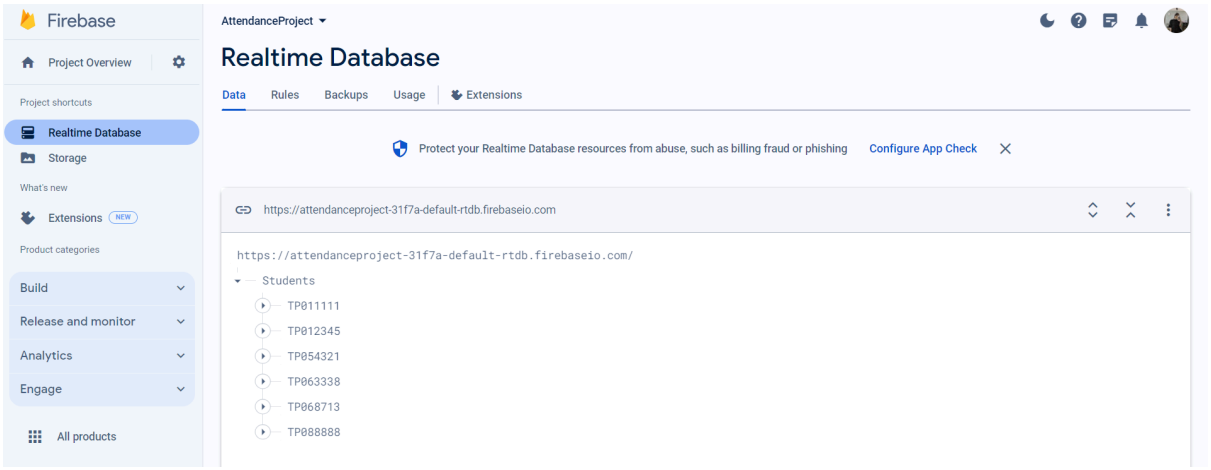
- In the screenshot above, I'm explaining how a dictionary works. Within your dictionary called DATA, you have created a new "item"/element which is named "TP011111". The information stored about this element is below it.
- TP011111 is the key
- Every information within it is known as the value
- Unzip the information within the data dictionary and store into your ref ("Students") the values and their children!
- RUN YOUR PROGRAM!



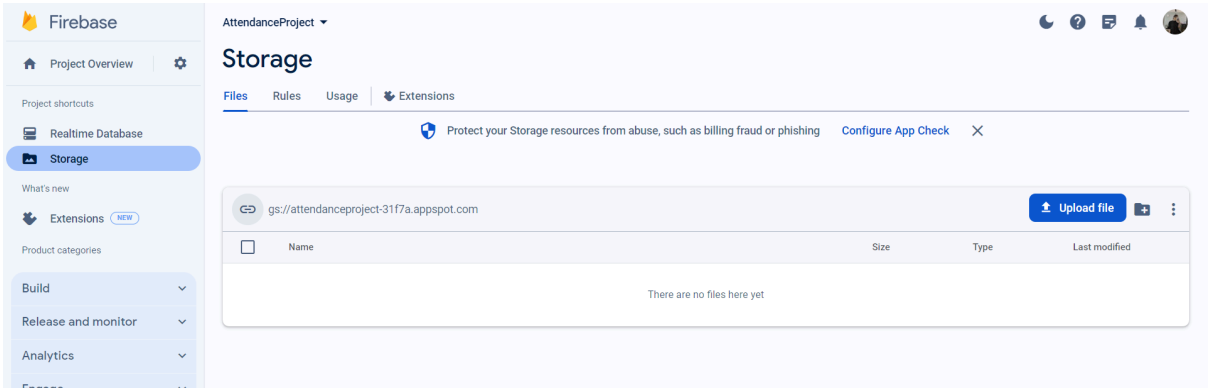
- REFRESH YOUR REALTIME DATABASE TAB AND TA DAH! Your dictionary should appear in it now!
- It's called realtime because it refreshes and updates instantly
- You can try playing around with the database details in your program and running the program to see the information change in your realtime database!

```
    },
    "TP054321":
    {
        "name" : "Elon Musk",
        "major" : "Ruining Lives",
        "starting_year": 2020,
        "total_attendance": 100,
        "grades" : "A+",
        "year" : 4,
        "last_attendance_taken" : "2024-01-22 01:30:30",
    },
    "TP063338":
    {
        "name" : "Dalton Gan",
        "major" : "SE",
        "starting_year": 2020,
        "total_attendance": 64,
        "grades" : "A+",
        "year" : 2,
        "last_attendance_taken" : "2024-01-22 01:30:30",
    },
    "TP068713":
    {
        "name" : "Suzanne Lai",
        "major" : "CS(AI)",
        "starting_year": 2021,
        "total_attendance": 10,
        "grades" : "A+",
        "year" : 2,
        "last_attendance_taken" : "2024-01-22 01:30:30",
    },
```

- You can go ahead and add all the details of your students now!

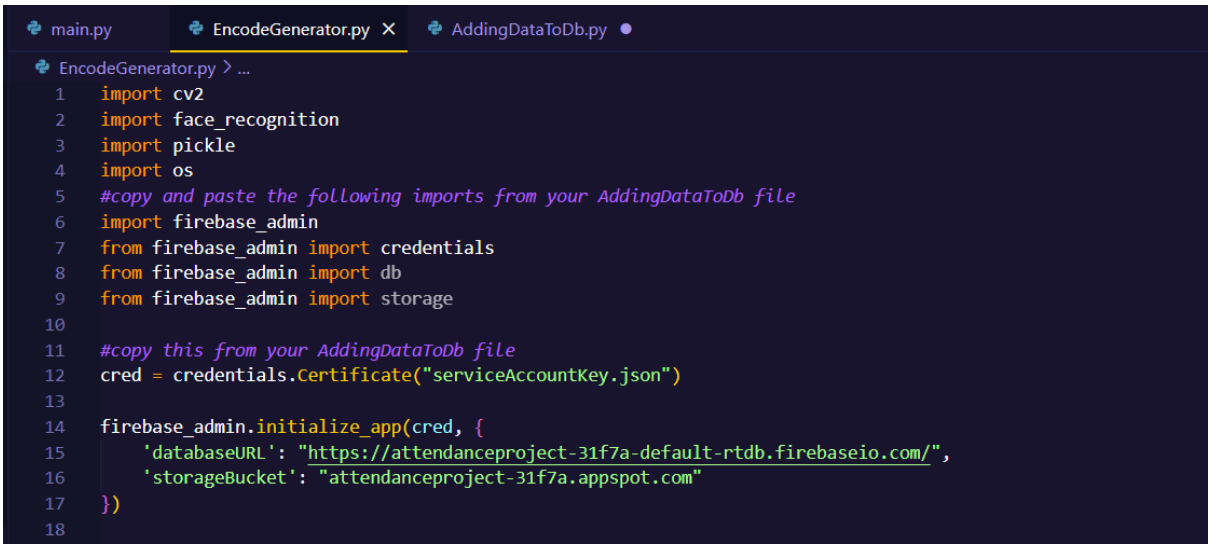


- Run the program and see the changes reflected in your database
- All my newly-added students are inside now



