**A Smart Attendance System**

**based on Facial Recognition**

**Presented by Team Liquid**

Chu Cheng Qing

Pong Jian Wei

Marcus Chai Jia Cher

**Table of Contents**

Table of Contents1

Introduction2

Problem Statement2

Purposes3

Project Scopes3

Impacts3

System Design4

Software Development6

Full code8

Explanation14

Direction of using16

Demonstration18

References23

**Facial Recognition System**

**Introduction**

The main purpose of this project is to build a facial recognition-based attendance monitoring system for educational institution to replace the ancient way of marking attendance. For example, recording manually by placing a mark or signature beside their name in a name list to indicate their presence, or even roll call by lecturer. This ancient system has a lot of ambiguity that caused inaccurate and inefficient of attendance taking because there might have some of the students tend to cheat for the attendance by asking friends to help them. None of it only, it will also cause time wasting. Many issues develop when the authority is unable to enforce the old system's regulations. The face recognition system will be the technology at work. The human face is one of the natural characteristics that can be used to individually identify a person. As a result, it is used to determine identify as the possibilities of a face deviating or being replicated is negligible. Face databases will be built as part of this research. Data should be fed into the recognizer algorithm. Faces will then be compared during the attendance taking session against the database in order to find out who you are. When an individual is identified, their attendance will be recorded automatically storing necessary data in a csv file.

**Problem Statement**

According to the previous attendance management system, the most serious issue is the accuracy of the data collected. This is because attendance may not be recorded individually by the original person; in other words, a specific person's attendance may be taken by a third party without the institution's knowledge, which violates the veracity of the data. For example, student A didn’t attend the class today, but the name of student A exist which means he or she was cheating for taking attendance. Supposing the institution establish an enforcement, it might need to waste a lot of human resource and time which in turn will not be practical at all. Thus, all the recorded attendance in the previous system is not reliable for analysis usage.

Next, the second problem of the previous system is to much time consuming. Assuming that the time it takes a pupil to sign his or her name on a 3-4 paged name list is approximately 1 minute. Only about 60 pupils can sign their attendance in one hour, which is plainly wasteful and time consuming.

Lastly, The third issue is the legitimate concerned party's access to the information. For example, most parents are quite concerned about tracking their child's actual whereabouts to ensure that their youngster actually attends college/school lessons. However, in the old system, parents had no access to such information. As a result, evolution of the prior system is required to increase efficiency, data quality, and enable access to information for those legitimate parties.

**Purposes**:

* To develop an advanced Smart facial recognition-based attendance system.
* To enhanced the speed of the attendance taking process for students and staff of educational institution.
* To make sure all the attendance taken is fairness.
* To help parents for tracking their child’s attendance.
* Provide a user-friendly interface for admins to access the attendance database and for non-admins (parents) to check their child’s attendance by mailing the attendance.

**Project scopes:**

* The targeted groups of the attendance monitoring system are the students and staff of an educational institution.
* The database of the attendance management system are not restricted as long as their facial photo is inside the folder.
* The facial recognition process can only be access for one person at a time
* The project must operate inside a Wi-Fi service area or via an Ethernet connection, as the system must constantly update the attendance system's database.

**Impact and significance of the system towards the targeted group**

Many of attendance management systems are inefficient and lack information sharing. As a result, the following restrictions will be overcome and further improved in this project:

* Students will be more conscientious about attending lessons. This is because a student's attendance can only be recorded by using their own face.
* Absences will be noted by the system if they are taken individually. This not only prepares the student to be punctual, but also to be responsible.
* They are punctual and avoid any immoral ethics, such as signing attendance for their buddies.
* The institution can save a lot of money because enforcement is now done through technology rather than by hand rather than human supervision will squander a lot of human resources for a minor process.
* The application can run on any device in any location as long as there is Wi-Fi coverage or an Ethernet connection, allowing the attendance system to be deployed in any planned location. For example, the gadget can be placed at the door of the classroom to take attendance, saving a lot of money because it eliminates all paperwork.
* The method is also very efficient because all calculations are automated. In short, the project was created to address existing flaws with the old attendance system.

**System Design**

The hardware and software components of the attendance tracking system are divided into two categories in the design. Prior to the software Before developing the design, the hardware must be finalised to offer a platform for the programme to run on. We must first install several libraries in order for the application to function properly. Our group are fully using Phyton to write the code for the software development. We also installed OpenCV and Dlib via Phyton.

