A Project report on

AUTO STUDENT ATTENDANCE USING IMAGE PROSESSING - A MACHINE LEARNING APPROACH

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by

B YAMUNA	**(17B91B0506)**
G SAI KIRAN	**(18B95B0504)**
S B KOUSHIK VARMA	**(17B91B0548)**
Y SRI SAI CHAND	**(18B95B0512)**

Under the Guidance of

Dr V CHANDRA SEKHAR
Associate Professor & Head

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SRKR ENGINEERING COLLEGE (A)
ChinnaAmiram, Bhimavaram, West Godavari Dt.,

[2020– 2021]

SRKR ENGINEERING COLLEGE (A)

BONAFIDE CERTIFICATE

This is to certify that the project work entitled " **AUTO STUDENT ATTENDANCE USING IMAGE PROSESSING - A MACHINE LEARNING APPROACH "** is the bonafide work of **B. YAMUNA, G. SAI KIRAN, S. BALAJI KOUSHIK VARMA, Y. SRI SAI CHAND** who carried out the project work under my supervision.

This is in partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering.

HEAD OF THE DEPARTMENT	**SUPERVISOR**
(Dr. V. Chandra Sekhar)	Dr. V. Chandra Sekhar
	Associate Professor & HOD

SELF DECLARATION

I hereby declare that the project work entitled "**AUTO STUDENT ATTENDANCE USING IMAGE PROSESSING - A MACHINE LEARNING APPROACH**", is a genuine work carried out by me in B.Tech, (Computer Science and Engineering) at SRKR Engineering College(A), Bhimavaram and has not been submitted either in part or full for the award of any other degree or diploma in any other institute or University.

S. Balaji Koushik Varma

17B91B0548

TABLE OF CONTENTS

ABSTRACT

Human face detection has become a major area of interest in the current research because there is no deterministic algorithm to find the face(s) in a given image. Furthermore, the algorithms that are there are very specific to the type of photos you will take as input and detect faces. The problem is detecting the faces in the colour group photo selected for the class. Our approach is a combination of heuristic and well-known algorithms. To detect faces, we can put a number of simple rejection blocks in series, until we get the faces. The deeper the rejection block, and more specifically it can be trained to eliminate non-faces. Various methods such as neural networks, template matching, maximum rejection, linear hunter features, and subjective destinations have been tried. Finally, a combination of skin colour segmentation, morphological (erosion) processes, and intrinsic orientations were used.

A The face detection algorithm is very specific to the type of problem and cannot be guaranteed to work unless it is applied, and results are obtained. We have taken a multi-algorithm approach to face detection, which is actually a series of simple rejection blocks. In the final algorithm design, many different schemes were tried. The face detection system area of the project with face recognition is image processing.

It's tricky because even though faces have commonalities, they can vary greatly in terms of age, skin colour, and facial expressions. The problem is further complicated by different lighting conditions, image characteristics and geometric shapes, as well as the possibility of partial blockage and disguise. Thus, the perfect face detector will be able to detect the presence of any face under any set of lighting conditions, on any background. There are a large number of applications of this face detection project, this project can be extended so that it can detect the different parts of the face which are in different directions and shapes.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Machine learning (ML) is a subset of artificial intelligence (AI). While the field of AI itself covers a lot of areas, it basically boils down to simulating human intelligence in machines. ML includes programming algorithms that can learn from themselves and even make their own predictions. ML algorithms learn from data to solve problems that are too complex to be solved using traditional programming.

It is difficult to understand how facial recognition technology works, and interpretation of the quality will exceed the standards of this article. For our purposes, we will be looking at the four cross-cutting problems that a machine needs to solve for facial recognition. They are face detection, face alignment, feature extraction, face recognition and face verification.

FACE DETECTION:

The device must first locate the face in the photo or video. At the moment, most cameras have a built-in face detection function. Face detection is also what Snapchat, Facebook, and other social media platforms use to allow users to add effects to the photos and videos they take with their apps.

FACE ALIGNMENT:

A face normalization algorithm is required to be consistent with the faces in the database. One way to achieve this is to use several general facial features. For example, the lower part of the chin, the top of the nose, the outer side of the eyes, various points around the eyes and mouth, etc. The next step is to train ML an algorithm to find these points on any face and rotate the face toward the centre.

FEATURE EXTRACTION:

Measure and extract various features from the face which will allow the algorithm to match the face with other faces in its database. However, it was not initially clear which features should be measured and extracted until the researchers discovered that the best way was to allow the ML algorithm to decide which measurements to collect for itself. This process is known as embedding.

FACE RECOGNITION:

Using the unique measurements of each face, a final ML algorithm will match the face measurements against known faces in the database. Any face in the database that is closer to the measurements of the face in question will be returned as the match.

FACE VERIFICATION:

Face verification compares the unique characteristics of one face with another. The ML algorithm will return a confidence value to assess whether the faces are identical or not.

There are many attendance systems that exist in today's world. Some popular systems are.

MANUAL RECORDING:

This is the most widely used method of attendance tracking even before the age of technology. It is a method of recording the incoming and outgoing time of an employee in the log. Since it's a manual recording, this can lead to employees asking for favours to get out and come back often.

TIMESHEETS:

These sheets keep track of the beginning and end of the tasks given to the employee. Thus, the employer will get a detailed breakdown of the tasks that the employee has accomplished throughout the total work. This information is used to determine grade and manage time. These schedules are widely used on the Internet. Some companies offer web-based services for these schedules. Therefore, employee working hours can be effectively tracked. Shifts also provide employee productivity in a company. This will also greatly reduce administrative work.

BIOMETRICS:

Provides employee verification and identification to the employer in the organization. The template is captured and saved either on a smart card or a database for the purpose of verification of the registered user (employee). As a result, users are identified from biometric characteristics alone without the use of smart cards, usernames, ID, etc. All records in the database are compared to the template and the closest matching result is returned. Therefore, the closest match authentication is within the permissible limit.

ACCESS CARDS:

These cards are used to provide the user with access after swiping the card. It stores data on magnetic tape, also called magnetic stripe card.

PROXIMITY CARD READERS:

They are contactless integrated circuit devices used for secure access. Proximity cards / proxy cards range from 0 to 3 inches, and allow the user to leave the card in the wallet, purse, etc. Therefore, this technique is used in applications like ID cards, key cards, etc. And so, based on their resources and no. any method of employee attendance tracking system can be specified by the company.

CHAPTER 2

PROBLEM STATEMENT

In today’s world, the procedure of monitoring running instances has continually been a complex task. Over the years, numerous answers were evolved to file the presence of employees. The maximum famous of those conventional guide attendance monitoring structures are the timecard, the perforation clock, and the time sheets. The essential risks of those conventional structures are as follows.

THERE IS A RISK OF HUMAN ERROR

When you rely upon punched playing cards or time sheets, there's usually a hazard of human error. Employees can also additionally report their hours of labour incorrectly in a time sheet or the records can be incorrectly entered into your payroll software. These mistakes regularly arise withinside the place of job while you operate guide time monitoring solutions. Your employees branch can spend a number of times correcting those mistakes. Unrecorded mistakes can result in wrong worker bills and different payroll issues.

MANUAL ENTRY IS VERY TIME CONSUMING

A lot of time is spent on attendance time sheets. Your personnel can also additionally queue to go into or go out and your human assets branch can also additionally spend time handling time-on-time price tag issues. This can also additionally consist of changing the scorecard, correcting time access mistakes and mistakes, and different troubles that arise whilst the usage of guide time recording systems.

TOO MUCH PAPERWORK

Managing this developing batch of records calls for a gadget that guarantees that facts is archived, found, and retrieved fast and efficiently. If you're nonetheless responding on paper primarily based totally at the attendance gadget, you're much more likely to pick attendance incorrectly. So, to keep away from those mistakes, we're looking to put in force an Auto Student Attendance System using image processing by Machine Learning algorithms. So that we are able to lessen the mistakes which might be maximum normally take place in current systems.

In this the facial reputation performs a key role. To make this gadget higher than current ones, we should minimise the mistakes and time.

CHAPTER 3

LITERATURE SURVEY

Nirmalya kar, Mrinal Kanti [1] used the eigen face approach for face recognition which was introduced by kirby and sirovich in 1988 at Brown University. The method works for analysing faces, images and computing eigenfaces. These are faces composed of eigen vectors. The comparison of eigen face is used to identify the presence of face and its identity.

Clyde Gomes, Sagar Chanchal [2] developed a system using raspberry pi camera module. When raspberry pi captures an image. This system makes use of DNN to detect the faces of students by PCA and LDA algorithms. The accuracy of 86% with database was created by extracting frames and store them in database.

Shrija Madhu, Anusha Adapa [3] proposed an automated attendance management system. This model incorporates a camera and an algorithm to detect a face from the image. The camera captures the image and sends it to the server when faces are recognized.

Mohammad Abdul Muqueet [4] developed an attendance management system that uses the faces of students as the feed input. To make it available to every platform he chooses raspberry pi module. More efficient automated student attendance system using recognition that leverages on cloud computing infrastructure called FACE CUBE. This FACE CUBE takes the attendance by using IP camera mounted in front of the classroom.

Edy Winarno, Henry Februariyanti [5] developed an attendance system using camera based on face recognition and it has been developed by several research to produce a system that is accurate and able to store a large scale for image database. The facial attendance system using camera is very sage and accurate for detecting users. This is based on image intensity, a method that works based on video sequences and a 3D information and infra – red image.

Serign Bah, Famg Ming [6] developed a computer application that is capable of detecting, tracking, identifying, or verifying human faces from an image or a video captured using a digital camera. They used a new method using Local Binary Pattern (LBP) algorithm combined with advanced image processing. To improve the LBP codes, thus improve the accuracy of the overall face recognition system.

CHAPTER 4

SOFTWARE REQUIREMENTS AND SPECIFICATIONS

4.1 PURPOSE

The main purpose of this project is to implement a student recognition system, which can be used to give attendance to the students. So that we can save time during the classes for taking the attendance. We try to develop a system with high accuracy, so that there would be no margin for errors.

4.2 SCOPE

The scope of this project is to solve the problem of time complexity and provide a system that can give results with high accuracy.

4.3 OBJECTIVE

The main objective of this project is to provide attendance to the students with face recognition system using image processing techniques with high accuracy and to store that attendance in excel sheet including the date and time.

4.4 EXISTING SYSTEMS

- **MANUAL RECORDING**

  This is the most widely used method of attendance tracking even before the age of technology. It is a method of recording the incoming and outgoing time of an employee in the log. Since it's a manual recording, this can lead to employees asking for favours to get out and come back often.

  DRAWBACKS:

  - ➢ Lot of paperwork.
  - ➢ Wastage of time.
- **TIMESHEETS**

  These sheets keep track of the beginning and end of the tasks given to the employee. Thus, the employer will get a detailed breakdown of the tasks that the employee has accomplished throughout the total work. This information is used to determine grade and manage time. These schedules are widely used on the Internet. Some companies offer web-based services for these schedules. Therefore, employee working hours can be effectively tracked. Shifts also provide employee productivity in a company. This will also greatly reduce administrative work.