- Copy the link to your storage

EncodeGenerator.py



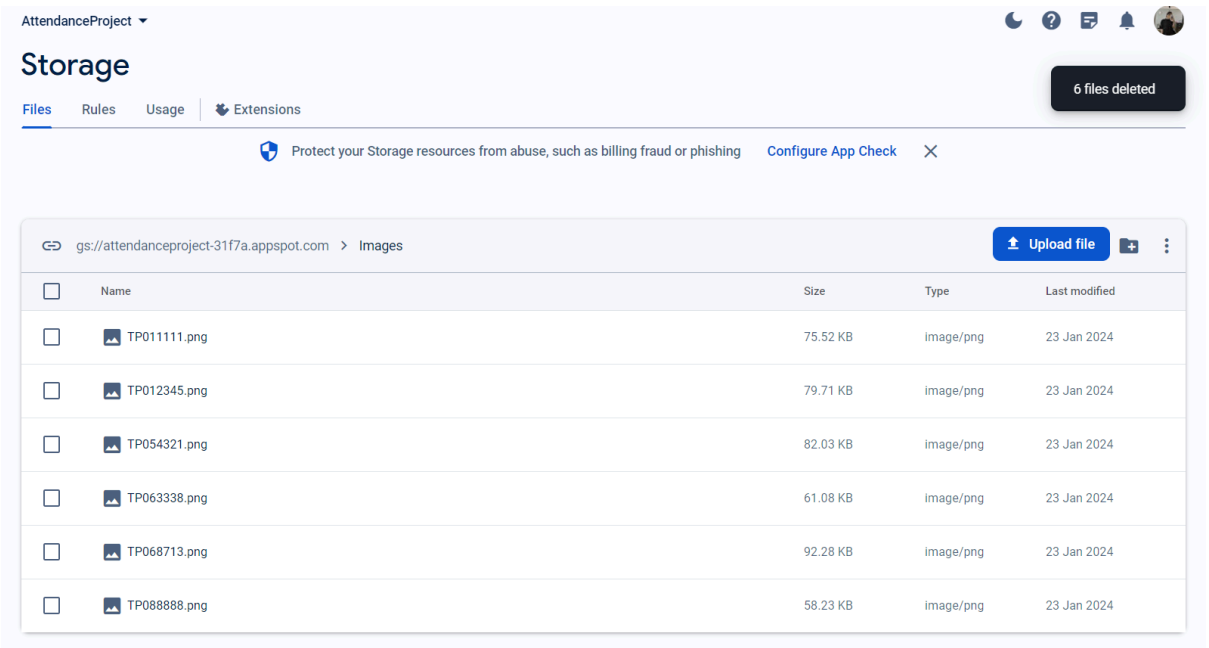
- You'll want to set up the connection to your firebase database and storage in your EncodeGenerator.py file
- Remove the "gs://" from the url you found and copied from your Storage and initialize it as your storageBucket

main.py | EncodeGenerator.py | AddingDataToDb.py

EncodeGenerator.py > ...

```python
#copy this from your AddingDataToDb file
cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred, {
    'databaseURL': "https://attendanceproject-31f7a-default-rtdb.firebaseio.com/",
    'storageBucket': "attendanceproject-31f7a.appspot.com"
})

#importing students' images from Images by using the same logic as earlier
folderPathImages = 'Images'
listPathImages = os.listdir(folderPathImages)
imgListImages = []


#Extracting the student ID from each picture
studentIDs = []

for path in listPathImages:
    imgListImages.append(cv2.imread(os.path.join(folderPathImages, path)))
    studentIDs.append(os.path.splitext(path)[0])

    #you're already going through the images one by one here, so you might as well use this loop to store them as well
    #join together the full filename using folderPathImages as well as each individual path
    fileName = f'{folderPathImages}/{path}'
    #the following code will retrieve a reference to your established cloud storage
    bucket = storage.bucket()
    #a reference to a specific object is created
    blob = bucket.blob(fileName)
    #the file with the fileName is uploaded to the reference "blob" created in your cloud storage
    blob.upload_from_filename(fileName)

```

- Still inside your EncodeGenerator file, you'll want to reuse the for loop that was used to iterate through the images in your Images folder
- Since it is going through each image anyway, it makes sense that you'll use this loop to upload the images onto your cloud storage
- You'll want to form the full fileName by joining the folder path and each individual path
- Create a reference to your established connection to a cloud storage
- Create a reference within the bucket using your fileName - this is called a "blob"
- The specific image is uploaded to the storage destination referenced by the "blob"



- Run the application and refresh the Storage page! The images should all be uploaded onto it now

You’re a natural babe 🤩

2.6 Update database

```python
import cv2
import face_recognition
import pickle
import os
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage

cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred, {
    'databaseURL': "https://attendanceproject-31f7a-default-rtdb.firebaseio.com/",
    'storageBucket': "attendanceproject-31f7a.appspot.com"
})
```

- Copy all of these lines from EncodeGenerator.py

main.py

```python
#HIIII This is Suzanne
import cv2
import os
import pickle
import face_recognition
import numpy as np
import cvzone
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage

cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred, {
    'databaseURL': "https://attendanceproject-31f7a-default-rtdb.firebaseio.com/",
    'storageBucket': "attendanceproject-31f7a.appspot.com"
})
```

- Paste it into main.py

```python
#create a new variable called modeType
modeType = 0
counter = 0
id = -1

while True:
    success, image = capture.read()

    backgroundImg[162:162+480, 55:55+640] = image
    #overlay the FIRST element within imgListMode on top of the background
    backgroundImg[44:44+633, 808:808+414] = imgListMode[0]

    #resize image because we will require a lot of computational power if we try to encode images that are large
    smallImage = cv2.resize(image, (0,0), None, 0.25, 0.25)
    smallImage = cv2.cvtColor(smallImage, cv2.COLOR_BGR2RGB)
```

- Right before your while loop starts, you want to create a variable called modeType and assign it with the value of 0
- Also create a new variable called counter and assign it the value of 0 as well

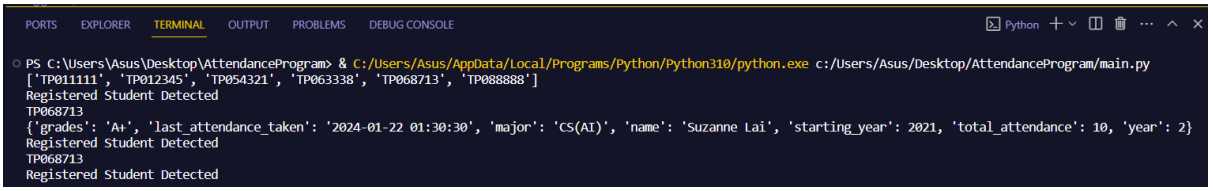
- Create a variable called id and equate it to -1

```
42  #create a new variable called modeType
43  modeType = 0
44  counter = 0
45  id = -1
46
47  while True:
48      success, image = capture.read()
49
50      backgroundImg[162:162+480, 55:55+640] = image
51      #overlay the FIRST element within imgListMode on top of the background
52      backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]
53
```

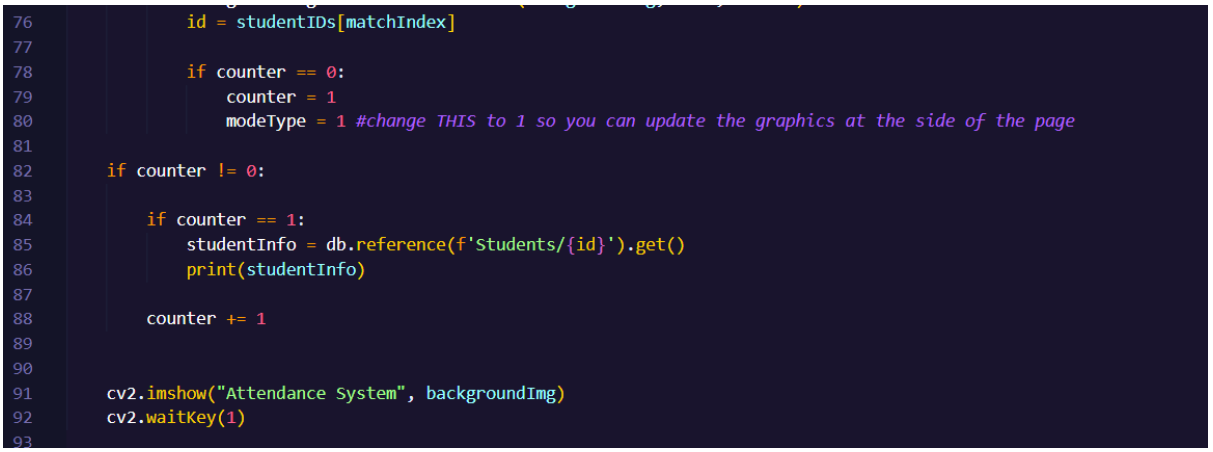
- Change imgListMode[0] to imgListMode[modeType] in line 52

```
58      #Locating the face in the current frame
59      faceCurrentFrame = face_recognition.face_locations(smallImage)
60      #converting facial features in the current frame into encodings
61      encodeCurrentFrame = face_recognition.face_encodings(smallImage, faceCurrentFrame)
62
63      for encodeFace, faceLocation in zip(encodeCurrentFrame, faceCurrentFrame):
64          matches = face_recognition.compare_faces(encodingsListKnown, encodeFace)
65          faceDistance = face_recognition.face_distance(encodingsListKnown, encodeFace)
66          matchIndex = np.argmin(faceDistance)
67
68          if matches[matchIndex]:
69              print("Registered Student Detected")
70              print(studentIDs[matchIndex])
71
72              y1, x2, y2, x1 = faceLocation
73              y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
74              bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
75              backgroundImg = cvzone.cornerRect(backgroundImg, bbox, rt = 0)
76              id = studentIDs[matchIndex]
77
78              if counter == 0:
79                  counter = 1
80
81      if counter != 0:
82
83          if counter == 1:
84              studentInfo = db.reference(f'Students/{id}').get()
85              print(studentInfo)
86
87          counter += 1
88
89
90      cv2.imshow("Attendance System", backgroundImg)
91      cv2.waitKey(1)
92
```

