UDC: 004.93'1:572.54.087
Professional paper

FACE RECOGNITION – ATTENDANCE SYSTEM

Florim Idrizi[1*], Jahi Ibrahimi[1], Agon Memeti[1], Florinda Imeri[1], Shpend Ismaili[1]

[1]Department of Informatics, Faculty of Natural Sciences and Mathematics, University of Tetova
**Corresponding author e-mail: florim.idrizi@unite.edu.mk*

Abstract

The face represents a unique part of the human body as well as a biometric identifier. Using facial features, we can achieve create and use applications or systems based on facial recognition. This project involves the construction of such a system which will use facial features to mark the presence, entry time, and finally the creation of an extracted document in the Excel application. The project was developed by researching and using libraries, models as well as Machine Learning based algorithms that are necessary for face detection, training, and recognition during system operation. Python programming language is used to create the system, and CSS is used to design interfaces. The system is built in the form that professors have access to the registration of new students, as well as the creation of relevant courses and obtaining the final attendance report after receiving notes from the system.

Keywords: Open CV, Biometric, Machine Learning, Python

1 Introduction

Face recognition technology approaches are generally classified as feature-based approaches and holistic (comprehensive) approaches. In holistic approaches, recognition is based on global facial features, while in feature-based approaches, recognition is performed using local facial features.

The face recognition system is the perfect way to solve and avoid the problems that the traditional system displays, the presence in this system will be obtained in real-time and fast, the data is safe at any time so there is no risk of loss by the lecturer, the data is always accurate as it is not accessible to students and is automatically generated by the system. It is a biometric technique that works in cases where the image of a certain person matches the images of the face which are stored in a database. The built-in system uses face recognition technology to automate the procedure for obtaining the presence of students or even employees without their direct involvement [1-5].

The main purpose is to assist lecturers, and administrative staff, to have efficient and automatic organization and management of reports related to the presence or absence of students, minimizing the errors and shortcomings of the manual (traditional) system.

2 Defining Problem & Project Goal

The traditional technique of marking the participation uses forms such as the signature of the participants and the checking of the respective identification cards, these two methods in themselves contain many problems starting from the concern of the progress of the learning process, but also the distraction of students during exam sessions. The main problem with the traditional attendance system is that it is subject to manipulation and is prone to human error when entering data [6-7].

To avoid these problems, a system based on face recognition has been proposed, which efficiently replaces the traditional system, performing processes in an automated form, which saves time, increases efficiency, and provides flexibility and security throughout the process. of taking notes for the presence within the educational institution.

This system aims to automate the traditional system for receiving presence, in which case the organization will be able to maintain its data at any time and digitally. With the digitalization of the

system, we enable better visualization of the data, in which case we have the opportunity to create final reports about student participation [8-10].

Given the needs and challenges mentioned above, the goals of the project are: Enable the enrollment of students and other members; Accurately detect the faces of students who are previously stored in a database; To enable separation between the respective subjects and depending on the choice to maintain the presence in those subjects; To take attendance of identified students and create a document with their data.

In general, the main scope of the system is the registration of participation based on facial recognition techniques as well as the preservation of the obtained results and providing the opportunity to access them at any time.

The basic principles of determining the scope of the system are:

- Enable enrollment of new students.
- Provide divisions for different subjects.
- Save data and display reports.

Guidelines defining the scope:

- The ability of the system to receive data from the registration process.
- Ability to transfer final reports to a database or .xslx file.
- Ability to build specific reports depending on the respective subjects.

The basic requirements of the present marking system, whether traditional or built-in, are the same, but the system through face recognition performs the same processes, more efficiently, in a short time, and automatically.

The main requirements of the system are: registration of students, their recognition by the system, division based on subjects and study programs, a note of presence, and generation of the final report after each teaching session held.

