DEVELOPING A ROBUST FACIAL RECOGNITION SYSTEM FOR SECURE EMPLOYEE IDENTIFICATION,FOCUSING ON OPTMIZING ACCURACY AND PERFORMANCE IN VARYING

A dissertation submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

In

Computer Science and Engineering

By

KOPPISETTY BHAVYASRI (23U61A0583)

Under the guidance of

Mrs. Noore Ilahi
B. Tech., M. Tech.
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(Approved by AICTE, New Delhi & Affiliated to JNTUH)
(Recognized under section 2(f) of UGC Act 1956)
An ISO:9001-2015 Certified Institution
CHILKUR (V), MOINABAD (M), R.R. DIST. T.S-501504
June 2025

GLOBAL
Institute of Engineering & Technology
Approved By AICTE, Affiliated To JNTUH EAMCET CODE: GLOB
A+ GRADE
EDUCATION EXCELLENCE AWARD 2022
ISO
info@globalhyd.edu.in www.globalhyd.edu.in

(Approved by AICTE & Affiliated to JNTUH)
(Recognized under Section 2(f) of UGC Act 1956)
An ISO:9001-2015 Certified Institution
Survey No. 179, Chilkur (V), Moinabad (M), Ranga Reddy Dist. TS.
JNTUH Code (U6) ECE –EEE-CSD-CSM – CSE - CIVIL – ME – MBA - M.Tech EAMCET Code - (GLOB)

Department of Computer Science and Engineering

Noore Ilahi
B. Tech., M. Tech.
Assistant Professor & Head

Date: 02-06-2025

CERTIFICATE

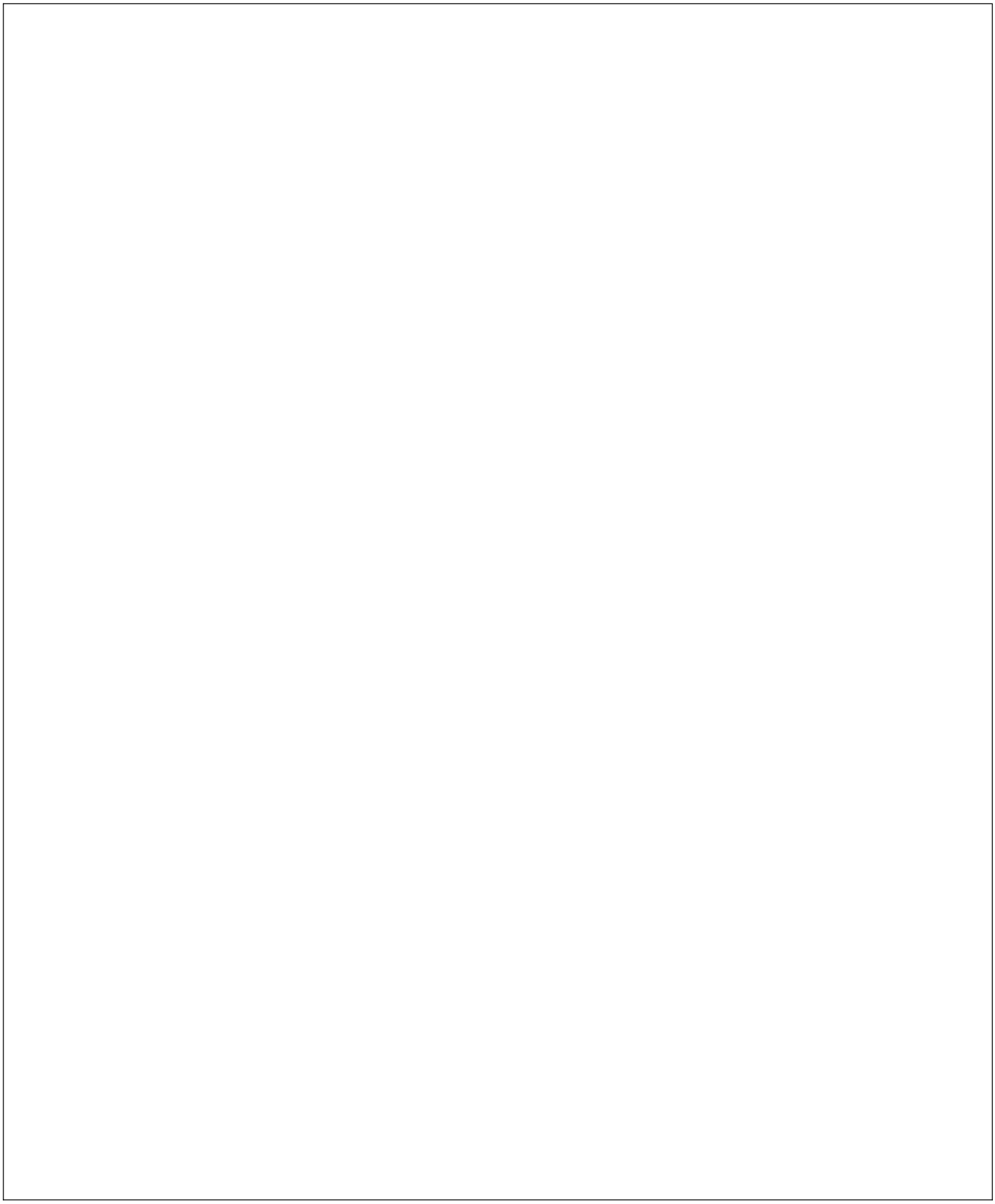
This is to certify that the project work entitled "**Developing a robust facial recognition system for secure employee identification,focusing on optimizing accuracy and performance in varying**", is a bonafide work of **Koppisetty Bhavyasri (HT.No:23U61A0583),** submitted in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering** during the academic year 2024-25. This is further certified that the work done under my guidance, and the results of this work have not been submitted elsewhere for the award of any other degree or diploma.