- Within the for loop for face encodings, you want to add line 76, which saves the current id as the match to the face in the current frame/webcam
- The counter is changed to 1 in line 79
- Check if the counter is not 0. If the counter is not 0:
    - Check if counter is equal to 1. If it is equal to 1, then retrieve the information of the student from the database and store it into the variable studentInfo
    - Print the studentInfo saved

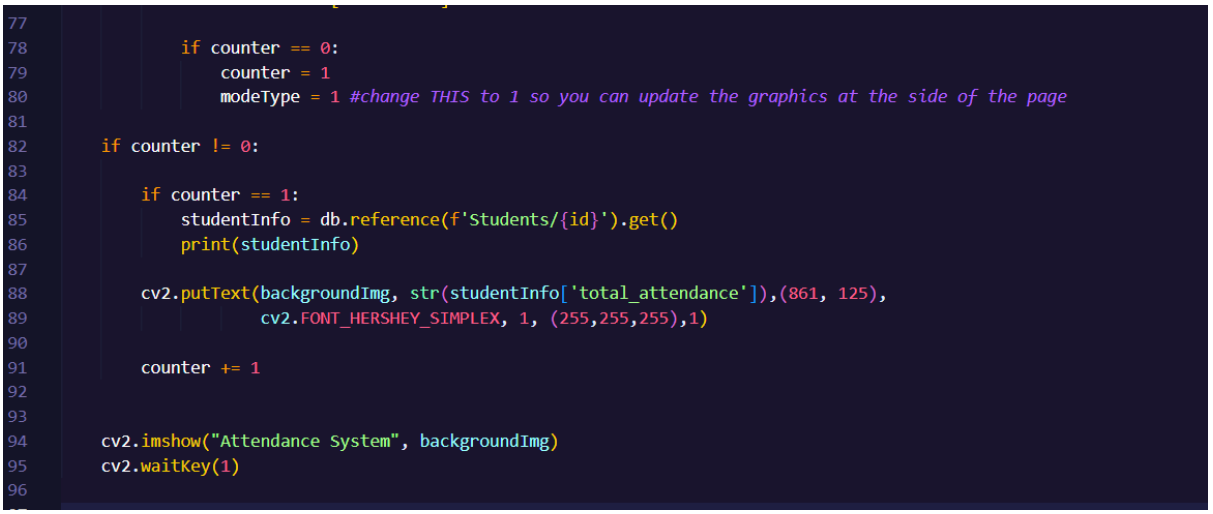


- As seen from the terminal after running the program, the system successfully identifies me as TP068173 and returns all of my information



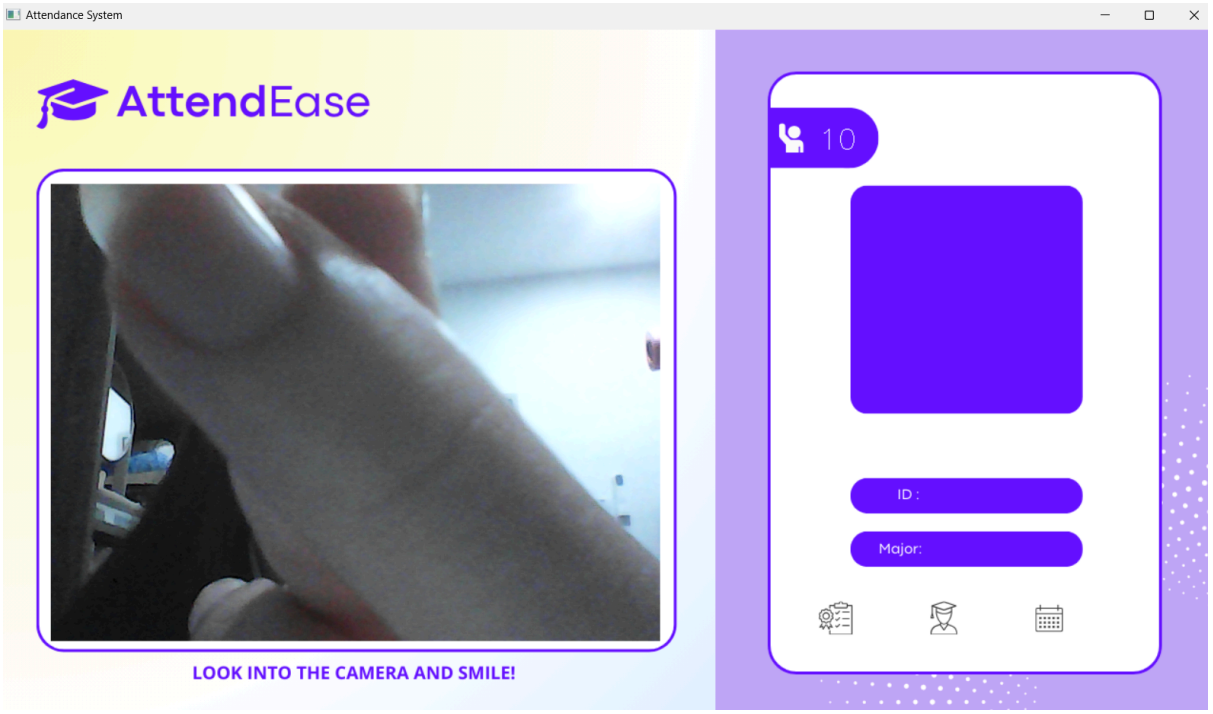
```
76              id = studentIDs[matchIndex]
77
78              if counter == 0:
79                  counter = 1
80                  modeType = 1 #change THIS to 1 so you can update the graphics at the side of the page
81
82      if counter != 0:
83
84          if counter == 1:
85              studentInfo = db.reference(f'Students/{id}').get()
86              print(studentInfo)
87
88          counter += 1
89
90
91      cv2.imshow("Attendance System", backgroundImg)
92      cv2.waitKey(1)
93
```

- In order to successfully change the graphics in the system according to the state, change modeType =1 when counter is 0
- This means that when counter = 1, modeType = 1
- This will directly affect the if loop in line 84



```
77
78              if counter == 0:
79                  counter = 1
80                  modeType = 1 #change THIS to 1 so you can update the graphics at the side of the page
81
82      if counter != 0:
83
84          if counter == 1:
85              studentInfo = db.reference(f'Students/{id}').get()
86              print(studentInfo)
87
88          cv2.putText(backgroundImg, str(studentInfo['total_attendance']),(861, 125),
89                      cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),1)
90
91          counter += 1
92
93
94      cv2.imshow("Attendance System", backgroundImg)
95      cv2.waitKey(1)
96
```

