ABSTRACT

The rapid advancement in biometric technologies has paved the way for more secure and efficient systems in various domains, including education. This project presents an Automated Attendance Management System (AAMS) that leverages face recognition to streamline the attendance process, eliminating the need for traditional methods such as manual roll calls or ID card swipes. The system employs Eigenface values and Principal Component Analysis (PCA) for dimensionality reduction, coupled with Convolutional Neural Networks (CNN) for accurate face detection and recognition.
The proposed system captures real-time images from a camera and processes these images to identify and verify student identities. The use of PCA ensures that the system operates efficiently by reducing the computational load without sacrificing accuracy, while CNNs enhance the model's ability to recognize faces under varying lighting conditions, angles, and expressions. Upon successful recognition, attendance is automatically marked in the database, providing an efficient and non-intrusive solution.
Additionally, the system is designed with a user-friendly interface that allows administrators and teachers to monitor attendance records, generate reports, and manage student data with ease. Security measures, including data encryption and access control, are implemented to ensure the privacy and integrity of sensitive information. The system also integrates with existing academic management systems for seamless operation.
To address the potential challenges of environmental variations, the system is equipped with adaptive algorithms that adjust to different classroom settings, ensuring consistent performance. Moreover, the inclusion of a backup manual attendance system ensures reliability even in cases of technical failures. The system supports scalability, allowing it to be deployed across institutions of varying sizes, from small classrooms to large universities.
The AAMS also features real-time notifications for students and parents, providing instant updates on attendance status, which can contribute to better academic performance by promoting regular class attendance. The implementation of advanced analytics allows for the generation of insightful reports, which can help educators identify patterns, address absenteeism, and make informed decisions.
By automating the attendance process, the system not only saves time for educators but also minimizes the risk of human error and enhances overall operational efficiency. The project also considers ethical implications, such as data privacy and consent, ensuring compliance with relevant regulations. The AAMS holds significant potential for educational institutions seeking to enhance their operational efficiency, uphold academic integrity, and foster a disciplined learning environment.

CHAPTER 1
INTRODUCTION

1.1 Introduction

The management of attendance in educational institutions is a critical yet time-consuming task that traditionally relies on manual processes. These manual processes are not only labour-intensive but also prone to inaccuracies and fraudulent practices, such as proxy attendance. To alleviate this burden and enhance the accuracy and reliability of attendance records, smart and automated attendance management systems are increasingly being adopted.

One of the most promising solutions to this problem is the use of biometric verification, specifically facial recognition. Facial recognition technology has been widely adopted in various applications, including video monitoring, human-computer interaction, access control systems, and network security, due to its non-intrusive nature and high accuracy.

This paper proposes a model for an automated attendance management system that leverages facial recognition to authenticate and record student attendance in a classroom setting. The system utilizes Eigenface values, Principal Component Analysis (PCA), and Convolutional Neural Networks (CNN) for effective face detection and recognition. By capturing and recognizing the faces of students, the system can accurately mark attendance, thereby eliminating the possibility of proxies and ensuring that only students who are physically present are marked as such.

The primary implementation steps of the proposed system include:

1. **Face Detection**: Identifying and locating the faces within an image.
2. **Face Recognition**: Matching the detected faces against a pre-existing database of student faces to verify identity.

The integration of these steps ensures a seamless and efficient attendance management process. The recognized faces are then compared with the database to mark attendance accurately. This approach not only streamlines the attendance management process but also maintains precise and reliable records of student attendance.

By implementing this model, educational institutions can significantly reduce the administrative burden on teachers, improve attendance accuracy, and enhance overall operational efficiency. This paper delves into the technical aspects of the system, detailing the methods and algorithms employed, and demonstrates the effectiveness of the proposed solution in managing student attendance records.

1.2 Scope of Project

The scope of the automated attendance management system using face recognition encompasses several key aspects to ensure a comprehensive and effective solution. The project aims to address the following areas:

The primary objective is to automate the process of recording student attendance, eliminating the need for manual attendance taking by teachers. This will significantly reduce the administrative burden and minimize the possibility of errors and fraudulent attendance practices such as proxies. The system will use advanced facial recognition technology, leveraging Eigenface values, Principal Component Analysis (PCA), and Convolutional Neural Networks (CNN) to accurately detect and recognize student faces.

The project also aims to develop a robust database system to store and manage student face data securely. This database will ensure quick and accurate matching of recognized faces against the stored records. Additionally, the system will be designed to be user-friendly, providing a straightforward interface for both students and administrators to interact with.

Furthermore, the scope includes ensuring the system's scalability and adaptability to different educational institutions' needs. This means the solution should be easily deployable in various environments, whether in small classrooms or large lecture halls, and should be capable of handling varying numbers of students efficiently.

Lastly, the project will focus on maintaining high standards of data privacy and security. This involves implementing stringent measures to protect student data and ensure compliance with relevant data protection regulations. By doing so, the system will not only provide an efficient attendance management solution but also safeguard the privacy and integrity of student information.

1.3 Existing System

Currently, many educational institutions rely on traditional manual methods for recording student attendance. These methods typically involve teachers calling out student names and marking their presence or absence on a register or using paper-based attendance sheets. While some institutions have adopted digital systems where students manually sign in using an electronic device, these methods still have significant limitations.

Disadvantages of Existing Systems

1. **Time-Consuming**:
    - Manual attendance taking is a time-consuming process that can take up a significant portion of class time. This reduces the time available for actual teaching and learning activities.

2. Prone to Errors:

- Human error is a major issue in manual attendance systems. Mistakes in marking attendance, such as marking a student present when they are absent or vice versa, are common and can lead to inaccurate records.

3.Susceptibility to Fraud:

- The existing systems are vulnerable to fraudulent practices such as proxy attendance, where a student answers or signs in for an absent peer. This undermines the integrity of attendance records.

4.Inefficiency in Large Classes:

- In large lecture halls or classes with many students, the manual process becomes even more cumbersome and inefficient. The larger the class size, the more time and effort it takes to accurately record attendance.

5. Difficulty in Tracking and Reporting:

- Managing and analyzing attendance records over time can be challenging with manual systems. Generating reports, identifying attendance patterns, and tracking individual student attendance history requires significant effort and is often prone to inaccuracies.

1.4 Proposed System

The proposed automated attendance management system leverages advanced face recognition technology to streamline and enhance the accuracy of attendance recording in educational institutions. By using Eigenface values, Principal Component Analysis (PCA), and Convolutional Neural Networks (CNN), the system can effectively detect and recognize student faces to automate attendance marking.

Advantages of Our System

1. **Time Efficiency**:
    - The automated system significantly reduces the time required to take attendance, freeing up valuable class time for teaching and learning activities.
2. **Accuracy and Reliability**:
    - By using advanced facial recognition technology, the system ensures high accuracy in identifying and marking student attendance, minimizing errors commonly associated with manual processes.
3. **Elimination of Fraud**:
    - The system effectively eliminates proxy attendance and other fraudulent practices, ensuring that only students who are physically present are marked as present.

4. **Scalability**:
    - The system is scalable and can be easily adapted to different class sizes and educational settings, from small classrooms to large lecture halls.
5. **Real-Time Data and Reporting**:
    - The system provides real-time attendance data, allowing administrators to quickly access attendance records, generate reports, and analyze attendance patterns.

1.5 Feasibility Study

The feasibility study for the proposed automated attendance management system involves evaluating its technical, economic, operational, and legal aspects to determine its viability and potential for successful implementation.

Technical Feasibility

1. **Technology and Infrastructure**:
    - The system leverages well-established technologies such as face recognition, Eigenface values, PCA, and CNN. The required hardware includes cameras for capturing images and servers for processing and storing data. These components are readily available and can be integrated into existing institutional infrastructure.
2. **System Development**:
    - Developing the system requires expertise in machine learning, computer vision, and software development. The availability of pre-trained models and open-source libraries can accelerate development and reduce complexity.
3. **Scalability and Integration**:
    - The system is designed to be scalable, capable of handling varying class sizes and institutional needs. It can be integrated with existing educational management systems to ensure seamless data flow and synchronization.

Economic Feasibility

1. **Cost Analysis**:
    - Initial costs include purchasing hardware (cameras and servers), software development, and integration. Ongoing costs involve maintenance, updates, and potential cloud storage fees. A detailed cost analysis shows that these expenses are manageable within typical institutional budgets.
2. **Funding and ROI**:

- Potential funding sources include institutional budgets, grants, and government support for educational technology initiatives. The return on investment (ROI) is achieved through increased efficiency, reduced administrative workload, and improved accuracy in attendance records, leading to better resource allocation and planning.

Operational Feasibility

1. **Ease of Use**:
    - The system features a user-friendly interface, making it accessible for teachers, students, and administrators. Training sessions and user manuals will ensure smooth adoption and minimal disruption to existing workflows.
2. **Implementation and Maintenance**:
    - The implementation process includes setting up hardware, integrating software, and migrating existing attendance data. Regular maintenance and updates will ensure the system remains functional and up-to-date with the latest advancements in technology.

Legal Feasibility

1. **Data Privacy and Security**:
    - The system will comply with relevant data protection regulations, such as GDPR and CCPA, ensuring the privacy and security of student data. Robust encryption and secure storage solutions will be implemented to protect against data breaches and unauthorized access.
2. **Regulatory Compliance**:
    - The system will adhere to educational and biometric regulations, ensuring that its deployment and operation are legally sound. This includes obtaining necessary approvals and certifications from relevant authorities.

CHAPTER 2
SYSTEM SPECIFICATIONS

2.1 Hardware Requirements

Operating System:

- **Windows**: A graphical operating system developed by Microsoft, essential for running the software components of the attendance management system. Windows provides a user-friendly interface for interacting with applications, managing files, and performing everyday tasks. Windows 10 or later is recommended for its modern features, security updates, and compatibility with various applications.

Processor:

- **Intel Core i5**: A mid-range processor suitable for handling the computational requirements of face recognition algorithms and running the attendance management software efficiently. It provides a good balance between performance and cost, supporting multitasking and data processing.

