***Image Processing – Attendance System***

***Project Description***

*The Attendance Sheet Management System aims to streamline and digitize the process of recording attendance for organizations, classes ,or events .*

*This project involves the development of an efficient and user-friendly digital platform/program that allows for the easy and accurate tracking of attendance.*

***Dependencies:***

* *OpenCV, or Open Source Computer Vision Library, is a powerful open-source computer vision and machine learning software library primarily developed in C++, but with interfaces for Python and other languages as well. It's widely used for tasks such as image and video analysis, object detection and recognition, face detection, and more.*
* *The* ***cv2*** *module in Python is the interface to the OpenCV library. It provides functions and classes to perform various image processing and computer vision tasks. cv2 is often used alongside other Python libraries like NumPy for efficient array manipulation and Matplotlib for visualization.*
* *The* ***os*** *module in Python provides a way to interact with the operating system. It allows you to perform various tasks such as navigating file systems, manipulating files and directories, accessing environment variables, and executing system commands.*
* ***DeepFace*** *is a deep learning-based face recognition system developed by Facebook AI Research (FAIR). It's designed to perform accurate face verification and recognition tasks using deep neural networks.*
* ***RetinaFace*** *is a state-of-the-art face detection and alignment algorithm that accurately locates and aligns faces in images with high precision. Developed by researchers from the Chinese University of Hong Kong and Alibaba Group, RetinaFace employs a single deep neural network architecture to simultaneously perform face detection and facial landmark localization.*
* ***Matplotlib.pyplot*** *is a Python library used for creating static, animated, and interactive visualizations in Python. It is part of the Matplotlib library, which is one of the most widely used plotting libraries in the Python ecosystem*.
* ***Pandas*** *is a powerful open-source Python library used for data manipulation and analysis. It provides easy-to-use data structures and functions for working with structured data, primarily in the form of tabular data (like spreadsheets or databases).*
* ***Tkinter*** *is a Python library used for creating graphical user interfaces (GUIs). It's a standard library included with most Python installations, making it a convenient and accessible choice for building desktop applications.*
* ***shutil*** *is a built-in Python module that provides a collection of high-level functions for working with files and directories. It simplifies common file management tasks, making your code more concise and readable compared to using lower-level* ***os*** *module functions directly*

