

A  
**Project Report**  
on  
**FACE RECOGNITION BASED ATTENDANCE SYSTEM**

Submitted to

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTAPURAMU**

in partial fulfillment of the requirements for the award of the Degree of

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

N SUNIL	(209E1A05F4)
N VAASAVI	(209E1A05E7)
N PAVANI	(209E1A05E8)
P THARUN TEJ	(209E1A05G8)
S UMAPATHI	(20381A0513)

Under the Guidance of  
**Mr. J. Sankar Babu M.Tech**  
Assistant Professor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**SRI VENKATESWARA ENGINEERING COLLEGE**

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)

Karakambadi Road, TIRUPATI – 517507

**2020-2024**

**SRI VENKATESWARA ENGINEERING COLLEGE**  
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)  
**Karakambadi Road, TIRUPATI – 517507**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

*This is to certify that the project report entitled **“FACE RECOGNITION BASED ATTENDANCE SYSTEM”** a bonafide record of the project work done and submitted by*

<b>N SUNIL</b>	<b>(209E1A05F4)</b>
<b>N VAASAVII</b>	<b>(209E1A05E7)</b>
<b>N PAVANI</b>	<b>(209E1A05E8)</b>
<b>P THARUN TEJ</b>	<b>(209E1A05G8)</b>
<b>S UMAPATHI</b>	<b>(20381A0513)</b>

*for the partial fulfillment of the requirements for the award of B.Tech Degree in **COMPUTER SCIENCE AND ENGINEERING**, JNT University Anantapur, Ananthapuramu.*

**GUIDE:**  
**Mr. J. SANKAR BABU, M.Tech .**  
Assistant Professor  
Dept. of. CSE

**HEAD OF THE DEPARTMENT:**  
**Dr.SANTHI, Ph.D.**  
Professor & HOD  
Dept. of. CSE

External Viva-Voce Exam Held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

# DECLARATION

We hereby declare that the project report entitled **“FACE RECOGNITION BASED ATTENDANCE SYSTEM”** done by us under the guidance of **Mr. J Sankar Babu**, and is submitted in partial fulfillment of the requirements for the award of the Bachelor’s degree in **Computer Science and Engineering**. This project is the result of our own effort and it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above.

<b>N SUNIL</b>	<b>(209E1A05F4)</b>
<b>N VAASAVII</b>	<b>(209E1A05E7)</b>
<b>N PAVANI</b>	<b>(209E1A05E8)</b>
<b>P THARUN TEJ</b>	<b>(209E1A05G8)</b>
<b>S UMAPATHI</b>	<b>(20381A0513)</b>

## ACKNOWLEDGEMENT

We are thankful to our guide **Mr. J Sankar Babu sir** for his valuable guidance and encouragement. His helping attitude and suggestions have helped us in the successful completion of the project.

We would like to express our gratefulness and sincere thanks to **Dr. K. Santhi**, Head of the Department of COMPUTER SCIENCE AND ENGINEERING, for her kind help and encouragement during the course of our study and in the successful completion of the project work.

We have great pleasure in expressing our hearty thanks to our beloved Principal **Dr. C. Chandrasekhar**, for spending his valuable time with us to complete this project.

Successful completion of any project cannot be done without proper support and encouragement. We sincerely thank the **Management** for providing all the necessary facilities during the course of study.

We would like to thank our parents and friends, who have the greatest contributions in all our achievements, for the great care and blessings in making us successful in all our endeavors.

<b>N SUNIL</b>	<b>(209E1A05F4)</b>
<b>N VAASAVII</b>	<b>(209E1A05E7)</b>
<b>N PAVANI</b>	<b>(209E1A05E8)</b>
<b>P THARUN TEJ</b>	<b>(209E1A05G8)</b>
<b>S UMAPATHI</b>	<b>(20381A0513)</b>

## TABLE OF CONTENTS

Chapter No.	Description	Page No.
	Abstract	i
	List of Figures	li
	List of Tables	lii
	List of Abbreviations	Iv
1	Introduction	1
2	Project Description	2
	2.1 Problem Definition	2
	2.2 Project Details	3
3	Computational Environment	5
	3.1 Software Specification	5
	3.2 Hardware Specification	5
	3.3 Software Features	5
4	Feasibility Study	31
	4.1 Technical Feasibility	31
	4.2 Social Feasibility	31
	4.3 Economical Feasibility	32
5	System Analysis	33
	5.1 Existing System	33
	5.1.1 Drawbacks of existing system	33
	5.2 Proposed System	33
	5.2.1 Advantages of proposed System	34
6	System Design	35
	6.1 UML Diagrams	35
	6.1.1 Class Diagram	36
	6.1.2 Use case Diagram	38
	6.1.3 Sequence Diagram	40
	6.1.4 Activity Diagram	42
7	System Implementation	45
	7.1 Implementation Process	45
	7.2 Modules	47

<b>8</b>	<b>Testing</b>	<b>50</b>
	8.1 Unit Testing	50
	8.2 Integration Testing	51
	8.3 System Testing	52
	8.4 Acceptance Testing	52
<b>9</b>	<b>Sample Source Code</b>	<b>55</b>
<b>10</b>	<b>Module List</b>	<b>66</b>
<b>11</b>	<b>Screen Layouts</b>	<b>69</b>
<b>12</b>	<b>Conclusion and Future Scope</b>	<b>71</b>
<b>13</b>	<b>Bibiliography</b>	<b>72</b>

## **ABSTRACT**

This project introduces a user-friendly attendance system using facial recognition technology. The aim is to simplify and improve traditional attendance tracking by automating the process. The system utilizes existing infrastructure commonly found in schools, offices, and public areas to create an efficient, accurate, and non-intrusive attendance management solution.

By analyzing facial features in real-time, the system quickly identifies individuals and cross-references them with a preregistered database. Privacy concerns are addressed through secure data handling practices. The research involves designing, developing, and testing the system in various environments, demonstrating its practicality and effectiveness. This project offers a straightforward and reliable alternative to traditional attendance methods, promising ease of deployment and improved attendance management for diverse organizations.

In conclusion, our facial recognition attendance system demonstrates a user-friendly and effective alternative to traditional methods. With seamless integration into existing networks, it offers a practical solution for organizations aiming to enhance attendance tracking accuracy and efficiency. This research lays the foundation for modernizing attendance management processes with simplicity and reliability.

<b>N SUNIL</b>	<b>(209E1A05F4)</b>
<b>N VAASAVII</b>	<b>(209E1A05E7)</b>
<b>N PAVANI</b>	<b>(209E1A05E8)</b>
<b>P THARUN TEJ</b>	<b>(209E1A05G8)</b>
<b>S UMAPATHI</b>	<b>(20381A0513)</b>

## LIST OF FIGURES

S.NO.	FIGURE NO.	DESCRIPTION	PAGE NO.
1	Fig 3.1	Python for windows	11
2	Fig 3.2	Installation of python	12
3	Fig 3.3	Python setup	12
4	Fig 3.4	Python in Terminal	13
5	Fig 3.5	Pycharm	14
6	Fig 3.6	Visual Studio Code	15
7	Fig 3.7	Pydev	16
8	Fig 3.8	Sublime Text	17
9	Fig 3.9	Spyder	18
10	Fig 3.10	Jupyter Notebook	19
11	Fig 3.11	Thonny	20
12	Fig 3.12	Customize install location	26
13	Fig 3.13	Running Python using Thonny	27
14	Fig 3.14	Simple program in terminal	28
15	Fig 3.15	Python IDLE	28
16	Fig 6.1	Class Diagram	37
17	Fig 6.2	Use Case Diagram	38
18	Fig 6.3	Sequence Diagram	40
19	Fig 6.4	Activity Diagram	42
20	Fig 11.1	Folders Creation	69
21	Fig 11.2	Command to run the application	69



## LIST OF FIGURES

22	Fig 11.3	Attendance Tracking	70
----	----------	---------------------	----

## **LIST OF ABBREVIATIONS**

<b>S.NO</b>	<b>NAME</b>	<b>ABBREVIATION</b>
<b>1</b>	IDLE	Integrated Development and Learning Environment
<b>2</b>	OpenCV	Open Computer Vision
<b>3</b>	NUMPY	Numerical Python
<b>4</b>	UML	Unified Modelling Language
<b>5</b>	SQL	Structured Query
<b>6</b>	HTML	Hyper Text Markup Language
<b>7</b>	CSS	Cascade Style Sheet

## 1. INTRODUCTION

Face recognition-based attendance systems are innovative solutions that leverage facial recognition technology to automate the attendance tracking process. This system utilizes advanced algorithms to detect and recognize unique facial features of individuals, allowing for accurate identification and recording of attendance without the need for manual intervention.

In practical terms, the system works by capturing images or videos of individuals' faces using cameras installed in designated areas, such as entry points or classrooms. These images are then processed by the facial recognition software, which analyzes key facial characteristics like the distance between eyes, nose shape, and mouth structure to create a unique biometric profile for each individual.

When a person approaches the system, their face is scanned, and the software compares it with the stored biometric profiles in its database. If a match is found, the system records the individual's attendance automatically, providing real-time data that can be accessed by administrators or supervisors.

Face recognition-based attendance systems offer several advantages over traditional methods. They eliminate the need for physical attendance registers or ID cards, reducing administrative workload and the likelihood of errors or fraudulent entries. Additionally, these systems provide a more secure and efficient way of tracking attendance, as they are less susceptible to manipulation or proxy attendance.

Overall, face recognition-based attendance systems represent a modern and effective approach to managing attendance in various settings, including schools, workplaces, and events, streamlining processes and enhancing security.

## **2.PROJECT DESCRIPTION**

### **2.1 PROBLEM DEFINITION**

The traditional attendance system, relying on methods like manual sign-in sheets or physical punch cards, poses several challenges in modern environments. One of the primary issues is its susceptibility to errors and manipulation. Human errors in recording attendance, such as illegible handwriting or accidental omissions, can lead to inaccurate data. Moreover, individuals may engage in proxy attendance, where someone else signs in on their behalf, undermining the integrity of attendance records.

Another problem with traditional attendance systems is their inefficiency and time-consuming nature. Collecting and processing attendance data manually can be a labor-intensive task, requiring significant administrative effort. Moreover, the process of tallying attendance records and generating reports may be prone to delays, hindering timely decision-making and analysis.

Additionally, traditional attendance systems lack flexibility and adaptability to dynamic environments. They often struggle to accommodate variations in scheduling, such as flexible work hours or remote work arrangements. This rigidity can result in inaccuracies and discrepancies in attendance tracking, leading to frustration among employees or students.

Furthermore, traditional attendance systems may compromise security and privacy. Physical sign-in sheets or punch cards can be misplaced, lost, or stolen, potentially exposing sensitive personal information. Moreover, the reliance on manual processes increases the risk of unauthorized access to attendance records, raising concerns about data security and confidentiality.

In summary, the traditional attendance system faces challenges related to accuracy, efficiency, flexibility, and security. To address these issues, organizations may need to explore more advanced solutions, such as automated attendance systems leveraging technologies like biometrics or RFID, to enhance reliability, streamline processes, and improve data security.

## 2.2 PROJECT DETAILS

The face recognition-based attendance system project aims to revolutionize the conventional methods of attendance tracking by implementing cutting-edge facial recognition technology. The project begins with a comprehensive analysis of the existing attendance systems in various settings, including educational institutions, corporate offices, and public events. This analysis highlights the shortcomings of traditional methods such as manual sign-in sheets, swipe cards, or biometric scanners, including inaccuracies, susceptibility to fraud, and inefficiencies.

To address these challenges, the project team embarks on designing and developing a sophisticated attendance system centered around facial recognition. The first phase involves researching and selecting the most suitable facial recognition algorithms and technologies based on factors such as accuracy, speed, scalability, and compatibility with existing infrastructure. This step ensures that the system can reliably identify individuals in real-time, even in challenging conditions such as varying lighting or facial expressions.

In parallel, the hardware components necessary for implementing the system are identified and procured. This includes high-resolution cameras capable of capturing clear facial images, processing units with sufficient computing power to perform complex facial recognition algorithms, and network infrastructure for data transmission and storage. The system architecture is meticulously designed to facilitate seamless integration with existing attendance management systems or databases, ensuring minimal disruption to workflow and operations.

Once the hardware and software components are in place, the development team focuses on building the user interface and backend functionalities of the attendance system. The user interface is designed to be intuitive and user-friendly, allowing administrators to easily configure settings, monitor attendance data in real-time, and generate reports. Meanwhile, the backend system handles tasks such as face detection, feature extraction, matching against the database of enrolled individuals, and recording attendance records securely.

Throughout the development process, rigorous testing and validation procedures are conducted to evaluate the system's performance, accuracy, and reliability. This includes testing the system with diverse datasets of facial images, simulating various scenarios and conditions to assess its robustness and scalability. Any

identified issues or bugs are promptly addressed through iterative development cycles, ensuring that the final product meets the highest standards of quality and reliability.

Once the development and testing phases are complete, the face recognition-based attendance system undergoes a pilot deployment in a controlled environment, such as a specific department within an organization or a single classroom in a school. This pilot phase allows for real-world testing and validation of the system's effectiveness in accurately tracking attendance, addressing any operational challenges or user feedback, and refining implementation strategies.

Following the successful completion of the pilot phase, the face recognition-based attendance system is rolled out across the entire organization or institution. Comprehensive training and support are provided to administrators, supervisors, and end-users to ensure smooth adoption and optimal utilization of the system. Continuous monitoring and evaluation mechanisms are put in place to assess the system's impact on attendance management practices, efficiency gains, and user satisfaction, with periodic updates and improvements implemented as needed to enhance system performance and functionality.

In summary, the face recognition-based attendance system project represents a significant advancement in attendance tracking technology, offering a reliable, efficient, and secure solution for organizations and institutions seeking to modernize their attendance management processes. By leveraging state-of-the-art facial recognition technology and best practices in system design and implementation, the project aims to improve accuracy, streamline operations, and enhance overall organizational effectiveness.

### **3.COMPUTATIONAL ENVIRONMENT**

#### **3.1 SOFTWARE SPECIFICATION**

- Operating system : Windows 11.
- Coding Language : Python, html
- Platform : Python IDE, Internet Explorer
- Technology : Machine Learning

#### **3.2 HARDWARE SPECIFICATION**

- System : i3 Processor
- Hard Disk : 512 GB.
- Ram : 4 GB

#### **3.3 SOFTWARE FEATURES**

Python, a high-level programming language, offers a plethora of features that contribute to its popularity among developers. One prominent feature is its simplicity and readability, making it an ideal choice for beginners and seasoned programmers alike. Python's syntax emphasizes code readability, with clear and concise code structures that are easy to understand and maintain.