Hard Disk:

- **512 GB**: Solid-state drive (SSD) with at least 512 GB of storage capacity. This ensures sufficient space for storing the operating system, application software, and the database of student face images and attendance records. SSDs are preferred for their faster data access speeds compared to traditional hard disk drives (HDDs).

RAM:

- **8 GB**: A minimum of 8 GB of RAM is recommended to support smooth operation of the software and handling of large datasets. Adequate RAM is crucial for running face recognition algorithms and maintaining overall system performance.

Camera or Sensor:

- **High-Resolution Camera**: Essential for capturing clear and detailed images of student faces. A camera with at least 1080p resolution is recommended to ensure accurate face detection and recognition. The camera should be positioned to cover the entire classroom effectively.

Connectivity:

- **Interfaces**: Data transfer and communication interfaces including Ethernet for network connectivity, USB for connecting peripherals, and possibly additional interfaces depending on specific system requirements.

Power Supply:

- **Reliable Power Supply**: Adequate power supply units to ensure uninterrupted operation of the system's components. This includes power adapters for cameras, servers, and any other electronic devices used in the system.

Mounting Hardware:

- **Mounting Fixtures**: Secure mounts and brackets to install cameras at optimal positions in the classroom. Proper mounting ensures that cameras are correctly aligned to capture all necessary angles for attendance detection.

Environmental Protection:

- **Protective Casings**: Enclosures or casings to shield electronic components from dust, moisture, and temperature variations, particularly for cameras and servers that may be exposed to environmental factors.

These hardware components are crucial for the successful deployment and operation of the automated attendance management system, ensuring efficient performance and reliability in a classroom setting.

2.2 Software Requirements

Python: Python is an interpreted, high-level, general-purpose programming language created by Guido van Rossum and first released in 1991. It emphasizes code readability through its significant use of whitespace, allowing for clear, logical coding. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Its dynamic typing and garbage collection features make it suitable for both small and large-scale projects, making it ideal for developing the algorithms and functionalities required for the automated attendance management system.

Jupyter Notebook: Jupyter Notebook is an interactive, web-based computational environment that allows for the creation and sharing of documents containing live code, equations, visualizations, and narrative text. It uses JSON as its document format and supports various programming languages, though it is most commonly used with Python. The .ipynb file extension represents Jupyter Notebook files, which are useful for data analysis, visualization, and iterative development. Jupyter Notebook facilitates the development and testing of computer vision algorithms and data processing tasks due to its interactive nature and support for visual outputs.

Python Libraries:

- **NumPy**: A fundamental library for numerical computing in Python, providing support for arrays, matrices, and mathematical functions. It is crucial for handling and processing data in the face recognition system.
- **Pandas**: A library used for data manipulation and analysis, offering data structures and operations for handling structured data. It is useful for managing and analysing attendance records and student data.
- **TensorFlow**: An open-source library for machine learning and deep learning, developed by Google. TensorFlow is used for building and training the Convolutional Neural Networks (CNNs) required for face recognition tasks.
- **Matplotlib**: A plotting library for creating static, animated, and interactive visualizations in Python. It is used for visualizing data, results, and performance metrics of the face recognition algorithms.

Web Server:

- **Command Prompt**: While Command Prompt itself is not a web server, it can be used to manage and execute web server-related commands and scripts for setting up and running web applications.

Database:

- **C Drive**: The database for storing student face images and attendance records is located on the C Drive. Ensure that the storage solution on the C Drive is adequately secured and backed up to prevent data loss.

CHAPTER 3
LITERATURE SURVEY

The Boosting Approach to Machine Learning: An Overview

- **Author:** R.E. Schapire
- **Methodology:** This work focuses on boosting as a method to enhance the accuracy of learning algorithms, with particular emphasis on the AdaBoost algorithm. The chapter reviews recent developments, including analyses of AdaBoost's training and generalization errors, connections to game theory and linear programming, incorporation of human knowledge, and experimental applications.
- **Advantages:** Boosting methods, including AdaBoost, offer several benefits over parametric classifiers such as discriminant analysis. They can significantly improve model accuracy by combining weak learners to create a strong classifier. The method also provides flexibility in incorporating various types of knowledge into the learning process.
- **Disadvantages:** One challenge in boosting is its computational demand. Effective boosting algorithms must be fast and accurate, but the trade-off between speed and accuracy can be problematic, especially in applications requiring real-time results.

2. Support Vector Clustering

- **Authors:** A. Ben-Hur, D. Horn, H. T. Siegelmann, V. Vapnik
- **Methodology:** The authors propose a clustering method that maps data points to a high-dimensional feature space using a Gaussian kernel. In this space, a minimal enclosing sphere is identified, which, when mapped back to the original space, reveals distinct clusters. This approach allows for the separation of clusters with arbitrary shapes.
- **Advantages:** This clustering algorithm can generate cluster boundaries of any shape, providing greater flexibility compared to other methods that are restricted to geometric shapes such as hyper-ellipsoids. It is effective for complex clustering problems where traditional methods might struggle.
- **Disadvantages:** A limitation of this approach is that it computes adjacencies only with support vectors rather than the entire adjacency matrix, potentially leading to less comprehensive clustering.

3. Land Cover Change Assessment Using Decision Trees, Support Vector Machines, and Maximum Likelihood Classification Algorithms

- **Authors:** J. R. Otukei, T. Blaschke
- **Methodology:** The study assesses land cover changes using remote sensing data and compares various classification algorithms, including Decision Trees (DT), Support Vector Machines (SVM), and Maximum Likelihood Classification. DTs are highlighted for their speed and lack of statistical assumptions about data distribution.
- **Advantages:** Decision Trees provide a computationally efficient way to classify land cover changes and do not rely on assumptions about data distribution, making them robust in diverse scenarios. They offer a clear and interpretable classification process.
- **Disadvantages:** Despite their advantages, the algorithms may not always deliver the desired accuracy in classification tasks. The challenge lies in optimizing the decision boundaries and tree structures to achieve accurate results.

Relevance to Automated Attendance Management System

- **Boosting Approach:** The principles of boosting, particularly the AdaBoost algorithm, can enhance the performance of face recognition systems by combining multiple weak classifiers to improve accuracy. This can be applied to improve the precision of face recognition in the automated attendance system.
- **Support Vector Clustering:** While primarily used for clustering, the concept of high-dimensional mapping and minimal enclosing spheres can inspire methods for handling complex patterns in face data, potentially improving the robustness of the recognition system.
- **Classification Algorithms:** Techniques like Decision Trees and Support Vector Machines are directly applicable to classification tasks in attendance management. Understanding their strengths and limitations helps in choosing the best approach for distinguishing between different faces and managing attendance records effectively.

CHAPTER 4
SOFTWARE REQUIREMENT ANALYSIS

The SRA for the automated attendance management system focuses on the software components, functionalities, and tools necessary for developing and deploying the face recognition-based attendance system.

1. Functional Requirements

User Interface (UI) Requirements:

- **Image Upload:** Users should be able to upload images or capture live photos for attendance marking.
- **Result Display:** The application should display attendance results, indicating whether the student is present or absent, along with timestamp information.
- **History and Reports:** Users should access a history of attendance records and generate downloadable reports.

Model Integration:

- **Image Processing:** The backend should preprocess images (e.g., resizing, normalization) before processing them with the face recognition model.
- **Face Recognition:** Implement the face recognition model using Eigenface values, PCA, and CNN to accurately identify students.

User Management:

- **Authentication and Authorization:** Support for user registration, login, and role-based access control (e.g., admin, teacher, student).
- **Profile Management:** Allow users to update their profile information and manage their account settings.

Notifications and Alerts:

- **Real-Time Alerts:** Provide real-time notifications for attendance events or anomalies.
- **Email Notifications:** Send email alerts for significant events, such as missing attendance or unauthorized access attempts.

2. Non-Functional Requirements

Performance Requirements:

- **Response Time:** The system should process and return attendance results within a few seconds of image capture.
- **Scalability:** Ensure the system can handle a growing number of users and attendance records without performance issues.

Reliability and Availability:

- **Uptime:** Maintain high availability with minimal downtime for updates and maintenance.
- **Error Handling:** Implement robust error handling to provide informative feedback to users.

Security Requirements:

- **Data Privacy:** Ensure compliance with data protection regulations for handling student images and personal information.
- **Data Encryption:** Encrypt sensitive data both in transit and at rest to protect user privacy.

Usability Requirements:

- **Intuitive Interface:** Design a user-friendly interface with clear instructions and easy navigation.
- **Accessibility:** Ensure the application is accessible to users with disabilities, following relevant accessibility standards.

Maintainability and Extensibility:

- **Modular Design:** Design the system with modular components to simplify updates and feature enhancements.
- **Documentation:** Provide detailed documentation for both developers and users, including API docs and user guides.

3. Technical Requirements

Frontend:

- **Framework:** Use a modern web development framework such as React, Angular, or Vue.js for building the user interface.
- **Responsive Design:** Ensure the application is responsive and functions well across different devices (smartphones, tablets, desktops).

Backend:

- **Framework:** Employ a robust backend framework like Django, Flask (Python), or Node.js for handling server-side logic and interactions.

- **API:** Develop RESTful APIs to facilitate communication between the frontend and backend components.

Database:

- **Database Management System (DBMS):** Choose a reliable DBMS like PostgreSQL, MySQL, or MongoDB for storing user data, attendance records, and model metadata.
- **Data Storage:** Implement efficient data storage solutions to manage large volumes of attendance data and images.

Model Deployment:

- **Framework:** Deploy the face recognition model using TensorFlow Serving, Flask, or a similar model serving framework.
- **Scalability:** Ensure the model deployment setup can handle multiple concurrent recognition requests.

Cloud Infrastructure:

- **Hosting:** Deploy the application on a cloud platform such as AWS, Google Cloud, or Azure to ensure scalability and reliability.
- **Storage:** Utilize cloud storage solutions (e.g., Amazon S3, Google Cloud Storage) for storing images and model files.