Internal Guide

Mrs. Noore Ilahi
Assistant Professor
Assistant Professor

Head of the Department

Mrs. Noore Ilahi



DECLARATION

I hereby declare that the project work entitled **Developing a robust facial recognition system for secure employee identification,focusing on optimizing accuracy and performance in varying,** submitted to **Department of Computer Science and Engineering, Global Institute of Engineering & Technology, Moinabad,** affiliated to **JNTUH, Hyderabad** in partial fulfillment of the requirement forthe award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** is the work done by me and has not been submitted elsewhere for the awardof any degree or diploma.

Koppisetty Bhavyasri (23U61A0583)

ACKNOWLEDGEMENT

I am thankful to my guide **Mrs. Noore Ilahi,** Assistant Professor of CSE Department for her valuable guidance for successful completion of this project.

I express my sincere thanks to **Mrs. G. Pavani,** Project Coordinator for giving me an opportunity to undertake the project "**Developing a robust facial recognition system for secure employee identification,focusing on optimizing accuracy and performance in varying**" and for enlightening me on various aspects of my project work and assistance in the evaluation of material and facts. She not only encouraged me to take up this topic but also given her valuable guidance in assessing facts and arriving at conclusions.

I am also most obliged and grateful to **Mrs. Noore Ilahi,** Assistant Professor and Head, Department of CSE for giving me guidance in completing this project successfully.

I express my heart-felt gratitude to our Vice-Principal **Prof. Dr. G Ahmed Zeeshan**, Coordinator Internal Quality Assurance Cell (IQAC) for his constant guidance, cooperation, motivation and support which have always kept me going ahead. I owe a lot of gratitude to him for always being there for me.

I also most obliged and grateful to our Principal **Dr. P. Raja Rao** for giving me guidancein completing this project successfully.

I also thank my parents for their constant encourage and support without which the projectwould have not come to an end.

Last but not the least, I would also like to thank all my class mates who have extended theircooperation during our project work.

Koppisetty Bhavyasri (23U61A0583)

VISION

The Vision of the Department is to produce professional Computer Science Engineers who can meet the expectations of the globe and contribute to the advancement of engineering and technology which involves creativity and innovations by providing an excellent learning environment with the best quality facilities.

MISSION

M1. To provide the students with a practical and qualitative education in a modern technical environment that will help to improve their abilities and skills in solving programming problems effectively with different ideas and knowledge.

M2. To infuse the scientific temper in the students towards the research and development in Computer Science and Engineering trends.

M3. To mould the graduates to assume leadership roles by possessing good communication skills, an appreciation for their social and ethical responsibility in a global setting, and the ability to work effectively as team members.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in MNC's and / or to pursue postgraduate studies with an appreciation for lifelong learning.