**Installation and components needed for software development**

a) Python

- Ensure you have Python installed on your system. The code is designed for Python 3.x.

b) Phyton Libraries

- Install the required Python libraries

Explanation of libraries:

1. face\_recognition :

* used for face recognition tasks

1. opencv-python-headless :

* OpenCV (Open Source Computer Vision) is used for capturing video frames from a webcam. The "headless" version does not require a graphical user interface.

1. Tkinter :

* It is the standard GUI toolkit for Python, used in this code to create the graphical user interface for the face recognition attendance system.

1. Dlib :

* Install Dlib, a dependency for "face\_recognition"

**Hardware Requirements**

Laptop webcam:

* Ensure your computer has a functional webcam. Grant necessary permissions for the code to access the webcam.

Code Configuration

1. File Paths:

* Update file paths in the code to match your preferred locations or ensure that the specified directories exist.

1. Capture Delay:

* Adjust the capture\_delay parameter in the code based on your preferences.

**Additional Considerations:**

Dlib Compatibility:

* Ensure compatibility between the installed version of Dlib and the face\_recognition library.

Python Virtual Environment (Optional):

* Consider using a virtual environment to isolate project dependencies.

Security and Privacy:

* Consider the security and privacy implications of capturing and storing images. Inform users if their images are being used for recognition.

Tolerance Parameter:

* Adjust the tolerance parameter in face comparison based on your recognition requirements.

Testing:

* Thoroughly test the system in a controlled

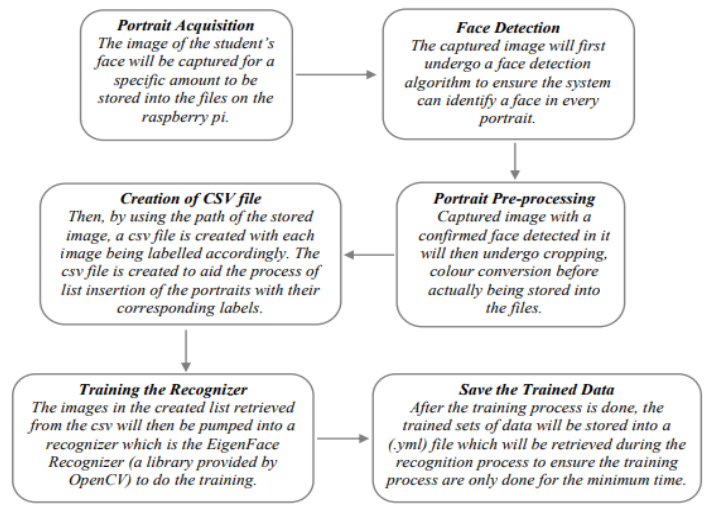
**Software Development**

There are two major system flows in the software development section as shown below:

* The creation of the face database
* The process of attendance taking

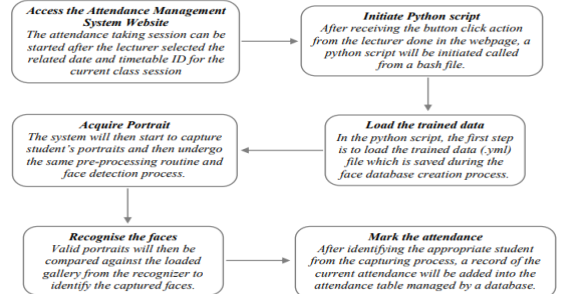
Both processes mentioned above are essential because they made up the backbone of the attendance management system.

1. **The creation of the face database**



The face database is a critical stage that must be completed before any further process can begin. This is since the face database serves as a comparison factor during the recognition process, which will be explored more below. Because each student will have more than one portrait stored, a csv file is prepared to facilitate the process of image labelling. Labels are used to distinguish them in order to group their portraits under the name of the same individual. Following that, the photos will be fed into a recognizer to train it. Because the training procedure has become increasingly time consuming as the face database has grown in size, training is only performed when there is a need.