Integration and Continuous Deployment:

- **CI/CD Pipeline:** Implement a Continuous Integration/Continuous Deployment (CI/CD) pipeline using tools like Jenkins, GitHub Actions, or GitLab CI to automate testing and deployment processes.

4.1Problem with the Existing System

Current attendance management systems face several issues that the proposed system aims to address:

1.Manual Attendance Systems:

- Manual methods for recording attendance are prone to errors and inefficiencies. Teachers often face difficulties in tracking and managing attendance accurately, leading to inconsistencies and potential disputes. The process is time-consuming and requires significant effort, which is impractical for large classes or institutions.

2. Proxy Attendance:

- Proxy attendance, where students mark others as present, undermines the integrity of the attendance records. Traditional systems rely on physical presence, which can be easily manipulated by students.

3. Lack of Real-Time Monitoring:

- Conventional attendance systems do not provide real-time monitoring, making it difficult to track student attendance instantaneously. This can result in delayed interventions and unresolved attendance issues.

4. Complexity in Record Keeping:

- Maintaining accurate attendance records over time can be challenging. Manual systems often involve cumbersome paperwork and data entry, which increases the risk of errors and makes data retrieval and analysis difficult.

5. Integration Issues:

- Many existing systems lack integration with other administrative functions, such as grading or scheduling, leading to fragmented data management and inefficient workflows.

Solutions in Proposed System

1. Automated Attendance Tracking:

- The proposed system uses Convolutional Neural Networks (CNNs) for face recognition to automate attendance tracking. This approach reduces manual effort and minimizes errors, providing accurate and efficient attendance management.

2. Enhanced Security:

- Face recognition technology ensures that attendance is recorded based on the actual presence of students, eliminating the issue of proxy attendance. The system verifies and records the identity of each student accurately.

3. Real-Time Attendance Monitoring:

- The system provides real-time monitoring and attendance updates. Teachers and administrators can view and manage attendance records instantly, enabling timely interventions and accurate tracking.

4. Streamlined Record Keeping:

- The automated system simplifies record keeping by digitizing attendance data. It provides easy access to historical attendance records and integrates with other administrative functions, improving overall data management.

5. Seamless Integration:

- The system can be integrated with other institutional systems, such as grading and scheduling, to provide a unified administrative solution. This integration enhances workflow efficiency and data coherence.

Implementation Steps

1. **Data Collection and Model Training:** Collect and preprocess a diverse dataset of student face images. Train the CNN model using this dataset to recognize and differentiate between individual students.
2. **System Development:** Develop the backend and frontend components of the attendance management system. Integrate the trained CNN model with the backend to handle face recognition and attendance recording.
3. **User Interface Design:** Create an intuitive user interface for teachers and administrators to manage and review attendance records. Ensure the interface supports real-time updates and easy navigation.
4. **Deployment and Testing:** Deploy the system on a suitable platform, conduct thorough testing to ensure accuracy and performance, and gather feedback from users to make necessary adjustments.
5. **Training and Support:** Provide training and documentation to users for effective adoption of the system. Offer ongoing support to address any issues and ensure smooth operation.

Modules

1. **Image Capture and Processing:** Capture live images or upload photos of students for face recognition. Process and preprocess these images for accuracy and consistency.
2. **Face Recognition and Verification:** Apply the CNN model to recognize and verify students based on their facial features. Match recognized faces with the database to record attendance.
3. **Attendance Recording:** Automatically record and update attendance based on face recognition results. Generate and store attendance records for further analysis and reporting

CHAPTER-5

SYSTEM DESIGN

5.1 Introduction

The system is designed to facilitate automated attendance management using face recognition technology. It integrates various components to capture, process, and analyse facial images for attendance tracking. The primary components of the system include Image Acquisition, Data Processing, Face Recognition and Classification, User Interface (UI), Backend Services, and Deployment and Maintenance.

Components and Architecture

Image Acquisition

- **Hardware:** High-resolution cameras or webcams are used to capture facial images. These devices should provide clear, consistent images under various lighting conditions. For classroom settings, cameras are typically mounted in a fixed position to capture images of students as they enter or are present in the room.
- **Software:** The system employs a camera interface or web-based interface to capture and upload images to the server. Real-time image capture is essential for accurate and timely attendance recording.

Data Processing

- **Preprocessing:** This step includes image normalization (scaling and aligning images), noise reduction (using filters), and data augmentation (techniques such as rotation, scaling, and flipping) to improve model robustness and handle variations in facial images.
- **Feature Extraction:** Convolutional Neural Networks (CNNs) are used to extract features from facial images. CNNs identify key facial features necessary for accurate recognition and classification.

Face Recognition and Classification

- **Model Selection:** The system uses CNN models like ResNet, VGG, or Inception. These pre-trained models can be fine-tuned or customized to meet the specific requirements of the attendance system. If pre-trained models are insufficient, a custom CNN model can be developed.
- **Evaluation:** The model's performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. The system supports real-time face recognition and attendance recording, ensuring timely updates.

User Interface (UI)

- **Web Interface:** Users can access a web-based interface to view attendance records, manage settings, and generate reports. Features include user login, real-time attendance updates, and historical data review.
- **Mobile App:** A mobile application is available for capturing images and managing attendance on the go. It provides notifications and reports directly to users' devices, enhancing accessibility and convenience.

Backend Services

- **Server and API Management:** APIs handle tasks such as image upload, preprocessing, and interaction with the face recognition model. The backend processes these requests and returns results to the frontend.
- **Database Management:** The system uses a database to store user information, attendance records, and image metadata. Reliable database management ensures efficient data retrieval and storage.
- **Cloud Integration:** Cloud services (e.g., AWS, Azure, Google Cloud) are used for scalable processing and storage. Cloud infrastructure supports large-scale data handling and model deployment.

Deployment and Maintenance

- **Deployment:** The system is hosted on a secure web server, with the mobile app distributed through app stores like Google Play and Apple App Store. Deployment involves configuring the server environment, setting up databases, and integrating with cloud services.
- **Maintenance:** Regular maintenance includes updating the model with new data, retraining to enhance accuracy, and monitoring system performance. User support is provided through a help desk, and comprehensive documentation is available, including user guides and FAQs.

This architecture ensures a robust, scalable, and user-friendly automated attendance management system, leveraging advanced face recognition technology to streamline and improve attendance tracking

5.2 Software Techniques

In this section, we delve into the software techniques employed in developing our web application for potato plant disease detection, focusing on the architecture, technologies, and methodologies used to ensure accurate and efficient disease identification.

1.Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a core component of our system, offering a powerful approach to image classification tasks. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images through multiple layers. The architecture of CNNs consists of several key components. Convolutional layers apply a set of filters to the input image, creating feature maps that detect specific features such as edges, textures, or patterns. Pooling layers follow, reducing the dimensions of the feature maps while retaining essential information, thus enhancing computational efficiency and reducing the risk of overfitting. Finally, fully connected layers take the flattened feature maps and perform the final classification based on the extracted features. The combination of these layers enables CNNs to achieve high accuracy in image classification, making them an ideal choice for our disease detection application.

5.3 Data Flow Diagrams

A data flow diagram is graphical tool used to describe and analyse movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams.

Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labelled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process. Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this led to the modular design. A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So, it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

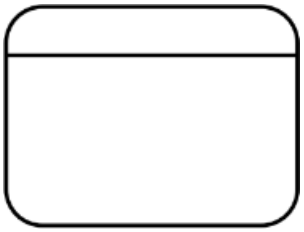
DFD Symbols: In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data.

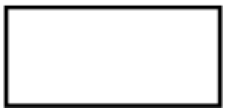
2. An arrow identifies data flow. It is the pipeline through which the information flows.

3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.

4. An open rectangle is a data store, data at rest or a temporary repository of data.



Process that transforms data flow.



Source or Destination of data

Data flow

Data Store

Constructing a Data Flow Diagram:

Several rules of thumb are used in drawing DFD'S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.

2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.

- When a process is exploded into lower-level details, they are numbered.
- The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized.

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

Salient Features of DFD'S

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.

2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.

3. The sequence of events is not brought out on the DFD.

Types of Data Flow Diagrams

1. Current Physical

2. Current Logical

3. New Logical

4. New Physical

Current Physical

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly, data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

Current Logical

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transforms them regardless of actual physical form.

New Logical

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

New Physical

The new physical represents only the physical implementation of the new system.

Rules Governing the DFD'S

Process

- No process can have only outputs.
- No process can have only inputs. If an object has only inputs than it must be a sink.
- A process has a verb phrase label.

Data Store

- Data cannot move directly from one data store to another data store, a process must move data.
- Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
- A data store has a noun phrase label.

Source

The origin:

- Data cannot move direly from a source to sink it must be moved by a process.
- A source and /or sink has a noun phrase land.