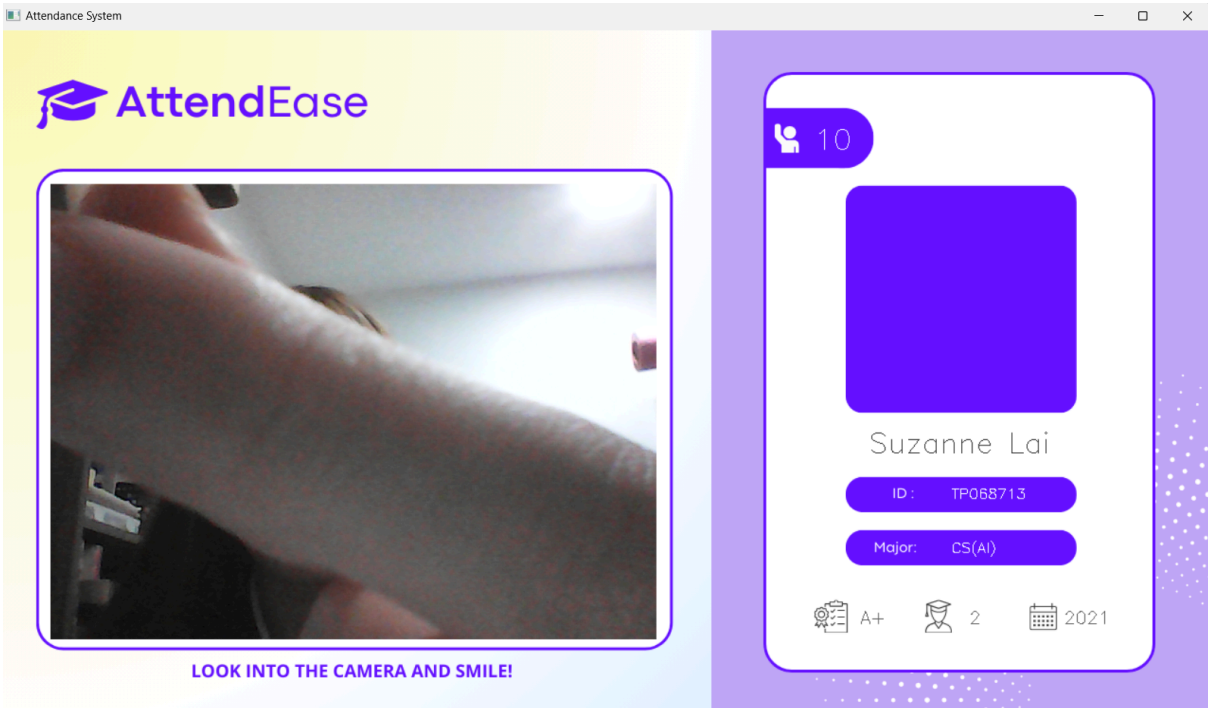
- Add the code from line 88 - 89 to display our total attendance on the background image



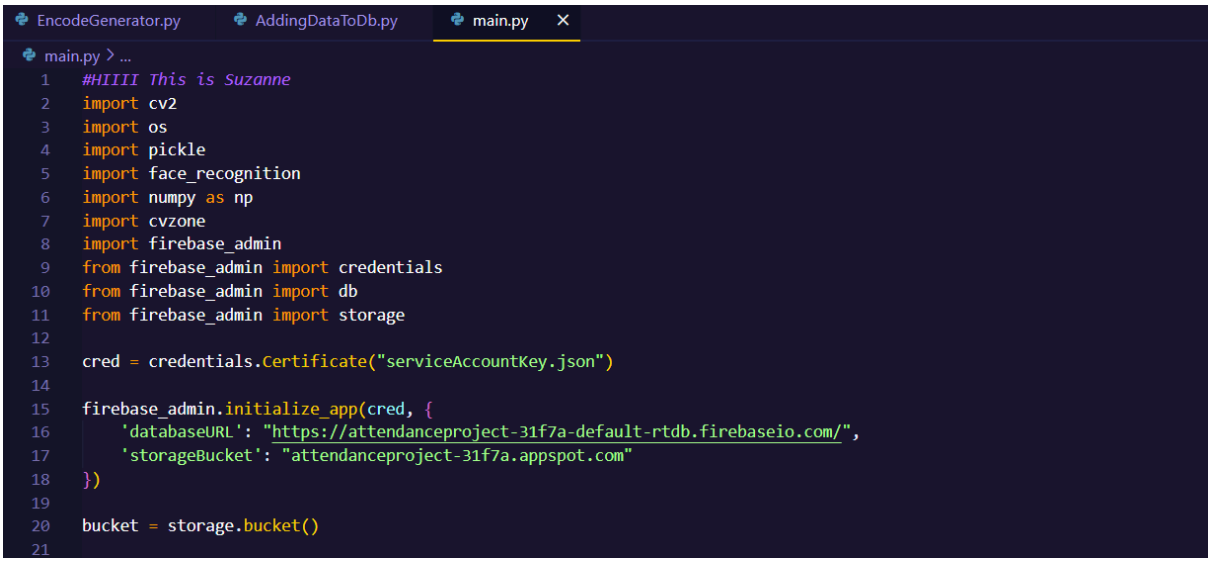
- After you run the program and have let the program detect and recognize your face for a while, your total attendance will be stored successfully, as seen in the number appearing on the screen that was previously not there

```
82      if counter != 0:
83
84          if counter == 1:
85              studentInfo = db.reference(f'Students/{id}').get()
86              print(studentInfo)
87
88          cv2.putText(backgroundImg, str(studentInfo['total_attendance']),(861, 125),
89                      cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),1)
90          cv2.putText(backgroundImg, str(studentInfo['major']), (1006,550),
91                      cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255),1)
92          cv2.putText(backgroundImg, str(id),(1006,493),
93                      cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),1)
94          cv2.putText(backgroundImg, str(studentInfo['grades']),(910,625),
95                      cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
96          cv2.putText(backgroundImg, str(studentInfo['year']),(1025,625),
97                      cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
98          cv2.putText(backgroundImg, str(studentInfo['starting_year']),(1125,625),
99                      cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
100
101         #in order to center the name, calculation needs to be performed
102         (width, height), _ = cv2.getTextSize(studentInfo['name'],cv2.FONT_HERSHEY_SIMPLEX,1,1)
103
104         #we are using a double // because that gets rid of trailing float numbers
105         center_offset = (414 - width)//2
106         cv2.putText(backgroundImg, str(studentInfo['name']),(808 + center_offset,445),
107                     cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,50),1)
108
109         counter += 1
110
```

- Add in these lines of code to ensure that all of these fields show up in the system
- Ensure that the name is centered by using the center_offset method



- As seen in the screenshot, when the program runs, all of my data is being fed into the system