PEO2: To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

PEO3: To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting edge technologies of multi disciplinary nature for societal development.

PROGRAMME OUTCOMES:

PO1: Engineering knowledge: An ability toApply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: An ability to Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.

PO3: Design/development of solutions: An ability to Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.

PO4: Conduct investigations of complex problems: An ability to Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: An ability to Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: An ability to Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment sustainability: An ability to Understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.

PO8: Ethics: An ability to Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and teamwork: An ability to Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: An ability to Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: An ability to Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Lifelong learning: An ability to Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

PROGRAMME SPECIFIC OUTCOMES

PSO1: An Ability to Apply the fundamentals of mathematics, Computer Science and Engineering Knowledge to analyze and develop computer programs in the areas related to Algorithms, System Software, Web Designing, Networking and Data mining for efficient Design of computer-based system to deal with Real time Problems.

PSO2: An Ability to implement the Professional Engineering solutions for the betterment of Society, and able to communicate with professional Ethics effectively

ABSTRACT

This project presents a robust facial recognition system designed for secure employee identification and attendance management, with a focus on optimizing recognition accuracy and system performance under varying environmental conditions. Leveraging the power of computer vision and deep learning-based facial encoding, the system captures facial features in real-time using a webcam, encodes them into numerical vectors, and performs face matching against a pre-registered employee dataset to authenticate identity.

The application is deployed through a user-friendly Flask-based web interface that supports two core functionalities: employee registration and secure login. During registration, the system captures a snapshot of the employee's face and stores both the image and associated metadata (ID, name) in a structured format. During login, it scans and matches live facial input with stored encodings to verify the employee and log attendance automatically. The backend processes handle image capture, encoding generation, face matching, and attendance logging in CSV format, ensuring lightweight and efficient operation.

The system architecture emphasizes modularity, enabling future scalability and enhancements such as multi-camera support, integration with employee management systems, or cloud-based storage. Emphasis is also placed on adaptability to different lighting conditions and angles, with potential for improving robustness through real-time feedback and dynamic encoding augmentation.

While the current implementation relies on static image capture and offline encoding using the `face_recognition` library, future enhancements may include real-time video-based detection, liveness checks for spoof prevention, and machine learning-based accuracy tuning. The project contributes to the growing demand for secure, touchless biometric systems in organizational environments and serves as a foundational platform for advanced identity verification solutions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In today's fast-paced digital work environments, organizations are increasingly seeking secure, efficient, and contactless methods for verifying employee identity and managing attendance. Traditional authentication mechanisms such as ID cards, passwords, or biometric fingerprints are either susceptible to fraud, easily misplaced, or non-hygienic. As businesses adopt hybrid and flexible work models, there is a growing need for intelligent systems that can accurately recognize personnel while minimizing administrative overhead.

Facial recognition technology has emerged as a compelling solution, offering rapid, non-intrusive identification by analyzing unique facial features. Powered by advancements in deep learning and computer vision, facial recognition systems are capable of delivering high accuracy under varying lighting, angles, and facial expressions. When integrated with real-time processing and secure storage mechanisms, these systems become powerful tools for identity verification, access control, and attendance tracking.

This project aims to develop a facial recognition-based employee identification system, designed to authenticate individuals and record their attendance with minimal human intervention. Deployed through a Flask-based web interface, the system captures and processes live webcam input, compares facial encodings against a pre-registered employee dataset, and logs attendance data into structured files. The platform emphasizes modularity, scalability, and accuracy, making it suitable for deployment in small to medium-scale enterprises.

1.1 EXISTING SYSTEM

Existing attendance systems primarily rely on manual sign-ins, RFID card swipes, or biometric fingerprint scanning. While functional, these systems face various limitations:

- **Manual Logs**: Prone to human error, time-consuming, and easily manipulated.
- **RFID Cards**: Require physical possession, can be lost or swapped between individuals.
- **Fingerprint Scanners**: Pose hygiene concerns and may fail due to physical damage or moisture on the skin.

With the advent of AI, modern systems are exploring facial recognition to address these limitations. Some commercial attendance solutions incorporate facial identification; however, they are often closed-source, costly, and lack flexibility for academic or development purposes. Additionally, many systems do not expose the internal facial encoding pipeline, limiting their customizability or integration with other employee management software.