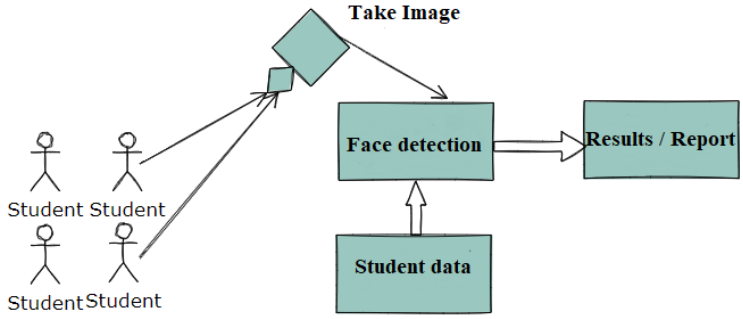

Figure 1. System operation scheme

After analyzing the traditional system, collecting requests, and analyzing and reviewing them, the system was built based on the main pillars of operation which are presented in fig 1.

The process of operating the system based on the figure starts with taking the image, then adjusting the system in such a way as to detect and control the face with those that are already known, and finally generating final reports for processes performed.

3 System Design

In the system design phase, I am determined to use UML diagrams, which provide us with a clear overview of the system even before we start the coding phase, as well as facilitate the coding process by defining functions, roles, classes, and interactions between users.

Using UML techniques, we built:

- Use Case diagram
- Activity diagram
- Status diagram
- Class diagram
- Component diagram
- Interaction diagram

3.1 *Use Case Diagram:*

The benefit of use case diagrams is mostly based on communication between the request team and the user group. A use case specification document should cover the following areas:

- Actors - participating in and interacting in this use case
- Preconditions - must be met for the use case to work
- Unconditional - defines the various states in which the system is expected to be after it is executed. The Use Case diagram lists the basic events that will occur when the system is executed. It includes all the primary actions that the system must perform.

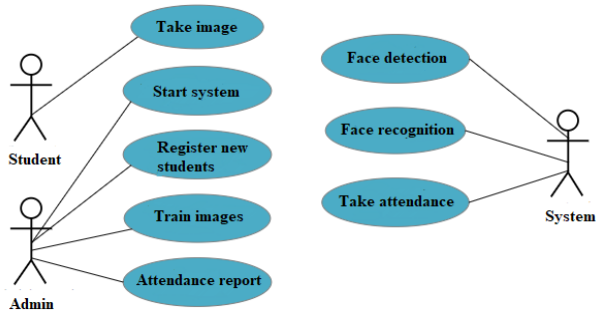

Figure 2. Use case diagram

In our system are defined three actors, as seen in Figure 3, as actors are: student, administrator, and system, each of them is presented as related to activities or functions which are planned to be executed in the system and which are the responsibility of relevant actors during the operation of the system.

3.2 *Activity Diagram:*

The activity diagram is a presentation in the form of diagrams keeping a hierarchy of activities. These diagrams are important to understand the way the system works as well as the flow of activities previously stated in the use case diagram. Activities are states of action that automatically switch to another state after the action is completed, but always starting with the "circles" that symbolize the beginning and the end as well as the "arrows" which display the transition from one activity to the next. For our system, we have presented three activity diagrams to create an overview of how the system functions flow.

Figure 3 shows the activity diagram, which shows the flow of events in the process when we are dealing with new students, who must first do the face registration in which case the system stores them, then

training, and then the presence with these students can be noted. And finally, we have the generation of the report which is the result of the preliminary step respectively the activity of marking the presence.