***Used Functions:***

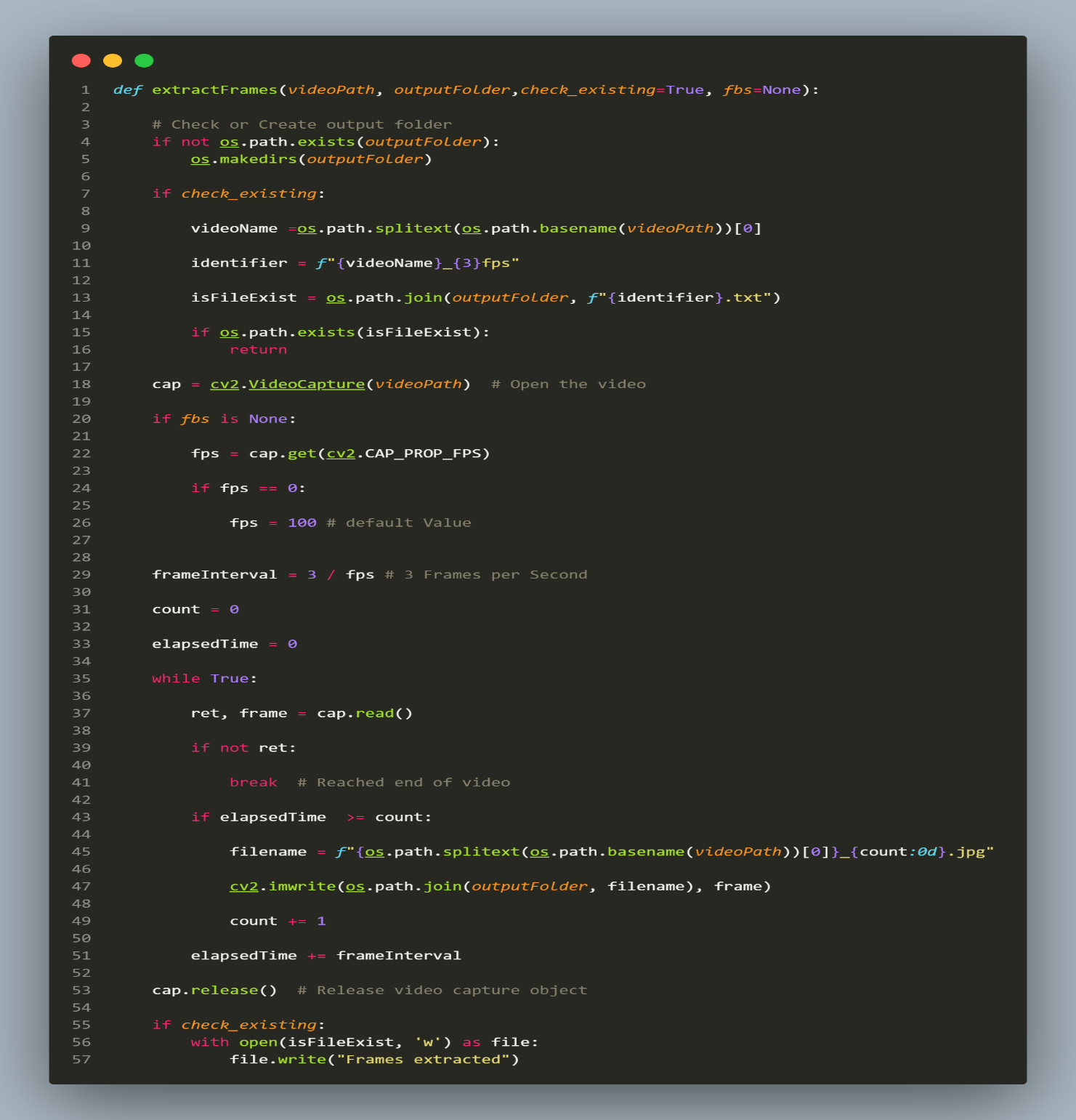
***A screen shot of a computer program

Description automatically generated***

* *getVideoPaths(folderPath)*
* *This function retrieves paths of video files (with specific extensions) from a given folder.*
* *Inputs:*

*folderPath (string): Path to the folder containing video files.*

* *Outputs:*
* *videos (list of strings): List of paths to video files.*
* *Process:*
  + - *Iterates through files in the specified folder.*
    - *Checks if the file ends with one of the specified video extensions (.mp4, .avi, .mkv, .wmv).*
    - *Appends the path of the video file to the videos list if it matches the criteria.*



* *extractFrames(videoPath, outputFolder, check\_existing=True, fbs=None)*

*This function takes in a video file path (videoPath), an output folder path (outputFolder), and two optional parameters: check\_existing and fbs.*

* *Inputs:*
* *videoPath (string): Path to the input video file.*
* *outputFolder (string): Path to the output folder where frames will be saved.*
* *check\_existing (bool, optional): If True, checks if frames have already been extracted from the video. Default is True.*
* *fbs (float, optional): Frames per second of the video. If None, it is automatically determined from the video. Default is None.*
  + *Process:*
    - *Check or Create Output Folder:*
* *Checks if the output folder exists. If not, creates the folder.*
  + - *Check Existing Frames (Optional):*
* *If check\_existing is True, it checks if frames have already been extracted from the video by looking for a corresponding text file in the output folder. If found, it exits the function without extracting frames.*
  + - *Open Video File:*
* *Opens the video file using OpenCV (cv2.VideoCapture()).*
  + - *Determine Frames per Second (FPS):*
* *If fbs is None, it determines the frames per second (FPS) of the video using cv2.VideoCapture.get(cv2.CAP\_PROP\_FPS). If FPS is zero, it sets it to a default value of 100.*
  + - *Calculate Frame Interval:*
* *Calculates the frame interval based on the desired frame rate (3 frames per second in this case).*
  + - *Extract Frames:*
* *Iterates through the video frames using a while loop.*
* *At each iteration, it reads a frame using cv2.VideoCapture.read().*
* *Checks if it's time to extract a frame based on the elapsed time and count.*
* *If it's time to extract a frame, it saves the frame as an image in the output folder.*
  + - *Release Resources:*
* *Releases the video capture object using cv2.VideoCapture.release().*
  + - *Write File Indicating Frames Extracted (Optional):*
* *If check\_existing is True, it writes a text file in the output folder to indicate that frames have been extracted.*

A screenshot of a computer program

Description automatically generated

*This part of the script is responsible for extracting frames from video files located in a specified folder and saving them into corresponding output folders*