Recent advances in libraries like `face_recognition`, OpenCV, and Dlib, paired with frameworks like Flask, have made it feasible to build custom, lightweight facial authentication systems. Despite this, challenges such as model robustness, image quality, and real-time performance remain active areas of exploration.

1.2 PROPOSED SYSTEM

The proposed **Facial Recognition-Based Employee Identification System** enables real-time identity verification and attendance logging using facial biometrics. The system architecture begins with a webcam-based image capture, where a user's facial image is acquired either during registration or authentication. This image is processed through a facial landmark detection and encoding phase, where a 128-dimensional face vector is generated using a deep neural network model.

During **employee registration**, facial encodings are extracted and stored alongside metadata such as the employee's name and ID. In the **login/attendance** phase, a new live capture is encoded and compared against the existing dataset. If a match is found, the system verifies the identity and logs the timestamped attendance into a CSV file.

The system's backend is implemented using Flask and Python, while core functionalities rely on open-source libraries like `face_recognition`, OpenCV for image handling, and NumPy/Pandas for data manipulation. The web frontend is built using HTML, CSS, and Bootstrap to ensure responsiveness and ease of use.

Key features of the proposed system include:

- Face detection and alignment using HOG/CNN models.
- Facial feature encoding and matching with Euclidean distance thresholds.
- Automatic attendance logging with date and time.
- Modular backend design for easy extension or migration to cloud environments.

1.3 ADVANTAGES OF THE PROPOSED SYSTEM

- **Contactless Authentication**: Eliminates the need for physical touch, improving hygiene and user convenience.
- **High Accuracy**: Utilizes deep learning-based facial encodings for reliable identification.
- **Real-Time Verification**: Processes and matches facial inputs instantly using live webcam streams.
- **Modular & Scalable Design**: Backend components are independently structured, supporting future upgrades (e.g., liveness detection or multi-user support).
- **Lightweight Deployment**: Built with Flask and open-source libraries, making it portable and easy to deploy on local or cloud servers.
- **User-Friendly Interface**: Bootstrap-based frontend with intuitive features for registration and attendance viewing.
- **Secure Session Handling**: No persistent storage of sensitive user data between sessions unless explicitly configured, preserving privacy.
- **Educational & Research Utility**: The open nature of the system makes it ideal for learning and experimentation with face recognition pipelines.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Facial recognition systems have rapidly advanced in recent years, driven by breakthroughs in deep learning, computer vision, and image processing. Traditionally, biometric systems relied on fingerprints or iris scans for identification; however, these methods often require physical contact and can suffer from environmental limitations. Facial recognition provides a contactless, efficient alternative, capable of functioning under real-world conditions such as varying illumination, occlusions, and changes in facial expressions.

This chapter surveys foundational and contemporary research in the domain of facial recognition, including algorithms for face detection and encoding, system architectures for identity verification, and tools used in building web-based biometric applications. These insights form the foundation for the design and implementation of the proposed employee identification system.

2.2 Existing Work on Facial Recognition Systems

Significant research efforts have been made to improve facial recognition accuracy and robustness. The key techniques can be grouped into detection, feature extraction, and classification:

- **Viola-Jones Face Detector (2001)**
  One of the earliest and fastest face detection algorithms using Haar-like features and an AdaBoost classifier. It was widely used in early surveillance and access control systems but struggles with non-frontal faces and low light.
- **HOG + SVM Pipeline (Dalal and Triggs, 2005)**
  Introduced Histogram of Oriented Gradients (HOG) features for object detection, including faces, using linear classifiers. It provides a balance between accuracy and speed and is still used in lightweight applications.
- **DeepFace (Taigman et al., 2014 – Facebook)**
  Introduced a deep neural network that reduced face verification errors to 3.9% on the LFW dataset. Utilized a 3D alignment process and nine-layer neural network, demonstrating that deep learning significantly outperforms traditional approaches.

- **FaceNet (Schroff et al., 2015 – Google)**
  Used triplet loss for training embeddings, enabling efficient face comparisons via Euclidean distance. This model set new benchmarks for face verification and is the basis for several open-source libraries, including `face_recognition`.
- **LFW (Labelled Faces in the Wild)**
  A benchmark dataset used to evaluate facial recognition systems in unconstrained environments. Most state-of-the-art models achieve >99% accuracy on this dataset.