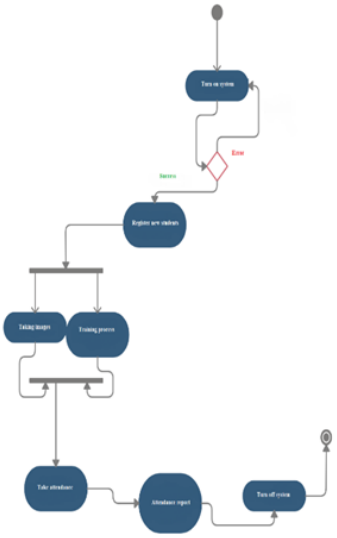

Figure 3. Activity diagram 1

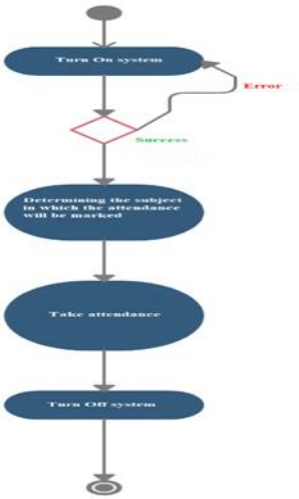

Figure 4. Activity diagram 2

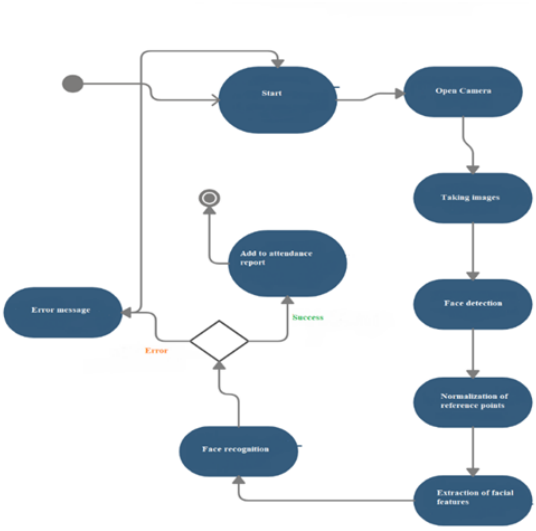

Figure 5. Activity diagram 3

Figure 4 shows a simple diagram of the activity in which the operation of the whole system is presented in only 4 basic activities that the system performs.

Figure 5 illustrates the activity diagram for how the system works in the facial features registration section, the processes which are performed before the presence marking process takes place. After going through the defined steps face image recognition can be performed successfully and, in this case, the assigned student is added to the presence preservation report otherwise, the error message is displayed on the screen, and the reason why it is not executed with order process success.

3.3 *State Diagram:*

The state diagram helps us to better understand the simple functions as well as the complex ones that the system offers. By using the state diagram, we identify the dynamic behavior of the system as a whole or even of the subsystems. Exactly with the help of the state diagram, we determine the different states of the participants during the operation with the system.

Figure 6 shows the situation diagram specifying all the necessary actions for the implementation of defined functions, as well as the states of the participants in the system.

3.4 *Class Diagram:*

The class diagram describes the structure of the system by presenting classes, attributes, operations, and relationships between objects. The class diagram is the main block of object-oriented modeling and is mainly used for general conceptual modeling of application structure and for detailed model translation modeling in coding. The class diagram can also be used for data modeling.