DRAWBACKS:

- ➢ Difficult to track intangible tasks.
- ➢ Human error is likely.

- **BIOMETRICS**

  Provides employee verification and identification to the employer in the organization. The template is captured and saved either on a smart card or a database for the purpose of verification of the registered user (employee). As a result, users are identified from biometric characteristics alone without the use of smart cards, usernames, ID, etc. All records in the database are compared to the template and the closest matching result is returned. Therefore, the closest match authentication is within the allowable limit.

  DRAWBACKS:

  - ➢ Physical traits are not changeable.
  - ➢ Error rate in some cases is so high.

4.5 PROPOSED SYSTEM

In this system we propose an auto student attendance system by image processing techniques to provide attendance to the students. In this we use machine learning approach so that there would less chance for errors. This system will automatically recognize the face of the students and process that with the given training dataset if that image matches with dataset, it will post the attendance in the excel sheet including with date and time. So that there would be no loss of date and is secured.

ADVANTAGES:

- **AVOID PROXY**
  - ➢ It can avoid proxy by finding the difference between the intime and out time of the user.
- **NO HUMAN ERROR**
  - ➢ It is purely a machine-driven approach.
- **HIGH PERFORMANCE**
  - ➢ On an average this system will respond to the end user within 5 seconds.

4.6 REQUIREMENTS

4.6.1 FUNCTIONAL REQUIREMENTS

- Transforming the data if required.
- Maintain the records safely.
- Admin can update the data user details at any point of time.
- The system will automatically recognise the faces and provides attendance to users.
- The admin can process & store the data in excel sheets.

4.6.2 NON – FUNCTIONAL REQUIREMENTS

Non-functional requirements are the limitations or constraints under which the system should provide its services to users.

PERFORMANCE

This system is developed in the high-level language using python. On an average this system will respond to the end user within 5 seconds of the request.

RELIABILITY

This system is more reliable. The reliability cab be up to 1 year.

USABILITY

This system is used by college offices. They can have access to all cameras. So that, maintenance would be easy.

2.6.3 PSEUDO REQUIREMENTS

- **SOFTWARE REQUIREMENTS**
  - ➢ **PYTHON**

    Python is a deciphered, significant level broadly used programming language.
  - ➢ **VISUAL STUDIO**

    Microsoft Visual Studio is an integrated development environment. It is used to develop computer programs.
- **HARDWARE REQUIREMENTS**

➢	Processor	: Intel core i3 or above
➢	RAM	: 2GB or above
➢	Hard Disk	: 16GB or above
➢	OS	: Windows 10

  - ➢ High Internet Connectivity.

CHAPTER 5

METHODOLOGY

In this project we are designing a facial recognition system using face-recognition library. To make that work we use dlib. So, we discuss each step at a time.

STEP 1: FINDING ALL THE FACES

- When the camera automatically selects faces, it makes sure that all faces are in focus before the photo is taken.
- We will use a method called Directed Gradient Histogram - HOG for short.
- To find faces in a photo, we will make our photos black and white. Then we will look at each pixel at once, we must look at the images that surround it directly.
- Our goal is to find out how dark the current pixel is compared to the pixel directly surrounding it. If you repeat this process, each pixel will be replaced with an arrow. These arrows are called gradients.
- There is a good reason to replace pixels with gradients. If we analyse the pixels directly, dark images and light images of the same person will have completely different pixel values.
- To find the faces in this HOG image, all we have to do is find a part of our image that looks like a known HOG pattern extracted from a bunch of other training faces.

STEP 2: POSING AND PROJECTING FACES

- We use the Landmark face estimation algorithm.
- The idea is to come up with 68 points (called landmarks) on each face. Then we will train an algorithm to find these specific points.
- We will use basic image transformations such as rotation and scale that preserve parallel lines (called affine transformations).
- Now no matter how the face is rotated, we can centre the eyes and mouth at roughly the same position in the image. This will make our next step appear accurate.

STEP 3: ENCODING FACES

- In this we will train the Convolutional Neural Network (CNN). We will train it to generate 128 measurements for each face.

- After repeating “one triple upload step” millions of times for millions of photos of different people, the neural network reliably generates 128 measurements for each person. This is called embedding.
- Training a CNN to produce weddings requires a lot of data and computational power to get good accuracy. So, we are using Open Face which has already done that and deployed several trained networks that we can use directly.
- Therefore, we need to run our faces through their pre-trained grid so that the grid generates roughly the same numbers when we look at two different images of the same person.

STEP 4: FINDING THE PERSON’S NAME FROM THE ENCODING

- We will use the SVM linear classifier algorithm in this step.
- All we need to do is train a classifier that can take measurements from a new test image and identify the person closest to it.
- The result of the classifier is the name of the person.

CHAPTER 6

SYSTEM DESIGN

DESIGN GOALS:

Design goals originate from the non-functional requirements specified during requirements elicitation and from technical and management goals specified by the project.

- **Response Time**: User request is acknowledged quickly within 0.3 sec after the request is issued.
- **Throughput**: Ability to accomplish the designed tasks within given period of time.
- **Memory**: System requires 1GB RAM and 120GB hard disk space.
- **Robustness**: System can survive from invalid user input.
- **Reliability**: System will model the specified and observed behaviour without any difference.
- **Availability**: System is available for all time (24X7) to accomplish any task provided if there is an internet connection and server is available.
- **Fault tolerance**: System can operate in erroneous conditions unless they are more.
- **Security**: System can withstand malicious attacks such as viruses, malwares, etc.
- **Extensibility**: It is easy to add the functionality or new class to our system.
- **Modifiability**: It is easy to change the existing functionality as per requirement in our system.
- **Portability**: It is easy to port the system to different platforms i.e., it is easy to change the frontend and the backend platforms.
- **Readability**: It is easy to understand the coding of the system. It is user friendly system.
- **Utility**: System supports the work of the user very well.
- **Usability**: User with basic knowledge can use the system.

6.1. SYSTEM ARCHITECTURE

A System architecture is a conceptual model that defines the structure, behaviour, and further views of the system. A description of architecture is a formal description and representation of a system, structured in a way that supports thinking about the structures and behaviours of the system. A system architecture can consist of developed system components and subsystems, which will work together to implement the system as a whole.

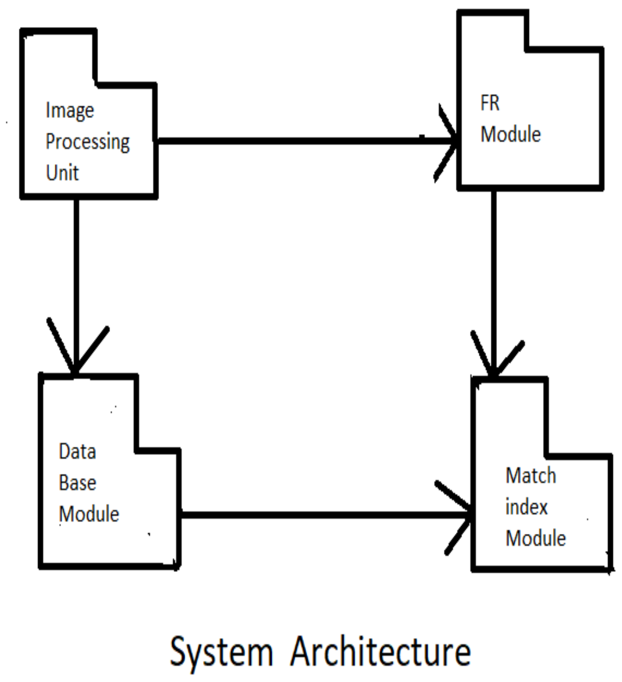


Fig 6.1. System Architecture

6.1.1. COMPONENT DIAGRAM

A component diagram is a special type of UML diagram. Hence, from this point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system.

A component diagram that breaks down the actual system under development into different high levels of functionality. Each component is responsible for one clear goal within the entire system and interacts with other essential components on a need-to-know basis.

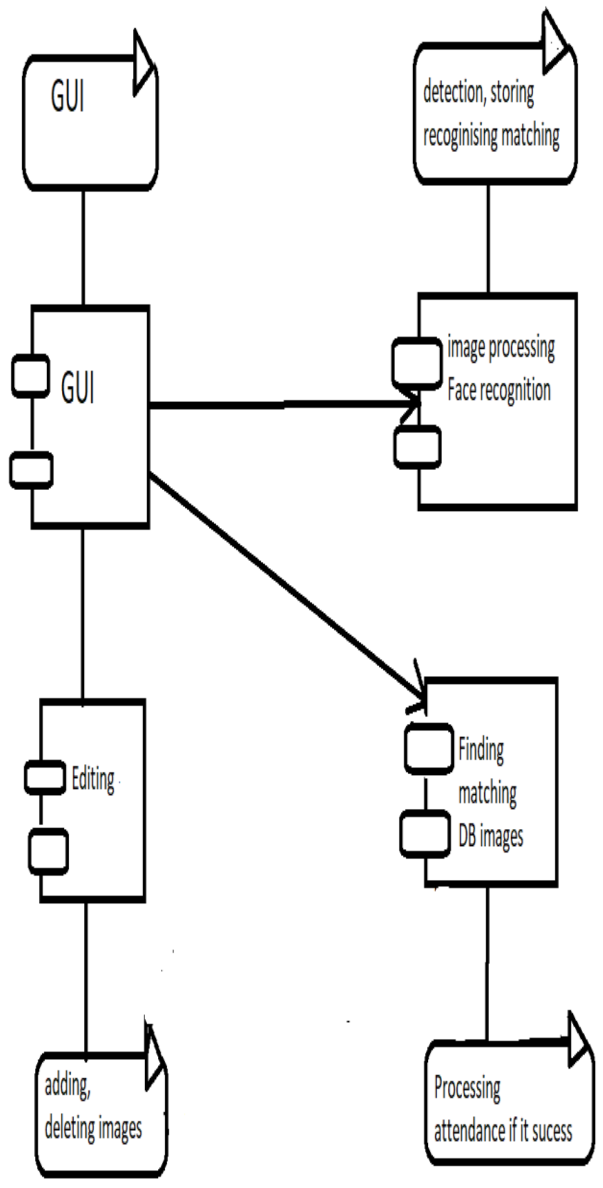


Fig 6.2. Component Diagram

6.1.2. DEPLOYMENT DIAGRAM

Deployment diagram is a schema type UML that describes the system implementation architecture, including the nodes such as hardware or software execution environments and the middleware that connects them. Deployment diagrams are typically used to visualize a system's

Hardware and software.