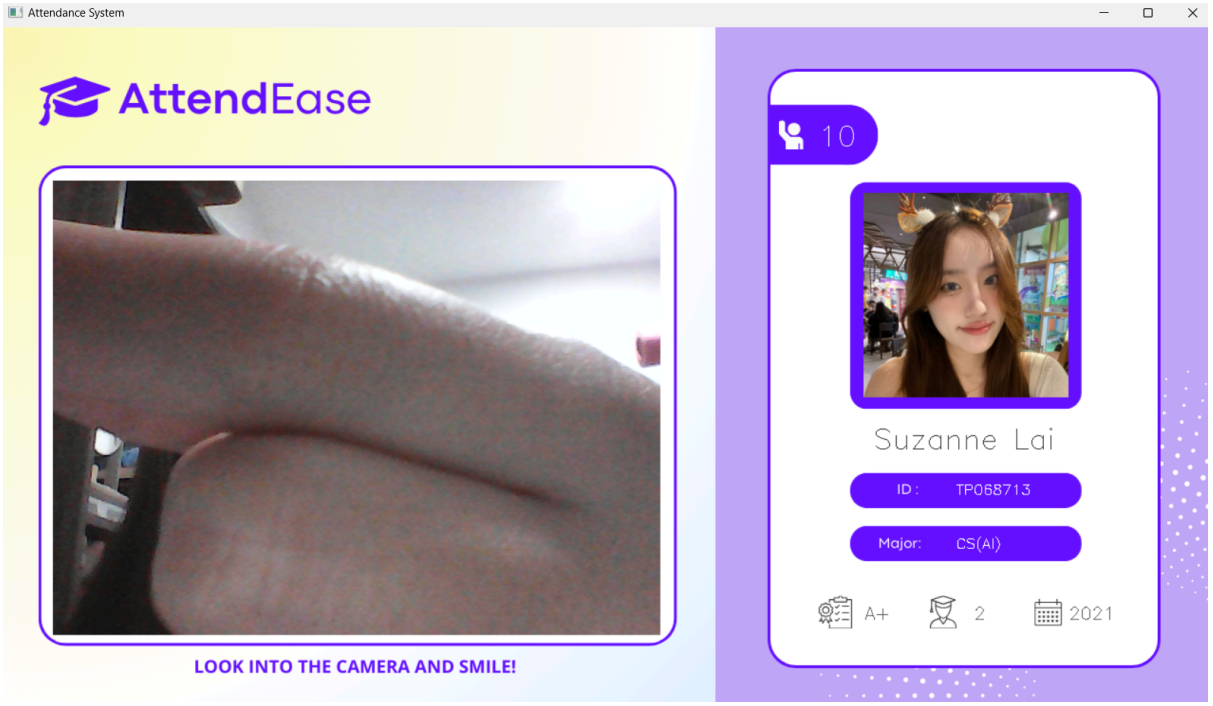
- Create a storage bucket after the initialization of the firebase app

```
83
84      if counter != 0:
85
86          if counter == 1:
87              #Retrieving students information from the database
88              studentInfo = db.reference(f'Students/{id}').get()
89              print(studentInfo)
90
91              #Retrieving student's image from the storage
92              blob = bucket.get_blob(f'Images/{id}.png')
93              array = np.frombuffer(blob.download_as_string(), np.uint8)
94              studentImg = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
95
96
97          cv2.putText(backgroundImg, str(studentInfo['total_attendance']),(861, 125),
98                      cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),1)
99          cv2.putText(backgroundImg, str(studentInfo['major']), (1006,550),
100                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255),1)
101         cv2.putText(backgroundImg, str(id),(1006,493),
102                     cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),1)
103         cv2.putText(backgroundImg, str(studentInfo['grades']),(910,625),
104                     cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
105         cv2.putText(backgroundImg, str(studentInfo['year']),(1025,625),
106                     cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
107         cv2.putText(backgroundImg, str(studentInfo['starting_year']),(1125,625),
108                     cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
109
```

- The lines from 92 to 94 tell you how to retrieve images from storage, download and convert them to numpy arrays, then decoded to become color-corrected images

```
97          cv2.putText(backgroundImg, str(studentInfo['total_attendance']),(861, 125),
98                      cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),1)
99          cv2.putText(backgroundImg, str(studentInfo['major']), (1006,550),
100                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255),1)
101         cv2.putText(backgroundImg, str(id),(1006,493),
102                     cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),1)
103         cv2.putText(backgroundImg, str(studentInfo['grades']),(910,625),
104                     cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
105         cv2.putText(backgroundImg, str(studentInfo['year']),(1025,625),
106                     cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
107         cv2.putText(backgroundImg, str(studentInfo['starting_year']),(1125,625),
108                     cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
109
110         #in order to center the name, calculation needs to be performed
111         (width, height), _ = cv2.getTextSize(studentInfo['name'],cv2.FONT_HERSHEY_SIMPLEX,1,1)
112
113         #we are using a double // because that gets rid of trailing float numbers
114         center_offset = (414 - width)//2
115         cv2.putText(backgroundImg, str(studentInfo['name']),(808 + center_offset,445),
116                     cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,50),1)
117
118         backgroundImg[175:175 + 216, 909:909 + 216 ] = studentImg
119
120         counter += 1
121
```

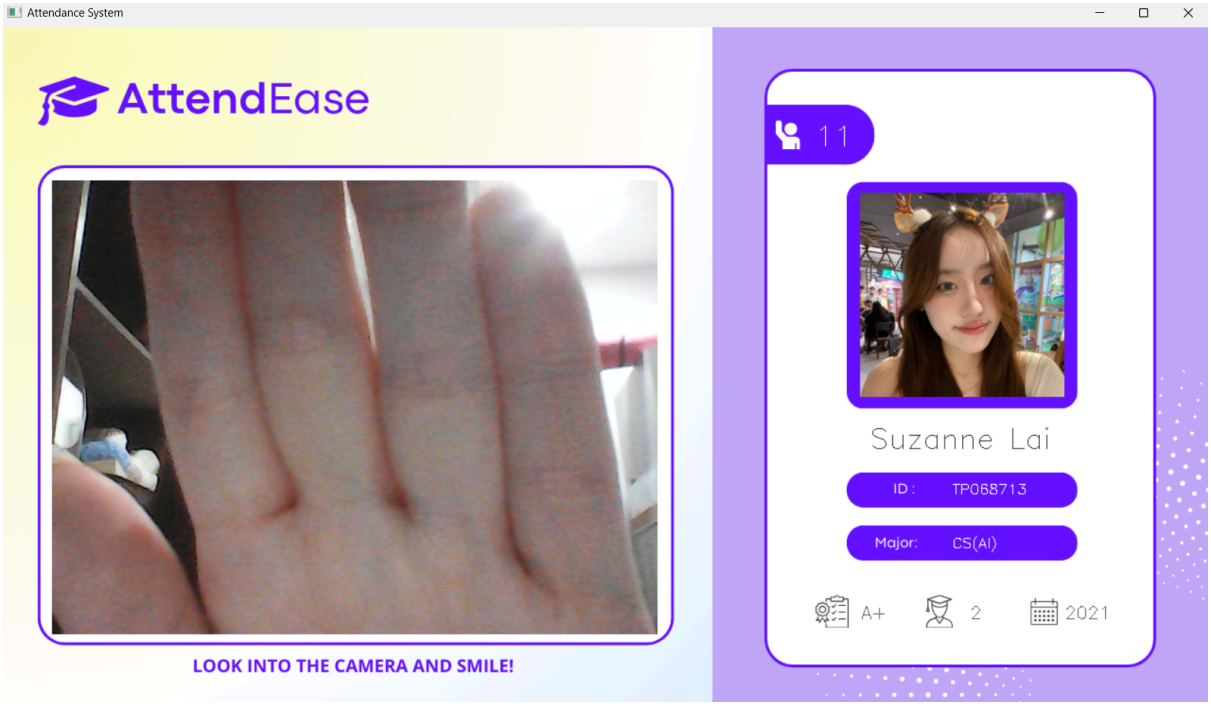
- In line 118, you are taught how to past the image onto our background