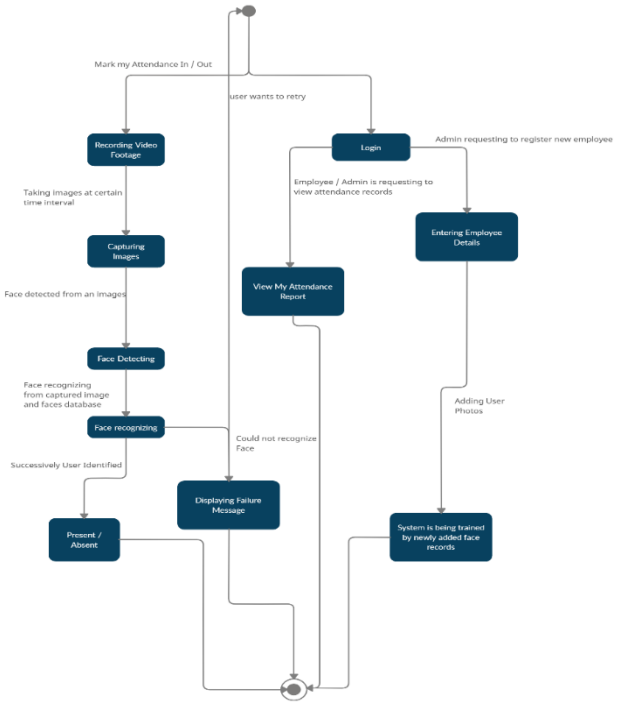

Figure 6. State diagram

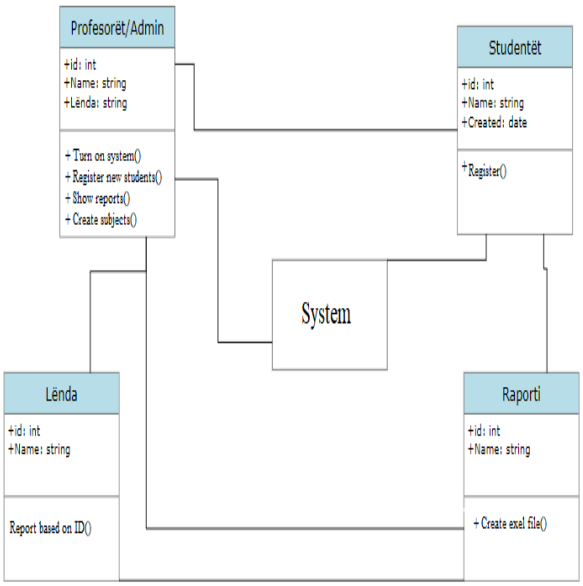

Figure 7. Class diagram

In figure 7 we have presented the class diagram in which we have managed to define four classes as well as the general one which is the system. Each class contains the name that characterizes it, each class has attributes as well as operations or functions. Most classes are interconnected with the system because there are functions that must be executed first by the system and then split between defined classes.

3.5 *Component Diagram:*

The primary difference between a component diagram and other diagrams is that component diagrams represent the implementation perspective of a system. Therefore, the components in a component diagram reflect by grouping different designs of system elements, such as system classes. Firstly, the component must be replaceable, and secondly, the component must provide interfaces to enable other elements to interact and provide the services provided by the component.

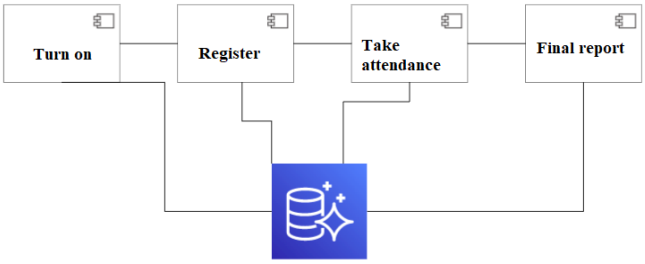

Figure 8. Component diagram

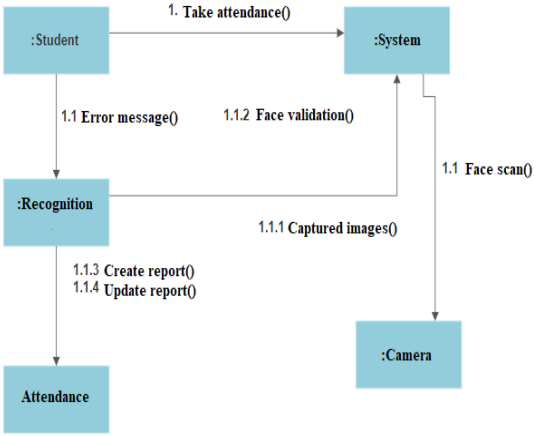

Figure 9. Interaction diagram

Figure 8 shows the diagram which contains four components: connection, registration, presence marking, and the final report, where the diagram shows how the components interact with each other and their interaction with the database.