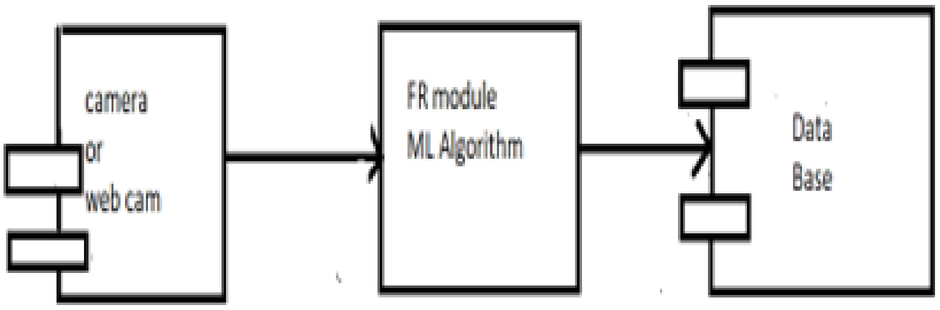


Fig 6.3. Deployment Diagram

6.1.3. SUB SYSTEM DECOMPOSITION

System decomposition begins with decomposing the system into coherent and well-defined subsystems. The subsystems then decompose into coherent and well-defined components.

Subsystem decomposition is usually done basing on the functional requirement by keeping functionally related objects. This is the main part of the system design. There are three subsystems identified for this current system.

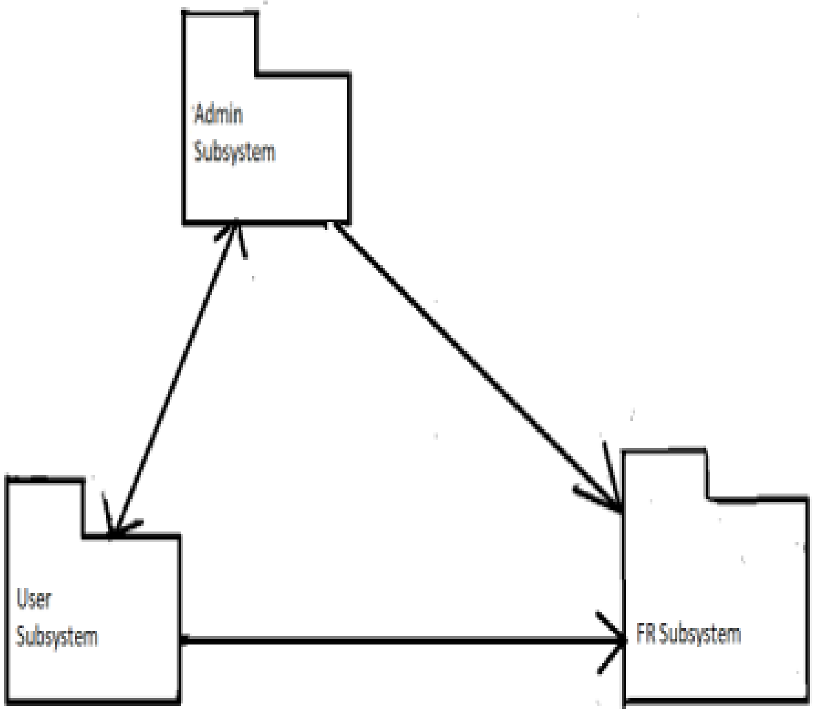


Fig 6.4. Subsystem Decomposition Diagram

6.2.UML INTRODUCTION

Unified Modelling Language (UML) is a general-purpose visual modelling language used to identify, visualize, construct, and document artifacts of a software system. It captures decisions and understanding about which systems to build. It is used to understand, design, browse, configure, maintain, and control information about these systems. The UML is a very important part of object-oriented software development and the software development process. UML mostly uses graphic symbols to express the design of software projects. Using the UML helps project teams to communicate, explore potential designs, and validate the software's architectural design. The primary objectives in UML design are:

- Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models. Provide extensibility and specialization mechanisms to extend the core concepts.
- To be independent of programming languages and special development processes.
- Provide a formal basis for understanding the modelling language. Encouraging the growth of the object-oriented tools market.
- Support high-level development concepts such as collaboration, frameworks, patterns, and components.
- Each UML diagram is designed to allow developers and customers to view a software system from a different perspective and with varying degrees of abstraction. UML diagrams commonly created in visual modelling tools include:

❖ Use Case Diagram
❖ Class Diagram
❖ Sequence Diagram
❖ Collaboration Diagram
❖ State Chart Diagram
❖ Activity Diagram

A use case diagram in UML is a type of behavioural diagram that was identified and created through a use case analysis. The purpose is to provide a graphical overview of the functionality provided by the system in terms of actors, their goals, and any dependencies between these two use cases. The main purpose of the use case diagram is to show the system functions that are implemented for any actor. The roles of the actors can be described in the system.

IDENTIFICATION OF ACTORS

The actors represent the users of the system. They help define the boundaries of the system and give a clearer picture of what the system should do. It is important to note that the subject interacts with but does not control the use cases.

- ❖ An actor is a person or thing:
- ❖ Interact with or use the system.
- ❖ Provides input and receives information from the system.
- ❖ Outside the system and has no control over the use cases.

IDENTIFICATION OF USE CASES

In its simplest form, a use case can be described as a specific way of using a system from the perspective of the user(s). A more detailed description might describe the use case as follows:

- ➢ The pattern of behaviour displayed by the system.
- ➢ The sequence of relevant transactions carried out by an actor in the system.
- ➢ Communicate something of value to the actor.

Use cases provide a means to:

- ➢ Capture system requirements.
- ➢ Communicate with end users and domain experts.
- ➢ Test the system.

It is best to discover use cases by examining the actors and deciding what the actor will do with the system. Since it is not possible to cover all the needs of a system in one use case, it is usual to have a set of use cases. The set of states together define all the ways the system is used.

CONSTRUCTION OF USE CASE DIAGRAM

Use-case diagrams graphically depict the system behaviour (use cases). These diagram's present a high-level view of how the system is used as viewed from an outsider's(actor's) perspective.

- A use-case diagram can contain:
- Actors ("things" outside the system)
- Use cases (system boundaries identifying what the system should do).

- Interactions or relationships between actors and use cases in the System include the associations, dependencies, and generalizations.

USES ASSOCIATION

The uses association occurs when we are describing our use-cases and notice that some of them have sub flows in common. To avoid describing a sub flow more than once in several use-cases, you can extract the common sub flow and make it a use-case of its own.

EXTENDS ASSOCIATION

An Extends association is a stereotyped association that specifies how the functionality of one of use case can be inserted into the functionality of another use-case.

INCLUDES ASSOCIATION

An Includes association is a stereotyped association that connects a base use case to an inclusion.

6.2.1 USECASE DIAGRAM

The purpose of the use case diagram is to capture the dynamic aspect of the system. However, this definition is too general to describe the purpose, as the other four schemas (Activity, Sequence, Collaboration, and State Diagram) have the same purpose as well. We will consider some specific purposes, which will distinguish it from the other four schemes.

State diagrams used to gather system requirements including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functions, use cases are prepared and actors are identified.

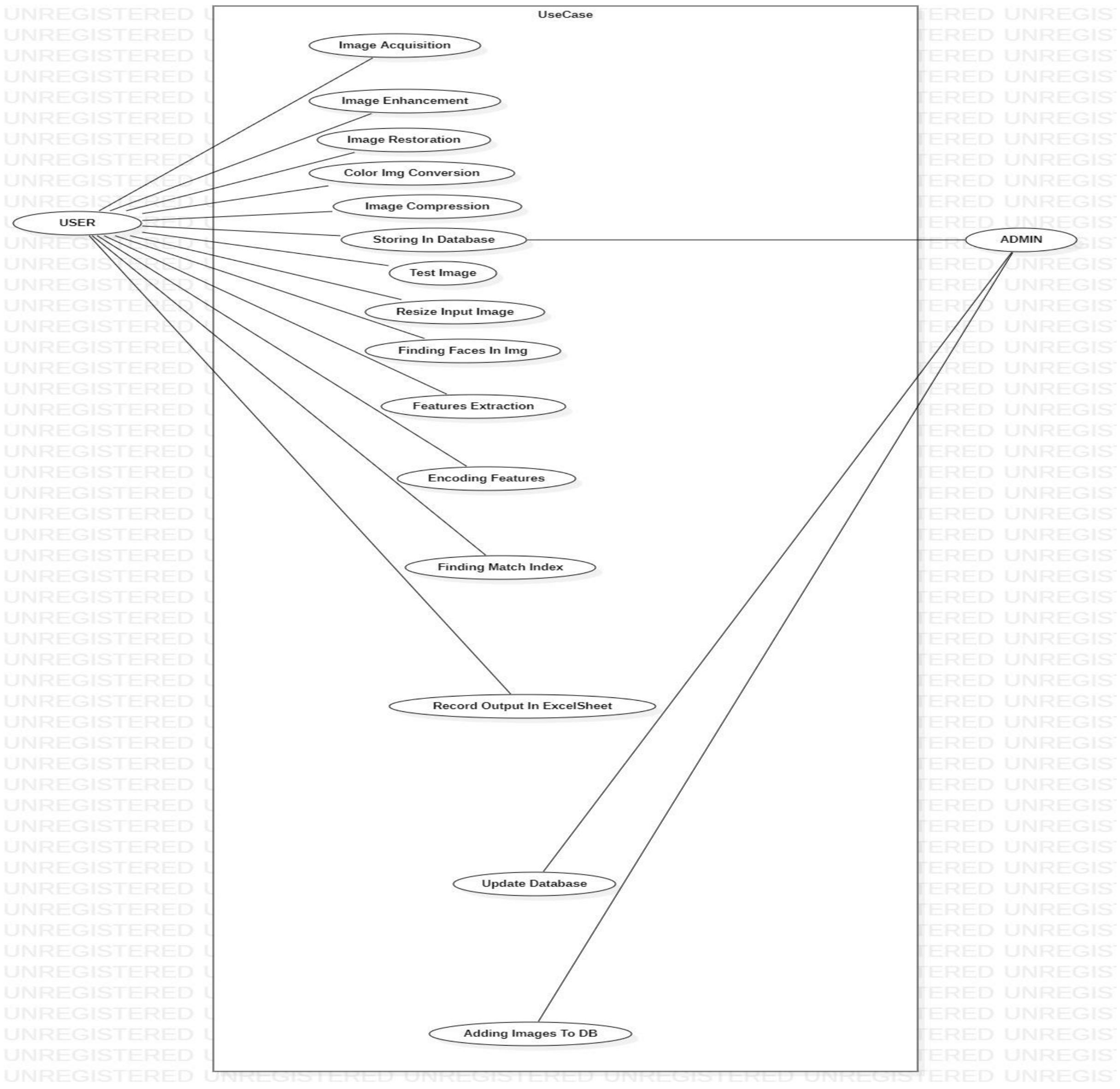


Fig 6.5. Use Case Diagram

6.2.2.CLASS DIAGRAM

The class diagram is a static diagram. ... Class diagrams are widely used in object-oriented modelling because they are the only UML diagrams, which can be mapped directly using object-oriented languages. The class diagram shows a set of classes, interfaces, associations, collaborations, and constraints.

An object model describes the structure of a system in terms of objects, attributes, associations, and processes. During requirements and analysis, the object model begins as the analysis object model and describes application concepts relevant to the system.