Another key feature of Python is its versatility and cross-platform compatibility. Python supports multiple operating systems, including Windows, macOS, and Linux, allowing developers to write code once and deploy it across different platforms without modification. This flexibility makes Python suitable for a wide range of applications, from web development to scientific computing and data analysis.

Python's extensive standard library is another notable feature that simplifies development by providing a rich set of pre-built modules and functions. These modules cover a wide range of tasks, including file I/O, networking, database access, and more, enabling developers to leverage existing code and resources to expedite development and reduce time-to-market.

Furthermore, Python's dynamic typing and automatic memory management contribute to its ease of use and productivity. Unlike statically-typed languages, Python does not require variable declarations, allowing developers to focus on solving problems rather than managing data types. Additionally, Python's garbage collection mechanism automatically deallocates memory when objects are no longer in use, eliminating the need for manual memory management and reducing the risk of memory leaks and other memory-related issues.

Python's support for object-oriented programming (OOP) is another valuable feature that enables developers to write modular, reusable code. Python allows for the creation of classes and objects, encapsulating data and behavior into cohesive units that can be easily extended and modified. This facilitates code organization, promotes code reuse, and enhances maintainability, particularly for large-scale projects.

Moreover, Python's extensive ecosystem of third-party libraries and frameworks further extends its capabilities, catering to a wide range of development needs. From web frameworks like Django and Flask to data science libraries like NumPy and pandas, Python offers a vast array of tools and resources that empower developers to tackle diverse challenges and build robust, feature-rich applications.

Overall, Python's simplicity, versatility, extensive standard library, dynamic typing, object-oriented programming support, and rich ecosystem of third-party libraries make it a compelling choice for developers across various domains and industries. Its combination of ease of use, flexibility, and powerful features continues to drive its widespread adoption and popularity in the software development community.

### **Why Python**

Python is favored by developers for several reasons, including its simplicity, versatility, extensive libraries, and vibrant community support.

One of Python's most appealing aspects is its readability and ease of use. Python's syntax is designed to be clear and concise, making it accessible to beginners and experienced programmers alike. Its simple and straightforward syntax reduces the time required for development and debugging, enabling developers to focus more on solving problems rather than dealing with complex syntax rules.

Python's versatility is another major factor driving its popularity. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, giving developers the flexibility to choose the most suitable



approach for their projects. Additionally, Python's cross-platform compatibility allows code to run seamlessly on various operating systems, further enhancing its versatility and practicality.

Furthermore, Python boasts an extensive standard library, providing a wide range of modules and functions for tasks such as file I/O, networking, database access, and more. This rich set of built-in tools accelerates development by eliminating the need to reinvent the wheel and simplifying common programming tasks.

Python's dynamic typing and automatic memory management contribute to its ease of use and productivity. Unlike statically-typed languages, Python does not require variable declarations, allowing developers to focus on solving problems rather than managing data types. Additionally, Python's garbage collection mechanism automatically deallocates memory when objects are no longer in use, reducing the risk of memory leaks and other memory-related issues.

Moreover, Python boasts a vibrant and supportive community of developers, educators, and enthusiasts who actively contribute to its growth and development. The Python community is known for its collaborative spirit, extensive documentation, and wealth of online resources, including tutorials, forums, and user groups. This thriving community fosters knowledge sharing, innovation, and continuous improvement, making Python an attractive choice for developers seeking community support and collaboration.

Overall, Python's simplicity, versatility, extensive libraries, dynamic typing, automatic memory management, and vibrant community support make it a compelling choice for a wide range of applications, from web development and data analysis to scientific computing and artificial intelligence. Its combination of ease of use, flexibility, and powerful features continues to drive its widespread adoption and popularity in the software development community.

### **The Python Programming Language**

Python is a high-level programming language known for its simplicity and readability. It's widely used for various purposes including web development, data analysis, artificial intelligence, and scientific computing. One of the key features of Python is its straightforward syntax, which makes it accessible to beginners and experts alike.

Python code is executed line by line, which makes it easy to understand and debug. Let's take a look at some basic Python code snippets:

```
```python
# Example 1: Hello World
print("Hello, World!")

# Example 2: Variables and Data Types
x = 5
y = "Hello"
print(x)
print(y)

# Example 3: Basic Arithmetic Operations
a = 10
b = 5
sum = a + b
difference = a - b
product = a * b
quotient = a / b
print("Sum:", sum)
print("Difference:", difference)
print("Product:", product)
print("Quotient:", quotient)

# Example 4: Control Structures - if-else
num = 10
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")

# Example 5: Loops - for loop
fruits = ["apple", "banana", "cherry"]
```

```
for fruit in fruits:
    print(fruit)

# Example 6: Functions
def greet(name):
    print("Hello, " + name)

greet("Alice")
greet("Bob")

# Example 7: Lists
my_list = [1, 2, 3, 4, 5]
print("Length of list:", len(my_list))
print("First element:", my_list[0])
print("Last element:", my_list[-1])
...

```

These examples cover some of the fundamental aspects of Python, including printing to the console, variable declaration, basic arithmetic operations, control structures (if-else), loops (for loop), defining functions, and working with lists. Python's versatility and simplicity make it an excellent choice for both beginners and experienced programmers alike.

### **The Python Platform**

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware.

Python is a versatile programming language that can be executed on various platforms, including Windows, macOS, and Linux. It's an interpreted language, meaning that Python code is executed by an interpreter, which converts the code into machine-readable instructions.

Python is supported by a wide range of integrated development environments (IDEs), text editors, and online platforms. Some popular Python development environments include:

1. PyCharm: Developed by JetBrains, PyCharm is a powerful IDE with features like code completion, debugging, and version control integration.

2. Visual Studio Code (VS Code): A lightweight and extensible code editor from Microsoft, VS Code offers excellent support for Python development through extensions.

3. Jupyter Notebook: Ideal for data analysis and scientific computing, Jupyter Notebook allows you to create and share documents containing live code, equations, visualizations, and narrative text.

4. Spyder: This Python IDE is designed for scientific computing and provides features such as a variable explorer, a profiler, and integration with scientific libraries like NumPy and Matplotlib.

5. IDLE: IDLE comes bundled with the Python installation and offers a simple integrated development environment for writing and executing Python code.

In addition to these development environments, Python code can also be executed directly from the command line or terminal using the Python interpreter. This provides a quick and easy way to run Python scripts without the need for a dedicated IDE or text editor.

Python's platform-independent nature, combined with its extensive standard library and vast ecosystem of third-party packages, makes it a popular choice for a wide range of applications, from web development and automation to machine learning and artificial intelligence.

### **Install the Python Desktop IDE**

We do have many operating systems. Namely:

- Windows
- Mac OS X
- Linux
- Portable IDE (Windows and Linux)
- ChromeOS for Individuals and for Education

Choose the operating system and start the procedure of installing an Python IDE. We are using Windows 10 operating system.

### **Install the Arduino Software (IDE) on Windows PCs**

The Python installation has two steps:

- Download the Python Software (IDE).
- Proceed with CLI specific instructions.

### **Download the Python Software (IDE).**

First, go to the Python [website](#) and click on the “Download Python” button. Select the latest version. As of this writing, that would be Python 3.11.4.



**Fig 3.1 Python for windows**

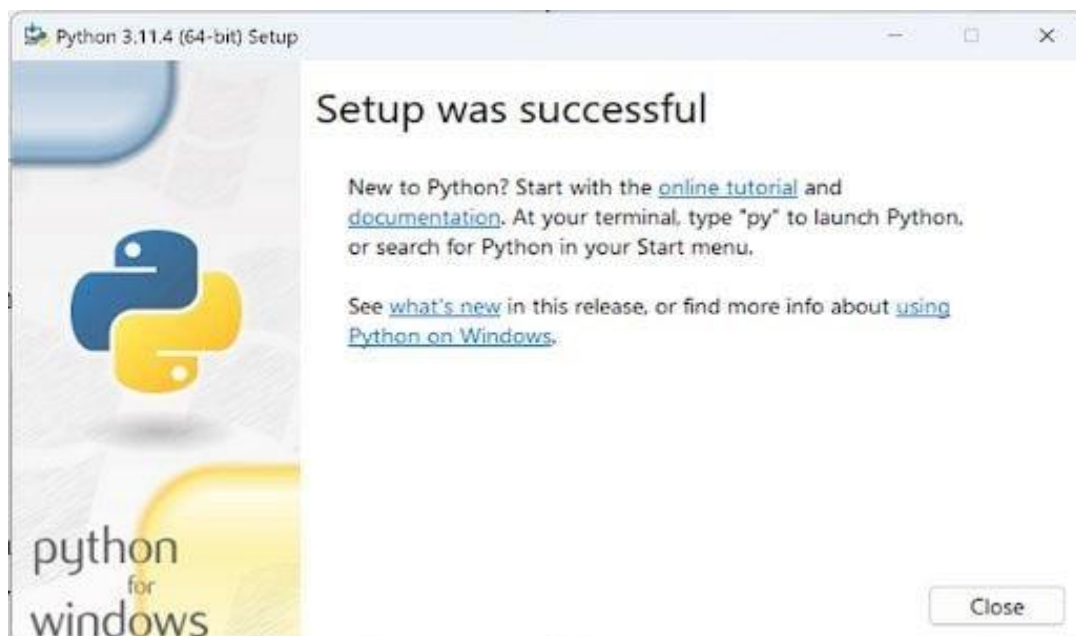
Select the “Windows installer” option and click on the “Download” button. Once the download is complete, run the installer.



**Fig 3.2 Installation of python**

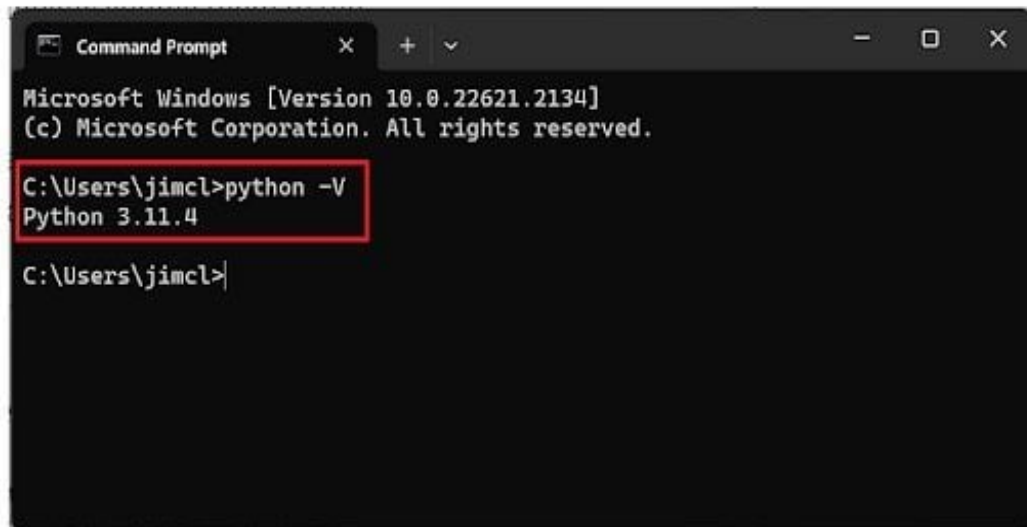
During the process, it's advisable to check the boxes "Use admin privileges when installing py.exe" and "Add python.exe to PATH" to save the trouble of manual adjustments in the environment variable later.

Then, follow the on-screen instructions to install Python. Once the installation is complete, you should see a success message like this:



**Fig 3.3 Python setup**

You can verify that Python has been installed by opening a command prompt and typing the following command: `python -V`.



```
Command Prompt
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jimcl>python -V
Python 3.11.4