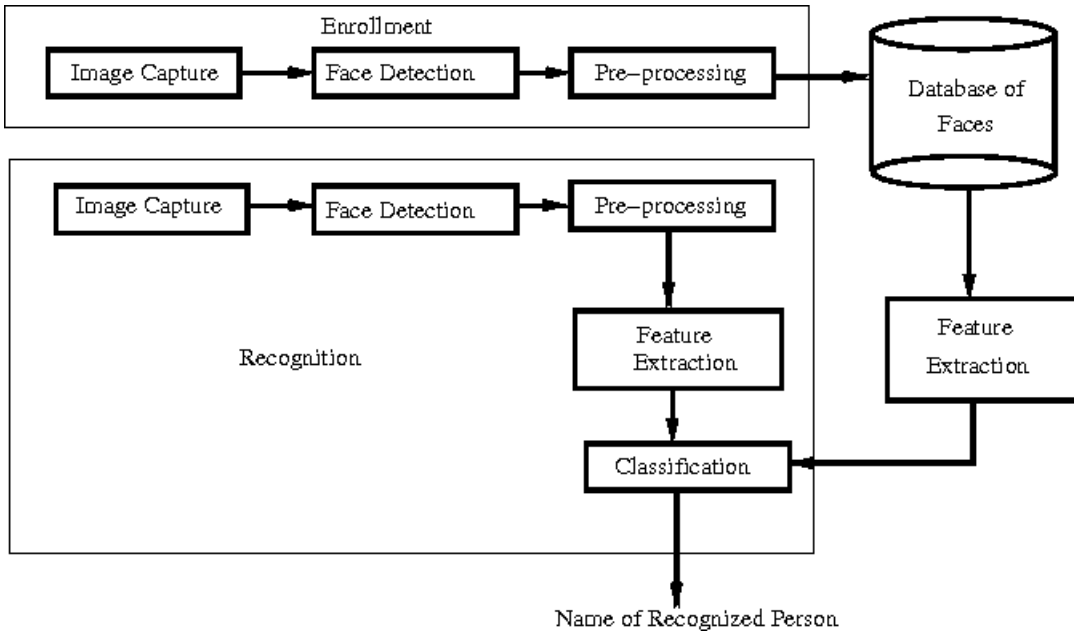
Data Flow

- A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated however by two separate arrows since these happen at different type.
- A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.

- A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
- A Data flow to a data store means update (delete or change).
- A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

Data Flow Diagram:



5.4 UML Diagrams

Unified Modelling Language Diagrams

The unified modelling language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows:

- User Model View
    i. This view represents the system from the user's perspective.
    ii. The analysis representation describes a usage scenario from the end-user's perspective.
- Structural model view
    i. In this model the data and functionality are arrived from inside the
    ii. This model view models the static structures.

- Behavioral Model View

  It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- Implementation Model View

  In this the structural and behavioral as parts of the system are represented as they are to be built.

- Environmental Model View

  In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

- UML Analysis modelling, this focuses on the user model and structural model views of the system.
- UML design modelling, which focuses on the behavioral modelling, implementation

modelling and environmental model views.

Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view. Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer ...etc., or another system like central database.

Unified Modelling Language

The Unified Modelling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software-intensive system. The UML is appropriate for modelling systems ranging from enterprise information systems to distributed Web-based applications and even to hard real time embedded systems. It is a very expressive language, addressing all the views needed to develop and then deploy such systems. The UML is only a language and so is just one part of a software development method. The UML is process independent, although optimally it should be used in a process that is use case driven, architecture-centric, iterative, and incremental.

The UML is a language for

- Visualizing.
- Specifying.
- Constructing.
- Documenting.

The artifacts of a software-intensive system. A modelling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system. Modelling yields an understanding of a system. The vocabulary of the UML encompasses three kinds of building blocks:

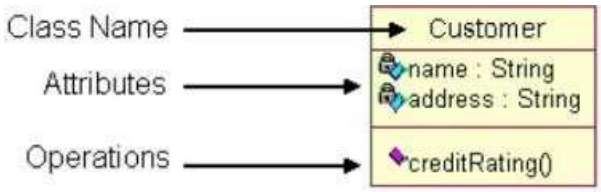
• Things

• Relationships

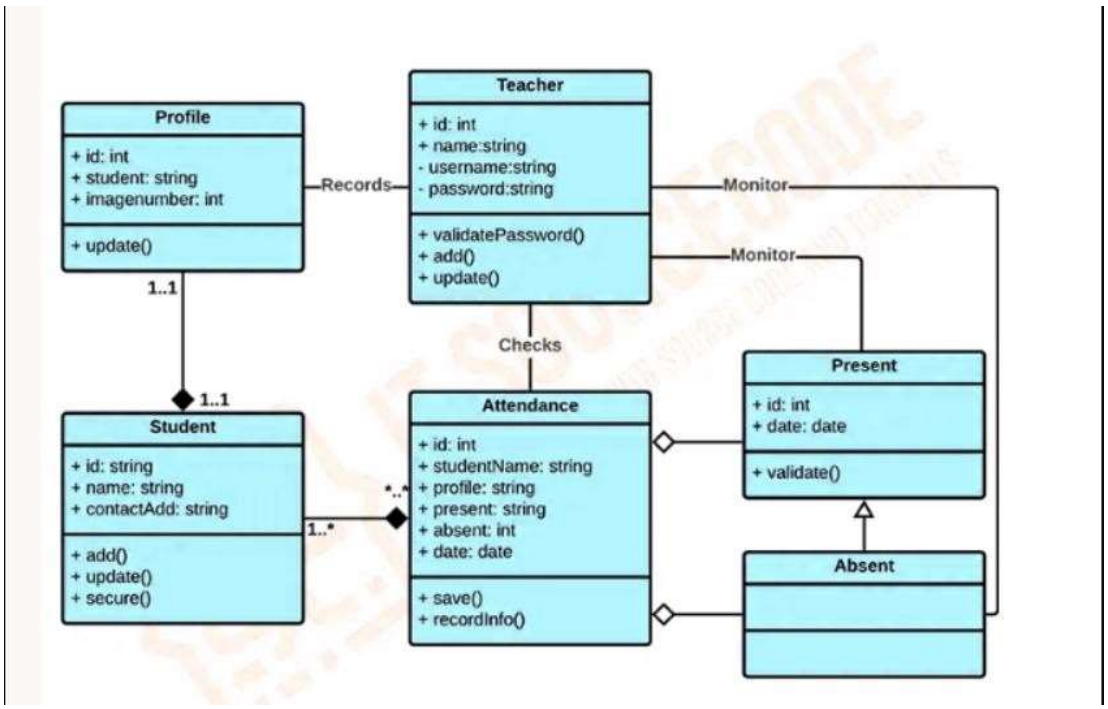
• Diagrams

5.4.1 Class Diagrams

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. Classes are composed of three things: a name, attributes, and operations. Below is an example of a class.

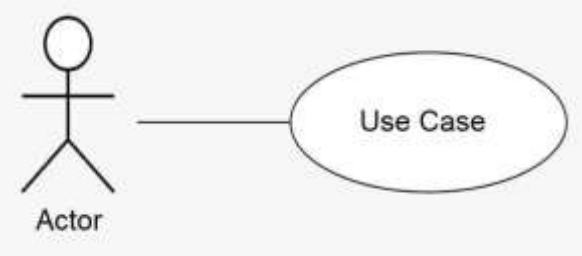


Class Diagram:



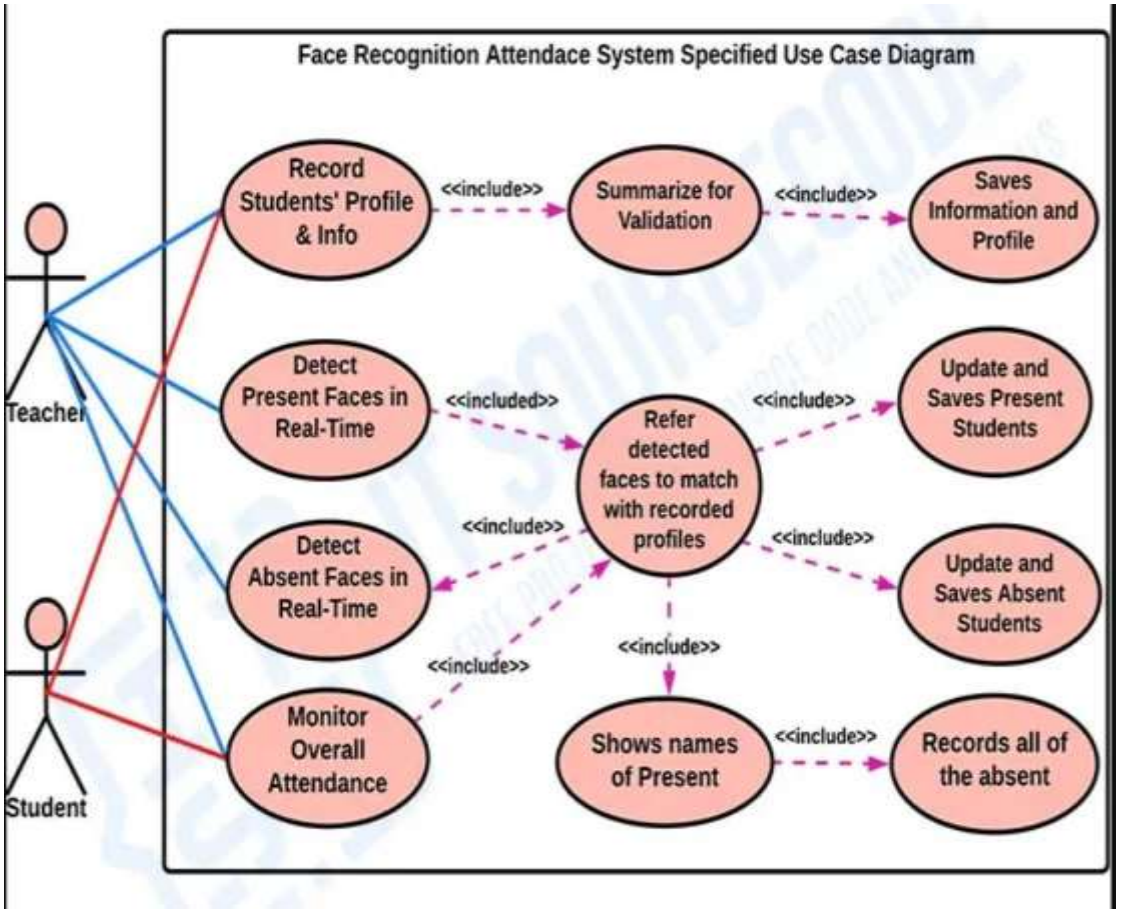
5.4.2 Use Case Diagrams

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of use case diagram are use cases and actors