Displays attributes, classes, functions, and relationships to give an overview of the software system. They form the names of classes, attributes, and functions in a separate compartment that helps in software development. Since it is a set of classes, interfaces, associations, collaborations, and constraints, it is called an organization diagram.

The purpose of the class diagram is to model the static view of an application. Class graphs are the only graphs that can be mapped directly using object-oriented languages and are therefore widely used at build time.

UML Diagrams such as activity diagram and sequence diagram can only give the application's sequence flow, but the class diagram is slightly different. It is the most popular UML diagram in the programmer community.

IDENTIFICATION OF ATTRIBUTES OF EACH CLASS:

Guidelines for identifying attributes of classes are as follows:

- Adjectives usually correspond to nouns followed by prepositional phrases. • Attributes may also correspond to attributes or circumstances.
- Keep the class simple with only enough attributes to define the state of the object.
- It is unlikely that the features will be fully described in the problem statement.
- Delete derived attributes.
- Don't have redundant discovery features.

IDENTIFICATION OF RELATIONS OF EACH CLASS:

ASSOCIATION:

This relationship represents a physical or conceptual connection between two or more objects.

GENERALISATION:

This relationship represents inheritance between a child classes and a parent class.

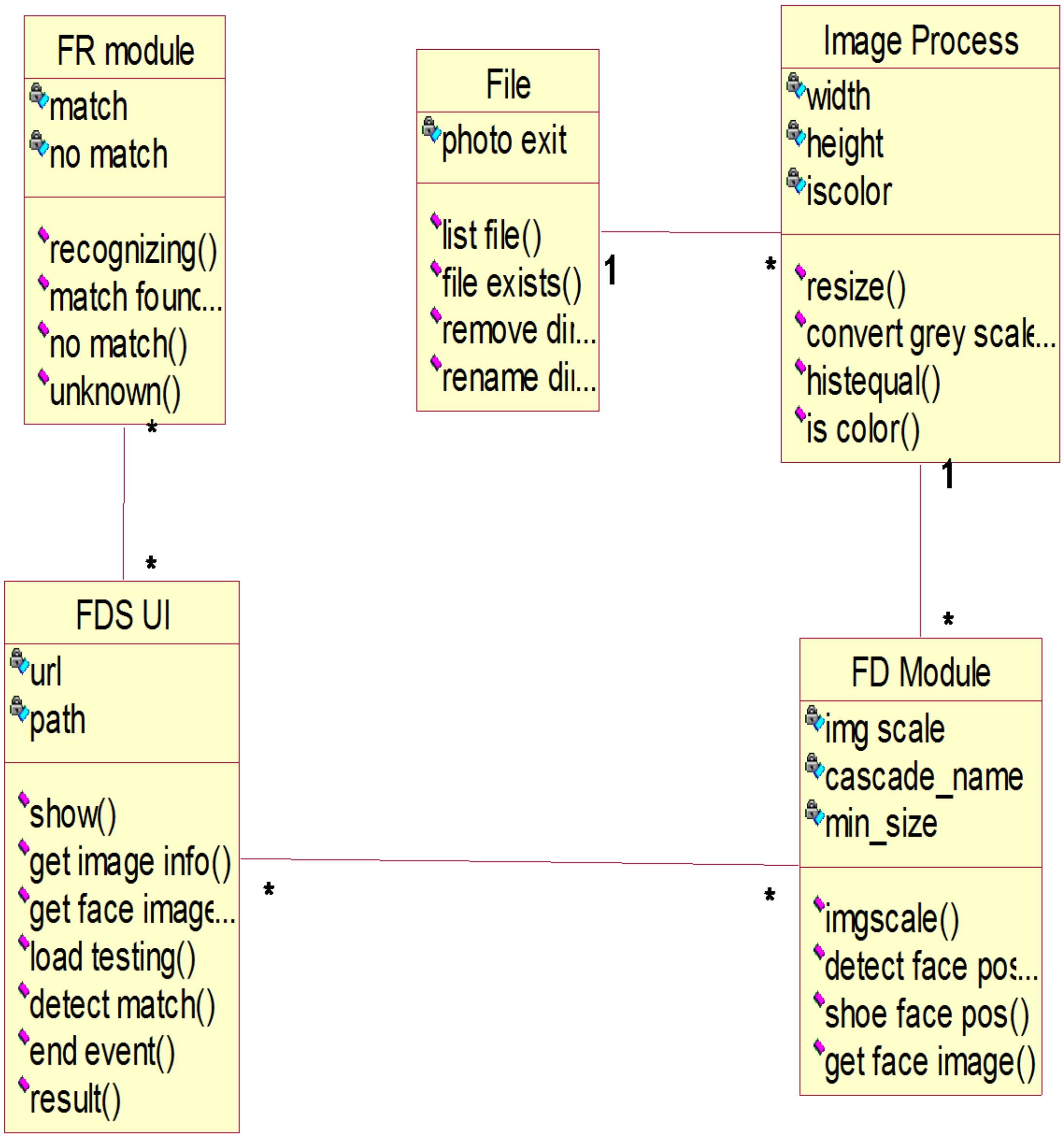


Fig 6.6. Class Diagram

6.2.3. SEQUENCE DIAGRAM

The sequence diagram A shows the interactions of the objects arranged in a chronological order. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functions of the scenario. Sequence diagrams are usually associated with realizing a use case in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

CONSTRUCTION OF SEQUENCE DIAGRAM

A sequence diagram is a graphic display of a scenario that shows the interaction of an object in a time-based sequence what happens first, and what happens next. Sequence diagrams define the roles of objects and help provide basic information for defining the responsibilities of a class and interfaces.

The dynamic model represented in UML with interaction diagrams, state diagrams, and activity diagrams describes the internal behaviour of the system.

The following tools are in the Sequence Diagram toolbox that enable the modelling of Sequence Diagrams.

- A The message icon represents communication between objects indicating that an action will be followed. The message symbol is a connected horizontal arrow that connects two lifelines together.
- Focus on Control (FOC) is an advanced notation technique that enhances sequence diagrams. Shows the period of time during which an object performs an action, either directly or through an underlying action.
- A message to self is a tool that sends a message from one object to the same object. It does not include other objects because the message goes back to the same object. The sender of the message is the same as the receiver.
- A note explaining the assumptions and decisions applied during the analysis and design. Notes may contain any information, including plain text, pieces of code, or references to another document.
- A link of note links the note with the element it affects.

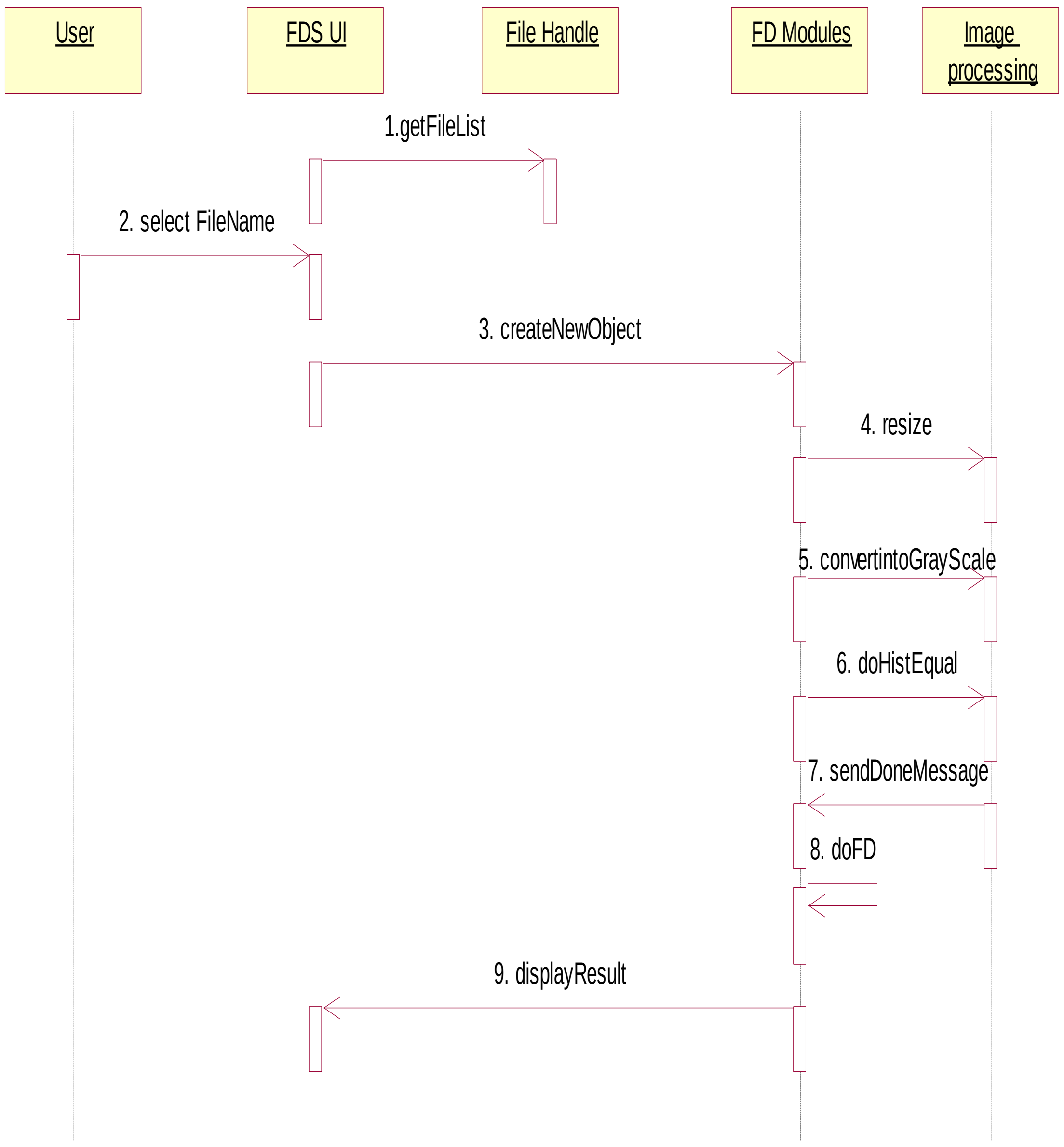


Fig 6.7. Sequence Diagram

DIFFERENCE BETWEEN SEQUENCE & COLLABORATION:

- Sequence diagrams show time-based object interactions graphs that show how objects relate to each other.
- Sequence diagrams are easier to read.

- Collaboration diagrams use plotting to indicate how objects are related statistically.
- The Create Collaboration Diagram command creates a collaboration diagram from the information in the sequence diagram. The Create Sequence Diagram command creates a sequence diagram from the information in the diagram for collaboration interactions.
- Sequence diagrams are closely related to collaboration diagrams, and both are alternative representations of interaction.

6.2.4. COLLABORATION DIAGRAM

Collaboration diagrams (known as a connection diagram) are used to show how objects interact to perform the behaviour of a particular use case, or part of a use case. Besides sequence diagrams, designers use collaboration to define and clarify the roles of objects that perform a specific flow of use case events.