C:\Users\jimcl>
```

**Fig 3.4 Python in terminal**

This will display the Python version currently installed.

The advantage of installing Python via an installer file is that it is easy to use but requires a few more clicks than installing Python via the command-line interface (CLI).

## WHAT IS AN IDE?

IDE stands for Integrated Development Environment, which is a type of software that combines many common software developer tools in one single, user-friendly interface. On top of the common features of a code editor, IDEs typically include additional features such as code autocompletion and debugging.

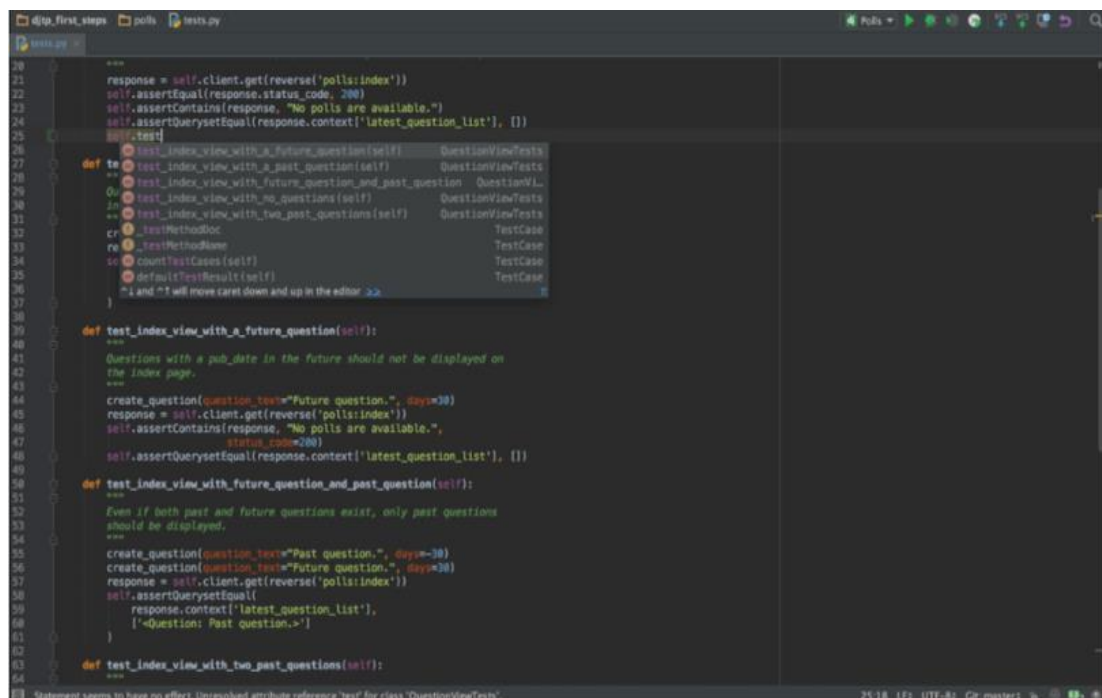
An IDE is a comprehensive tool that you can use throughout the complete process of software development. IDEs provide code building, editing, testing and running capabilities into one central application, so you rarely have to use other programs. IDEs can also include features like syntax highlighting, code autocompletion and more to improve workflows.

A code editor is typically just a software application used for editing source code, which means that you will likely need some other program to compile/interpret the code and run it. This does give the code editor some advantages though. It can be

more lightweight and customizable through plugins and add-ons. A code editor can also be utilized as an individual application or implemented as part of an IDE application.

Here, I have gathered a list of the most widely used and popular Python IDEs and code editors. They aren't necessarily in best-to-worst order. Instead, I give each a short description to make it easy for you to choose the one that suits your needs best!

## 1. PYCHARM (IDE)



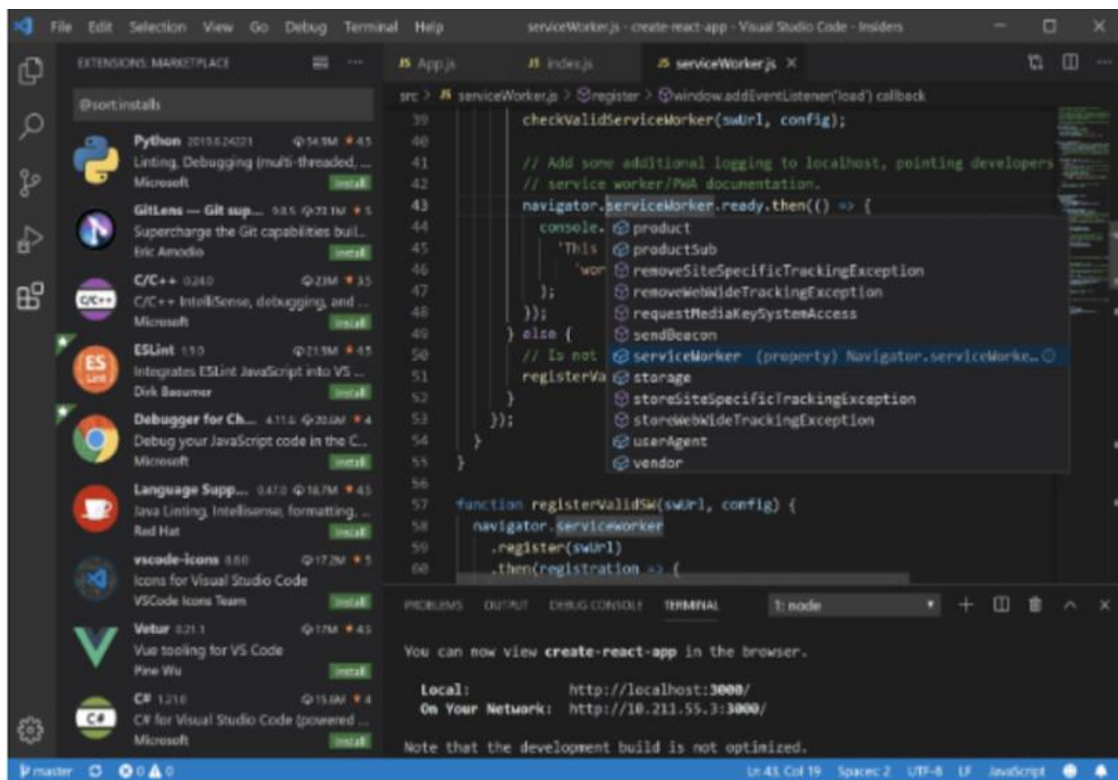


The professional version comes with a couple of added advanced features. These include database management and Python web frameworks like Django, Google App Engine, and Pyramid.

The downsides of PyCharm come from its comprehensiveness. It has somewhat long loading times, and you might have to tweak some settings to run existing projects.

All in all, PyCharm is a very good choice for anyone looking for a comprehensive Python development tool. It helps with the quality of the code you write and boosts your efficiency.

## 2. VISUAL STUDIO CODE (CODE EDITOR)

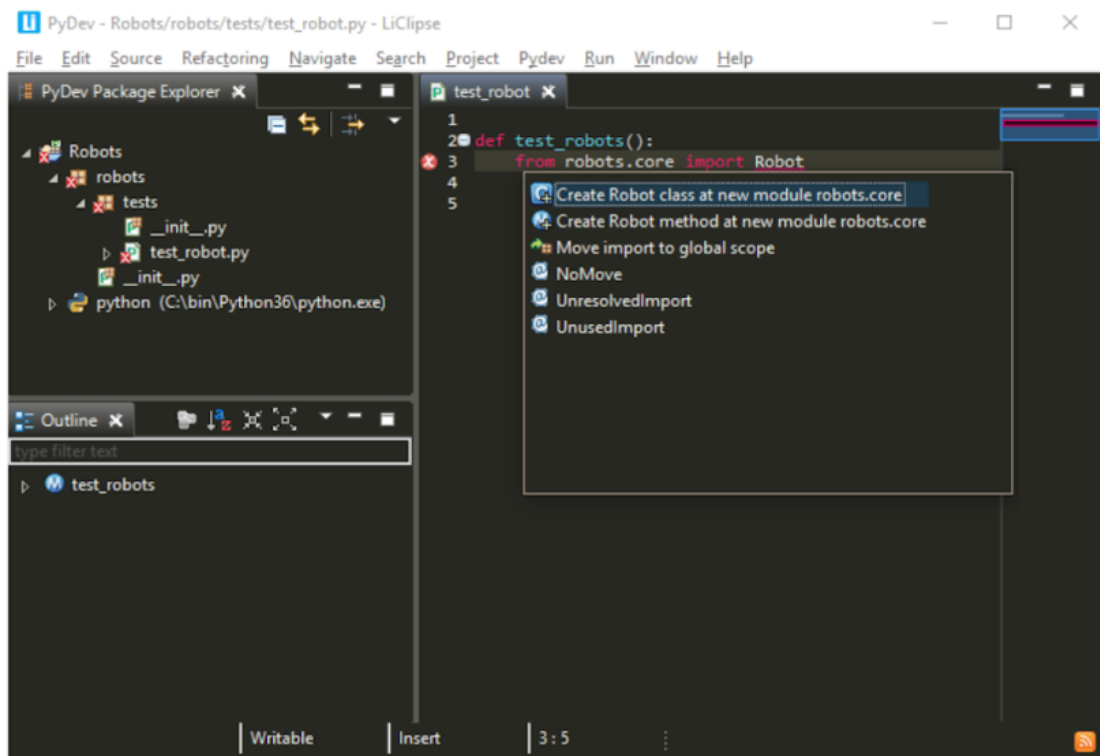


**Fig 3.6 Visual Studio Code**

Visual Studio Code (or VS Code for short) is a free and open-source code editor developed by Microsoft. It supports many programming languages, including Python, through an extension. It is relatively lightweight and comes with many useful features like syntax highlighting, very good code autocompletion, Git integration, and advanced code debugging. Visual Studio Code is often confused with Visual Studio. These are different programs, however. VS Code is also very customizable through a huge number

of extensions. It is a good choice if you are looking for a lightweight, fast, and customizable code editor for Python.

### 3. PYDEV (IDE)



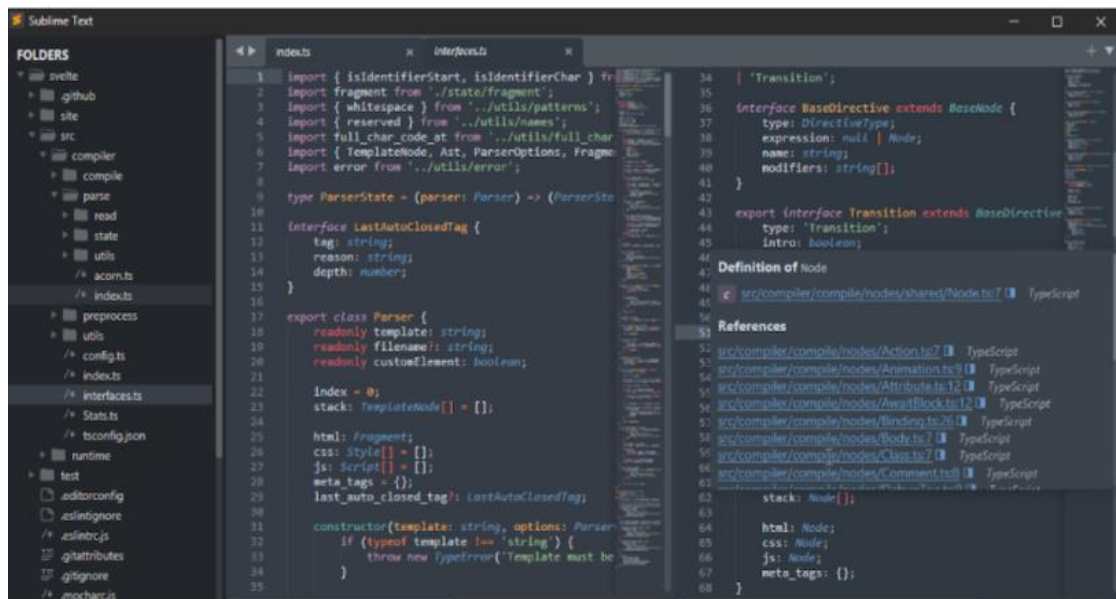
**Fig 3.7 Pydev**

Eclipse is a popular IDE designed for software development in Java. Through an extension, however, you can use it for other languages as well, including Python. PyDev is the plugin that allows you to use Eclipse as a Python IDE. Eclipse and PyDev are both free and open-source.

Notable features include syntax highlighting, code autocompletion, refactoring, debugging, code analysis, interactive console, and good support for Python web development.

Eclipse + PyDev is a good choice if you are looking for a complete IDE that is free and open-source. This combination is also great if you need to work on multiple languages. You can use the same IDE for the different languages and have robust features for all of them.

#### 4. SUBLIME TEXT (CODE EDITOR)

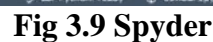


**Fig 3.8 Sublime Text**

Sublime Text is a popular code editor with support for multiple programming languages, including Python. According to the site; “Text may be downloaded and evaluated for free, however, a license must be purchased for continued use. There is currently no enforced time limit for the evaluation.”

Features of Sublime Text include high customizability through plugins, speed, minimal, discreet and powerful user interface, syntax highlighting, code auto-completion, and powerful text editing features.

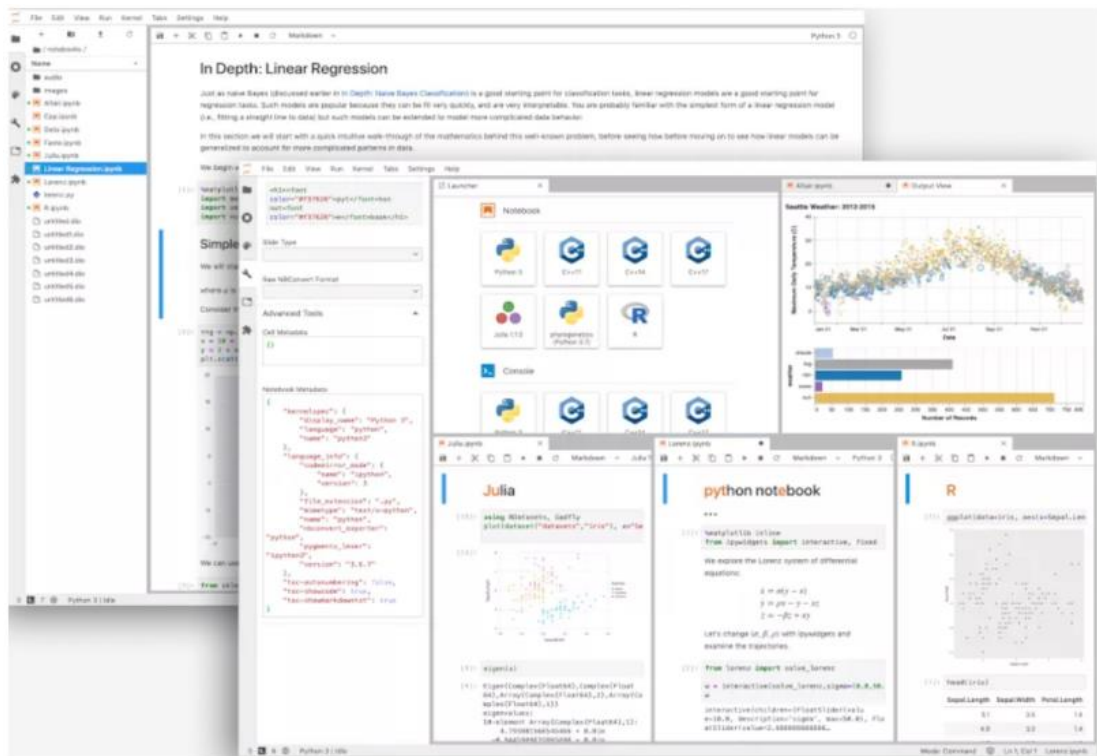
Sublime Text can be a good choice for you if you are looking for a lightweight code editor that you can customize, and that also has a minimal and powerful interface and text editing features.



Features of Spyder include syntax highlighting, code autocompletion, debugging, code analysis, interactive console, plotting all kinds of charts and graphs, data manipulation, and integration of many data science libraries such as NumPy, Pandas, Scipy, and Matplotlib. Spyder also has great community support.

Spyder is a comprehensive IDE that especially shines in the fields of machine learning or data science.

## 6. JUPYTER NOTEBOOK



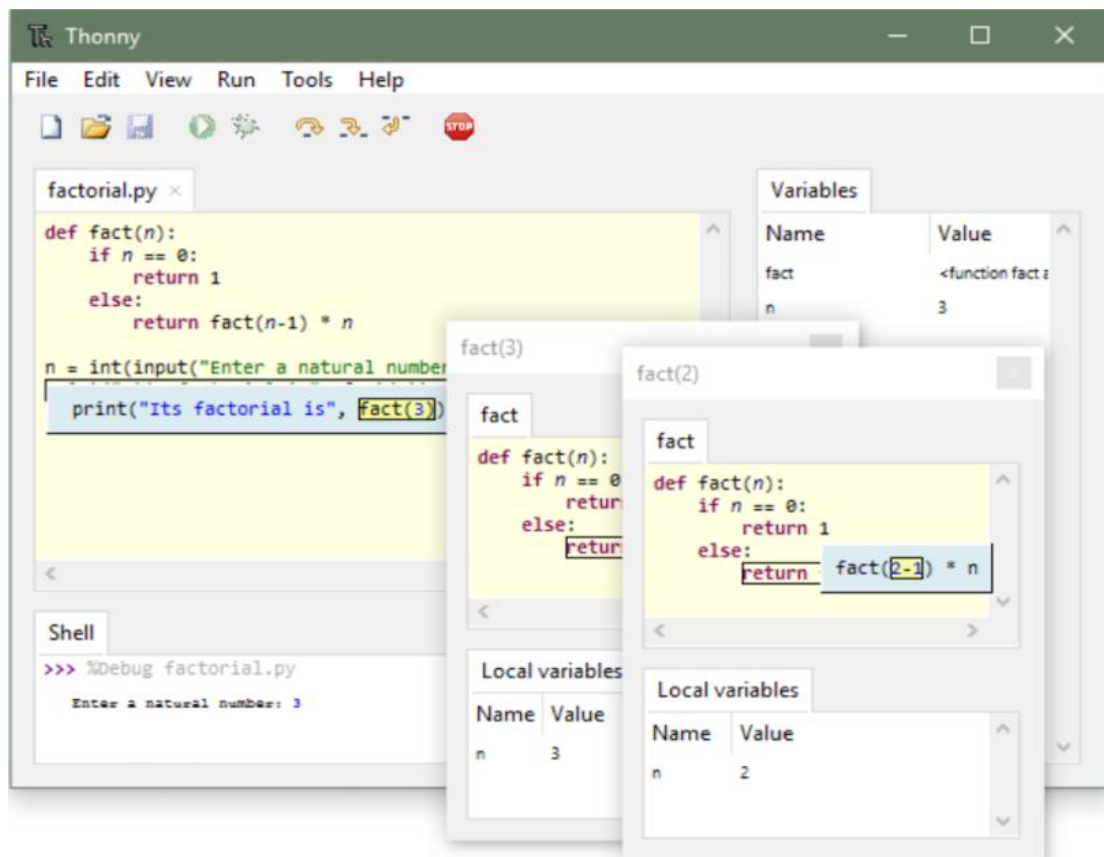
**Fig 3.10 Jupyter Notebook**

Jupyter Notebook is a web-based interactive development environment. Therefore, you can access it from almost anywhere, from any computer, and continue working on your project as long as you have an internet connection. For this same reason, it is also a great tool for presenting information and sharing your work.

Jupyter Notebook has support for multiple programming languages, including Python. It is also easy to use and open source.

Jupyter Notebook is well known in the data science community for analyzing, sharing, and presenting information. It is great for machine learning, simulation, and data science applications as well as visualizations. Jupyter Notebook is also great if you just need a quick tool to run some Python since you don't need to set up anything on your computer.

## 7. THONNY (IDE)



**Fig 3.11 Thonny**

Thonny is a free and open-source Python IDE with an educational focus and aimed at beginners.

The benefit is mainly ease of use. Installation should be easy and the interface should be clear and easy to navigate. It has syntax highlighting, simple code completion, and a simple debugger. It also shows you the variables saved in memory, so you can easily keep track of how the code you are running affects them.

Thonny is a solid choice if you are just starting to learn Python and want to gain insight into basic Python operations.



## PYTHON MODULES

### 1. OpenCV (`cv2``):

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a wide range of functionalities for image and video analysis, including face detection and recognition. Installation via pip: ``pip install opencv-python``

### 2. Flask:

Flask is a lightweight WSGI web application framework in Python. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. Installation via pip: ``pip install Flask``

### 3. Scikit-learn:

Scikit-learn is a machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It includes various algorithms for classification, regression, clustering, dimensionality reduction, and more. Installation via pip: ``pip install scikit-learn``

### 4. Pandas:

Pandas is a fast, powerful, and flexible data analysis and manipulation library built on top of the Python programming language. It provides data structures and functions to efficiently manipulate large datasets. Installation via pip: ``pip install pandas``

### 5. NumPy:

NumPy is a fundamental package for scientific computing in Python. It provides support for multi-dimensional arrays and matrices, along with a large collection of mathematical functions to operate on these arrays. Installation via pip: ``pip install numpy``

### 6. Joblib:

Joblib is a set of tools to provide lightweight pipelining in Python. It is particularly useful for efficiently saving and loading Python objects (especially large arrays) to disk. Installation via pip: ``pip install joblib``

## MODULES EXPLANATION

### 1. cv2 (OpenCV):

OpenCV is a library of programming functions mainly aimed at real-time computer vision. It includes algorithms for face detection, image processing, object detection, and more. In the code, it's used for face detection, face recognition, and image capture.

### 2. os:

The `os` module provides a way to interact with the operating system. It's used in the code to perform operations such as directory creation, listing files in directories, and removing files and directories.

### 3. Flask:

Flask is a micro web framework written in Python. It provides tools, libraries, and technologies for building web applications. In the code, Flask is used to create web routes, handle HTTP requests, and render HTML templates.

### 4. request:

The `request` module is part of Flask and is used to handle HTTP requests sent by the client. It allows access to form data, file uploads, cookies, and other request data.

### 5. render\_template:

`render_template` is a function provided by Flask to render HTML templates. It allows generating HTML content dynamically by combining static HTML with data from Python code.

### 6. date, datetime:

These modules provide functions for working with dates and times. They're used in the code to get the current date and time for recording attendance.

### 7. numpy:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices. In the code, it's used for numerical computations and data manipulation.

### 8. sklearn.neighbors.KNeighborsClassifier:

This is a machine learning model provided by scikit-learn. It's a classification algorithm based on k-nearest neighbors. In the code, it's used for face recognition after training on images of registered users.

### 9. pandas:

Pandas is a library for data manipulation and analysis. It provides data structures like



DataFrames, which are used to store and manipulate tabular data efficiently. In the code, it's used to read and write attendance data from/to CSV files.

#### 10. joblib:

Joblib is a set of tools to provide lightweight pipelining in Python. It's used in the code to save and load the trained machine learning model for face recognition.

These modules and libraries enable the functionality of the face recognition-based attendance management system implemented in the provided code. They provide tools for image processing, web development, machine learning, data manipulation, and more.

## **FLASK**

Flask is a micro web framework for Python based on Werkzeug, Jinja2, and other libraries. It's designed to be lightweight, flexible, and easy to use, making it a popular choice for developing web applications and APIs in Python.

Here's an overview of Flask and its key features:

#### 1. Minimalism:

- Flask is known for its minimalistic approach. It provides only the essential tools needed to build a web application, allowing developers to add features as needed.

#### 2. Routing:

- Flask uses routes to define how URLs correspond to functions within the application.
- Routes are defined using decorators (`@app.route('/path')`) and specify which function should be executed when a particular URL is accessed.

#### 3. HTTP Request Handling:

- Flask provides built-in support for handling various HTTP methods like GET, POST, PUT, DELETE, etc.
- It offers request and response objects to access data sent by the client and send data back to the client.

#### 4. Template Rendering:

- Flask uses Jinja2 templating engine for rendering HTML templates.
- Templates allow developers to generate HTML dynamically by combining static HTML content with data from Python code.

#### 5. Extensibility:

- Flask is highly extensible and allows integration with various extensions to add functionality such as user authentication, database integration, and more.
- Extensions are available for common tasks, reducing the need to reinvent the wheel.

#### 6. Development Server:

- Flask comes with a built-in development server that can be used to run the

application during development.

- The development server automatically reloads the application when code changes are detected, making the development process efficient.

#### 7. URL Building:

- Flask provides URL building capabilities to generate URLs dynamically based on route names and parameters.
- This helps in maintaining consistency and flexibility in URL structure across the application.

#### 8. Testing Support:

- Flask includes support for testing web applications using the Werkzeug test client.
- Developers can write unit tests and integration tests to ensure the correctness and reliability of their applications.

### Typical Usage:

#### - Web Development:

Flask is commonly used for developing web applications, ranging from simple websites to complex web platforms.

#### - API Development:

Flask can be used to build RESTful APIs (Application Programming Interfaces) for serving data and interacting with other services.

#### - Prototyping:

Its simplicity makes Flask ideal for prototyping and building minimal viable products (MVPs) quickly.

#### - Microservices:

Flask's lightweight nature makes it suitable for developing microservices that work together to build larger systems.

Example:

```
```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello, World!'

@app.route('/hello/<name>')
def hello(name):
    return render_template('hello.html', name=name)

if __name__ == '__main__':
    app.run(debug=True)
```
```

In this example:

- We import Flask and create an instance of the Flask class.

- We define routes using the `@app.route`` decorator.
- The ``index`` function is executed when the root URL ``/`` is accessed.
- The ``hello`` function is executed when a URL like ``/hello/John`` is accessed, and it renders an HTML template (``hello.html``) with a dynamic parameter (``name``).

Overall, Flask provides a flexible and easy-to-use framework for building web applications and APIs in Python, making it a popular choice among developers.

## PYTHON

Python is a cross-platform programming language, which means that it can run on multiple platforms like Windows, macOS and Linux.

Even though most of today's Linux and Mac have Python pre-installed in it, the version might be out-of-date. So, it is always a good idea to install the most current version.

### The Easiest Way to Run Python

The easiest way to run Python is by using **Thonny IDE**.

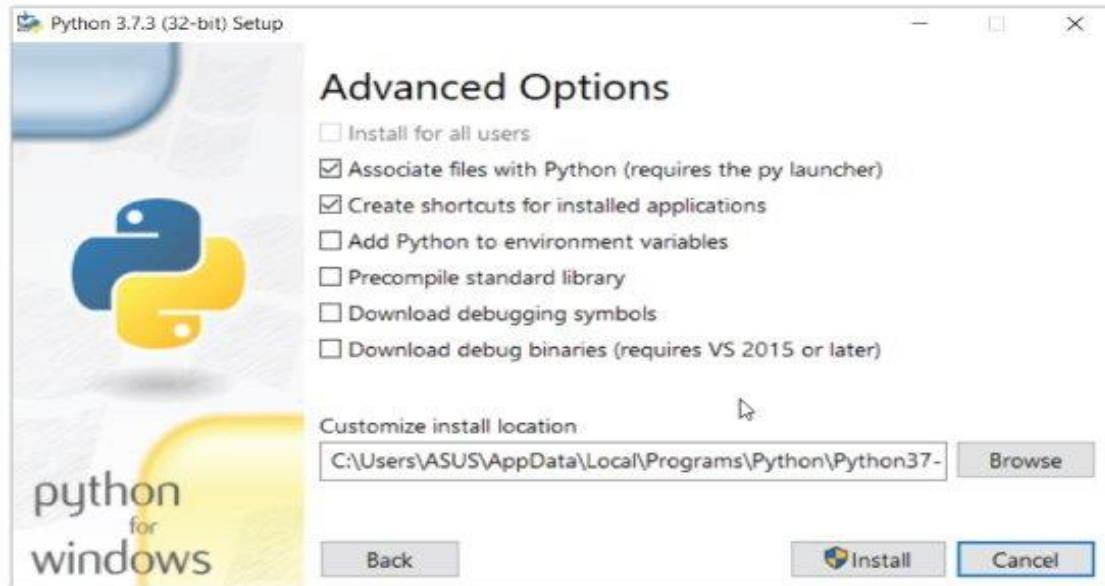
The easiest way to run Python is by using **Thonny IDE**.

The Thonny IDE comes with the latest version of Python bundled in it. So you don't have to install Python separately.

Follow the following steps to run Python on your computer.

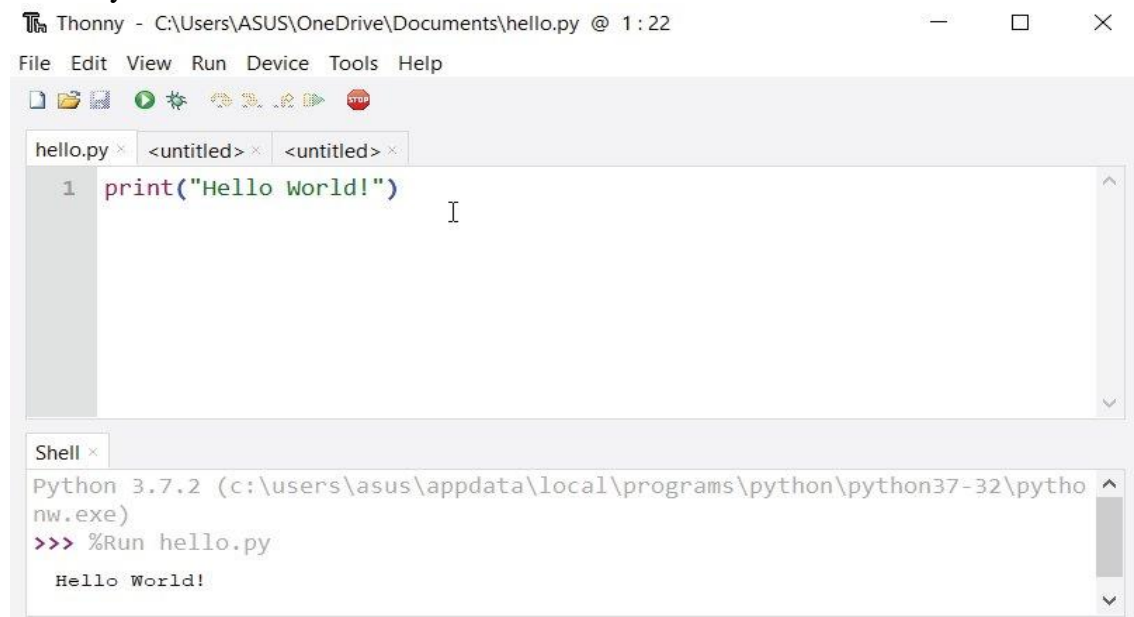
1. Download [Thonny IDE](#).
2. Run the installer to install **Thonny** on your computer.
3. Go to: **File > New**. Then save the file with `.py` extension. For example, `hello.py`, `example.py`, etc.

You can give any name to the file. However, the file name should end with **.py**



**Fig 3.12 Customize install location**

Write Python code in the file and save it.



**Fig 3.13 Running Python using Thonny IDE**

4. Then Go to **Run > Run current script** or simply click **F5** to run it.

### Install Python Separately

If you don't want to use Thonny, here's how you can install and run Python on your computer.

1. Download the [latest version of Python](#).

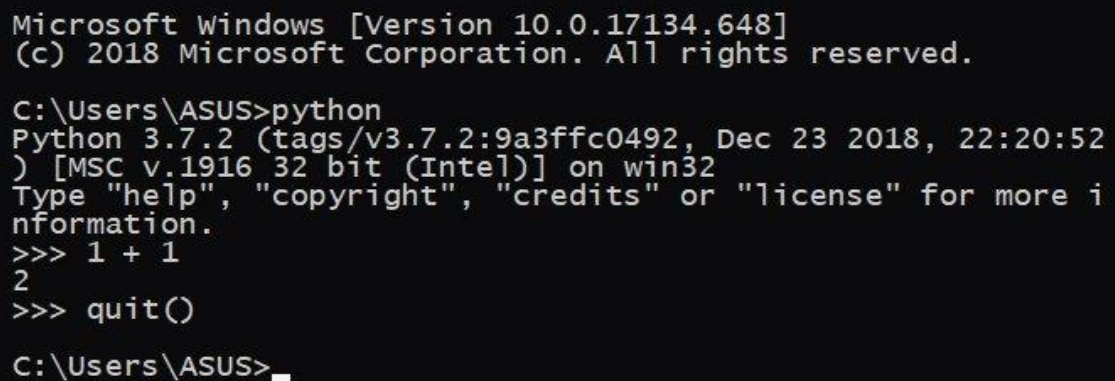
2. Run the installer file and follow the steps to install Python. During the install process, check **Add Python to environment variables**. This will add Python to environment variables, and you can run Python from any part of the computer. Also, you can choose the path where Python is installed. Installing Python on the computer

Once you finish the installation process, you can run Python.

### 1. Run Python in Immediate mode

Once Python is installed, typing `python` in the command line will invoke the interpreter in immediate mode. We can directly type in Python code, and press Enter to get the output.

Try typing in `1 + 1` and press enter. We get `2` as the output. This prompt can be used as a calculator. To exit this mode, type `quit()` and press enter.