1. **The process of attendance taking**



**Conclusion**

In short, the facial recognition-based attendance system is a great tool for schools. It replaces the old way of taking attendance, like marking lists or passing around name lists, with a smarter way. This system uses face recognition, making it hard to cheat attendance. It solves problems like wrong data and wasting time on manual attendance. Plus, it lets parents check their kid's attendance, creating trust between schools and families.

This new system does more than just use cool technology. It makes students more responsible and helps schools save time and money. It's easy to use, works in different places, and does all the attendance work automatically. This system isn't just about technology; it's about making attendance better for everyone involved in education.

**Full coding:**

import face\_recognition as fr

import cv2

import tkinter as tk

import os

import csv

from datetime import datetime

class FaceRecognitionApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Face Recognition Attendance System")

self.cap = cv2.VideoCapture(0)

if not self.cap.isOpened():

print("Error: Could not open webcam.")

self.root.destroy()

return

self.canvas = tk.Canvas(root)

self.canvas.pack()

self.result\_label = tk.Label(root, text="")

self.result\_label.pack()

self.attendance\_text = tk.Text(root, height=10, width=50)

self.attendance\_text.pack()

self.start\_button = tk.Button(root, text="Start/Stop Recognition", command=self.toggle\_recognition)

self.start\_button.pack()

self.clear\_records\_button = tk.Button(root, text="Clear Records", command=self.clear\_records)

self.clear\_records\_button.pack()

self.register\_face\_button = tk.Button(root, text="Register New Face", command=self.register\_new\_face)

self.register\_face\_button.pack()

self.n = ""

self.known\_faces = []

self.known\_names = []

self.capture\_path = "C:\Photo\Registered\_Images" #Can edit your path here

os.makedirs(self.capture\_path, exist\_ok=True)

self.attendance\_file\_path = "attendance.csv"

self.recognition\_active = False

self.current\_user = None

if not os.path.exists(self.attendance\_file\_path):

with open(self.attendance\_file\_path, 'w', newline='') as csvfile:

csv\_writer = csv.writer(csvfile)

csv\_writer.writerow(['Name', 'Timestamp'])

self.capture\_delay = 5 # Set the delay time in seconds

self.last\_capture\_time = datetime.now()

def toggle\_recognition(self):

if not self.recognition\_active:

self.recognition\_active = True

self.start\_recognition()

else:

self.recognition\_active = False

self.result\_label.config(text="Recognition Stopped")

def start\_recognition(self):

while self.recognition\_active:

ret, frame = self.cap.read()

if not ret:

print("Error: Failed to grab frame.")

break

cv2.imshow("Webcam", frame)

key = cv2.waitKey(1)

if key == ord('q'):

self.recognition\_active = False

elif key == ord('s') and (datetime.now() - self.last\_capture\_time).seconds >= self.capture\_delay:

self.n = os.path.join(self.capture\_path, "captured\_image.jpg")

cv2.imwrite(self.n, frame)

print(f"Image saved to {self.n}")

self.update\_attendance()

self.last\_capture\_time = datetime.now()

cv2.destroyAllWindows()

def update\_attendance(self):

if self.n:

self.load\_known\_faces()

unknown\_image = fr.load\_image\_file(self.n)

try:

unknown\_face\_encoding = fr.face\_encodings(unknown\_image)[0]

except IndexError:

print("I wasn't able to locate any faces in the image. Check the image file. Aborting...")

return

results = fr.compare\_faces(self.known\_faces, unknown\_face\_encoding, tolerance=0.45)

if any(results):

index = results.index(True)

recognized\_name = self.known\_names[index]

self.current\_user = recognized\_name

result\_text = f"Recognized: {recognized\_name}"

self.result\_label.config(text=result\_text)

self.mark\_attendance()

self.display\_attendance()

def mark\_attendance(self):

if self.current\_user:

timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

with open(self.attendance\_file\_path, 'a', newline='') as csvfile:

csv\_writer = csv.writer(csvfile)

csv\_writer.writerow([self.current\_user, timestamp])

print(f"Attendance marked for {self.current\_user} at {timestamp}")

def load\_known\_faces(self):

self.known\_faces = []

self.known\_names = []

for file in os.listdir(self.capture\_path):

if file.endswith("\_registered\_image.jpg"):

image\_path = os.path.join(self.capture\_path, file)

known\_image = fr.load\_image\_file(image\_path)

try:

known\_face\_encoding = fr.face\_encodings(known\_image)[0]

self.known\_faces.append(known\_face\_encoding)

self.known\_names.append(os.path.splitext(file)[0])

except IndexError:

print(f"I wasn't able to locate any faces in {file}. Check the image file. Aborting...")