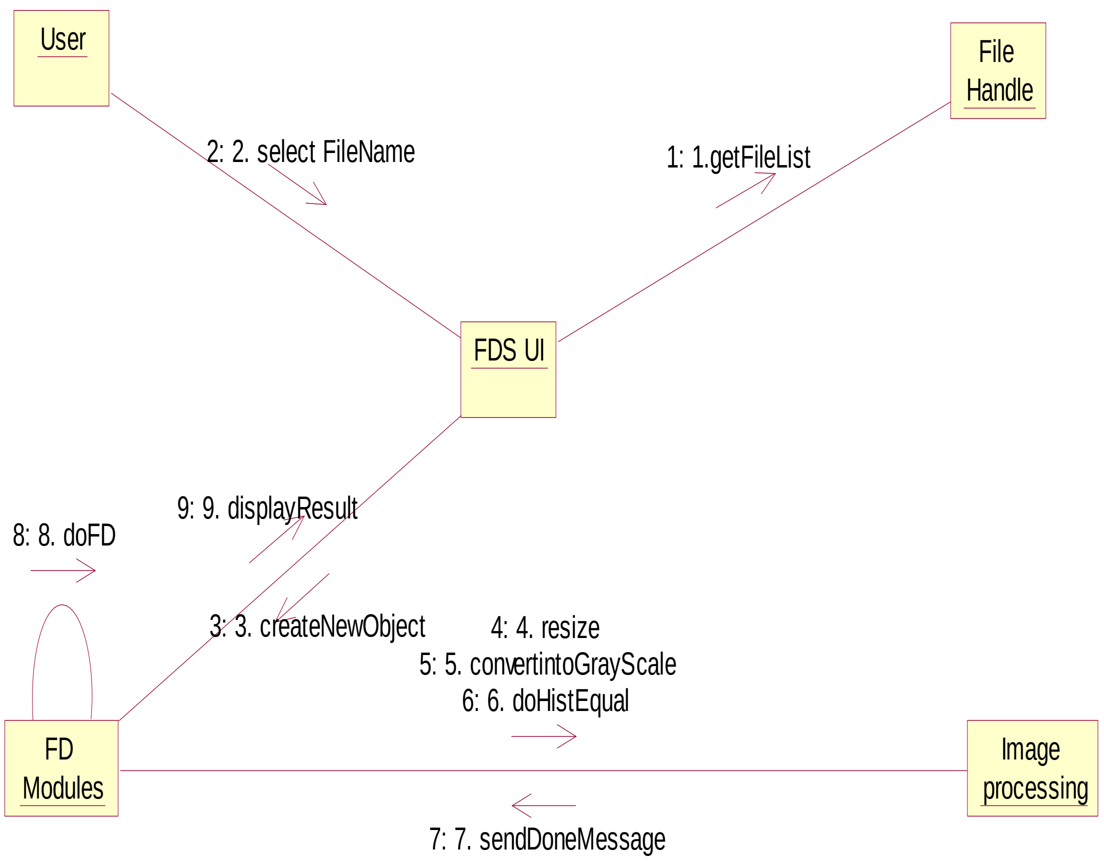


Fig 6.8. Collaboration Diagram

6.2.5. STATECHART DIAGRAM

State diagrams model the dynamic behaviour of individual classes or any other type of object. Shows the sequence of states that an object goes through events that cause the transition from one state to another and actions that result from a state change. State diagrams are closely related to activity diagrams. The main difference between the two diagrams is that state diagrams are state cantered, while activity diagrams are activity centred. A Status Charts graphs centre around the activity. A The state diagram is usually used to model the discrete stages of an object's life, while the activity diagram is more suitable for modelling the sequence of activities in a process. Each state represents a named condition during the life of the object during which it satisfies some condition or is waiting for an event. A state chart diagram usually contains one start state and two ending states. The transition connects the different states to the diagram.

The following tools are used on the state chart diagram toolbox to model state chart diagrams:

- **Decisions**: A decision represents a specific location on state chart diagram where the work flow may branch based upon guard conditions.
- **Synchronization**: Synchronizations visually define forks and joins representing parallel work flow.
- **Forks and Joins**: A fork construct is used to model a single flow of control that divides into two or more separate, but simultaneous flows. A join consists of two of more flows of control that unite into single flows of control that unite into single flow of control.
- **States**: A state represents a condition or situation during the life of an object during which it satisfies some conditions or waits for some event.
- **Transitions**: A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied.
- **Start states**: A start state (also called an "initial state") explicitly shows the beginning of a work flow.
- **End States**: An end state represents a final state or terminal state.

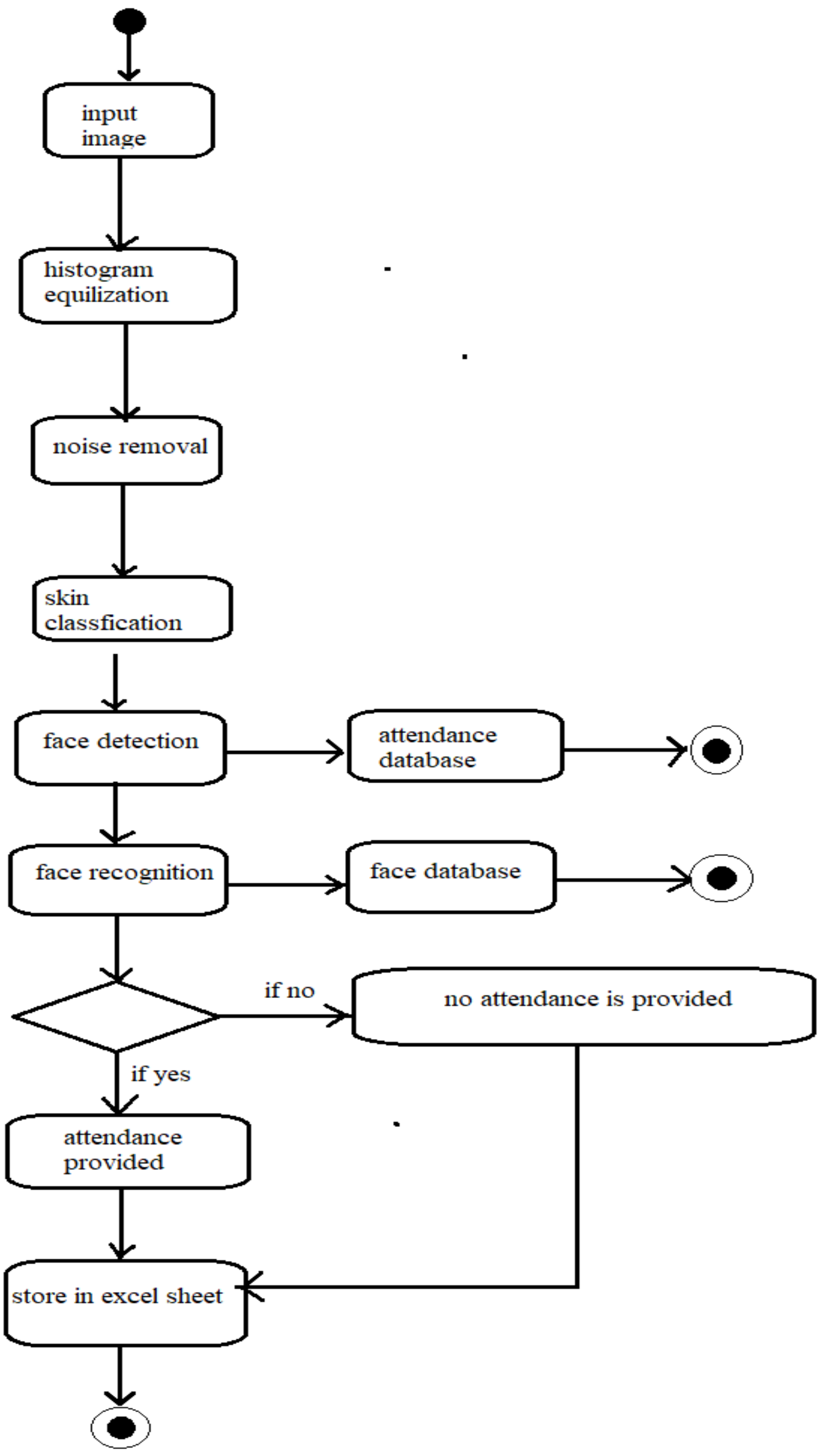


Fig 6.9. State Chart Diagram

6.2.6. ACTIVITY DIAGRAM

An activity diagram is basically a flowchart to represent the flow from one activity to another. Activity can be described as running the system. The flow of control is dragged from one process to another. This flow can be sequential, branched, or synchronous. Activity diagrams deal with all types of flow control using different elements like fork, hook, etc.

Activity diagrams provide a way to model the workflow of a business process. An activity diagram is usually used to model the sequence of workflows. A software company can use activity diagrams to model the software development process.

The following tools are used in the Activity Diagram toolbox for modelling activity diagrams:

- **Decisions**: A decision represents a specific location on activity diagram where the workflow may branch based upon guard conditions.
- **Synchronizations**: Synchronizations visually define forks and joins representing parallel workflow.
- **Forks and Joins**: A fork construct is used to model a single flow of control that divides into two or more separate, but simultaneous flows. A join consists of two or more flows of control that unite into a single flow of control.
- **States**: A state represents a condition or situation during the life of an object during which it satisfies some condition or waits for some event.
- **Transitions**: A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied.
- **Start states**: A start state (also called "initial state") explicitly shows the beginning of a workflow.
- **End States**: An end state represents a final state or terminal state.
- **Swim lane**: A unique activity diagram feature that defines who or what is responsible for carrying out the activity or state.