```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\ASUS>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more
>>> 1 + 1
2
>>> quit()
C:\Users\ASUS>
```

**Fig 3.14 Simple program in terminal**

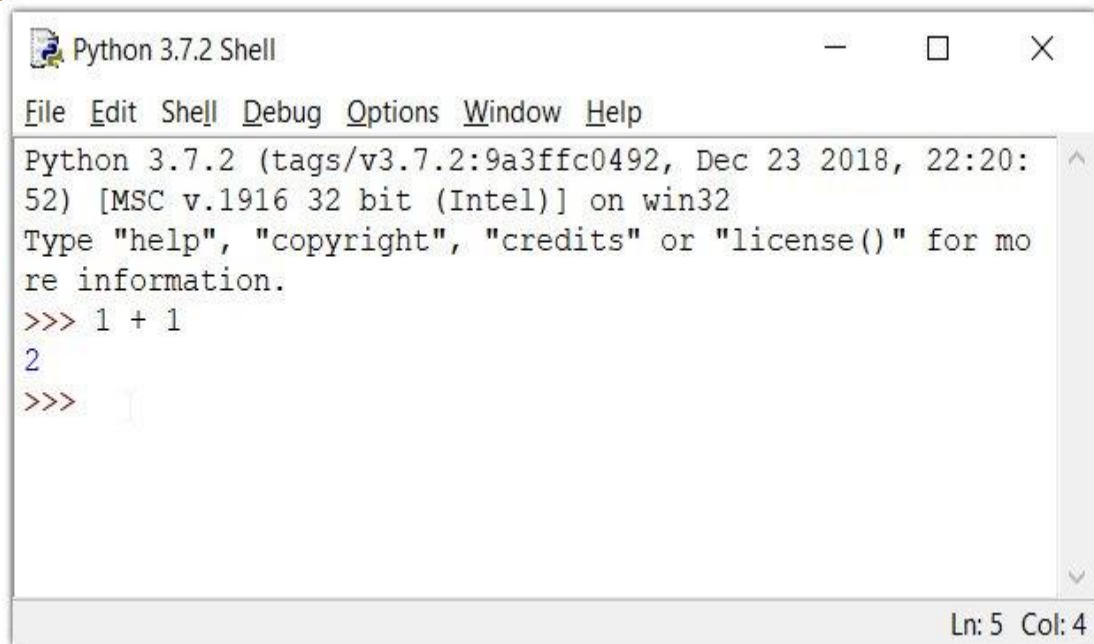
### 2. Run Python in the Integrated Development Environment (IDE)

We can use any text editing software to write a Python script file.

We just need to save it with the `.py` extension. But using an IDE can make our life a lot easier. IDE is a piece of software that provides useful features like code hinting, syntax highlighting and checking, file explorers, etc. to the programmer for application development.

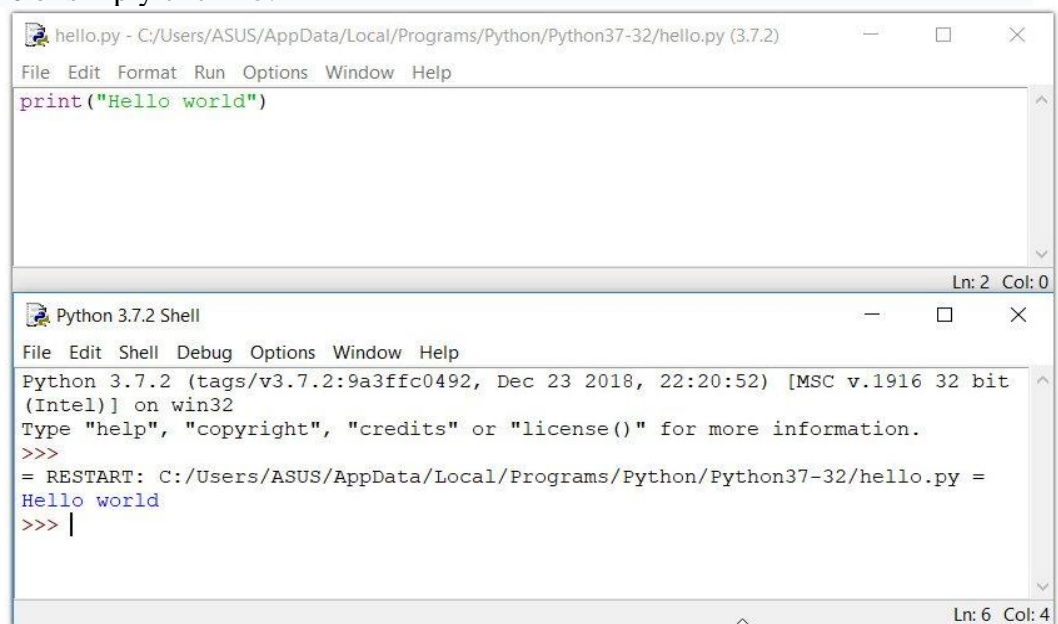
By the way, when you install Python, an IDE named **IDLE** is also installed. You can use it to run Python on your computer. It's a decent IDE for beginners.

When you open IDLE, an interactive Python Shell is opened.

**Fig 3.15 Python IDLE**

Now you can create a new file and save it with **.py** extension. For example, **hello.py**

Write Python code in the file and save it. To run the file, go to **Run > Run Module** or simply click **F5**.

**Fig 3.16 Running a Python program in IDLE**

The provided code is a Python Flask application for a face recognition-based attendance management system. Several programming concepts and techniques are involved in its implementation. Let's break down the concepts used in the code:

1. Web Development with Flask:

- Flask is a micro web framework for Python used to build web applications.
- Routes (`@app.route()`) are defined to handle different URLs and HTTP methods.
- Rendering HTML templates (`render_template()`) is used to generate dynamic web pages.

2. File and Directory Operations:

- The `os` module is used to perform file and directory operations, such as creating directories, checking for file existence, listing directory contents, and deleting files and directories.

3. Image Processing with OpenCV:

- OpenCV (`cv2`) library is utilized for image processing tasks such as face detection and recognition.
- Functions like `cv2.CascadeClassifier()` and `detectMultiScale()` are used for face detection.
- Image capture and resizing are performed using OpenCV functions like `cv2.VideoCapture()` and `cv2.resize()`.

4. Machine Learning (K-nearest Neighbors) :

- Scikit-learn library (`sklearn`) is used for machine learning tasks.
- The `KNeighborsClassifier` algorithm is employed for face recognition.
- Model training (`fit()`) and prediction (`predict()`) are performed using scikit-learn functions.

5. Data Manipulation with Pandas:

- Pandas library (`pandas`) is used for reading and writing data to CSV files and manipulating DataFrame objects.
- Functions like `pd.read_csv()` and DataFrame manipulation are employed to extract attendance information and manage user data.

6. Date and Time Handling:

- The `date` and `datetime` modules are used to handle dates and times.
- Functions like `date.today()` and `datetime.now()` are used to get the current date and time.

7. HTML Templating:

- Flask's `render_template()` function is used to render HTML templates.
- HTML templates are used to generate dynamic web pages with placeholders for dynamic content.

8. Error Handling:

- Basic error handling is implemented using `try-except` blocks to catch exceptions and handle errors gracefully.

9. Web Forms and User Input:

- HTML forms are used to capture user input (e.g., new user details).
- Flask's request object (`request`) is used to access form data submitted by the user.

10. Control Flow:

- Control flow structures like `if`, `for`, and `while` loops are used to manage the flow of execution based on conditions and iterate over data.

11. Function Definitions:

- Functions are defined to encapsulate reusable code and improve code organization and readability.
- Functions are used for tasks like face extraction, model training, attendance recording, etc.

12. Dynamic Content Rendering:

- Dynamic content is rendered in HTML templates using placeholders and variables passed from the Python code



## **4.FEASIBILITY STUDY**

Feasibility study is the initial design stage of any project, which brings together the elements of knowledge that indicate if a project is possible or not. A feasibility study includes an estimate of the level of expertise required for a project and who can provide it, quantitative and qualitative assessments of other essential resources, identification of critical points, a general timetable, and a general cost estimate. Whether a project is viable or not, i.e. whether it can generate an equal or a higher rate of return during its lifetime requires a thorough investigation of the investment per se as well as the level of current expenditure. The preliminary design is the simple description of the conceived idea with an indication of the main factors to be considered in the study.

The feasibility of a proposed solution is evaluated in terms of its components. These components are

- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY
- ECONOMICAL FEASIBILITY
- OPERATIONAL FEASIBILITY

### **4.1 TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### **4.2 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to

educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

#### **4.3 ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

#### **4.4 OPERATION FEASIBILITY**

The purpose of this project is to develop software named Criminal Activity Detection in Social Network which facilitates quick allocation process. The activities of the system such as data entry, information retrieval, updating and deletion of records from various tables etc. are made easily. All the operations of this project are trained in this area, so this project is operationally feasible.

## **5.SYSTEM ANALYSIS**

### **5.1 EXISTING SYSTEM**

Manual Attendance Tracking:

- In the existing system, attendance is typically tracked manually using paper-based methods or basic electronic systems like swipe cards or biometric scanners.
- Time-Consuming Process:
  - Manual attendance tracking is time-consuming and prone to errors, especially in large organizations.
- Limited Security:
  - Basic electronic systems may lack robust security measures, making them susceptible to fraudulent activities such as buddy punching.
- Lack of Real-Time Insights:
  - The existing system often lacks real-time reporting capabilities, making it challenging for administrators to monitor attendance trends and patterns effectively.

#### **5.1.1 DRAWBACKS OF EXISTING SYSTEM**

- Inefficiency:
  - Manual attendance tracking is inefficient and labor-intensive, requiring significant time and effort from administrators.
- Error-Prone:
  - Manual data entry increases the risk of errors, leading to inaccuracies in attendance records.
- Security Risks:
  - Basic electronic systems may be vulnerable to security breaches and unauthorized access.
- Limited Scalability:
  - The existing system may struggle to scale with the growing needs of the organization, leading to operational challenges.

### **5.2 PROPOSED SYSTEM**

- Automated Attendance Management:

- The proposed system introduces a face recognition-based attendance management system, eliminating the need for manual data entry.
- Real-Time Tracking:
  - The system captures attendance data in real-time, providing administrators with immediate insights into attendance patterns.
- Enhanced Security:
  - Face recognition technology offers enhanced security features, reducing the risk of fraudulent activities such as buddy punching.
- Scalability:
  - The proposed system is scalable and can accommodate the needs of organizations of all sizes, from small businesses to large enterprises.

### **5.2.1 ADVANTAGES OF PROPOSED SYSTEM**

- Efficiency:
  - Automated attendance tracking improves efficiency by eliminating manual processes and reducing administrative burden.
- Accuracy:
  - Face recognition technology ensures accurate attendance tracking, minimizing errors and discrepancies.
- Security:
  - Enhanced security features, such as biometric authentication, enhance data security and prevent unauthorized access.
- Real-Time Insights:
  - Real-time reporting capabilities provide administrators with actionable insights into attendance trends and patterns, facilitating better decision-making.

## **6.SYSTEM DESIGN**

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development, then design is the act of taking the marketing information and creating the design of the product to be manufactured.

Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

### **6.1 UML DIAGRAMS**

The Unified Modelling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things

simpler, has exacerbated these architectural problems. The Unified Modelling Language (UML) was designed to respond to these needs. Simply, Systems design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

### Contents of UML

In general, a UML diagram consists of the following features:

- **Entities:** These may be classes, objects, users or systems behaviors.
- Relationship Lines that model the relationships between entities in the system.
- **Generalization** -- a solid line with an arrow that points to a higher abstraction of the present item.
- **Association** -- a solid line that represents that one entity uses another entity as part of its behaviour.
- **Dependency** -- a dotted line with an arrowhead that shows one entity depends on the behaviour of another entity.

In this project four basic UML diagrams have been explained

- 1) Class Diagram
- 2) Use Case Diagram
- 3) Sequence Diagram
- 4) Activity Diagram
- 5) Deployment Diagram

#### 6.1.1 CLASS DIAGRAM

UML class diagrams model static class relationships that represent the fundamental architecture of the system. Note that these diagrams describe the relationships between classes, not those between specific objects instantiated from those classes. Thus the diagram applies to all the objects in the system.

A class diagram consists of the following features:

- **Classes:** These titled boxes represent the classes in the system and contain information about the name of the class, fields, methods and access specifies. Abstract roles of the Class in the system can also be indicated.
- **Interfaces:** These titled boxes represent interfaces in the system and contain information about the name of the interface and its methods. Relationship Lines that model the relationships between classes and interfaces in the system.
- **Dependency:** A dotted line with an open arrowhead that shows one entity depends on the behavior of another entity. Typical usages are to represent that one class instantiates another or that it uses the other as an input parameter
- **Aggregation:** Represented by an association line with a hollow diamond at the tail end. An aggregation models the notion that one object uses another object without "owning" it and thus is not responsible for its creation or destruction.
- **Inheritance:** A solid line with a solid arrowhead that points from a sub-class to a super class or from a sub-interface to its super-interface.
- **Implementation:** A dotted line with a solid arrowhead that points from a class to the interface that it implements
- **Composition:** Represented by an association line with a solid diamond at the tail end. A composition models the notion of one object "owning" another and thus being responsible for the creation and destruction of another object.

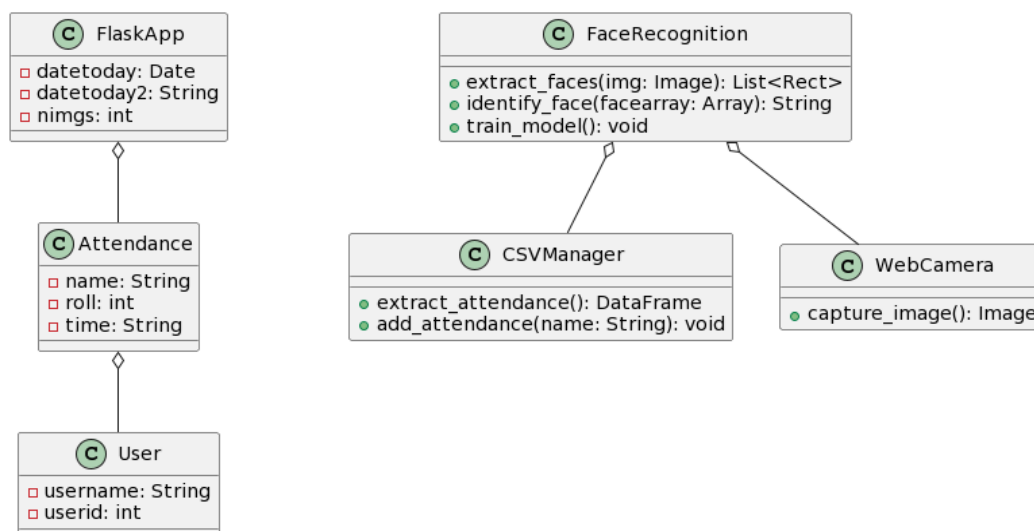


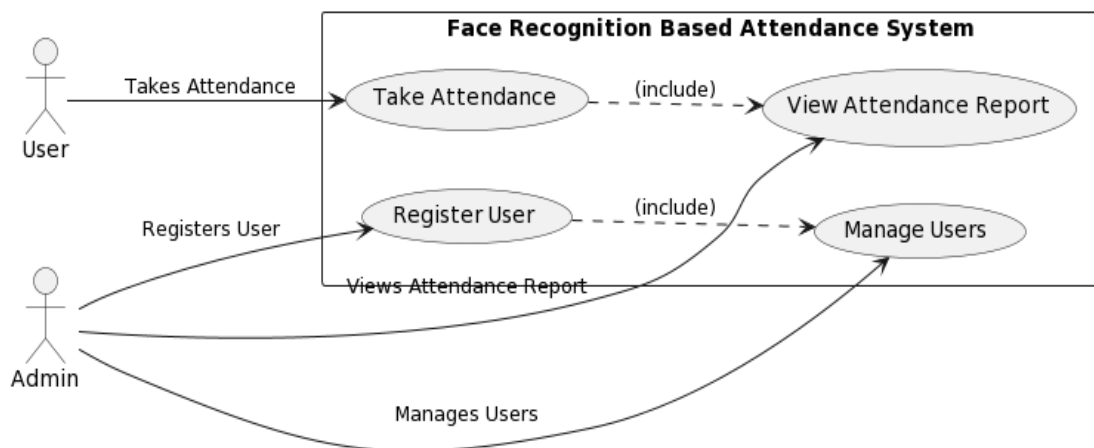
Fig 6.1: Class Diagram

### 6.1.2 USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms.

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**Fig 6.2: Use case Diagram**

#### ➤ Parts of Use cases

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.



➤ **Actors**

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

➤ **System boundary boxes (optional)**

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not **Relationships.**

➤ **Include**

In one form of interaction, a given use case may include another. “Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case”.

The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviours from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". This usage resembles a macro expansion where the included use case behavior is placed inline in the base use case behavior. There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.

➤ **Extend**

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". The notes or constraints may be associated with this relationship to illustrate the conditions under which this behavior will be executed. Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case. Depending on the modeler's approach "optional" may mean "potentially not executed with the base use case" or it may mean "not required to achieve the base use case goal".

➤ **Generalization**

In the third form of relationship among use cases, a generalization/

specialization relationship exists. A given use case may have common behaviours, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation).

➤ **Associations**

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modelled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicate the direction of the initial invocation of the relationship or to indicate the primary actor within the use case. The arrowheads imply control flow and should not be confused with data flow.

➤ **STEPS TO DRAW USE CASES**

- Identifying Actor
- Identifying Use cases
- Review your use case for completeness

### **6.1.3 SEQUENCE DIAGRAM**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behaviour of a system.

➤ **Elements of sequence diagram**

The sequence diagram is an element that is used primarily to showcase the interaction that occurs between multiple objects. This interaction will be shown over certain period of time. Because of this, the first symbol that is used is one that symbolizes the object.

➤ **Lifeline**

A lifeline will generally be generated, and it is a dashed line that sits vertically, and the top will be in the form of a rectangle. This rectangle is used to indicate both the instance and the class. If the lifeline must be used to denote an object, it will be underlined.

➤ **Messages**

To showcase an interaction, messages will be used. These messages will come in the form of horizontal arrows, and the messages should be written on top of the arrows. If the arrow has a full head, and it's solid, it will be called a synchronous call. If the solid arrow has a stick head, it will be an asynchronous call. Stick heads with dash arrows are used to represent return messages.

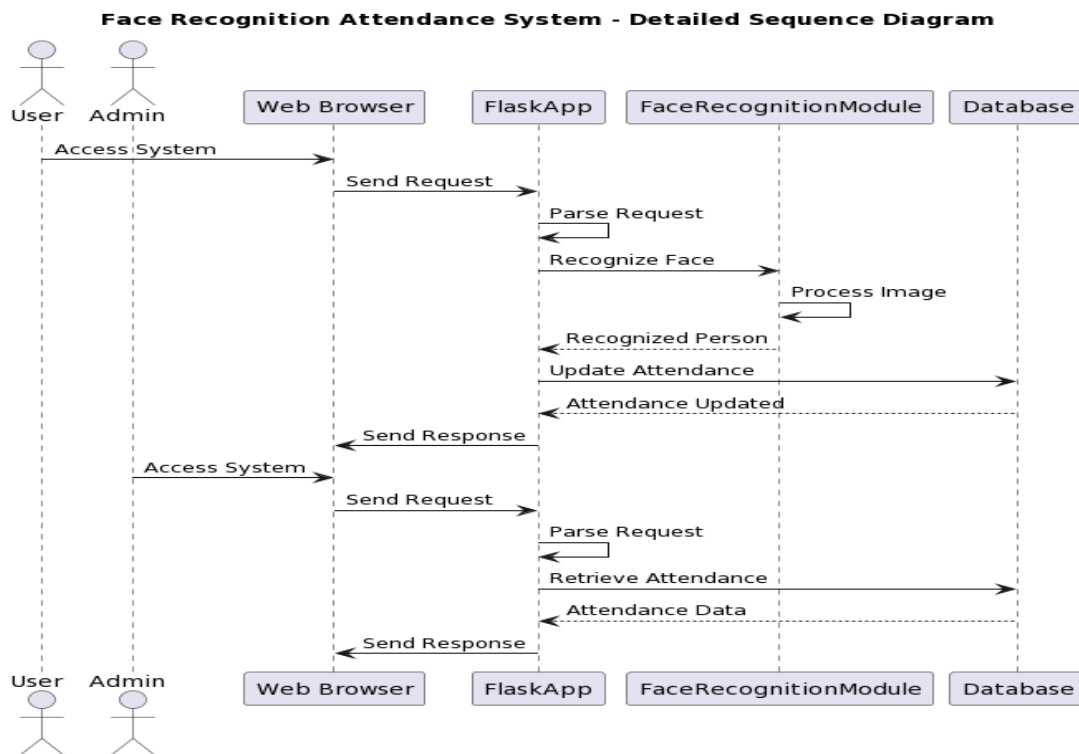
➤ **Objects**

Objects will also be given the ability to call methods upon themselves, and they can add net activation boxes. Because of this, they can communicate with others to show multiple levels of processing. Whenever an object is eradicated or erased from memory, the "X" will be drawn at the lifeline's top, and the dash line will not be drawn beneath it. This will often occur as a result of a message. If a message is sent from the outside of the diagram, it can be used to define a message that comes from a circle that is filled in. Within a UML based model, a Super step is a collection of steps which result from outside stimuli.

**Steps to Create a Sequence Diagram**

- Set the context for the interaction, whether it is a system, subsystem, operation or class.
- Set the stage for the interaction by identifying which objects play a role in interaction.
- Set the lifetime for each object.
- Start with the message that initiates the interaction.
- Visualize the nesting of messages or the points in time during actual computation.
- Specify time and space constraints, adorn each message with timing mark and attach suitable time or space constraints.

- Specify the flow of control more formally, attach pre and post conditions to each message.



**Fig 6.3: Sequence Diagram**

#### 6.1.4 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

##### How to draw Activity Diagram?

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagram is not exactly a flow chart as they have some additional capabilities. These additional capabilities include branching, parallel flow, swim lane etc. Before drawing an activity diagram, we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system.