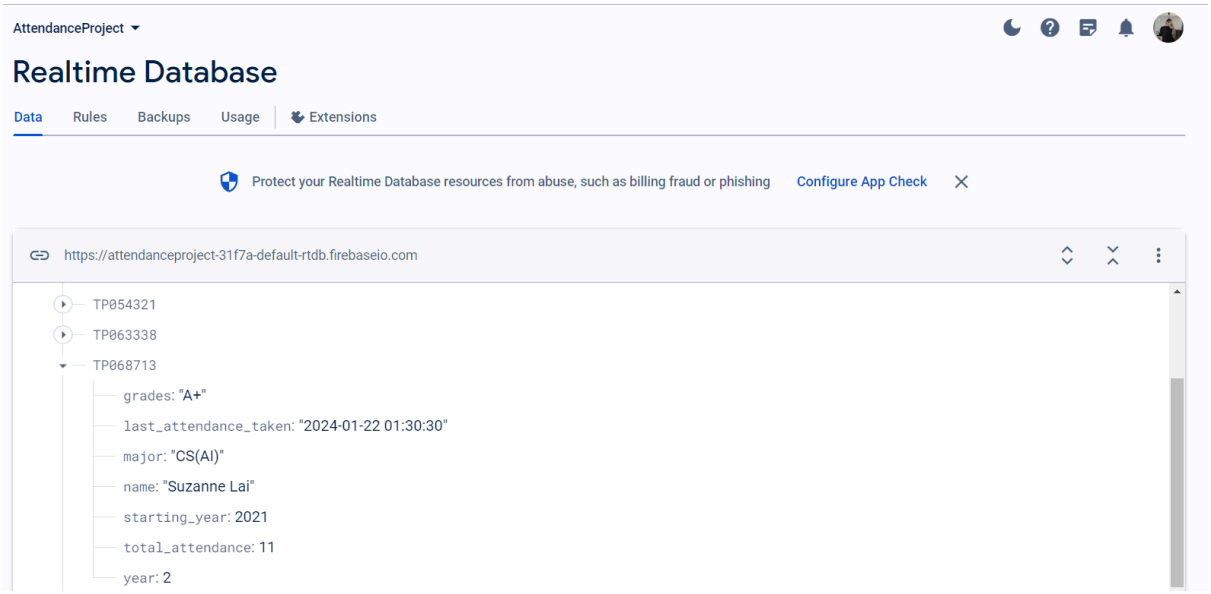


- Now, your picture shows up when the system recognizes you!
- The next step is to ensure that the system can help you UPDATE you attendance in real time

```
84      if counter != 0:
85
86          if counter == 1:
87              #Retrieving students information from the database
88              studentInfo = db.reference(f'Students/{id}').get()
89              print(studentInfo)
90
91              #Retrieving student's image from the storage
92              blob = bucket.get_blob(f'Images/{id}.png')
93              array = np.frombuffer(blob.download_as_string(), np.uint8)
94              studentImg = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
95
96              #Update attendance record and data
97              ref = db.reference(f'Students/{id}')
98              studentInfo['total_attendance'] += 1
99
100             ref.child('total_attendance').set(studentInfo['total_attendance'])
101
```

- At the segment of the code where you are checking if the counter is equal to 1, you can add some more lines to update your attendance record.
- As usual, you'll need to create a reference point within your database
- We have to update the 'total_attendance' field/column within studentInfo to be incremented by 1
- Finally, the new data is sent to the database





- Upon running the program again, you'll notice that, while my previous attendance record was 10, its is now 11!

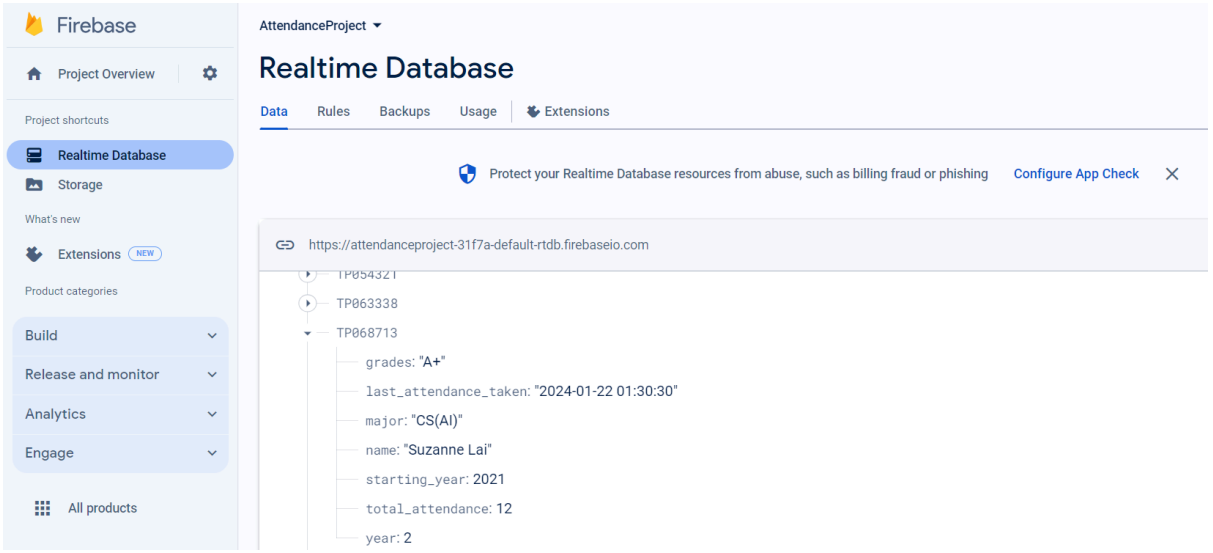
```
101
102         if counter < 10:
103
104             #indent this properly
105             #all will execute while modeType = 1
106
107             cv2.putText(backgroundImg, str(studentInfo['total_attendance']),(861, 125),
108                         cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),1)
109             cv2.putText(backgroundImg, str(studentInfo['major']), (1006,550),
110                         cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255),1)
111             cv2.putText(backgroundImg, str(id),(1006,493),
112                         cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),1)
113             cv2.putText(backgroundImg, str(studentInfo['grades']),(910,625),
114                         cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
115             cv2.putText(backgroundImg, str(studentInfo['year']),(1025,625),
116                         cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
117             cv2.putText(backgroundImg, str(studentInfo['starting_year']),(1125,625),
118                         cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
119
120             #in order to center the name, calculation needs to be performed
121             (width, height), _ = cv2.getTextSize(studentInfo['name'],cv2.FONT_HERSHEY_SIMPLEX,1,1)
122
123             #we are using a double // because that gets rid of trailing float numbers
124             center_offset = (414 - width)//2
125             cv2.putText(backgroundImg, str(studentInfo['name']),(808 + center_offset,445),
126                         cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,50),1)
127
128             backgroundImg[175:175 + 216, 909:909 + 216 ] = studentImg
129
```

- Before the section where you put in a bunch of Texts to fill up the graphics on the right, you'll want to INDENT lines 107 - 128
- Add an if condition that controls this loop so that the contents are only executed if the counter is > 20

```
90
91             #Retrieving student's image from the storage
92             blob = bucket.get_blob(f'Images/{id}.png')
93             array = np.frombuffer(blob.download_as_string(), np.uint8)
94             studentImg = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
95
96             #Update attendance record and data
97             ref = db.reference(f'Students/{id}')
98             studentInfo['total_attendance'] += 1
99
100            ref.child('total_attendance').set(studentInfo['total_attendance'])
101
102
103        if 10 < counter < 20:
104            modeType = 2
105
106        backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]
107
108
109        if counter < 10:
110
111            #indent this properly
112            #all will execute while modeType = 1
113
114            cv2.putText(backgroundImg, str(studentInfo['total_attendance']),(861, 125),
115                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),1)
116            cv2.putText(backgroundImg, str(studentInfo['major']), (1006,550),
117                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255),1)
```

- To ensure that the system can show dynamic change in the graphics so the users know what their attendance status is, a new if condition is added such that if counter is between 10 and 20, the modeType is changed to 2

- Now, upon marking my attendance, the graphics on the right will show “Attendance Marked” instead of my student details

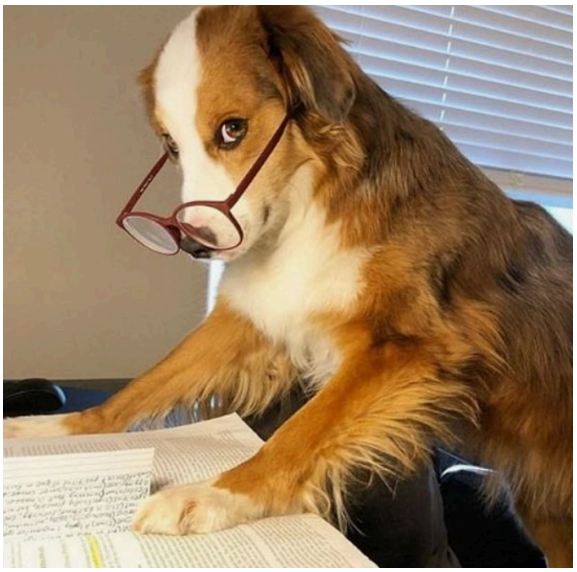


- Also, my realtime database will show that I have 12 attendances taken now since I ran the program and it recognized me again

```
                    cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,50),1)

            backgroundImg[175:175 + 216, 909:909 + 216 ] = studentImg

        counter += 1

        if counter > 20:
            counter = 0
            modeType = 0
            studentInfo = []
            studentImg = []
            backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]


    cv2.imshow("Attendance System", backgroundImg)
    cv2.waitKey(1)

```

- If the counter is more than 20, have your system "reset" by clearing all previously held data out

Challenge: run it again and see what happens when you let the program to continue running for ~2 minutes!



What does it feel like to always be the smartest person in the room?