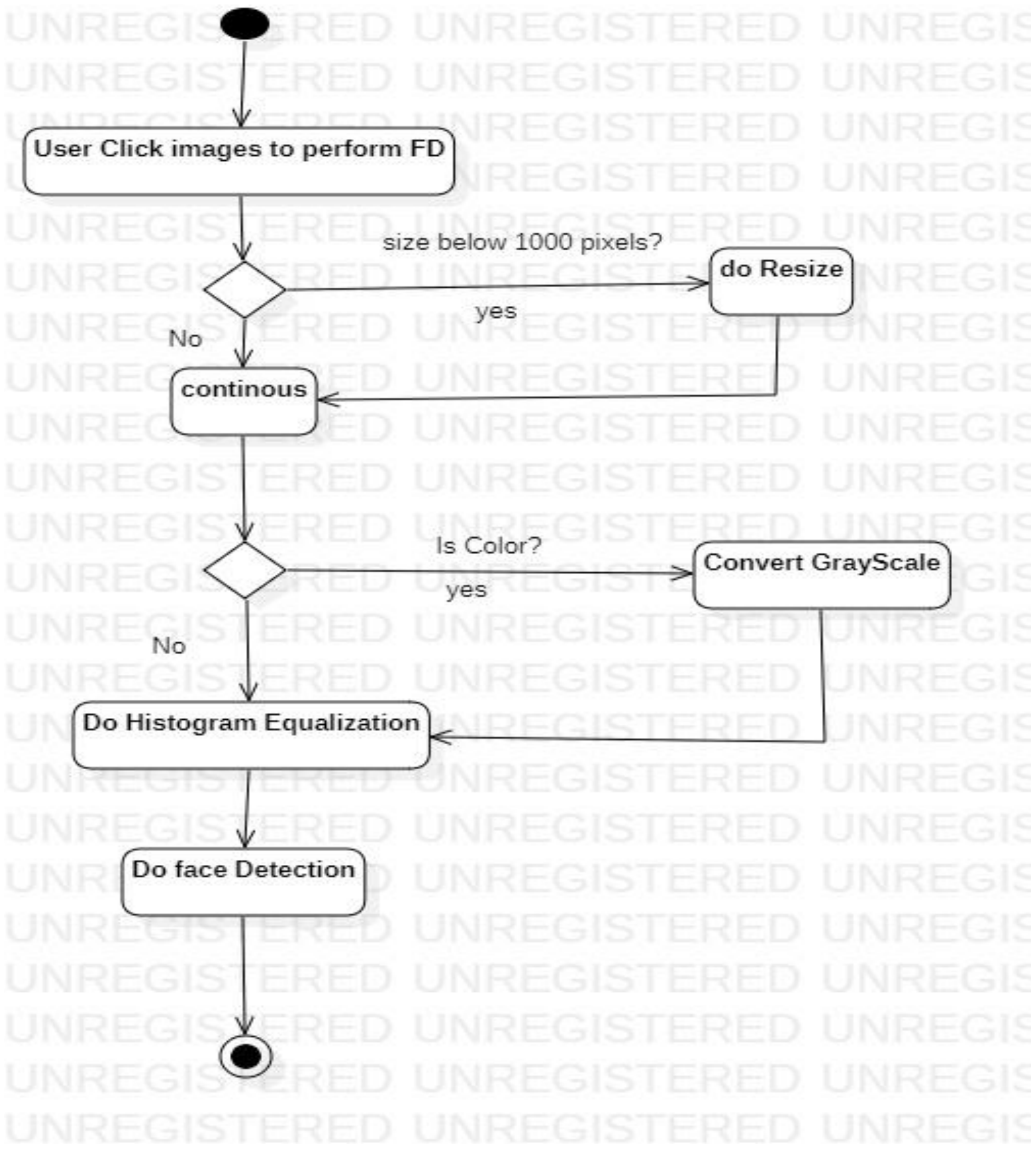


Fig 6.10. Activity Diagram

CHAPTER 7

SYSTEM TESTING

7.1. TESTING

The purpose of testing is to detect errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a method for verifying the functionality of components, sub-assemblies, assemblies, and/or the final product. It is the process of practicing software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. To find flaws in the developed system, we will use test cases, using test cases, we will find out if all user requirements are satisfied or not. Test case means a series of steps or actions that are performed to verify a feature or functionality of a software application. Test case A usually contains test steps, data, a precondition, and a post condition that are developed for a specific test scenario to validate the requirements. The different types of testing a particular system are listed below.

7.1.1 UNIT TESTING:

In computer programming, unit testing is a software testing method in which individual units of source code sets of one or more computer software units are tested together with associated control data, usage procedures, and operating procedures to determine if they are suitable for use.

7.1.2 INTEGRATION TESTING:

Integration testing is the stage in software testing in which individual software modules are combined and tested as a group. Integration testing is performed to assess the compliance of a system or component with specific functional requirements. It occurs after unit testing and before validation testing.

7.1.3 SYSTEM TESTING:

System testing is a test performed on a complete integrated system to assess the compliance of the system with its specified requirements. The system test takes, as its input, all the integrated components that have passed the integration test.

7.1.4 ACCEPTANCE TESTING:

Acceptance test is a test performed to determine whether the requirements of a specification or contract are satisfied. May include chemical tests, physical tests or performance tests.

7.1.5 BLACK BOX TESTING:

As we know in white box testing, the tester knows the internal structure of the item. Its major goal is to identify the workflow of the system by providing various test cases. But in the black box testing the implementation is ignored by the tester. Here the tester only concentrates whether the output is appropriate or not for the provided input data to the system.

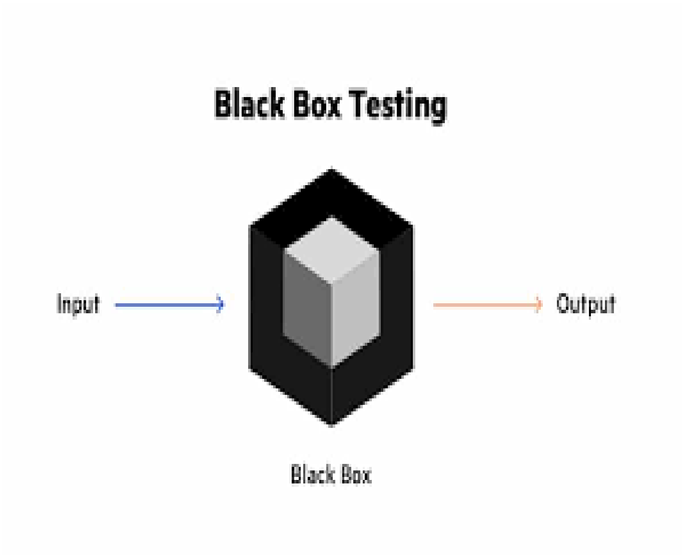


Fig 7.1 Black Box Testing

Our system is so small that we use black box testing to test our system, because black box testing helps find gaps in functionality, usability and other features. This type of test gives an overview of the program's performance and output. Improves software quality and reduces time to market. This type of testing mitigates the risk of software failure on the user's part.

Serial No	Input	Expected Output	Obtained Output	Result
1		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.1 Black Box Testing 1

Serial No	Input	Expected Output	Obtained Output	Result
2		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.2 Black Box Testing 2

Serial No	Input	Expected Output	Obtained Output	Result
3		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.3 Black Box Testing 3

Serial No	Input	Expected Output	Obtained Output	Result
4		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.4 Black Box Testing 4

Serial No	Input	Expected Output	Obtained Output	Result
5		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.5 Black Box Testing 5

Serial No	Input	Expected Output	Obtained Output	Result
6		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.2 Black Box Testing 2

Serial No	Input	Expected Output	Obtained Output	Result
7		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.7 Black Box Testing 7

Serial No	Input	Expected Output	Obtained Output	Result
8		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.8 Black Box Testing 8

Serial No	Input	Expected Output	Obtained Output	Result
9		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.9 Black Box Testing 9

Serial No	Input	Expected Output	Obtained Output	Result
10		Recognise the face and post the name to excel sheet	Face recognised and posted attendance to excel sheet	success

Table 7.10 Black Box Testing 10

7.2 RESULT ANALYSIS

The Auto Student Attendance system is proven to recognize images in different lighting and angle conditions. Faces that are not in our training data set are marked as unknown. Student photo recognition attendance is determined in real time. And import to Excel file and saved automatically by the system. The face detection feature worked very well even when people wore glasses, had a beard, or any other facial features, the real-time video speed was satisfactory and free of noticeable frame lag. All of these cases are considered and tested for a high level of accuracy and efficiency.

CHAPTER 8

CONCLUSION AND FUTURE WORK

CONCLUSION

The automatic attendance system explains to students the techniques used in the project and the methodologies used. Finally, it shows the result and how it was resolved followed by a discussion. The face detection feature worked very well even when people wore glasses, had a beard, or any other facial features, the real-time video speed was satisfactory and free of noticeable frame lag. All of these cases are considered and tested for a high level of accuracy and efficiency. Thus, it can be concluded from the above discussion that a reliable, safe, fast, and efficient system has been developed to replace a manual and unreliable system. The system will save time, reduce the amount of work that the management has to do, replace stationery items with electronic devices and reduce the number of human resources required for this purpose. It can be implemented as a cost-effective platform for facial recognition. The main working principle of the project is video captured data is converted into image to detect and recognize it.

FUTURE WORK

Currently, this system is developed by keeping student's attendance in mind to make administrative easier. So, we would like to extend this application to various Platforms like offices, malls, etc.

The time taken to train the data set is more. So would like to reduce the training time by using latest conventional methods. It can further be improved to obtain high accuracy levels and better results.

CHAPTER 9

REFERENCES

[1] Study of Implementing Automated Attendance System using Face Recognition – Nirmalya kar, Mrinal Kanti, Ashim Saha, Dweijen Rudra Pal.

[2] Class Attendance Management System using Facial Recognition – Clyde Gomes, Sagar Chanchal, Tanmay Desai.

[3] Face Recognition Based Attendance Management System Using Raspberry Pi – Dr. Mohd. Abdul Muqeet.

[4] Image Analysis for Face Recognition – Edy Winarno, Henry Februariyanti.

[5] Face Recognition Based Attendance System Using Machine Learning – Dr. Shrija Madhu, Ms.Anusha Adapa, Ms. Visrutha Vatsavaya, Ms. Padmini Pothula.

[6] Image - Based Face Detection System - Tsang Tat Man.