After identifying the activities, we need to understand how they are associated with constraints and conditions. So before drawing an activity diagram we should identify the following elements.

- Activities
- Association
- Conditions
- Constraints

The following are the basic notational elements that can be used to make up a diagram:

**Initial state**

An initial state represents a default vertex that is the source for a single transition to the default state of a composite state. There can be at most one initial vertex in a region. The outgoing transition from the initial vertex may have a behavior, but not a trigger or guard. It is represented by Filled circle, pointing to the initial state.

**Final state**

A special kind of state signifying that the enclosing region is completed. If the enclosing region is directly contained in a state machine and all other regions in the state machine also are completed, then it means that the entire state machine is completed. It is represented by Hollow circle containing a smaller filled circle, indicating the final state.

**Rounded rectangle**

It denotes a state. Top of the rectangle contains a name of the state. Can contain a horizontal line in the middle, below which the activities that are done in that state are indicated.

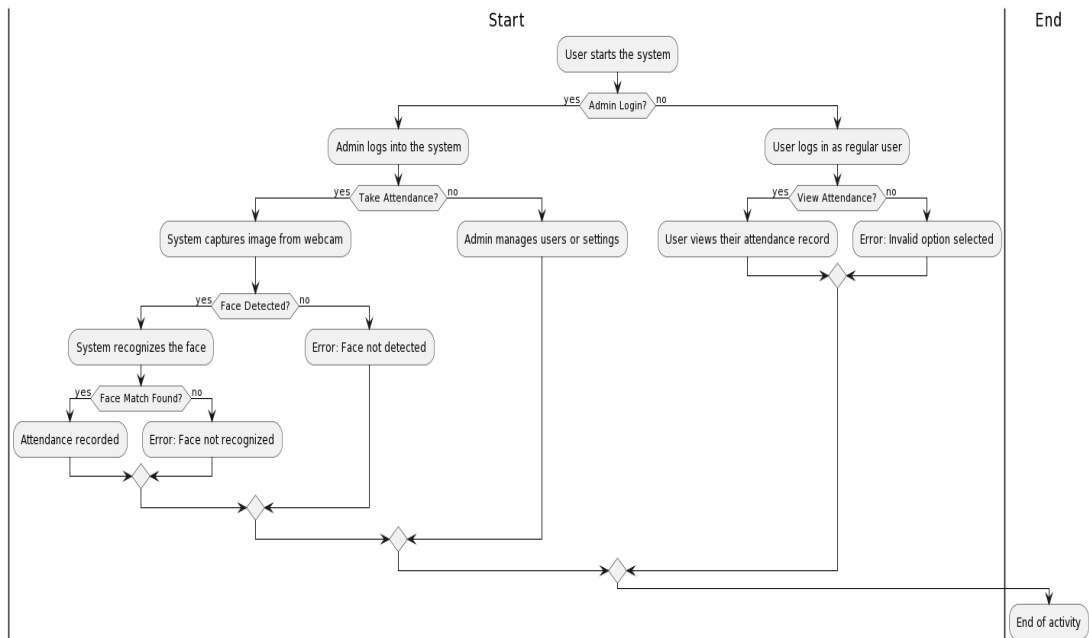
**Arrow**

It denotes transition. The name of the event (if any) causing this transition labels the arrow body.

**Steps To Construct Activity Diagram**

- Identify the preconditions of the workflow
- Collect the abstractions that are involved in the operations

- Beginning at the operation's initial state, specify the activities and actions.
- Use branching to specify conditional paths and iterations
- Use forking & joining to specify parallel flows of control.



**Fig 6.4: Activity Diagram**

## **7.SYSTEM IMPLEMENTATION**

### **7.1 IMPLEMENTATION PROCESS**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

#### **Step 1: Set Up Development Environment**

1. Install Python and required libraries: Python, Flask, OpenCV, scikit-learn, Pandas.
2. Set up a development environment using your preferred text editor or IDE.

#### **Step 2: Design Database Schema**

1. Design a database schema to store user information and attendance records.
2. Choose a database system (e.g., SQLite, MySQL) and create the necessary tables.

#### **Step 3: Collect Training Data**

1. Collect a dataset of facial images for training the face recognition model.
2. Ensure diversity in the dataset to account for variations in lighting, pose, and facial expressions.

#### **Step 4: Train Face Recognition Model**

1. Preprocess the training data (e.g., face detection, normalization).
2. Train a face recognition model using machine learning algorithms like K-nearest

neighbors or deep learning frameworks like TensorFlow or PyTorch.

3. Evaluate the model's performance using validation data and fine-tune as necessary.

#### Step 5: Develop Flask Web Application

1. Create Flask routes for various functionalities (e.g., user registration, attendance tracking).

2. Implement user interfaces using HTML templates with Jinja2 templating.

3. Handle form submissions and user interactions using Flask request objects.

4. Integrate face recognition functionality into the Flask application.

#### Step 6: Implement User Registration and Management

1. Create routes for user registration, login, and profile management.

2. Implement functionality to capture user images for registration and update user information in the database.

#### Step 7: Implement Attendance Tracking

1. Design routes and functionality for capturing attendance data.

2. Integrate face recognition functionality to identify users during attendance capture.

3. Update attendance records in the database with timestamps and user information.

#### Step 8: Implement Reporting and Analytics

1. Develop routes for generating attendance reports and analytics.

2. Implement functionality to retrieve attendance data from the database and visualize it using libraries like Matplotlib or Plotly.

#### Step 9: Testing and Validation

1. Write unit tests and integration tests to ensure the correctness of each component.

2. Conduct user acceptance testing (UAT) to validate the system's functionality and usability.



### Step 10: Deployment

1. Deploy the Flask application to a web server (e.g., Heroku, AWS, DigitalOcean).
2. Configure the server environment and database settings for production use.
3. Monitor system performance and handle any issues that arise.

### Step 11: Maintenance and Updates

1. Regularly update the system with new features, bug fixes, and security patches.
2. Monitor user feedback and make improvements based on user suggestions and requirements.

## 7.2 MODULES

In the face recognition-based attendance management system project, modules are self-contained units of code that encapsulate specific functionality. These modules help organize the codebase, promote reusability, and facilitate better maintenance and collaboration among developers. Let's discuss the modules used in the project:

### 1. Flask Application Module:

**Purpose:** This module contains the main Flask application responsible for handling HTTP requests, rendering HTML templates, and managing routes.

**Functionality:** It defines routes for various functionalities such as user registration, attendance tracking, and reporting. It interacts with other modules to coordinate the system's behavior.

**Example Functions:** ``home``, ``listusers``, ``deleteuser``, ``start``, ``add``.

**File:** ``app.py`` or ``flask_app.py``.

### 2. Face Recognition Module:

**Purpose:** This module handles face detection, recognition, and identification using machine learning algorithms.

Functionality: It encapsulates the logic for preprocessing images, training the face recognition model, and predicting the identity of detected faces.

Example Functions: ``extract_faces``, ``identify_face``, ``train_model``.

File: ``face_recognition.py`` or ``face_recognition_module.py``.

### 3. Database Interaction Module:

Purpose: This module manages interactions with the database, including CRUD (Create, Read, Update, Delete) operations for user information and attendance records.

Functionality: It abstracts away the details of database queries and operations, providing an interface for accessing and manipulating data.

Example Functions: ``add_user``, ``update_attendance``, ``get_attendance``.

File: ``database.py`` or ``db_interaction.py``.

### 4. User Interface Module:

Purpose: This module contains HTML templates, CSS stylesheets, and client-side JavaScript code for building the user interface.

Functionality: It defines the layout and appearance of web pages, handles user interactions, and communicates with the Flask application via HTTP requests.

Example Files: ``index.html``, ``listusers.html``, ``add_user.html``.

Folder: ``templates`` or ``static``.

### 5. Utility Module:

Purpose: This module contains utility functions and helper classes used across the project for common tasks.

Functionality: It provides reusable code snippets for tasks such as image processing, file operations, date/time handling, and data validation.

Example Functions: ``totalreg``, ``extract_attendance``, ``deletefolder``.

File: ``utils.py`` or ``utilities.py``.

Benefits of Modularization:

1. Code Organization: Modules help organize code into logical units, making it easier

to navigate and understand the project structure.

2. Reusability: Modular code can be reused in other projects or within the same project, reducing duplication and promoting consistency.
3. Maintenance: Modular code is easier to maintain and update, as changes made to one module have minimal impact on other modules.
4. Collaboration: Modularization facilitates collaboration among developers, as different team members can work on separate modules concurrently.
5. Testing: Modules can be tested independently, allowing for more targeted and efficient testing of specific functionalities.

By modularizing the project, developers can build a scalable, maintainable, and well-structured system that meets the project requirements and facilitates future enhancements and updates.

## 8. TESTING

A “program unit” stands for a routine or a collection of routines implemented by an individual programmer. It might even be a stand-alone program or a functional unit a larger program.

### 8.1 UNIT TESTING

Unit testing is performed prior to integration of the unit into a larger system. It is like coding and debugging ->unit testing ->integration testing. A program unit must be tested for functional tests, performance tests, stress tests and structure tests.

Functional tests refer to executing the code with standard inputs, for which the results will occur within the expected boundaries. Performance test determines the execution time spent in various parts of the unit, response time, device utilization and throughput. Performance testing will help the tuning of the system.

Stress tests drive the system to its limits. They are designed to intentionally break the unit. Structure tests verify logical execution along different execution paths. Functional, performance and stress tests are collectively known as “black box testing”. Structure testing is referred to as “white box” or “glass box” testing. Program errors can be classified as missing path errors, computational errors and domain errors.

Even if it looks like all possible execution paths have been tested, there might still exist some more paths. A missing path error occurs, when a branching statement and the associated computations are accidentally omitted. Missing paths can be detected only by functional specifications. A domain error occurs when a program traverses the wrong path because of an incorrect predicate in a branching statement. When a test case fails to detect a computational error there is said to be a coincidental error.

### Debugging

Debugging is eliminating the cause of known errors. Commonly used debugging techniques are induction, deduction and backtracking. Debugging by induction involves the following steps:

- Collect all the information about test details and test results
- Look for patterns
- Form one or more hypotheses and rank/classify them.
- Prove/disprove hypotheses. Re examine
- Implement appropriate corrections
- Verify the corrections. Re run the system and test again until satisfactory
- Debugging by deduction involves the following steps:
  - List possible causes for observed failure.
  - Use the available information to eliminate various hypotheses.
  - Prove/disprove the remaining hypotheses.
  - Determine the appropriate correction.
  - Carry out the corrections and verify.

Debugging by backtracking involves working backward in the source code from point where the error was observed. Run additional test cause and collect more information.

## **8.2 INTEGRATION TESTING**

Integration testing strategies include bottom-up (traditional), top-down and sandwich strategies. Bottom-up integration consists of unit testing, followed by testing entire system. Unit testing tries to discover errors in modules. Modules are tested independently in an artificial environment known as “test harness”. Test harnesses provide data environments and calling sequences for the routines and subsystem that are being tested in isolation.

Disadvantages of bottom-up testing include that harness preparation, which can sometimes take about 50% or more of the coding and debugging effort for a smaller product. After testing all the modules independently and in isolation, they are linked and executed in one single integration run. This is known as “Big bang” approach to integration testing. Isolating sources of errors is difficult in “big bang” approach.

Top-down integration starts with main routine and one or two immediately next lower level routines. After a through checking the top level becomes a test harness to its immediate subordinate routines. Top-down integration offers the following advantages.

System integration is distributed throughout the implementation phase. Modules are integrated as they are developed.

- Top-level interfaces are first test.
- Top-level routines provide a natural test harness for lower-level routines.
- Errors are localized to the new modules and interfaces that are being added.

Though top-down integrations seem to offer better advantages, it may not be applicable in certain situations. Sometimes it may be necessary to test certain low- level modules first. In such situations, a sandwich strategy is preferable. Sandwich integration is mostly top-down, but bottom-up techniques are used on some modules and sub systems. This mixed approach retains the advantages of both strategies.

### **8.3 SYSTEM TESTING**

System testing involves two activities: Integration testing and Acceptance testing. Integration strategy stresses on the order in which modules are written, debugged and unit tested. Acceptance test involves functional tests, performance tests and stress tests to verify requirements fulfillment. System checking checks the interface, decision logic, control flow, recovery procedures and throughput, capacity and timing characteristics of the entire system.

### **8.4 ACCEPTANCE TESTING**

Acceptance testing involves planning and execution of functional tests, performance tests and stress tests in order to check whether the system implemented satisfies the requirements specifications. Quality assurance people as well as customers

may simultaneously develop acceptance tests and run them. In addition to functional and performance tests, stress tests are performed to determine the limits/limitations of the system developed. For example, a compiler may be tested for its symbol table overflows or a real-time system may be tested for multiple interrupts of different/same priorities.

Acceptance test tools include a test coverage analyzer, and a coding standards checker. Test coverage analyzer records the control paths followed for each test case. A timing analyzer reports the time spent in various regions of the source code under different test cases. Coding standards are stated in the product requirements. Manual inspection is usually not an adequate mechanism for detecting violations of coding standards.

## **TESTING OBJECTIVES**

Testing is a process of executing a program with the intent of finding errors. A good test is one that has a high probability of finding undiscovered errors. Testing is vital to the success of the system. System testing is the state of implementation, which ensures that the system works accurately before live operations commence. System testing makes a logical assumption that the system is correct and that the system is correct and that the goals are successfully achieved.

### **Effective Testing Prerequisites**

#### **Integration testing**

An overall test plan for the project is prepared before the start of coding.

#### **Validation testing**

This project will be tested under this testing sample data and produce the correct sample output.

#### **Recovery testing**

This project will be tested under this testing using correct data input and its product and the correct valid output without any errors.

#### **Security testing**

This project contains password to secure the data.

### **Test Data and Input**

Taking various types of data we do the above testing. Preparation of test data plays a vital role in system testing. After preparing the test data the system under study is treated using the test data. While testing the system by using the above testing and correction methods. The system has been verified and validated by running with both.

- Run with live data
- Run with test data

#### **Run with test data**

In the case the system was run with some sample data. Specification testing was also done for each conditions or combinations for conditions.

#### **Run with live data**

The system was tested with the data of the old system for a particular period. Then the new reports were verified with the old.



## 9.SAMPLE SOURCE CODE

### PYTHON