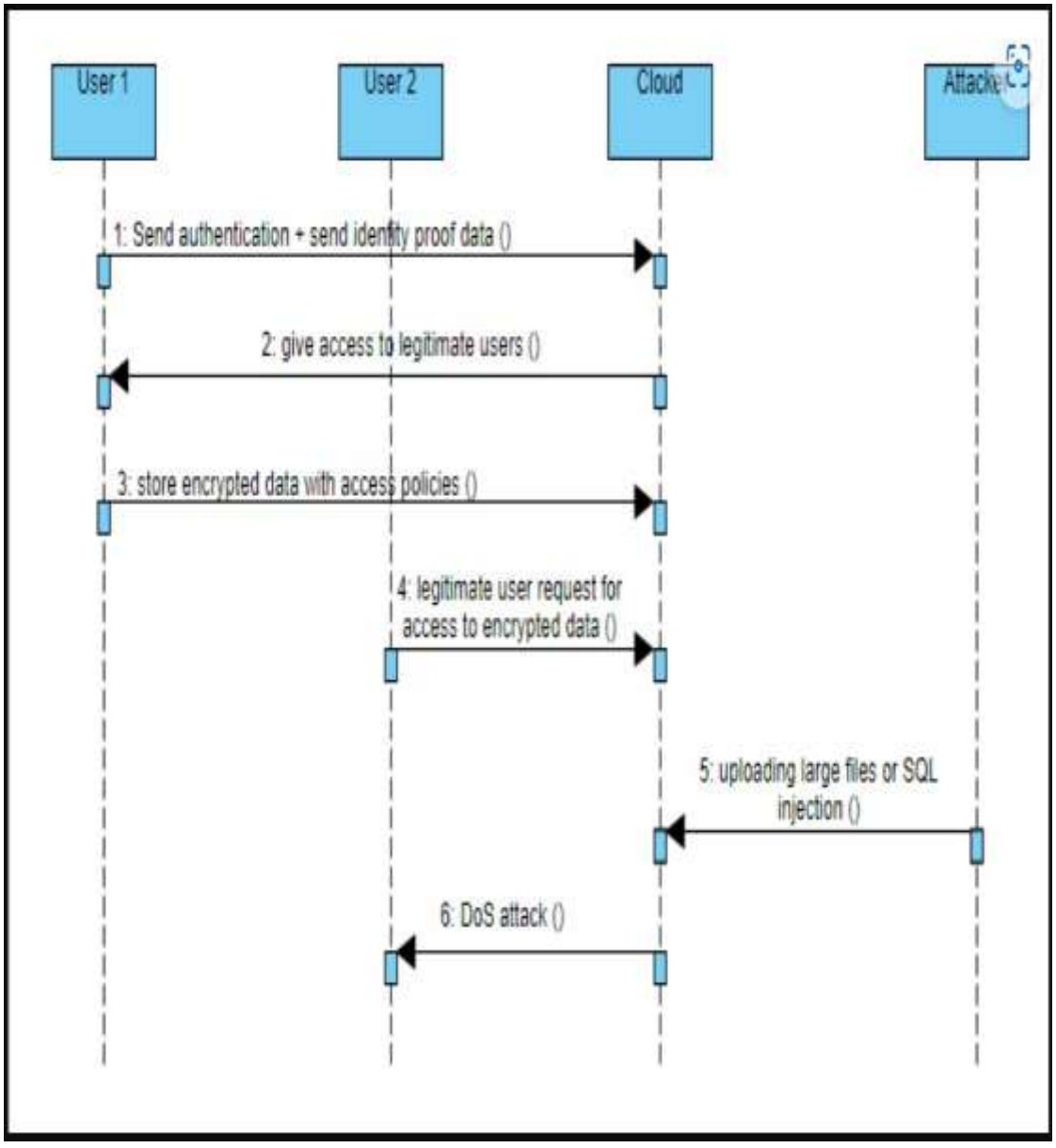
An actor is representing a user or another system that will interact with the system you are modelling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

Customer Use Case:

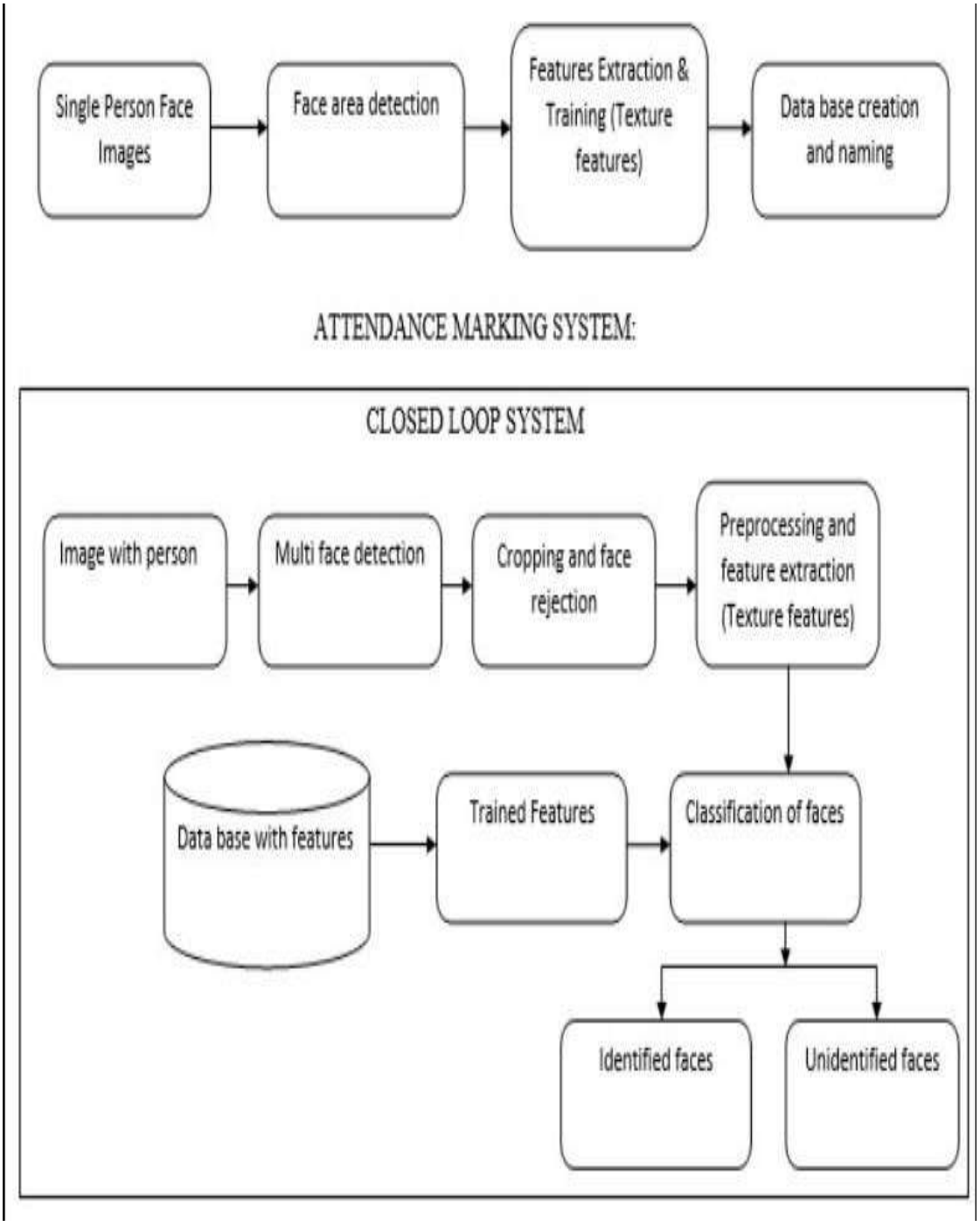


5.4.3 Sequence Diagrams

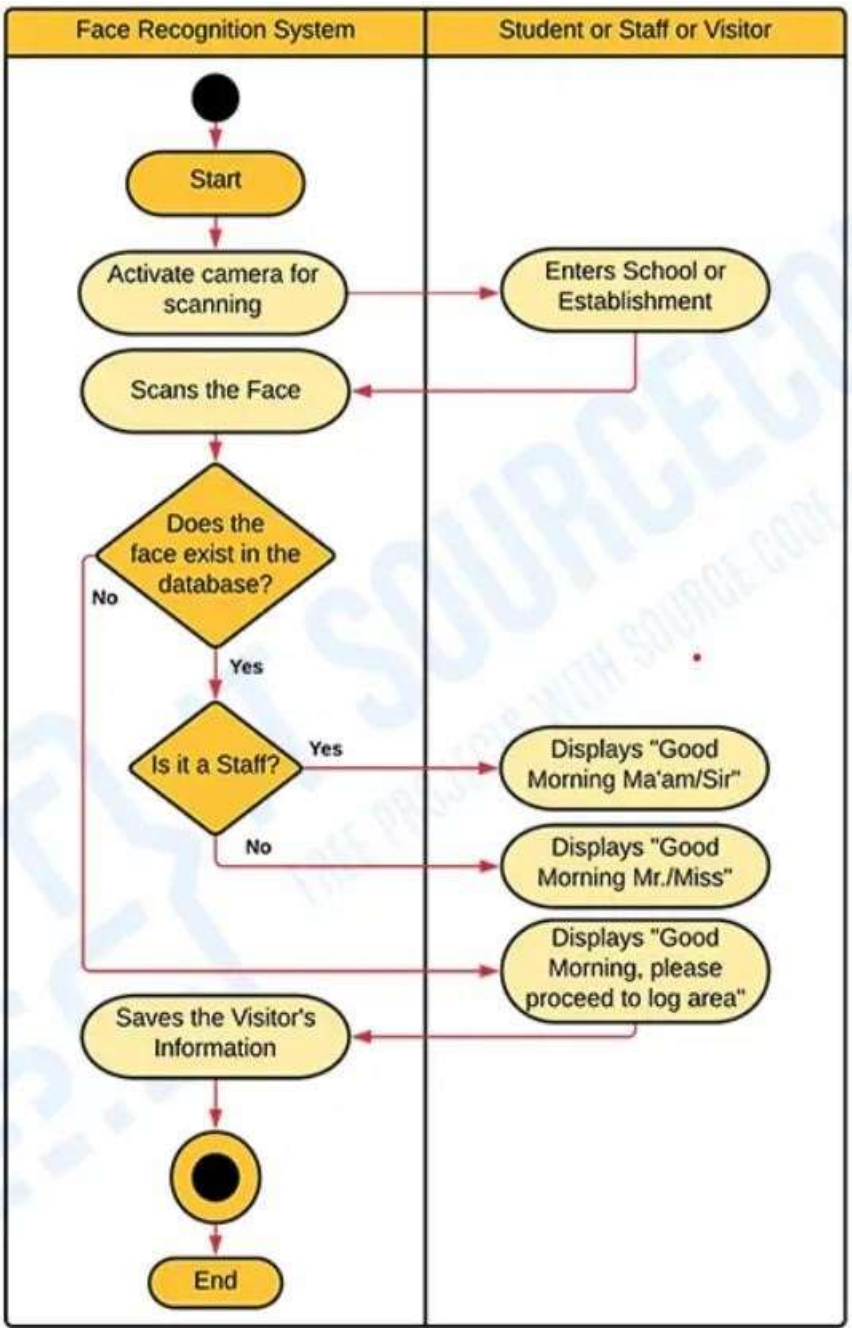
The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. With the advanced visual modelling capability, you can create complex sequence diagram in few clicks. Besides, VP-UML can generate sequence diagram from the flow of events which you have defined in the use case description.