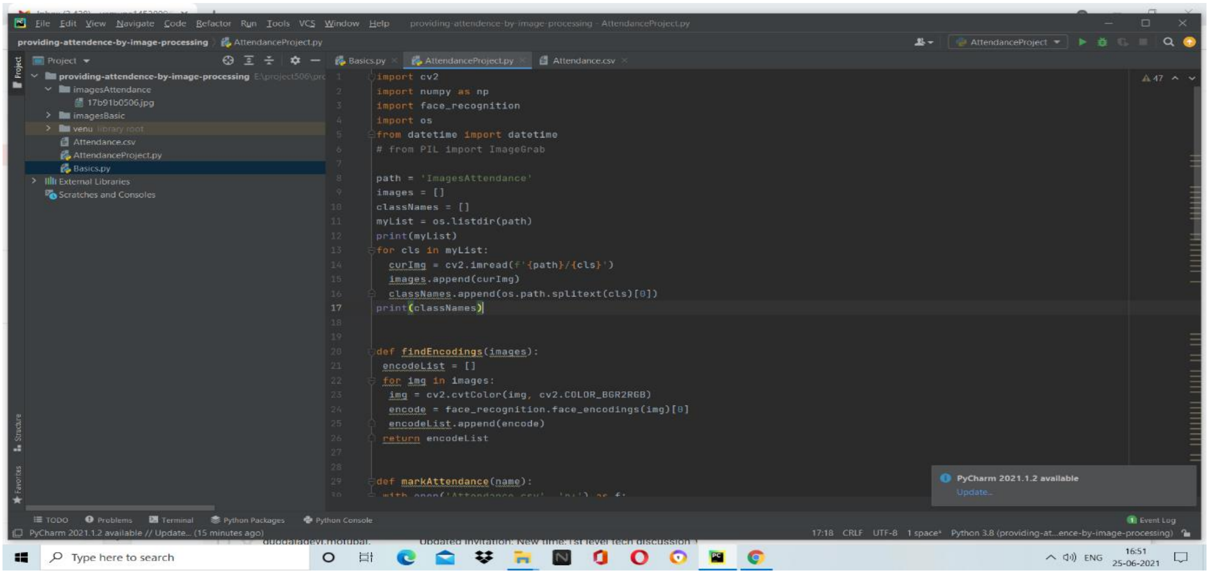
CHAPTER 10

APPENDIX

10.1. SAMPLE CODE:

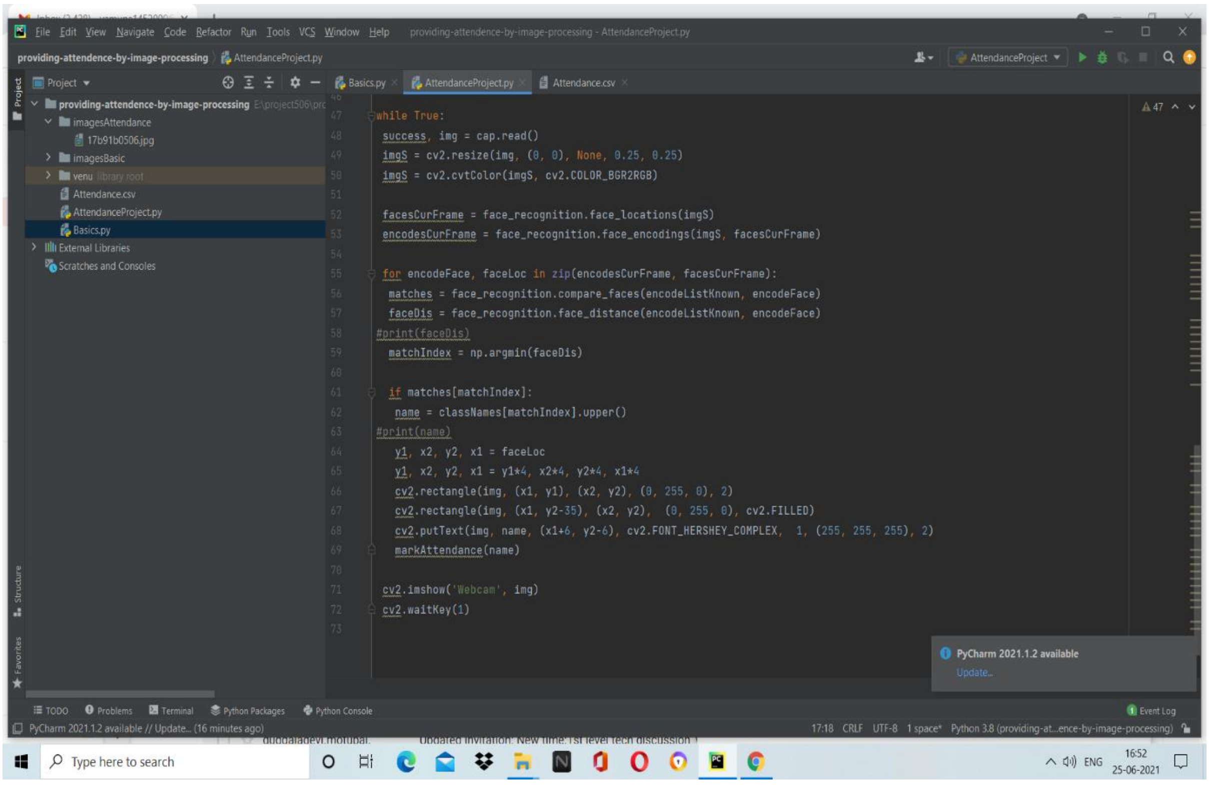
```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime
# from PIL import ImageGrab

path = 'ImagesAttendance'
images = []
classNames = []
myList = os.listdir(path)
print(myList)
for cl in myList:
  curImg = cv2.imread(f'{path}/{cl}')
  images.append(curImg)
  classNames.append(os.path.splitext(cl)[0])
print(classNames)


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


def markAttendance(name):
 with open('Attendance.csv', 'r+') as f:
  myDataList = f.readlines()
  nameList = []
  for line in myDataList:
   entry = line.split(',')
   nameList.append(entry[0])
  if name not in nameList:
    now = datetime.now()
    dtString = now.strftime('%H:%M:%S')
    f.writelines(f'\n{name},{dtString}')


encodeListKnown = findEncodings(images)
print('Encoding Complete')

cap = cv2.VideoCapture(0)

while True:
 success, img = cap.read()
 imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
 imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

 facesCurFrame = face_recognition.face_locations(imgS)
 encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

 for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
  matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
  faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
#print(faceDis)
  matchIndex = np.argmin(faceDis)

  if matches[matchIndex]:
   name = classNames[matchIndex].upper()
#print(name)
   y1, x2, y2, x1 = faceLoc
   y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
   cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
   cv2.rectangle(img, (x1, y2-35), (x2, y2),  (0, 255, 0), cv2.FILLED)
   cv2.putText(img, name, (x1+6, y2-6), cv2.FONT_HERSHEY_COMPLEX,  1, (255, 255,
255), 2)
   markAttendance(name)

 cv2.imshow('Webcam', img)
 cv2.waitKey(1)
```

10.2. IMPLEMENTATION SCREENS:



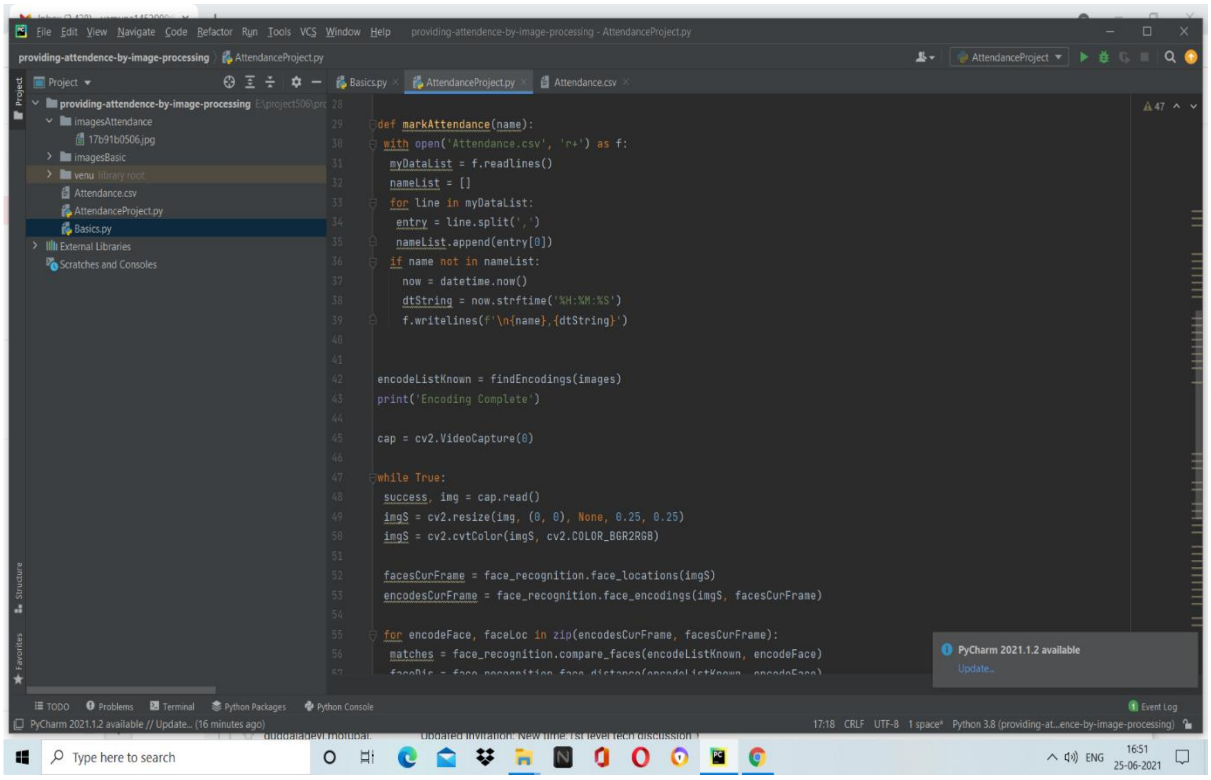
```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime
# from PIL import ImageGrab

path = 'ImagesAttendance'
images = []
classNames = []
myList = os.listdir(path)
print(myList)
for cls in myList:
  curImg = cv2.imread(f'{path}/{cls}')
  images.append(curImg)
  classNames.append(os.path.splitext(cls)[0])
print(classNames)


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


def markAttendance(name):
```

Fig 10.1 Implementation Screen 1



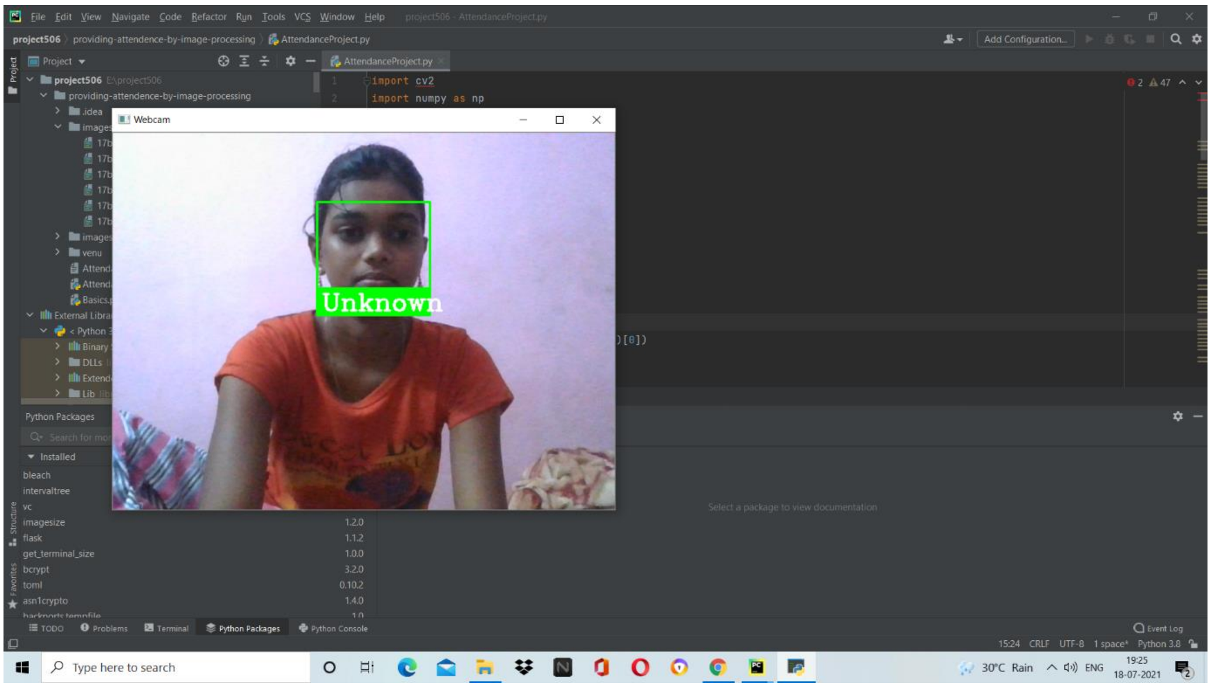
```
while True:
 success, img = cap.read()
 imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
 imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

 facesCurFrame = face_recognition.face_locations(imgS)
 encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

 for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
  matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
  faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
#print(faceDis)
  matchIndex = np.argmin(faceDis)

  if matches[matchIndex]:
   name = classNames[matchIndex].upper()
#print(name)
   y1, x2, y2, x1 = faceLoc
   y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
   cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
   cv2.rectangle(img, (x1, y2-35), (x2, y2),  (0, 255, 0), cv2.FILLED)
   cv2.putText(img, name, (x1+6, y2-6), cv2.FONT_HERSHEY_COMPLEX,  1, (255, 255, 255), 2)
   markAttendance(name)

 cv2.imshow('Webcam', img)
 cv2.waitKey(1)
```