return

def clear\_records(self):

self.attendance\_text.delete(1.0, tk.END)

print("Attendance records cleared.")

self.clear\_historical\_records() # Clear historical records

def display\_attendance(self):

with open(self.attendance\_file\_path, 'r', newline='') as csvfile:

csv\_reader = csv.reader(csvfile)

attendance\_records = [row for row in csv\_reader]

self.attendance\_text.delete(1.0, tk.END)

self.attendance\_text.insert(tk.END, "Attendance Records:\n")

for record in attendance\_records[1:]:

self.attendance\_text.insert(tk.END, f"{record[0]} - {record[1]}\n")

def clear\_historical\_records(self):

with open(self.attendance\_file\_path, 'w', newline='') as csvfile:

csv\_writer = csv.writer(csvfile)

csv\_writer.writerow(['Name', 'Timestamp'])

print("Historical attendance records cleared.")

def register\_new\_face(self):

name = input("Enter the name of the person: ")

self.n = os.path.join(self.capture\_path, f"{name}\_registered\_image.jpg")

print(f"Please capture an image for {name} registration.")

input("Press Enter when ready...")

ret, frame = self.cap.read()

cv2.imwrite(self.n, frame)

print(f"Image for {name} registration saved to {self.n}")

def quit\_app(self):

self.cap.release()

cv2.destroyAllWindows()

self.root.destroy()

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = FaceRecognitionApp(root)

root.mainloop()

**Explanation**:

This code is a Python script that implements a simple face recognition attendance system using the face\_recognition library, OpenCV (cv2), and Tkinter for the graphical user interface. The script captures images from a webcam, recognizes faces, and marks attendance in a CSV file.

* Importing Libraries:
  + ‘face\_recognition’: A face recognition library.
  + ‘cv2’: OpenCV library for computer vision tasks.
  + ‘tkinter’: Tkinter library for creating a GUI.
  + ‘os’: Operating system interface to interact with the file system.
  + ‘csv’: Module for reading and writing CSV files.
  + ‘datetime’: Module to work with dates and times.
* Class Definition - ‘FaceRecognitionApp’:
  + The class initializes the GUI components, webcam, file paths, and other necessary variables.
  + It defines methods to toggle face recognition, start recognition, update attendance, mark attendance, load known faces, clear records, display attendance, clear historical records, register new faces, and quit the application.
* ‘\_\_init\_\_ ’method:
  + Initializes the Tkinter GUI with buttons, labels, and text boxes.
  + Sets up the webcam using OpenCV.
  + Defines paths for capturing images, storing attendance records, and sets up initial variables.
* ‘toggle\_recognition’ method:
  + Toggles the face recognition system on/off based on the current state.
* ‘start\_recognition’ method:
  + Captures video frames from the webcam, displays them, and waits for user input to stop recognition (q key).
  + Saves a captured image every few seconds and calls update\_attendance method.
* ‘update\_attendance ‘method:
  + Loads known faces, compares the captured face with known faces using the face\_recognition library, and updates the GUI accordingly.
  + Calls mark\_attendance and display\_attendance methods.
  + mark\_attendance method:
  + Marks the attendance in a CSV file with the current user's name and timestamp.
* ‘load\_known\_faces’ method:
  + Loads known faces from registered images and stores their encoding.
* ‘clear\_records’ method:
  + Clears the attendance records displayed in the GUI and calls clear\_historical\_records to clear the historical records in the CSV file.
* ‘display\_attendance’ method:
  + Reads the attendance records from the CSV file and displays them in the GUI.
* ‘clear\_historical\_records’ method:
  + Clears the historical attendance records in the CSV file.
* ‘register\_new\_face’ method:
  + Allows the user to register a new face by capturing an image from the webcam and saving it with a given name.
* ‘capture\_registration\_images’ method:
  + Display the video and capture frames from the webcam to allow user to adjust their faces angle for encoding process.
* ‘quit\_app’ method:
  + Releases the webcam, closes OpenCV windows, and destroys the Tkinter GUI.
* if \_\_name\_\_ == "\_\_main\_\_": block:
  + Creates an instance of the FaceRecognitionApp class and starts the Tkinter main loop.