5.4.4 Collaboration Diagrams:



5.4.5 Activity Diagram:



5.5 E-R Diagram

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. An entity-relationship model (ERM) in software engineering is an abstract and conceptual representation of data. Entity-relationship modelling is a relational schema database modelling method, used to produce a type of conceptual schema or semantic data model of a system, of a relational database.

After carefully understanding the requirements of the client the entire data storage requirements are divided into tables. The below tables are normalized to avoid any anomalies during the course of data entry.

Normalization

It is a process of converting a relation to a standard form. The process is used to handle the problems that can arise due to data redundancy i.e. repetition of data in the database, maintain data integrity as well as handling problems that can arise due to insertion, updating, deletion anomalies.

Decomposing is the process of splitting relations into multiple relations to eliminate anomalies and maintain anomalies and maintain data integrity. To do this we use normal forms or rules for structuring relation.

Insertion anomaly: Inability to add data to the database due to absence of other

data.

Deletion anomaly: Unintended loss of data due to deletion of other data.

Update anomaly: Data inconsistency resulting from data redundancy and partial

update

Normal Forms: These are the rules for structuring relations that eliminate

anomalies.

FIRST NORMAL FORM:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

SECOND NORMAL FORM:

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

1) Primary key is a not a composite primary key

2) No non key attributes are present

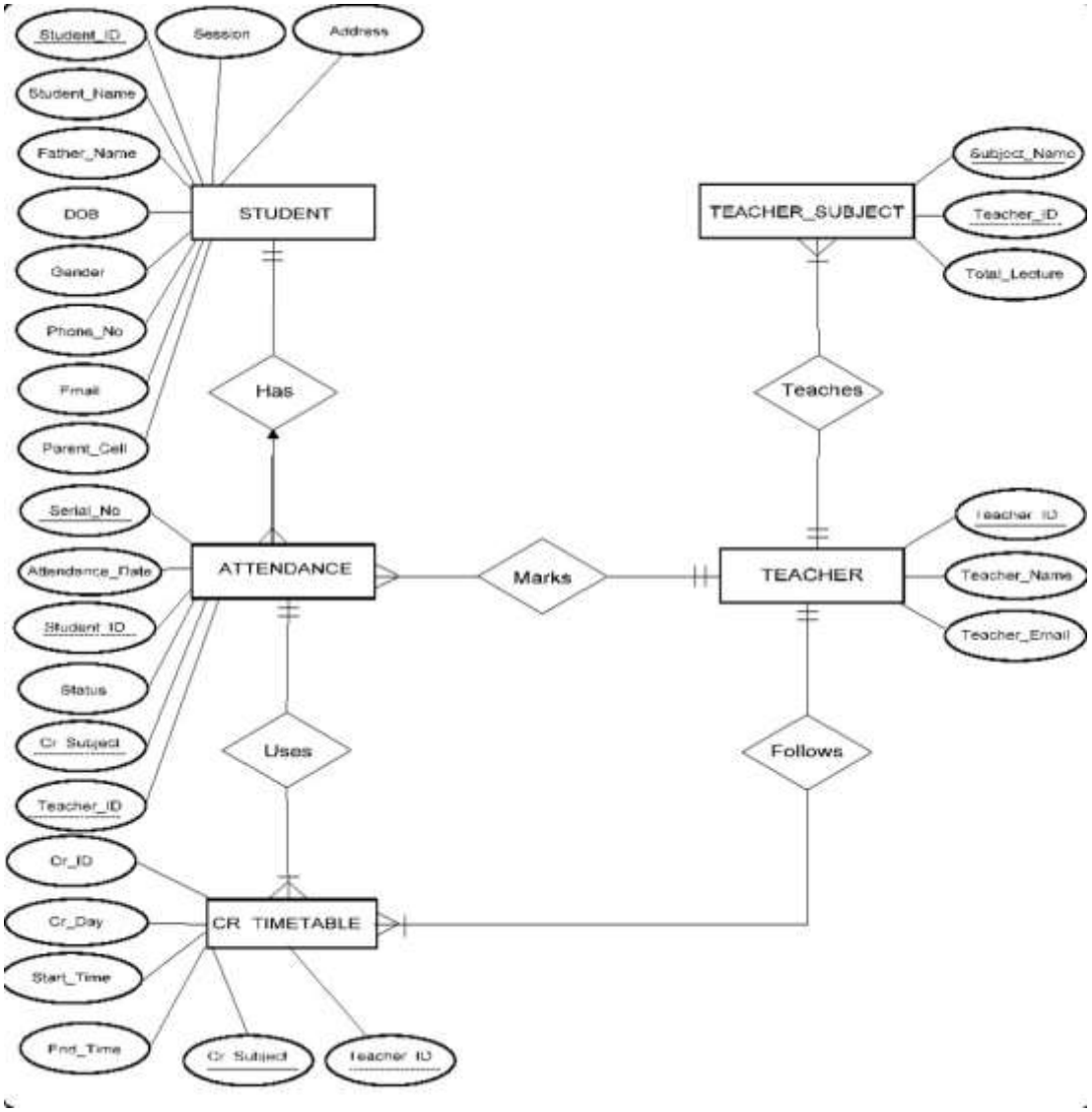
3) Every non key attribute is fully functionally dependent on full set of primary

Key.

THIRD NORMAL FORM:

A relation is said to be in third normal form if their exits no transitive dependencies.

Transitive Dependency: If two non-key attributes depend on each other as well as on the primary key then they are said to be transitively dependent. The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.



CHAPTER-6

TECHNOLOGY DESCRIPTION AND CODING

6.1 Technology Description

The attendance management system using face recognition technology leverages various technologies and tools to provide accurate and efficient automated attendance tracking. Here's a description of the key technologies used in this project:

Python Programming Language

Python is an interpreted, high-level programming language that is widely used for its readability and ease of use. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, making it versatile for developing complex systems. It is the primary language used for implementing the algorithms and integrating the various components of the attendance management system.

Jupyter Notebook

Jupyter Notebook is an interactive web-based tool that supports creating and sharing documents containing live code, equations, visualizations, and narrative text. It is used for prototyping and experimenting with machine learning algorithms, data analysis, and visualization. In this project, Jupyter Notebook is employed for developing and testing the face recognition models and conducting exploratory data analysis.

TensorFlow and Keras

TensorFlow is an open-source machine learning framework developed by Google, providing robust infrastructure for building and deploying machine learning models. Keras is a high-level API running on top of TensorFlow, simplifying the process of creating and training neural networks. In the context of face recognition, TensorFlow and Keras are used for developing Convolutional Neural Networks (CNNs) to detect and recognize faces accurately.

NumPy

NumPy is a fundamental Python library for numerical computations. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is crucial for handling and processing image data efficiently in the face recognition pipeline.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source library that provides tools for image processing and computer vision. It is used for tasks such as image acquisition, preprocessing, face detection, and feature extraction. OpenCV's pre-trained face detection models and real-time processing capabilities are integral to the system's functionality.

Matplotlib

Matplotlib is a data visualization library used to create static, animated, and interactive visualizations in Python. In this project, Matplotlib is used to visualize training and evaluation metrics, such as accuracy and loss curves, as well as to display sample images for debugging and analysis.

Pandas

Pandas is a data manipulation and analysis library that provides data structures like Series and DataFrame for handling and analyzing structured data. In the attendance management system, Pandas is used for managing attendance records, processing data from the database, and generating reports.

Convolutional Neural Networks (CNNs)

CNNs are a class of deep learning algorithms specifically designed for image recognition and classification tasks. CNNs are employed in the face recognition system to learn and extract features from facial images, enabling the accurate identification of individuals. Key components of CNNs include convolutional layers, pooling layers, and fully connected layers.

Eigenfaces and Principal Component Analysis (PCA)

Eigenfaces and PCA are used for dimensionality reduction in facial recognition tasks. PCA helps reduce the complexity of face images by projecting them onto a lower-dimensional space, while Eigenfaces represent the principal components of face images that capture the most variance in the data. These techniques improve the efficiency and accuracy of the face recognition model.

Flask

Flask is a lightweight web framework for Python used to build web applications. In this project, Flask is employed to create the web interface for the attendance management system, allowing users to upload images, view attendance records, and interact with the system.

SQL Database

A SQL database is used to store and manage user data, attendance records, and other relevant information. The database allows for efficient querying and retrieval of data, supporting the backend services of the attendance management system.

Cloud Infrastructure

Cloud infrastructure is used for hosting and scaling the application. Cloud platforms like AWS, Google Cloud, or Azure provide resources for deploying the application, managing storage, and handling computational requirements. Cloud-based storage solutions are used for saving images and model files, while cloud computing resources enable efficient model training and inference.