2.7 Validation and time elapsed

If you've already tried letting the program run for a while, you will notice that although the program is already doing what it's intended to do - which is to take your attendance if it sees and recognizes your face, there is absolutely no validation.

I could stand in front of the webcam all day and farm my attendance! This can't do in real-life, so here's your fix!

```
import cvzone
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
from datetime import datetime #NEW IMPORT

cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred, {
```

- Import datetime into your program

```
96
97              #Update attendance record and data
98
99              datetimeObject = datetime.strptime(studentInfo['last_attendance_taken'],
100                                                "%Y-%m-%d %H:%M:%S")
101             timeInSecsElapsed = (datetime.now() - datetimeObject).total_seconds()
102             print(timeInSecsElapsed)
103
104             ref = db.reference(f'Students/{id}')
105             studentInfo['total_attendance'] += 1
106
107             ref.child('total_attendance').set(studentInfo['total_attendance'])
108             ref.child('last_attendance_taken').set(datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
109
110
```

- In the section of the code where you are updating your attendance record, create a datetimeObject which stores the time in which your last attendance was taken
- The time elapsed in seconds was calculate by using the current datetime to subtract with the datetimeObject
- The last_attendance_taken field is updated with the current datetime, formatted accordingly

```
97              #Update attendance record and data
98
99              datetimeObject = datetime.strptime(studentInfo['last_attendance_taken'],
100                                                "%Y-%m-%d %H:%M:%S")
101             timeInSecsElapsed = (datetime.now() - datetimeObject).total_seconds()
102             if timeInSecsElapsed > 30:
103
104                 ref = db.reference(f'Students/{id}')
105                 studentInfo['total_attendance'] += 1
106
107                 ref.child('total_attendance').set(studentInfo['total_attendance'])
108                 ref.child('last_attendance_taken').set(datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
109             else:
110                 modeType = 3
111                 counter = 0
112                 backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]
113
```

- The new line 102 indicates that if the time elapsed has exceeded a certain threshold, you will be able to take your attendance and update the database accordingly.
- Otherwise, the modeType is changed to 3 (“Already Marked”) and the counter is reset while the background is changed appropriately
-

```
113
114         if modeType != 3:
115
116         if 10 < counter < 20:
117             modeType = 2
118
119         backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]
120
121
122         if counter < 10:
123
124             #indent this properly
125             #all will execute while modeType = 1
126
127             cv2.putText(backgroundImg, str(studentInfo['total_attendance']),(861, 125),
128                         cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),1)
129             cv2.putText(backgroundImg, str(studentInfo['major']), (1006,550),
130                         cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255),1)
131             cv2.putText(backgroundImg, str(id),(1006,493),
132                         cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),1)
133             cv2.putText(backgroundImg, str(studentInfo['grades']),(910,625),
134                         cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
135             cv2.putText(backgroundImg, str(studentInfo['year']),(1025,625),
136                         cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
137             cv2.putText(backgroundImg, str(studentInfo['starting_year']),(1125,625),
```

```
                    cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),1)
            cv2.putText(backgroundImg, str(studentInfo['grades']),(910,625),
                    cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
            cv2.putText(backgroundImg, str(studentInfo['year']),(1025,625),
                    cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
            cv2.putText(backgroundImg, str(studentInfo['starting_year']),(1125,625),
                    cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)

            #in order to center the name, calculation needs to be performed
            (width, height), _ = cv2.getTextSize(studentInfo['name'],cv2.FONT_HERSHEY_SIMPLEX,1,1)

            #we are using a double // because that gets rid of trailing float numbers
            center_offset = (414 - width)//2
            cv2.putText(backgroundImg, str(studentInfo['name']),(808 + center_offset,445),
                    cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,50),1)

            backgroundImg[175:175 + 216, 909:909 + 216 ] = studentImg

        counter += 1

        if counter > 20:
            counter = 0
            modeType = 0
            studentInfo = []
            studentImg = []
            backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]


    cv2.imshow("Attendance System", backgroundImg)
    cv2.waitKey(1)
```

- In order to ensure that there are no other images/texts interrupting my whole screen to the right which tells me my attendance has already been marked, we can include a new if condition
- If the modeType is NOT 3, the highlight the above lines and INDENT IT ONCE

```
while True:
    success, image = capture.read()

    backgroundImg[162:162+480, 55:55+640] = image
    #overlay the FIRST element within imgListMode on top of the background
    backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]

    #resize image because we will require a lot of computational power if we try to encode images that are large
    smallImage = cv2.resize(image, (0,0), None, 0.25, 0.25)
    smallImage = cv2.cvtColor(smallImage, cv2.COLOR_BGR2RGB)

    #locating the face in the current frame
    faceCurrentFrame = face_recognition.face_locations(smallImage)
    #converting facial features in the current frame into encodings
    encodeCurrentFrame = face_recognition.face_encodings(smallImage, faceCurrentFrame)


    if faceCurrentFrame:

    for encodeFace, faceLocation in zip(encodeCurrentFrame, faceCurrentFrame):
        matches = face_recognition.compare_faces(encodingsListKnown, encodeFace)
        faceDistance = face_recognition.face_distance(encodingsListKnown, encodeFace)
        matchIndex = np.argmin(faceDistance)
```

```
66
67      if faceCurrentFrame:
68
69          for encodeFace, faceLocation in zip(encodeCurrentFrame, faceCurrentFrame):
70              matches = face_recognition.compare_faces(encodingsListKnown, encodeFace)
71              faceDistance = face_recognition.face_distance(encodingsListKnown, encodeFace)
72              matchIndex = np.argmin(faceDistance)
73
74              if matches[matchIndex]:
75                  print("Registered Student Detected")
76                  print(studentIDs[matchIndex])
77
78                  y1, x2, y2, x1 = faceLocation
79                  y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
80                  bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
81                  backgroundImg = cvzone.cornerRect(backgroundImg, bbox, rt = 0)
82                  id = studentIDs[matchIndex]
83
84                  if counter == 0:
85                      counter = 1
86                      modeType = 1
87
88      if counter != 0:
```

```
EncodeGenerator.py   AddingDataToDb.py   main.py 2 ×
main.py > ...
141                     cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
142
143                 #in order to center the name, calculation needs to be performed
144                 (width, height), _ = cv2.getTextSize(studentInfo['name'],cv2.FONT_HERSHEY_SIMPLEX,1,1)
145
146                 #we are using a double // because that gets rid of trailing float numbers
147                 center_offset = (414 - width)//2
148                 cv2.putText(backgroundImg, str(studentInfo['name']),(808 + center_offset,445),
149                     cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,50),1)
150
151                 backgroundImg[175:175 + 216, 909:909 + 216 ] = studentImg
152
153             counter += 1
154
155             if counter > 20:
156                 counter = 0
157                 modeType = 0
158                 studentInfo = []
159                 studentImg = []
160                 backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]
161
162
163     cv2.imshow("Attendance System", backgroundImg)
164     cv2.waitKey(1)
165
166
```

- We want the program to reset if there are no faces detected
- To do this, we add a new line before the zipping of our face encoding and face mapping while sets a condition that only IF faces are attacked do the following lines of code get executed
- Everything below that if statement is indented except for the last two

```
152
153             counter += 1
154
155             if counter > 20:
156                 counter = 0
157                 modeType = 0
158                 studentInfo = []
159                 studentImg = []
160                 backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]
161
162     else:
163         modeType =  0
164         counter = 0
165
166     cv2.imshow("Attendance System", backgroundImg)
167     cv2.waitKey(1)
168
169
```

- Finally, add the else statement that resets the system if no faces are detected.