**How to use it?**

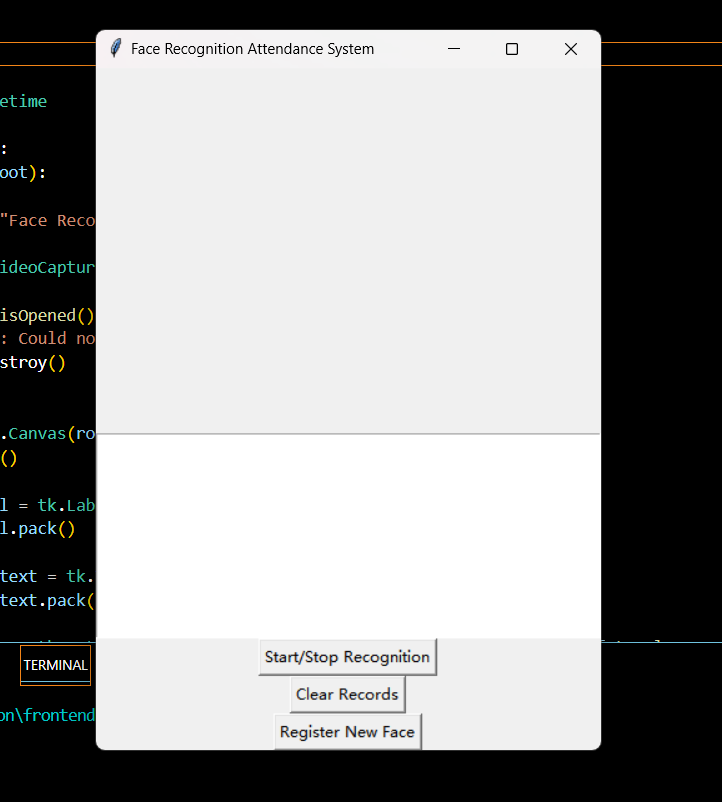
1. Install Required Libraries:
   1. Make sure you have the necessary Python libraries installed. You can install them using the following command: “pip install face\_recognition opencv-python tkinter”
2. Run the Script:
   1. Save the provided code in a file with a .py extension (e.g., face\_recognition\_app.py). Then, run the script using a Python interpreter: “python face\_recognition\_app.py”

1. GUI Interface:
   1. A Tkinter window will appear with buttons and a text box.
   2. Click on the "Start/Stop Recognition" button to start the face recognition system.
   3. You can use the "Clear Records" button to clear displayed attendance records.
2. Register a New Face:
   1. Click on the "Register New Face" button.
   2. Enter the name of the person when prompted and press Enter.
   3. Follow the instructions to capture an image for face registration.
3. Recognition:
   1. While recognition is active, the webcam feed will be displayed in a window.
   2. When a recognized face is detected, the name will be displayed in the Tkinter window, and attendance will be marked.
4. Quit the Application:
   1. Press the "q" key to stop face recognition.
   2. You can close the Tkinter window or use the "Quit" button to exit the application.
5. View Attendance:
   1. The attendance records will be displayed in the Tkinter window & the csv file.
   2. You can use the "Clear Records" button to clear displayed attendance records.

To make sure that the code can function properly, please ensure that your system has a working webcam. Additionally, the face recognition system relies on good lighting and clear visibility of faces for accurate results. The system saves registered faces in the "Registered\_Images" folder by default. Adjust the paths and settings in the script if needed.

**Demonstration**:

When you start running the codes, a GUI interface will pop up like this:



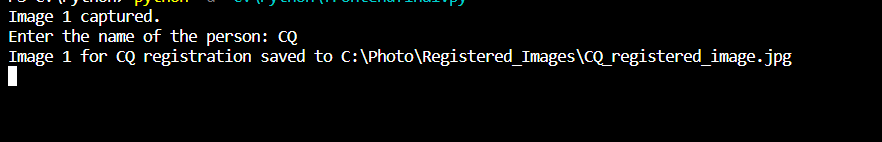
To start with the recognition process, click “register new face” button:



A webcam will show out as the above.

Press “s” key from your keyboard if you satisfied with the picture shown.

Key in your name at the prompt, and save your image.