By integrating these technologies, the attendance management system aims to provide an automated, accurate, and user-friendly solution for tracking attendance using face recognition.

6.2 Coding

```
# importing required libraries

from tkinter import. *

import tkinter as tk

from tkinter import Message ,Text

import cv2,os

import shutil

import csv

import numpy as np

from PIL import Image, ImageTk

import pandas as pd

import datetime

import time

import tkinter.ttk as ttk

import tkinter.font as font

window = tk.Tk()

#helv36 = tk.Font(family='Helvetica', size=36, weight='bold')
```

```
window.title("Face_Recogniser")

dialog_title = 'QUIT'

dialog_text = 'Are you sure?'

#answer = messagebox.askquestion(dialog_title, dialog_text)

#window.geometry('1280x720')

window.configure(background="blue")

##bg = PhotoImage(file = r"C:\Users\mouni\Desktop\Face-Recognition-Based-Attendance-System-master 1\bg.jpg")

##bg = PhotoImage(file = "Your_image.png")

#filename = PhotoImage(file = r"C:\Users\mouni\Desktop\Face-Recognition-Based-Attendance-System-master 1\bg.jpg")

#background_label = window(top, image=filename)

# Show image using label

##label1 = Label( root, image = bg)

##label1.place(x = 0, y = 0)

#window.attributes('-fullscreen', True)

window.grid_rowconfigure(0, weight=1)

window.grid_columnconfigure(0, weight=1)

#path = "profile.jpg"

#Creates a Tkinter-compatible photo image, which can be used everywhere Tkinter expects an image object.

#img = ImageTk.PhotoImage(Image.open(path))

#The Label widget is a standard Tkinter widget used to display a text or image on the screen.

#panel = tk.Label(window, image = img)

#panel.pack(side = "left", fill = "y", expand = "no")

#cv_img = cv2.imread("img541.jpg")

#x, y, no_channels = cv_img.shape

#canvas = tk.Canvas(window, width = x, height =y)
```

```
#canvas.pack(side="left")

#photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(cv_img))

# Add a PhotoImage to the Canvas

#canvas.create_image(0, 0, image=photo, anchor=tk.NW)

#msg = Message(window, text='Hello, world!')

# Font is a tuple of (font_family, size_in_points, style_modifier_string)

message =tk.Label(window, text="Face-Recognition-Based-Attendance-Management-System" ,bg="Green" ,fg="white" ,width=50 ,height=3,font=('times', 30, 'italic bold underline'))

message.place(x=200, y=20)

lbl = tk.Label(window, text="Enter ID",width=20  ,height=2  ,fg="red"  ,bg="yellow" ,font=('times', 15, ' bold ') )

lbl.place(x=400, y=200)

txt = tk.Entry(window,width=20  ,bg="yellow" ,fg="red",font=('times', 15, ' bold '))

txt.place(x=700, y=215)

lbl2 = tk.Label(window, text="Enter Name",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold '))

lbl2.place(x=400, y=300)

txt2 = tk.Entry(window,width=20  ,bg="yellow"  ,fg="red",font=('times', 15, ' bold ')  )

txt2.place(x=700, y=315)

lbl3 = tk.Label(window, text="Notification : ",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold underline '))

lbl3.place(x=400, y=400)

message = tk.Label(window, text="" ,bg="yellow"  ,fg="red"  ,width=30  ,height=2, activebackground = "yellow" ,font=('times', 15, ' bold '))

message.place(x=700, y=400)

lbl3 = tk.Label(window, text="Attendance : ",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold  underline'))

lbl3.place(x=400, y=650)
```

```
message2 = tk.Label(window, text="" ,fg="red"      ,bg="yellow",activeforeground = "green",width=30  ,height=2  ,font=('times', 15, ' bold '))

message2.place(x=700, y=650)

 def clear():

    txt.delete(0, 'end')

    res = ""

    message.configure(text= res)

def clear2():

    txt2.delete(0, 'end')

    res = ""

    message.configure(text= res)

def is_number(s):

    try:

        float(s)

        return True

    except ValueError:

        pass

    try:

        import unicodedata

        unicodedata.numeric(s)

        return True

    except (TypeError, ValueError):

        pass

    return False

def TakeImages():

    Id=(txt.get())

    name=(txt2.get())

    if(is_number(Id) and name.isalpha()):
```

```
cam = cv2.VideoCapture(0)
harcascadePath = "haarcascade_frontalface_default.xml"
detector=cv2.CascadeClassifier(harcascadePath)
sampleNum=0
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        #incrementing sample number
        sampleNum=sampleNum+1
        #saving the captured face in the dataset folder TrainingImage
        cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg",
gray[y:y+h,x:x+w])
        #display the frame
        cv2.imshow('frame',img)
    #wait for 100 miliseconds
if cv2.waitKey(100) & 0xFF == ord('q'):
        break
    # break if the sample number is morethan 100
elif sampleNum>15:
        break
cam.release()
cv2.destroyAllWindows()
res = "Images Saved for ID : " + Id +" Name : "+ name
row = [Id , name]
with open('StudentDetails\StudentDetails.csv','a+') as csvFile:
```

```
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message.configure(text= res)
    else:
        if(is_number(Id)):
            res = "Enter Alphabetical Name"
            message.configure(text= res)
        if(name.isalpha()):
            res = "Enter Numeric Id"
            message.configure(text= res)
    def TrainImages():
    print('start1')
                    recognizer=cv2.face_LBPHFaceRecognizer.create()#recognizer =
cv2.face.LBPHFaceRecognizer_create()#$cv2.createLBPHFaceRecognizer()
    print('start2')
    harcascadePath = "haarcascade_frontalface_default.xml"
    print('start3')
    detector =cv2.CascadeClassifier(harcascadePath)
    print('start4')
    faces,Id = getImagesAndLabels("TrainingImage")
    print('start5')
    recognizer.train(faces, np.array(Id))
    print('start6')
    recognizer.save("TrainingImageLabel\Trainner.yml")
    print('completed')
    res = "Image Trained"#+",".join(str(f) for f in Id)
    message.configure(text= res)
```

```
def getImagesAndLabels(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #print(imagePaths)
    #create empth face list
    faces=[]
    #create empty ID list
    Ids=[]
    #now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        #loading the image and converting it to gray scale
        pilImage=Image.open(imagePath).convert('L')
        #Now we are converting the PIL image into numpy array
        imageNp=np.array(pilImage,'uint8')
        #getting the Id from the image
        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(Id)
    return faces,Ids
def TrackImages():

        recognizer=cv2.face.LBPHFaceRecognizer_create()#cv2.createLBPHFaceRec
        ognizer()
    recognizer.read("TrainingImageLabel\Trainner.yml")
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);
```

```
df=pd.read_csv("StudentDetails\StudentDetails.csv")
cam = cv2.VideoCapture(0,cv2.CAP_DSHOW)
font = cv2.FONT_HERSHEY_SIMPLEX
col_names =  ['Id','Name','Date','Time']
attendance = pd.DataFrame(columns = col_names)
while True:
    ret, im =cam.read()
    gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    faces=faceCascade.detectMultiScale(gray, 1.2,5)
    for(x,y,w,h) in faces:
        cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)
        Id, conf = recognizer.predict(gray[y:y+h,x:x+w])
        if(conf < 50):
            ts = time.time()
            date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
            timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
            aa=df.loc[df['Id'] == Id]['Name'].values
            tt=str(Id)+"-"+aa
            attendance.loc[len(attendance)] = [Id,aa,date,timeStamp]
        else:
            Id='Unknown'
            tt=str(Id)
        if(conf > 75):
            noOfFile=len(os.listdir("ImagesUnknown"))+1
                        cv2.imwrite("ImagesUnknown\Image"+str(noOfFile)  +  ".jpg",
im[y:y+h,x:x+w])
        cv2.putText(im,str(tt),(x,y+h), font, 1,(255,255,255),2)
    attendance=attendance.drop_duplicates(subset=['Id'],keep='first')
```

```
        cv2.imshow('im',im)
        if (cv2.waitKey(1)==ord('q')):
            break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
    timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
    Hour,Minute,Second=timeStamp.split(":")
                    fileName="Attendance\Attendance_"+date+"_"+Hour+"-"+Minute+"-
"+Second+".csv"
    attendance.to_csv(fileName,index=False)
    cam.release()
    cv2.destroyAllWindows()
    #print(attendance)
    res=attendance
    message2.configure(text= res)
clearButton = tk.Button(window, text="Clear", command=clear ,fg="red"
 ,bg="yellow"  ,width=20  ,height=2 ,activebackground = "Red" ,font=('times', 15, '
bold '))
clearButton.place(x=950, y=200)
clearButton2 = tk.Button(window, text="Clear", command=clear2 ,fg="red"
 ,bg="yellow"  ,width=20  ,height=2, activebackground = "Red" ,font=('times', 15, '
bold '))
clearButton2.place(x=950, y=300)
takeImg = tk.Button(window, text="Take Images", command=TakeImages  ,fg="red"
 ,bg="yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, '
bold '))
takeImg.place(x=200, y=500)
trainImg = tk.Button(window, text="Train Images", command=TrainImages  ,fg="red"
 ,bg="yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, '
bold '))
trainImg.place(x=500, y=500)
```

```
trackImg = tk.Button(window, text="Track Images", command=TrackImages ,fg="red" ,bg="yellow" ,width=20 ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

trackImg.place(x=800, y=500)

quitWindow = tk.Button(window, text="Quit", command=window.destroy ,fg="red" ,bg="yellow" ,width=20 ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

quitWindow.place(x=1100, y=500)

copyWrite = tk.Text(window, background=window.cget("background"), borderwidth=0,font=('times', 30, 'italic bold underline'))

copyWrite.tag_configure("superscript", offset=10)

copyWrite.insert("insert", "Developed by lohith","", "TEAM", "superscript")

copyWrite.configure(state="disabled",fg="red" )

copyWrite.pack(side="left")

copyWrite.place(x=800, y=750)

window.mainloop()

# SET UP

from cx_Freeze import setup, Executable

import sys,os

PYTHON_INSTALL_DIR = os.path.dirname(os.path.dirname(os.__file__))

os.environ['TCL_LIBRARY'] = os.path.join(PYTHON_INSTALL_DIR, 'tcl', 'tcl8.6')

os.environ['TK_LIBRARY'] = os.path.join(PYTHON_INSTALL_DIR, 'tcl', 'tk8.6')

base = None

if sys.platform == 'win32':

    base = None

executables = [Executable("train.py", base=base)]

packages = ["idna","os","sys","cx_Freeze","tkinter","cv2","setup",

          "numpy","PIL","pandas","datetime","time"]

options = {
```

```
    'build_exe': {
        'packages':packages,
    },
}
setup(
    name = "ToolBox",
    options = options,
    version = "0.0.1",
    description = 'Vision ToolBox',
    executables = executables
)
```