3.6 *Interaction Diagram:*

The interaction diagram, otherwise known as the communication diagram, enables the illustration of connections and interactions between software objects. Otherwise, the interaction diagram represents the use case behavior describing how the set of objects interacts to complete the task. The types of interaction diagrams are sequential diagrams and cooperative diagrams. Figure 9 shows the interaction diagram, in which I tried to show the communication between the components of the system together with the action sequences which start from the first sequence respectively from the "presence note". The communication diagram provides a clear illustration of the sequence of processes in sequential steps which define the functioning of the system.

4 System Development

The application is built using Visual Studio Code software as well as the Python programming language. Special emphasis during the development of the system has been the design of the interface always seems to be "user friendly" and as easy to use, so that it can be easily used even by people who do not have a computer background.

The structure of the project from the software side which includes the necessary coding parts which are arranged in VS code, as well as other assets which are used in the development of the system are shown in the following picture:

Such ranking of directories is made based on my selection, making it easier to manage coding components as well as easier to access them.

The main project file is "attendance.py" in which the connection is made between all other coding parts, as well as the execution of the basic functions of the system. The execution of the system is done exactly with this file by executing the command "py attendance.py"

In "show_attendance.py" is the part which enables the lecturer to publish the reports of previously received presences. "TakeImage.py" accepts inputs or images during the registration process of new students. "TrainImage.py" is one of the most important parts of the system with which after taking the images the system deals with the training so that in the future it can do the detection without making mistakes in recognizing the faces that it has previously registered.

The execution of the system is done by executing the command "py attendance.py" in the terminal, and this command then initiates the start of the system and the screen should appear the part that looks like displayed in figure 11.

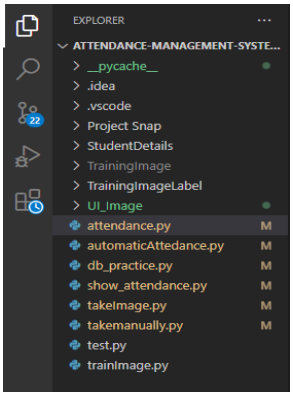

Figure 10. Project Files

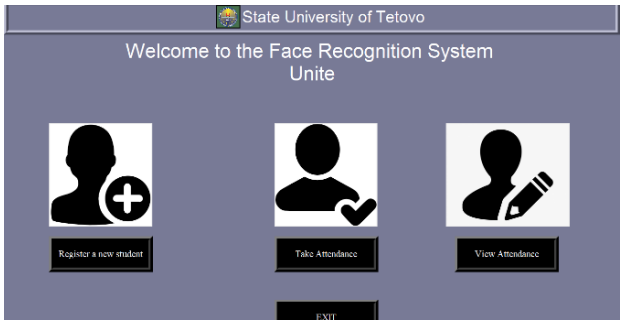

Figure 11. System UI

The opening interface of the system is built from the ribbons that name it, and below is divided into 3 parts which are the main functions of the system, the first part enables the registration of new students in case you click on the button, the second part enables the start of the present acquisition process, while the third part provides reports from previously received presences. As well as at the end we have the exit button that enables system shutdown. The coding part which has enabled the construction of this interface will be displayed in figure 12. Whereas by clicking the "Register new student" button, a new interface has been designed which will accept inputs that have previously been defined as necessary for the specification of new students, where each student in addition to the name will also contain an Id which is unique to all. The design of this part is shown below in figure 13.

```
logo = Image.open("UI_Image/ut-logo.png")
logo = logo.resize((50, 47), Image.ANTIALIAS)
logo1 = ImageTk.PhotoImage(logo)
titl = tk.Label(window, bg="#787a96", relief=RIDGE, bd=10, font=("arial", 35))
titl.pack(fill=X)
l1 = tk.Label(window, image=logo1, bg="black",)
l1.place(x=470, y=10)

titl = tk.Label(
    window, text="State University of Tetovo", bg="#787a96", fg="white", font=("arial", 27),
)
titl.place(x=525, y=12)

a = tk.Label(
    window,
    text="Welcome to the Face Recognition System \nUnite",
    bg="#787a96",
    fg="white",
    bd=10,
    font=("arial", 33),
)
```