```
import cv2
import os
from flask import Flask, request, render_template
from datetime import date
from datetime import datetime
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import joblib

# Defining Flask App
app = Flask(__name__)

nimgs = 10

# Saving Date today in 2 different formats
datetoday = date.today().strftime("%m_%d_%y")
datetoday2 = date.today().strftime("%d-%B-%Y")

# Initializing VideoCapture object to access WebCam
face_detector =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# If these directories don't exist, create them
if not os.path.isdir('Attendance'):
    os.makedirs('Attendance')
if not os.path.isdir('static'):
    os.makedirs('static')
if not os.path.isdir('static/faces'):
    os.makedirs('static/faces')
if f'Attendance-{datetoday}.csv' not in os.listdir('Attendance'):
    with open(f'Attendance/Attendance-{datetoday}.csv', 'w') as f:
        f.write('Name,Roll,Time')
```

```
# get a number of total registered users
def totalreg():
    return len(os.listdir('static/faces'))

# extract the face from an image
def extract_faces(img):
    try:
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        face_points = face_detector.detectMultiScale(gray, 1.2, 5,
minSize=(20, 20))
        return face_points
    except:

return []

# Identify face using ML model
def identify_face(facearray):
    model = joblib.load('static/face_recognition_model.pkl')
    return model.predict(facearray)

# A function which trains the model on all the faces available in faces
folder
def train_model():
    faces = []
    labels = []
    userlist = os.listdir('static/faces')
    for user in userlist:
        for imgname in os.listdir(f'static/faces/{user}'):
            img = cv2.imread(f'static/faces/{user}/{imgname}')
            resized_face = cv2.resize(img, (50, 50))
            faces.append(resized_face.ravel())
            labels.append(user)
    faces = np.array(faces)
    knn = KNeighborsClassifier(n_neighbors=5)
    knn.fit(faces, labels)
    joblib.dump(knn, 'static/face_recognition_model.pkl')

# Extract info from today's attendance file in attendance folder
```

```
def extract_attendance():
    df = pd.read_csv(f'Attendance/Attendance-{datetoday}.csv')
    names = df['Name']
    rolls = df['Roll']
    times = df['Time']
    l = len(df)
    return names, rolls, times, l

# Add Attendance of a specific user
def add_attendance(name):
    username = name.split('_')[0]
    userid = name.split('_')[1]
    current_time = datetime.now().strftime("%H:%M:%S")

    df = pd.read_csv(f'Attendance/Attendance-{datetoday}.csv')
    if int(userid) not in list(df['Roll']):
        with open(f'Attendance/Attendance-{datetoday}.csv', 'a') as f:
            f.write(f'\n{username},{userid},{current_time}')

## A function to get names and rol numbers of all users
def getallusers():
    userlist = os.listdir('static/faces')
    names = []
    rolls = []
    l = len(userlist)

    for i in userlist:
        name, roll = i.split('_')
        names.append(name)
        rolls.append(roll)

    return userlist, names, rolls, l

## A function to delete a user folder
def deletefolder(duser):
    pics = os.listdir(duser)
    for i in pics:
        os.remove(duser+'/' + i)
```

```
os.rmdir(duser)

##### ROUTING FUNCTIONS #####

# Our main page
@app.route('/')
def home():
    names, rolls, times, l = extract_attendance()
    return render_template('home.html', names=names, rolls=rolls,
times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2)

## List users page
@app.route('/listusers')
def listusers():
    userlist, names, rolls, l = getallusers()
    return render_template('listusers.html', userlist=userlist,
names=names, rolls=rolls, l=l, totalreg=totalreg(),
datetoday2=datetoday2)

## Delete functionality
@app.route('/deleteuser', methods=['GET'])
def deleteuser():
    duser = request.args.get('user')
    deletefolder('static/faces/'+duser)

    ## if all the face are deleted, delete the trained file...
    if os.listdir('static/faces/')==[]:
        os.remove('static/face_recognition_model.pkl')

    try:
        train_model()
    except:
        pass

    userlist, names, rolls, l = getallusers()
```

```
    return render_template('listusers.html', userlist=userlist,
names=names, rolls=rolls, l=1, totalreg=totalreg(),
datetoday2=datetoday2)

# Our main Face Recognition functionality.
# This function will run when we click on Take Attendance Button.
@app.route('/start', methods=['GET'])
def start():
    names, rolls, times, l = extract_attendance()

    if 'face_recognition_model.pkl' not in os.listdir('static'):
        return render_template('home.html', names=names, rolls=rolls,
times=times, l=1, totalreg=totalreg(), datetoday2=datetoday2,
mess='There is no trained model in the static folder. Please add a new
face to continue.')