Fig 10.2 Implementation Screen 2



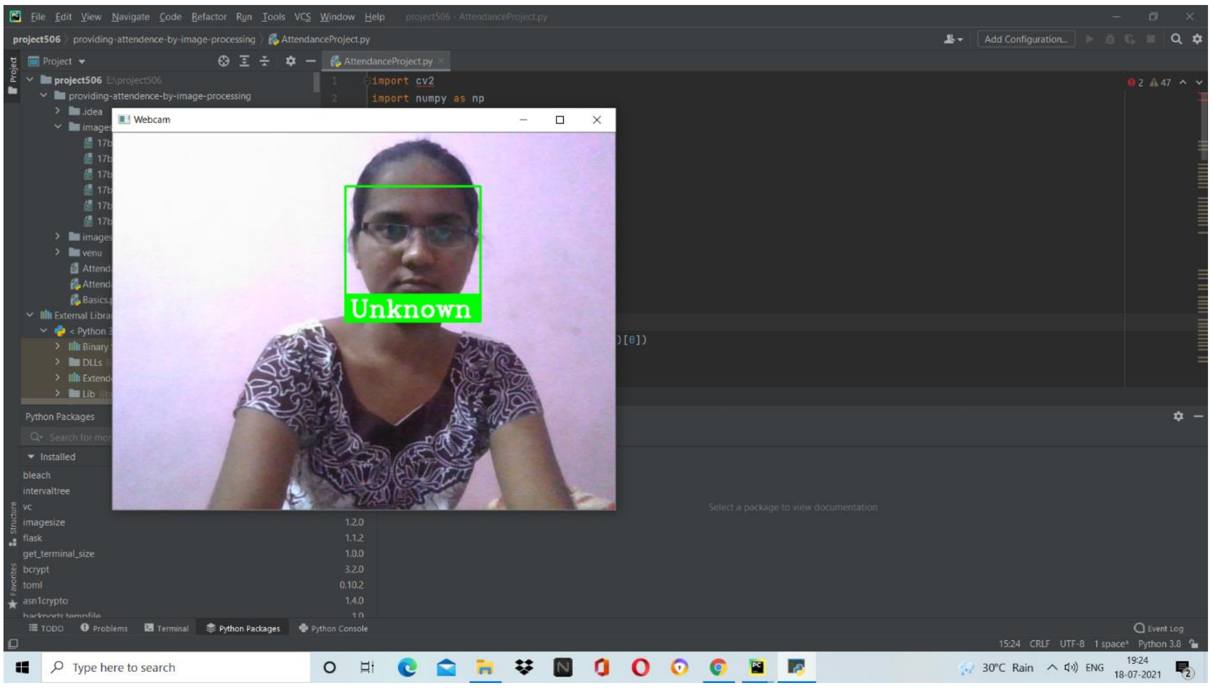
```
def markAttendance(name):
 with open('Attendance.csv', 'r+') as f:
  myDataList = f.readlines()
  nameList = []
  for line in myDataList:
   entry = line.split(',')
   nameList.append(entry[0])
  if name not in nameList:
   now = datetime.now()
   dtString = now.strftime('%H:%M:%S')
   f.writelines(f'\n{name},{dtString}')


encodeListKnown = findEncodings(images)
print('Encoding Complete')

cap = cv2.VideoCapture(0)

while True:
 success, img = cap.read()
 imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
 imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

 facesCurFrame = face_recognition.face_locations(imgS)
 encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

 for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
  matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
```

Fig 10.3 Implementation Screen 3



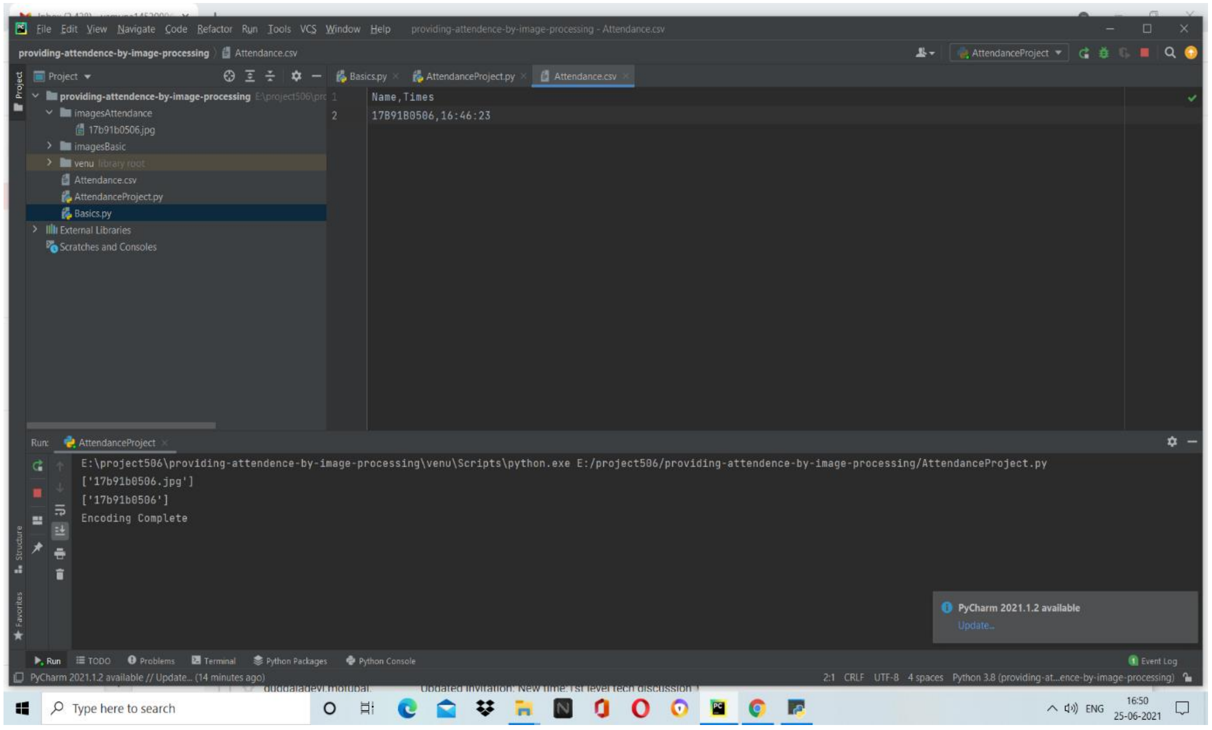
```
import cv2
import numpy as np
```

Fig 10.4 Implementation Screen 4



```
import cv2
import numpy as np
```

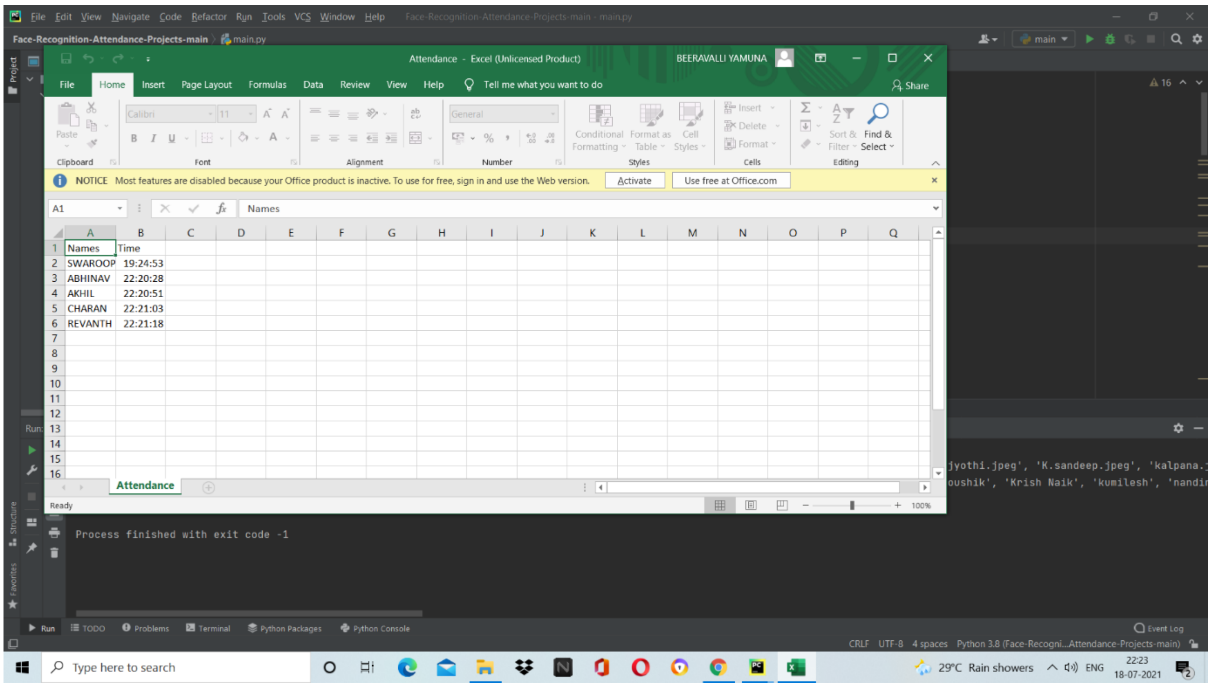
Fig 10.5 Implementation Screen 5



```
Name,Times
17B91B0506,16:46:23
```

```
E:\project506\providing-attendence-by-image-processing\venu\Scripts\python.exe E:/project506/providing-attendence-by-image-processing/AttendanceProject.py
['17b91b0506.jpg']
['17b91b0506']
Encoding Complete
```

Fig 10.6 Implementation Screen 6



	A	B
1	Names	Time
2	SWAROOP	19:24:53
3	ABHINAV	22:20:28
4	AKHIL	22:20:51
5	CHARAN	22:21:03
6	REVANTH	22:21:18

```
jyothi.jpeg', 'K.sandeep.jpeg', 'kalpana.
oushik', 'Krish Naik', 'kumilesh', 'nandi
```

```
Process finished with exit code -1
```

Fig 10.7 Implementation Screen 7