Figure 12. Code Extract

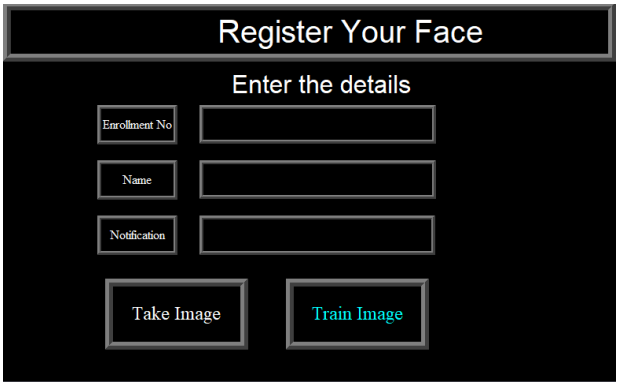

Figure 13. Register new students UI

After filling in the required data in the above fields then it is necessary to click the Take Image button in which case the system opens the camera and then captures images and finally gives the message if the images were taken successfully. After taking the images the same process happens with the Train Image button which in the background trains the captured images but the users do not interact with it, except at the end when the system notifies if the training was successful, or if it failed. The coding part for building this interface is displayed below in figure 14.

```
    lbl2 = tk.Label(
        ImageUI,
        text="Name",
        width=10,
        height=2,
        bg="black",
        fg="white",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 12),
    )
```

Figure 14. Input field design

```
    lbl1 = tk.Label(
        ImageUI,
        text="Enrollment No",
        width=10,
        height=2,
        bg="black",
        fg="white",
        bd=5,
        relief=RIDGE,
        font=("times new roman", 12),
    )
    lbl1.place(x=120, y=130)
    txt1 = tk.Entry(
        ImageUI,
        width=17,
        bd=5,
        validate="key",
        bg="black",
        fg="white",
        relief=RIDGE,
        font=("times", 25, "bold"),
    )
```

Figure 15. Input field design

The function that is called on the take image button enables the camera to be opened, and is defined as follows:

```python
def TakeImage(l1, l2, haarcasecade_path, trainimage_path, message, err_screen,text_to_speech
    if (l1 == "") and (l2==""):
        t='Please Enter the your Enrollment Number and Name.'
        text_to_speech(t)
    elif l1=='':
        t='Please Enter the your Enrollment Number.'
        text_to_speech(t)
    elif l2 == "":
        t='Please Enter the your Name.'
        text_to_speech(t)
    else:
        try:
            cam = cv2.VideoCapture(0)
            detector = cv2.CascadeClassifier(haarcasecade_path)
            Enrollment = l1
            Name = l2
            sampleNum = 0
            directory = Enrollment + "_" + Name
            path = os.path.join(trainimage_path, directory)
            os.mkdir(path)
```

Figure 16. Function that opens camera

```python
# Train Image
def TrainImage(haarcasecade_path, trainimage_path, trainimagelabel_path, message,text_to_speech):
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    detector = cv2.CascadeClassifier(haarcasecade_path)
    faces, Id = getImagesAndLables(trainimage_path)
    recognizer.train(faces, np.array(Id))
    recognizer.save(trainimagelabel_path)
    res = "Image Trained successfully"  # +",".join(str(f) for f in Id)
    message.configure(text=res)
    text_to_speech(res)

```

Figure 17. Train image function

After capturing the images, the system training is required, a function which is defined in the file "trainImage.py" and which is called in the Train Image function in the interface that was presented above.