CHAPTER 7

TESTING

7.1 Testing

The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied.

potato plant disease detection project, incorporating various types of testing ensures that the system functions correctly and meets user expectations. Here's how each testing type applies to your project:

7.2 Testing Types

7.2.1 Unit Testing

Unit testing in your project focuses on validating the smallest testable parts of the code, such as functions for image preprocessing, model training, and inference. These tests are designed to check if individual components, like image augmentation or CNN layers, perform as expected in isolation. For example, you might test functions responsible for resizing images or applying data augmentation to ensure they work correctly. Tools like pytest for Python can be used to automate these tests, allowing you to quickly identify and fix bugs during development. The primary benefits include early bug detection and simplifying debugging.

7.2.2 Integration Testing

Integration testing ensures that various components of your system, such as data preprocessing, feature extraction, and model inference, work together seamlessly. This involves testing interactions between modules to detect issues that might arise when these components are combined. For instance, you would test if the output from the image preprocessing module is correctly fed into the CNN model and if the model's predictions are accurately processed and displayed by the user interface. Tools like Selenium can be used for testing web interfaces, while TestNG can help manage tests for integrated components.

7.2.3 System Testing

System testing evaluates the complete and integrated software to ensure it meets all specified requirements. For your project, this would involve testing the entire workflow from image capture to disease classification and result display. System testing checks both functional aspects (e.g., accurate disease detection) and non-functional aspects (e.g., performance and usability). Tools like HP ALM or JIRA can

manage and execute these comprehensive tests, ensuring the entire system operates as expected and is ready for deployment.

7.2.4 Acceptance Testing

Acceptance testing, particularly User Acceptance Testing (UAT), is conducted to verify that the system meets the business requirements and user expectations. In your project, this would involve having real users (e.g., farmers or agricultural experts) test the system to ensure it provides accurate disease predictions and is user-friendly. UAT ensures the software is fit for its intended purpose and ready for production deployment. Tools like TestRail or QAComplete can facilitate the planning and execution of these tests, ensuring the system aligns with user needs and expectations.

7.2.5 Performance Testing

Performance testing assesses how well the system performs under various conditions, including load, stress, and scalability. For your project, this involves testing how the system handles multiple concurrent image uploads and model inference requests. Load testing ensures the system can handle the expected user load, stress testing evaluates its performance under extreme conditions, and scalability testing checks its ability to handle increased demand. Tools like JMeter and LoadRunner can be used to simulate different usage scenarios and identify performance bottlenecks, ensuring the system remains reliable and efficient under varying loads.

By applying these testing types, we had ensured that our potato plant disease detection system is robust, reliable, and ready to provide accurate and useful insights to users.

Test Results: All the test cases mentioned above passed successfully. No defects

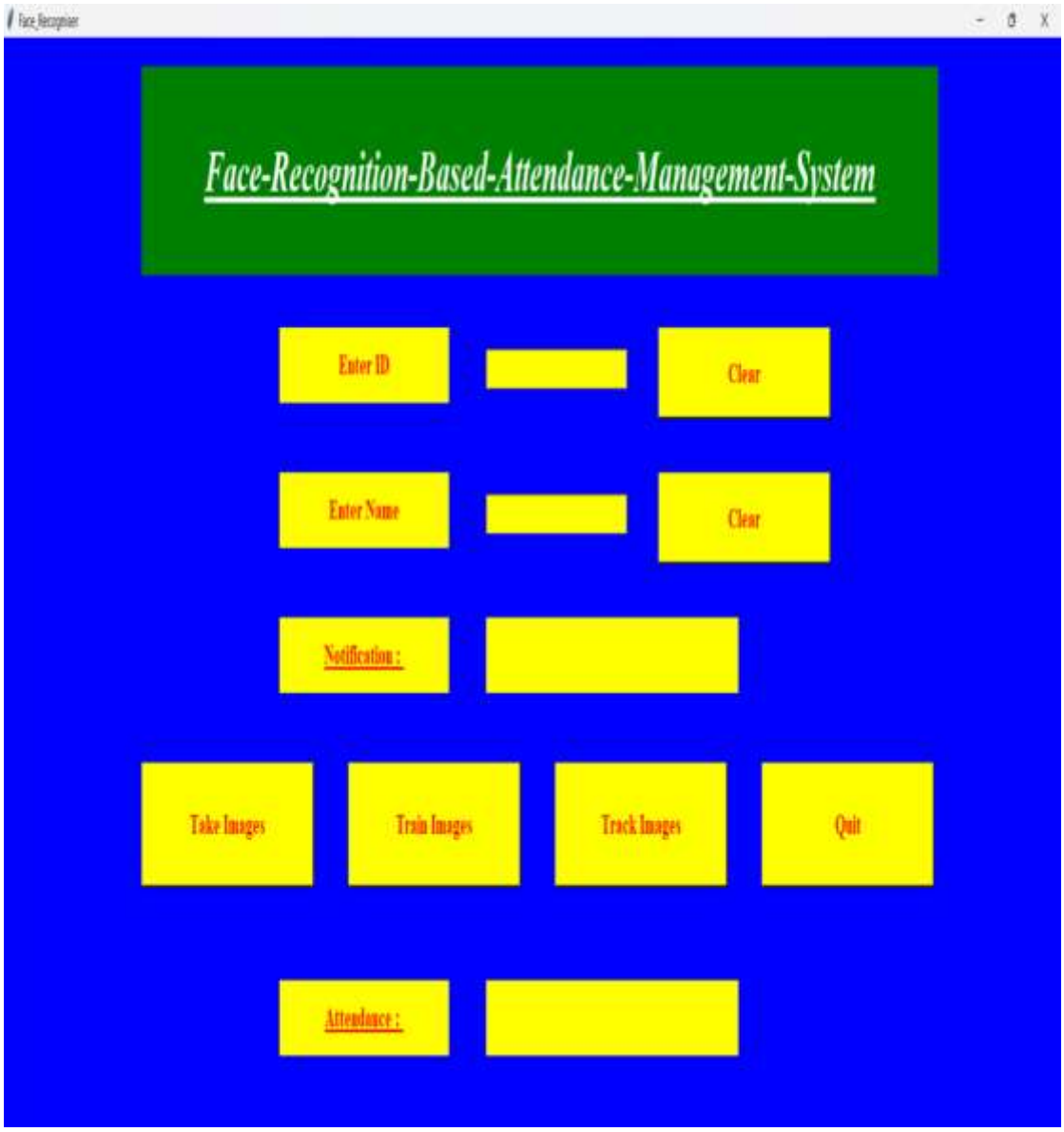
encountered.

7.3 Test Cases

Test Case ID	Test Case Description	Expected Result	Actual Result	Status
TC-01	Unit testing of face detection module	Correct identification and bounding of faces in images	Correct identification and bounding	Pass
TC-02	Unit testing of face recognition model	Model accurately recognizes and matches faces	Accurate recognition and matching	Pass
TC-03	Integration test between image capture and recognition	Smooth data flow from image upload to face recognition	Smooth data flow and recognition	Pass
TC-04	System testing with various images (different lighting, angles, etc.)	Consistent accuracy in recognizing faces across conditions	Consistent accuracy	Pass
TC-05	Performance testing with large datasets	System maintains acceptable performance and accuracy	Acceptable performance and accuracy	Pass
TC-06	User Acceptance Testing (UAT)	User satisfaction with the UI, features, and functionality	User satisfaction and readiness	Pass

CHAPTER 8

SCREEN SHOTS



CONCLUSION AND FUTURE SCOPE

Conclusion

The attendance management system project marks a notable advancement in automating attendance tracking through state-of-the-art face recognition technology. Utilizing Convolutional Neural Networks (CNNs), the system ensures precise and efficient face recognition, significantly reducing manual errors. Its real-time processing capability allows for instantaneous attendance recording, keeping records up-to-date without delays. The system's design features a user-friendly web interface and mobile app, which enhance accessibility and convenience. Additionally, robust security measures, including secure data handling and encryption, safeguard sensitive user information and adhere to data privacy regulations.

Future Scope

Looking ahead, the attendance management system presents several avenues for enhancement. Integration with other organizational systems such as payroll, HR management, and student information systems could streamline administrative processes. Further improvements in the face recognition model could enhance accuracy and robustness, particularly under varying conditions. Multi-modal authentication methods, such as fingerprint recognition or RFID, could provide additional security layers. The application of predictive analytics could offer insights into attendance patterns and assist in resource planning. Expanding scalability to accommodate large institutions and implementing real-time notifications for attendance anomalies would increase the system's effectiveness. Additionally, leveraging cloud technologies for improved scalability and ease of maintenance could further optimize the system.

Bibliography

1. "Deep Learning for Computer Vision" by Rajalingappaa Shanmugamani
2. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron
3. "Computer Vision: Algorithms and Applications" by Richard Szeliski
4. "Learning OpenCV 4: Computer Vision with Python" by Gary Bradski and Adrian Kaehler
5. "Digital Image Processing" by Rafael C. Gonzalez and Richard E.