    ret = True
    cap = cv2.VideoCapture(0)
    while ret:
        ret, frame = cap.read()
        if len(extract_faces(frame)) > 0:
            (x, y, w, h) = extract_faces(frame)[0]
            cv2.rectangle(frame, (x, y), (x+w, y+h), (86, 32, 251), 1)
            cv2.rectangle(frame, (x, y), (x+w, y-40), (86, 32, 251), -
1)

            face = cv2.resize(frame[y:y+h, x:x+w], (50, 50))
            identified_person = identify_face(face.reshape(1, -1))[0]
            add_attendance(identified_person)
            cv2.putText(frame, f'{identified_person}', (x+5, y-5),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255),
2)

            cv2.imshow('Attendance', frame)
            if cv2.waitKey(1) == 27:
                break
    cap.release()
    cv2.destroyAllWindows()
    names, rolls, times, l = extract_attendance()
```

```
    return render_template('home.html', names=names, rolls=rolls,
times=times, l=1, totalreg=totalreg(), datetoday2=datetoday2)

# A function to add a new user.
# This function will run when we add a new user.
@app.route('/add', methods=['GET', 'POST'])
def add():
    newusername = request.form['newusername']
    newuserid = request.form['newuserid']
    userimagefolder = 'static/faces/'+newusername+'_'+str(newuserid)
    if not os.path.isdir(userimagefolder):
        os.makedirs(userimagefolder)
    i, j = 0, 0
    cap = cv2.VideoCapture(0)
    while 1:
        _, frame = cap.read()
        faces = extract_faces(frame)
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 20), 2)
            cv2.putText(frame, f'Images Captured: {i}/{nimgs}', (30,
30),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 20), 2,
cv2.LINE_AA)
            if j % 5 == 0:
                name = newusername+'_'+str(i)+'.jpg'
                cv2.imwrite(userimagefolder+'/'+name, frame[y:y+h,
x:x+w])
                i += 1
                j += 1
            if j == nimgs*5:
                break
        cv2.imshow('Adding new User', frame)
        if cv2.waitKey(1) == 27:
            break
    cap.release()
    cv2.destroyAllWindows()
    print('Training Model')
    train_model()
```

```
names, rolls, times, l = extract_attendance()
    return render_template('home.html', names=names, rolls=rolls,
times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2)
```

```
# Our main function which runs the Flask App
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

## HTML

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title>Face Recognition Based Attendance System</title>
    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&disp
lay=swap" rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta3/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-
eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"
crossorigin="anonymous">
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            background-color: #f3f4f6;
            padding-top: 40px;
        }
        .container {
            max-width: 1200px;
            margin: auto;
            padding: 20px;
        }

        .card {
```

```
background-color: #fff;
border-radius: 20px;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
padding: 30px;
margin-bottom: 30px;
}
```

```
.btn-primary {
background-color: #0b4c61;
border: none;
border-radius: 10px;
padding: 12px 24px;
font-size: 18px;
font-weight: bold;
color: #fff;
transition: background-color 0.3s ease;
}
```

```
.btn-primary:hover {
background-color: #083542;
}
```

```
h1 {
font-size: 36px;
font-weight: bold;
color: #0b4c61;
text-align: center;
margin-bottom: 30px;
}
```

```
h2 {
font-size: 24px;
font-weight: bold;
color: #0b4c61;
margin-bottom: 20px;
}
```

```
table {
```



```
        width: 100%;
        border-collapse: collapse;
        margin-top: 20px;
    }

    th,
    td {
        padding: 12px;
        text-align: left;
        border-bottom: 1px solid #ddd;
    }

    tr:nth-child(even) {
        background-color: #f2f2f2;
    }

    form {
        margin-top: 30px;
    }

    label {
        font-weight: bold;
        color: #0b4c61;
        font-size: 18px;
        margin-bottom: 8px;
        display: block;
    }

    input[type="text"],
    input[type="number"] {
        width: 100%;
        padding: 10px;
        margin-bottom: 20px;
        border-radius: 8px;
        border: 1px solid #ccc;
        transition: border-color 0.3s ease;
    }
```

```
        input[type="text"]:focus,
        input[type="number"]:focus {
            outline: none;
            border-color: #0b4c61;
        }
    </style>
</head>

<body>

    <div class="container">
        <h1>Face Recognition Based Attendance System</h1>

        {% if mess %}
        <div class="alert alert-danger" role="alert">
            {{ mess }}
        </div>
        {% endif %}

        <div class="row">
            <div class="col-lg-6">
                <div class="card">
                    <h2>Today's Attendance</h2>
                    <a href="/start" class="btn btn-primary">Take
Attendance</a>

                    <table>
                        <tr>
                            <th>S No</th>
                            <th>Name</th>
                            <th>ID</th>
                            <th>Time</th>
                        </tr>
                        {% if 1 %}
                        {% for i in range(1) %}
                        <tr>
                            <td>{{ i+1 }}</td>
                            <td>{{ names[i] }}</td>
                            <td>{{ rolls[i] }}</td>
```

```
        <td>{{ times[i] }}</td>
    </tr>
    {% endfor %}
    {% endif %}
</table>
</div>
</div>
<div class="col-lg-6">
    <div class="card">
        <h2>Add New User</h2>
        <form action="/add" method="POST"
enctype="multipart/form-data">
            <label for="newusername">Enter New User
Name*</label>
            <input type="text" id="newusername"
name='newusername' required>
            <label for="newuserid">Enter New User
Id*</label>
            <input type="number" id="newusereid"
name='newuserid' required>
            <button type="submit" class="btn btn-
primary">Add New User</button>
            <p>Total Users in Database: {{totalreg}}</p>
        </form>
    </div>
</div>
</div>
</div>
</body>

</html>
```

## 10. MODULE LIST

### Module 1: Overall module

Certainly! Let's delve into each module used in the face recognition-based attendance management system project and explain their functionality in detail:

#### 1. `cv2` (OpenCV):

Explanation: OpenCV (Open Source Computer Vision Library) is a widely-used open-source library for computer vision and image processing tasks.

Functionality:

Image Input/Output: `cv2.imread()` and `cv2.imwrite()` functions are used to read and write images.

Image Display: `cv2.imshow()` is used to display images in windows.

Image Processing: OpenCV provides various functions for image processing tasks like resizing, blurring, and thresholding.

Object Detection: OpenCV includes pre-trained models for detecting objects like faces, eyes, and smiles.

Example Usage:

```
```python
import cv2

# Read an image
image = cv2.imread('image.jpg')

# Display image
cv2.imshow('Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Detect faces in an image
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1,
minNeighbors=5)
```
```

#### 2. `os`:

Explanation: The `os` module provides a portable way of interacting with the operating system. It allows Python programs to perform various operating system-related tasks such as file and directory operations.

Functionality:

File Operations: `os.path` submodule provides functions for file-related operations like checking file existence, joining paths, and file statistics.

Directory Operations: `os` module provides functions for creating, listing, and deleting directories.

Process Control: `os` module allows interacting with the underlying operating system processes.

Example Usage:

```
```python
import os

# Check if a file exists
if os.path.isfile('filename.txt'):
    print('File exists')

# Create a directory
os.makedirs('directory')

# List directory contents
files = os.listdir('directory')

# Delete a file
os.remove('filename.txt')
```
```

### 3. `Flask`:

Explanation: Flask is a lightweight web framework for Python that allows developers to build web applications quickly and with minimal boilerplate code.

Functionality:

Routing: Flask provides decorators like `@app.route()` to define routes and associate them with view functions.

Templating: Flask integrates with Jinja2 templating engine for rendering HTML templates with dynamic content.

Request Handling: Flask provides request and response objects to handle HTTP requests and responses.

Example Usage:

```
```python
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit_form():
    name = request.form['name']
    return f'Hello, {name}!'

if __name__ == '__main__':
    app.run(debug=True)
```
```

### 4. `datetime`:

Explanation: The `datetime` module provides classes and functions for manipulating dates and times in Python.

Functionality:

Date and Time Representation: `datetime.date` and `datetime.datetime` classes

represent dates and times respectively.

Date Formatting and Parsing: Functions like ``strftime()`` and ``strptime()`` are used for formatting and parsing date/time strings.

Arithmetic Operations: ``timedelta`` objects can be used to perform arithmetic operations on dates and times.

Example Usage:

```
```python
from datetime import datetime, date, timedelta

# Get current date and time
current_date = date.today()
current_time = datetime.now()

# Format date and time
formatted_date = current_date.strftime('%Y-%m-%d')
formatted_time = current_time.strftime('%H:%M:%S')

# Parse date from string
parsed_date = datetime.strptime('2022-01-01', '%Y-%m-%d')

# Perform arithmetic operations
future_date = current_date + timedelta(days=7)
```
```

## 5. ``numpy``:

Explanation: NumPy is a fundamental package for scientific computing in Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

Functionality:

Array Creation: NumPy provides functions for creating arrays, including arrays filled with zeros, ones, or random values.

Array Manipulation: Functions like ``reshape()``, ``transpose()``, and ``concatenate()`` are used for manipulating arrays.

Mathematical Operations: NumPy includes a wide range of mathematical functions for operations like addition, multiplication, exponentiation, and trigonometry.









Example Usage:

```
```python
import numpy as np

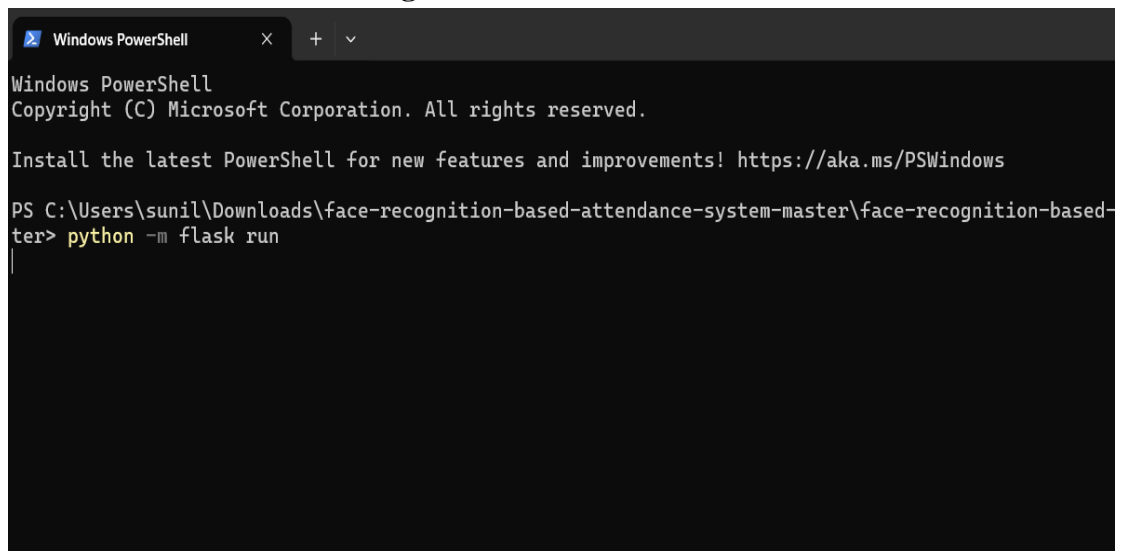
# Create an array
array = np.array([[1, 2, 3], [4, 5, 6]])

# Perform arithmetic operations
result = np.sqrt(array) + np.exp(array)
```
```

## 11.SCREEN LAYOUTS

|                                                                                                                       |                  |                     |          |
|-----------------------------------------------------------------------------------------------------------------------|------------------|---------------------|----------|
|  Attendance                          | 15-03-2024 11:47 | File folder         |          |
|  templates                           | 12-03-2024 14:43 | File folder         |          |
|  static                              | 12-03-2024 14:20 | File folder         |          |
|  __pycache__                         | 12-03-2024 14:19 | File folder         |          |
| ▼ A long time ago                                                                                                     |                  |                     |          |
|  app                                 | 07-11-2023 21:55 | Python Source File  | 8 KB     |
|  haarcascade_frontalface_default.xml | 07-11-2023 21:55 | xmlfile             | 909 KB   |
|  README                              | 07-11-2023 21:55 | Markdown Source ... | 1 KB     |
|  ss                                  | 07-11-2023 21:55 | PNG File            | 1,217 KB |

**Fig 11.1 Folders Creation**



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sunil\Downloads\face-recognition-based-attendance-system-master\face-recognition-based-ter> python -m flask run
```

**Fig 11.2 Command to run the application**

## Face Recognition Based Attendance System

### Today's Attendance

Take Attendance

| S No | Name  | ID | Time     |
|------|-------|----|----------|
| 1    | uma   | 2  | 11:48:02 |
| 2    | krish | 3  | 14:32:09 |
| 3    | sunil | 1  | 14:33:02 |

### Add New User

Enter New User Name\*

Enter New User Id\*

Add New User

Total Users in Database: 3

**Fig 11.3 Attendance Tracking**



## **12.CONCLUSION AND FUTURE ENHANCEMENT**

In conclusion, the face recognition-based attendance management system offers a modern and efficient solution for automating attendance tracking processes in organizations. By leveraging technologies like Flask, OpenCV, and machine learning algorithms, the system streamlines attendance capture, enhances security, and provides real-time reporting capabilities. Future enhancements could include improving face recognition accuracy through advanced training techniques, integrating with biometric devices for multiple authentication options, enhancing reporting and analytics features, developing a mobile application version for greater accessibility, integrating with existing HR systems, deploying on cloud platforms for scalability, and continuously iterating on the system based on user feedback and performance monitoring. These enhancements would further solidify the system's role as a comprehensive and reliable tool for organizations seeking to optimize their attendance management processes.

### 13. BIBLIOGRAPHY

Here are the book references without descriptions:

1. "Flask Web Development: Developing Web Applications with Python" by Miguel Grinberg
2. "Programming Computer Vision with Python: Tools and algorithms for analyzing images" by Jan Erik Solem
3. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems" by Aurélien Géron
4. "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython" by Wes McKinney
5. "OpenCV 4 with Python Blueprints: Build creative computer vision projects with the power of OpenCV and Python" by Gabriel Garrido Calvo, Prateek Joshi
6. "Biometrics: Advanced Identity Verification: The Complete Guide" by John D. Woodward Jr., Nicholas M. Orlans, Peter T. Higgins

# Base Papers