Take attendance" is the function that enables the lecturer to start directly with the process of taking attendance but first by specifying the subject in question and then has the opportunity to start the process, but also to see the reports from past lectures, the interface of this process as well as the coding part and function are displayed below.

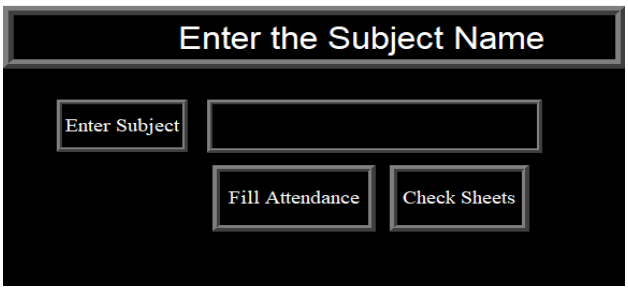

Figure 18. Take attendance UI

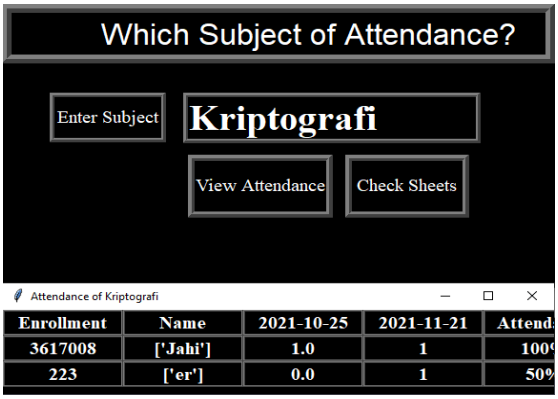

Figure 19. Showing attendance UI

The "View Attendance" function enables lecturers to view reports from previously received presences, in tables that are generated within the system, but also in excel files which are created automatically. The following figures show the graphical representation of the function, but also the coding part with which the construction of this function has been achieved.

The coding part includes two functions that are related to the buttons shown in the figure, by clicking the "view attendance" button displays the table which is shown in the figure, while by clicking the "check sheets" button then the system directs us to excel documents in which the registered presence is also stored.

```python
def subjectchoose(text_to_speech):
    def calculate_attendance():
        Subject = tx.get()
        if Subject=="":
            t='Please enter the subject name.'
            text_to_speech(t)
        os.chdir(
            f"C:\\Users\\Jahi\\Desktop\\Face\\Attendance\\{Subject}"
        )
        filenames = glob(
            f"C:\\Users\\Jahi\\Desktop\\Face\\Attendance\\{Subject}\\{Subject}*.csv"
        )
        df = [pd.read_csv(f) for f in filenames]
        newdf = df[0]
        for i in range(1, len(df)):
            newdf = newdf.merge(df[i], how="outer")
        newdf.fillna(0, inplace=True)
        newdf["Attendance"] = 0
        for i in range(len(newdf)):
            newdf["Attendance"].iloc[i] = str(int(round(newdf.iloc[i, 2:-1].mean() * 100)))+'%'
            #newdf.sort_values(by=['Enrollment'],inplace=True)
        newdf.to_csv("attendance.csv", index=False)
```

Figure 20. View attendance coding part

The development of the coding part throughout the process is accompanied by comments within the code which can make the process of changes that may occur in the future even easier. The whole process has been tried to make the coding understandable and easy to reuse, and for this reason, such a structure has been built so that the code is divided into components, as it offers flexibility in cases when we want to use only certain parts in the future if the system were to be further developed.

5 Testing

During the development of the system from the first stages, unit testing is constantly practiced, which means testing small pieces of code, but which is important to detect any failures or errors that may occur in the system.

After successfully passing the unit tests, the system finally went through the Integration testing, after the process of attaching the components, in which case the functionality of all components of the system was tested, and throughout this testing process, errors were identified, and then it has been intervened and improved and so the system is now fully functional by meeting those requirements which I have identified since the analysis stage.