If this show up means the image has been successfully registered.

Press “q” key to quit and move to the recognition process.

To start taking attendance, press the “start recognition” button at the GUI interface.

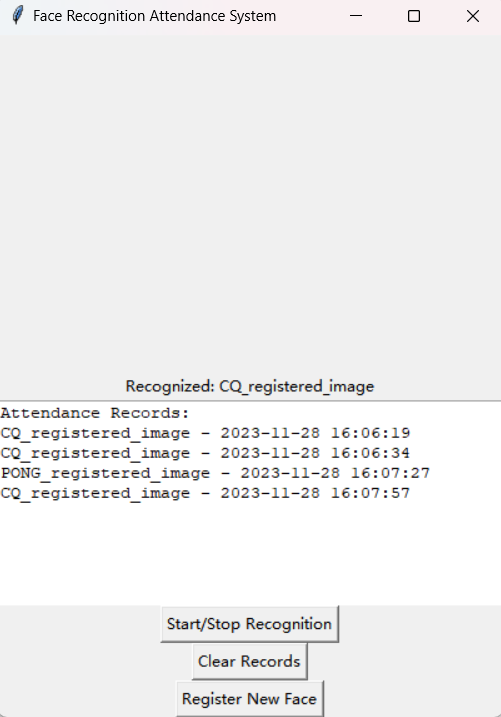


A webcam will show up, press “s” key when you are satisfied with the picture shown.

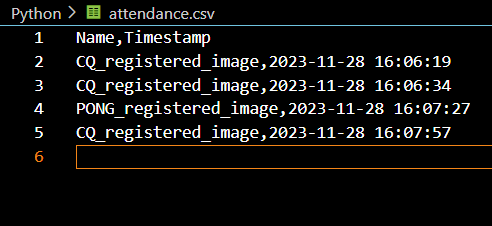
Screenshot 2023-11-28 155819

Attendance will be marked if the pictures saved are match with registered pictures in your storage.

Finally, attendance will be shown out at the GUI interface.

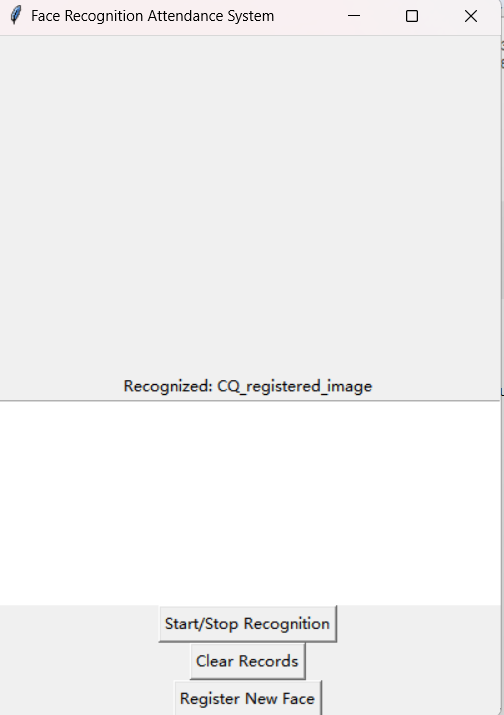


You can also find your attendance at the csv file.



To quit, press “q” key.

To clear historical records, press “clear records” button at the GUI interface.



All attendance record will be cleared then.



Conclusion

In short, the facial recognition-based attendance system is a great tool for schools. It replaces the old way of taking attendance, like marking lists or passing around namelists, with a smarter way. This system uses face recognition, making it hard to cheat attendance. It solves problems like wrong data and wasting time on manual attendance. Plus, it lets parents check their kid's attendance, creating trust between schools and families.

This new system does more than just use cool technology. It makes students more responsible and helps schools save time and money. It's easy to use, works in different places, and does all the attendance work automatically. This system isn't just about technology; it's about making attendance better for everyone involved in education.

**References**

OpenCV. (n.d.). OpenCV Documentation. Retrieved from <https://docs.opencv.org/>

Ageitgey, A. (n.d.). face\_recognition. GitHub Repository. Retrieved from <https://github.com/ageitgey/face_recognition>

Python Software Foundation. (n.d.). Tkinter Documentation. Retrieved from https://docs.python.org/3/library/tkinter.html