LET'S DO A FINAL CHECK!

main.py

main.py > ...

```python
#HIIII This is Suzanne
import cv2
import os
import pickle
import face_recognition
import numpy as np
import cvzone
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
from datetime import datetime #NEW IMPORT

cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred, {
    'databaseURL': "https://attendanceproject-31f7a-default-rtdb.firebaseio.com/",
    'storageBucket': "attendanceproject-31f7a.appspot.com"
})

bucket = storage.bucket()

capture = cv2.VideoCapture(0)
capture.set(3, 640)
capture.set(4, 480)
backgroundImg = cv2.imread('Resources/Background.png')

#importing images from Modes
folderPathMode = 'Resources/Modes'
listPathMode = os.listdir(folderPathMode)
imgListMode = []

for path in listPathMode:
    imgListMode.append(cv2.imread(os.path.join(folderPathMode, path)))

```

```python
#importing encodings into main.py
encodingsFile = open("EncodingsFile.p", "rb")
encodingsListWithIDs = pickle.load(encodingsFile)
encodingsFile.close()

encodingsListKnown, studentIDs = encodingsListWithIDs
print(studentIDs)

#create a new variable called modeType
modeType = 0
counter = 0
id = -1

while True:
    success, image = capture.read()

    backgroundImg[162:162+480, 55:55+640] = image
    #overlay the FIRST element within imgListMode on top of the background
    backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]

    #resize image because we will require a lot of computational power if we try to encode images that are larg
    smallImage = cv2.resize(image, (0,0), None, 0.25, 0.25)
    smallImage = cv2.cvtColor(smallImage, cv2.COLOR_BGR2RGB)

    #locating the face in the current frame
    faceCurrentFrame = face_recognition.face_locations(smallImage)
    #converting facial features in the current frame into encodings
    encodeCurrentFrame = face_recognition.face_encodings(smallImage, faceCurrentFrame)
```

```python
    encodeCurrentFrame = face_recognition.face_encodings(smallImage, faceCurrentFrame)


    if faceCurrentFrame:

        for encodeFace, faceLocation in zip(encodeCurrentFrame, faceCurrentFrame):
            matches = face_recognition.compare_faces(encodingsListKnown, encodeFace)
            faceDistance = face_recognition.face_distance(encodingsListKnown, encodeFace)
            matchIndex = np.argmin(faceDistance)

            if matches[matchIndex]:
                print("Registered Student Detected")
                print(studentIDs[matchIndex])

                y1, x2, y2, x1 = faceLocation
                y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
                bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
                backgroundImg = cvzone.cornerRect(backgroundImg, bbox, rt = 0)
                id = studentIDs[matchIndex]

                if counter == 0:
                    counter = 1
                    modeType = 1

        if counter != 0:

            if counter == 1:
                #Retrieving students information from the database
                studentInfo = db.reference(f'Students/{id}').get()
                print(studentInfo)
```

main.py > ...

```
87
88          if counter != 0:
89
90              if counter == 1:
91                  #Retrieving students information from the database
92                  studentInfo = db.reference(f'Students/{id}').get()
93                  print(studentInfo)
94
95                  #Retrieving student's image from the storage
96                  blob = bucket.get_blob(f'Images/{id}.png')
97                  array = np.frombuffer(blob.download_as_string(), np.uint8)
98                  studentImg = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
99
100                 #Update attendance record and data
101
102                 datetimeObject = datetime.strptime(studentInfo['last_attendance_taken'],
103                                                    "%Y-%m-%d %H:%M:%S")
104                 timeInSecsElapsed = (datetime.now() - datetimeObject).total_seconds()
105                 if timeInSecsElapsed > 30:
106
107                     ref = db.reference(f'Students/{id}')
108                     studentInfo['total_attendance'] += 1
109
110                     ref.child('total_attendance').set(studentInfo['total_attendance'])
111                     ref.child('last_attendance_taken').set(datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
112                 else:
113                     modeType = 3
114                     counter = 0
115                     backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]
116
117             if modeType != 3:
118
119                 if 10 < counter < 20:
120                     modeType = 2
121
122                 backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]
```

```
            backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]


            if counter < 10:

                #indent this properly
                #all will execute while modeType = 1

                cv2.putText(backgroundImg, str(studentInfo['total_attendance']),(861, 125),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),1)
                cv2.putText(backgroundImg, str(studentInfo['major']), (1006,550),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255),1)
                cv2.putText(backgroundImg, str(id),(1006,493),
                            cv2.FONT_HERSHEY_SIMPLEX,0.5,(255,255,255),1)
                cv2.putText(backgroundImg, str(studentInfo['grades']),(910,625),
                            cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
                cv2.putText(backgroundImg, str(studentInfo['year']),(1025,625),
                            cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)
                cv2.putText(backgroundImg, str(studentInfo['starting_year']),(1125,625),
                            cv2.FONT_HERSHEY_SIMPLEX,0.6,(100,100,100),1)

                #in order to center the name, calculation needs to be performed
                (width, height), _ = cv2.getTextSize(studentInfo['name'],cv2.FONT_HERSHEY_SIMPLEX,1,1)

                #we are using a double // because that gets rid of trailing float numbers
                center_offset = (414 - width)//2
                cv2.putText(backgroundImg, str(studentInfo['name']),(808 + center_offset,445),
                            cv2.FONT_HERSHEY_SIMPLEX,1,(50,50,50),1)

                backgroundImg[175:175 + 216, 909:909 + 216 ] = studentImg

        counter += 1
```

```
            counter += 1

            if counter > 20:
                counter = 0
                modeType = 0
                studentInfo = []
                studentImg = []
                backgroundImg[44:44+633, 808:808+414] = imgListMode[modeType]

    else:
        modeType =  0
        counter = 0

    cv2.imshow("Attendance System", backgroundImg)
    cv2.waitKey(1)
```

EncodeGenerator.py

EncodeGenerator.py ● AddingDataToDb.py main.py

EncodeGenerator.py > generateEncodings > encodingsList

```python
import cv2
import face_recognition
import pickle
import os
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage

cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred, {
    'databaseURL': "https://attendanceproject-31f7a-default-rtdb.firebaseio.com/",
    'storageBucket': "attendanceproject-31f7a.appspot.com"
})

#importing students' images from Images by using the same logic as earlier
folderPathImages = 'Images'
listPathImages = os.listdir(folderPathImages)
imgListImages = []


#Extracting the student ID from each picture
studentIDs = []

for path in listPathImages:
    imgListImages.append(cv2.imread(os.path.join(folderPathImages, path)))
    studentIDs.append(os.path.splitext(path)[0])

    fileName = f'{folderPathImages}/{path}'
    bucket = storage.bucket()
    blob = bucket.blob(fileName)
    blob.upload_from_filename(fileName)
```

```python
    blob.upload_from_filename(fileName)

def generateEncodings(images):
    encodingsList = []

    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodingsList.append(encode)

    return encodingsList

encodingsListKnown = generateEncodings(imgListImages)
encodingsListWithIDs = [encodingsListKnown, studentIDs]

encodingFile = open("EncodingsFile.p", "wb")
pickle.dump(encodingsListWithIDs, encodingFile)
encodingFile.close()

```

AddingDataToDb.py

```python
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db



cred = credentials.Certificate("serviceAccountKey.json")

firebase_admin.initialize_app(cred, {
    'databaseURL': "https://attendanceproject-31f7a-default-rtdb.firebaseio.com/"
})

ref = db.reference("Students")

data = {
    "TP011111":
    {
        "name" : "Lee Wen Han",
        "major" : "CS(AI)",
        "starting_year": 2021,
        "total_attendance": 8,
        "grades" : "A",
        "year" : 2,
        "last_attendance_taken" : "2024-01-23 16:30:30",
    },
    "TP012345":
    {
        "name" : "Shi Yu Qi",
        "major" : "PE",
        "starting_year": 2018,
        "total_attendance": 4,
        "grades" : "F",
        "year" : 1,
        "last_attendance_taken" : "2023-09-15 17:30:30",
    },
```

```python
        "total_attendance": 10,
        "grades" : "A+",
        "year" : 2,
        "last_attendance_taken" : "2024-01-22 01:30:30",
    },
    "TP088888":
    {
        "name" : "Karina",
        "major" : "Slayage",
        "starting_year": 2021,
        "total_attendance": 6,
        "grades" : "A+",
        "year" : 2,
        "last_attendance_taken" : "2024-01-22 01:30:30",
    },

}

#this is how to "unzip a dictionary in Python"
for key, value in data.items():
    #You are storing the value in the dictionary "children" for the key in the ref you created ("Students")
    ref.child(key).set(value)

```

By now, everything should be up and running! Congratulations and I hope you manage to submit on time to be eligible for the workshop certificate <3