6 System Maintenance

Given the fact that building new systems is always costly and time consuming. As a result, more efforts are being made to upgrade existing systems, so that work on managing applications is increasing.

Software maintenance and upgrading is the process of dealing with newly discovered problems or new requirements that need to be added to the system and that can sometimes take longer than starting to build software. About 2/3 of software engineering work is maintenance, but this statistic can be confusing. A small part of it is about correcting mistakes. Most of the maintenance is the reconstruction of the system to do new functions, which in many ways can be considered as new work.

In the software maintenance process, the term re-engineering or re-building of various pieces of software is used. The re-engineering paradigm is a cyclical model. This means that each of the activities presented as part of the model can be reviewed. For any particular cycle, the process can be completed after each of these activities.

Starting from the use of the iterative model we have used in the system we can say that we have an advantage in implementing the re-engineering process, as we can go back and add or modify certain parts at certain stages of development.

Development of applications based on software components is also a great help for software developers, as it enables the creation of new components but without affecting the current system, i.e., development as independent, in this case, avoids the potential risk that the system could have.

Conclusions

In this paper, we have elaborated, researched, and developed a system for registration of presence, through which lecturers or professors can record student participation in electronic form. Using the software system saves time and effort, especially for large groups of students. The automated presence registration system is built to reduce the shortcomings and problems that appear in the traditional (manual) system.

Thus, the purpose of the system and this paper is to build and clarify the system which enables the recognition of the facial expressions of the enrolled students, and the same preserves it and creates reports efficiently and with a high accuracy which the system achieves using image training. After extensive

research into the Python programming language and libraries needed to build such systems, I began developing the system by performing all the steps required by software engineering for the system to be fully functional and effective.

The completion of the project resulted in a software product that can automate the registration process of participation in the university facility. The system requires minimal human intervention, even only during the start-up and initial enrollment of students and then the process is completely automatic, saving the time consumed by the traditional system. In addition, the development of graphical interfaces makes the system very easy to use and understandable for all users who need to interact with it.

References

[1]. V. a. R. Tokas, "Fast Face Recognition Using Eigen Faces," IJRITCC, vol. 2, no. 11, pp. 3615-3618, November 2014.

[2]. N. J. M. M. K. a. H. A. Mayank Agarwal, "Face Recognition Using Eigenface approach," IRCSE, vol. 2, no. 4, pp. 1793-8201, August 2010.

[3]. Book Face Recognition Based Smart Attendance System (ISBN-978-3-659-68627-6) January 2018.

[4]. 10. Vinay Hermath, Ashwini Mayakar, "Face Recognition Using Eigen Faces and," IACSIT, vol. 2, no. 4, pp. 1793-8201, August 2010

[5]. Lirie K., Florim I.; Detection, identification and tracking of objects during the motion, IEEE 3rdInternational Symposium Multidisciplinary Studies and Innovative Technologies, Ankara, Turkey

[6]. Radhika C.Damale, Bageshree.V.Pathak., "Face Recognition Based Attendance System Using Machine Learning Algorithms." Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS 2018) IEEE Xplore Compliant Part Number: CFP18K74-ART; ISBN:978-1-5386- 2842-3. IEEE 2018.

[7]. Roy Shilkrot, David Millan Escriva, "Mastering OpenCV 4: A Comprehensive Guide to Building Computer Vision and Image Processing Applications", December 27, 2018

[8]. Stan Z. Li, Anil K. Jain, "Handbook of Face Recognition", 2005.

[9]. Handbook of Face Recognition (The Second Edition) Editors: Stan Z. Li & Anil K. Jain, October 2004

[10]. B. Cyganek, Object Detection and Recognition in Digital Images: Theory and Practice Boguslaw Cyganek. New Jersey, USA: John Wiley and Sons, Ltd, 2013, p. 540