* *Process:*
* *Specify Videos Folder:*
* *Defines the path to the folder containing input video files (videosFolder).*
* *Retrieve Video Paths:*
* *Calls the getVideoPaths() function to retrieve paths of video files from the specified folder (videosFolder).*
* *Iterate Over Video Paths:*
* *Checks if there are any video files found (if videoPaths:).*
* *If video files are found, iterates over each video file path (for videoPath in videoPaths:).*
* *Define Output Folder Path:*
* *Constructs the output folder path for each video file by removing the extension and appending it to the "./dataset/" directory.*
* *The os.path.splitext() function splits the file path into root and extension, and [0] selects the root part (without extension).*
* *Assigns this output folder path to the variable outputFolder.*
* *Extract Frames:*
* *Calls the extractFrames() function with the video file path (videoPath) and output folder path (outputFolder) as arguments.*
* *The extractFrames() function extracts frames from the video file and saves them into the specified output folder.*
* *Handle No Video Files:*
* *If no video files are found in the specified folder, it prints a message indicating that no video files were found (print("No video files found in the folder.")).*

A screenshot of a computer program

Description automatically generated

* *get\_photos(folderPath)*

*This function takes in a folder path (folderPath) and returns a list of*

*paths to image files (photos) found in the folder.*

* *Inputs:*
* *folderPath (string): Path to the folder containing image files.*
* *Outputs:*
* *photos (list of strings): List of paths to image files.*
* *Process:*
* *Iterate Through Files:*
* *Iterates through files in the specified folder using os.listdir(folderPath).*
* *Check File Extensions:*
* *Checks if the file name ends with one of the specified image extensions (".jpg", ".jpeg", ".png", ".bmp", ".gif").*
* *Converts the filename to lowercase before checking to ensure case-insensitive matching.*
* *Construct Photo Paths:*
* *Constructs the full path to the image file by joining the folder path and the filename using os.path.join(folderPath, filename).*
* *Append to List:*
* *Appends the path of the image file to the photos list if it matches the criteria.*
* *Return List:*
* *Returns the list of paths to image files (photos).*

A screenshot of a computer screen

Description automatically generated

*This section of the script aims to process photos located in a specified folder, detect faces in the photos using RetinaFace, and attempt to recognize the faces using DeepFace.*

*Process:*

* *Specify Photos Folder:*
* *Defines the path to the folder containing photo files (photosFolder).*
* *Retrieve Photo Paths:*
* *Calls the get\_photos() function to retrieve paths of photo files from the specified folder (photosFolder).*
* *Process Each Photo:*
* *If photo files are found, iterates over each photo file path (for photoPath in photoPaths:).*
* *Face Detection and Recognition:*
* *For each photo, extracts faces using RetinaFace (RetinaFace.extract\_faces(photoPath)).*
* *Prints the number of detected faces in the photo.*
* *For each detected face:*
* *Displays the face using matplotlib.pyplot.imshow().*
* *Saves the displayed face as "detectedFace.jpg" using matplotlib.pyplot.savefig().*
* *Attempts to recognize the face using DeepFace (DeepFace.find()).*
* *If the face is recognized:*
* *Extracts the recognized face number from the DeepFace result.*
* *Appends the recognized face and its corresponding number to lists (recognizedFaces and recognizedNumbers).*
* *Displays the recognized face with its assigned number as the title.*
* *If the face is not recognized:*
* *Displays a message indicating that the face was not recognized.*
* *Appends the unrecognized face to a list (unknownFaces).*
* *Displays the image using matplotlib.pyplot.show(), waits for 4 seconds (matplotlib.pyplot.pause(4)), and then closes the plot (matplotlib.pyplot.close()).*
* *Handle No Photo Files:*
* *If no photo files are found in the specified folder, it prints a message indicating that no photo files were found.*

A screen shot of a computer program

Description automatically generated

*This section of the script involves cleaning up temporary files and saving data to an Excel file.*

*Process:*

* *Clean Up Temporary Files:*

*Removes the temporary file "detectedFace.jpg" using os.remove('./detectedFace.jpg').*

* *Save Data to Excel:*

*Constructs a pandas DataFrame named AttendanceData containing the recognized face numbers (recognizedNumbers).*

*Saves the DataFrame to an Excel file named "Attendance.xlsx" using AttendanceData.to\_excel('Attendance.xlsx', index=False).*

*index=False parameter is used to exclude row indices from being saved to the Excel file.*