These advancements led to highly accurate facial embedding systems that can extract robust feature vectors from raw face images, enabling fast and reliable identity matching.

2.3 Applications of Facial Recognition in Identity Systems

Facial recognition is widely used in a variety of domains:

- **Access Control and Attendance Management**
  Common in workplaces, schools, and high-security facilities, facial recognition systems replace conventional ID cards for secure, contactless entry and attendance logging.
- **Surveillance and Law Enforcement**
  Real-time facial scanning helps identify suspects or monitor crowd movement in public spaces.
- **Consumer Devices**
  Mobile devices (e.g., Apple's Face ID) and laptops use facial recognition for unlocking and secure app access.

Challenges such as occlusions (e.g., masks), age progression, and varying lighting conditions remain active areas of research. However, the integration of convolutional neural networks (CNNs) and pre-trained models like ResNet, VGG-Face, and MobileNet has significantly mitigated these limitations.

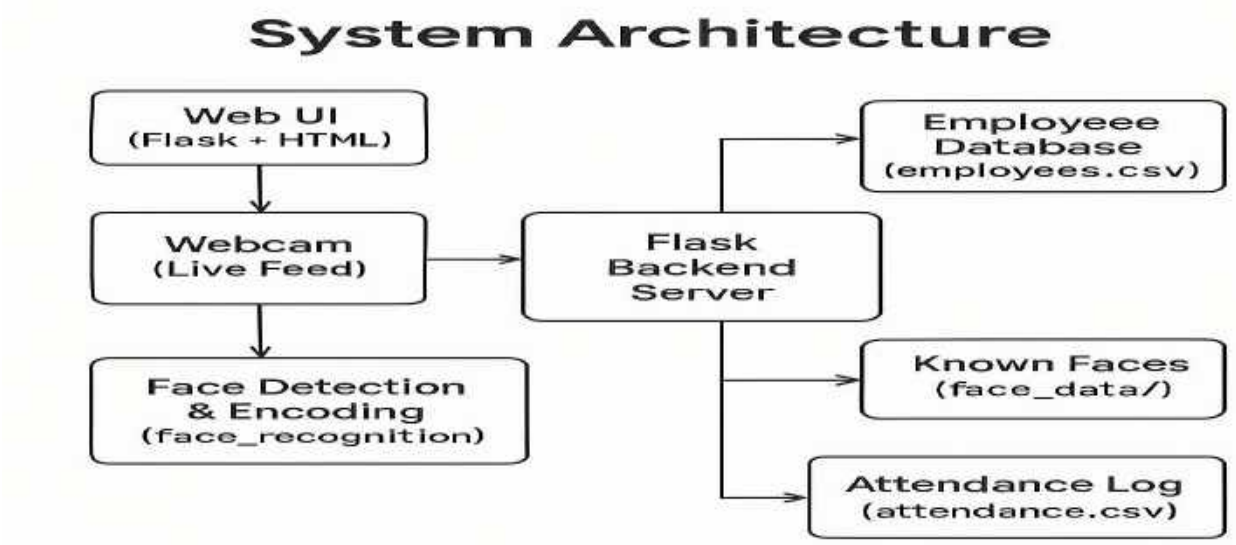
2.4 Tools and Technologies Used

- **Python**
  The primary programming language used for this project, valued for its readability and rich ecosystem of libraries.

- **Flask**
  A lightweight web framework used to build the system's backend and API endpoints, enabling real-time user interaction for login, registration, and attendance logging.
- **face_recognition Library**
  Built on dlib and FaceNet, it provides pre-trained deep learning models for face detection, encoding, and matching with high accuracy.
- **OpenCV**
  Used for webcam integration, image preprocessing, and real-time face detection during both registration and authentication phases.
- **NumPy & Pandas**
  Support numerical operations and structured data storage (e.g., employee records and attendance logs in CSV format).
- **Bootstrap**
  A responsive frontend toolkit used for creating intuitive and mobile-friendly user interfaces.

2.5 Summary

The literature reveals a clear shift from traditional biometric systems toward deep learning-based facial recognition, driven by the need for higher accuracy, user convenience, and contactless operation. This project builds on these findings by integrating real-time facial recognition with a Flask-based web interface. The selected tools and models offer a balance between performance, scalability, and accessibility, making the system suitable for deployment in small to medium-scale organizations.



CHAPTER 3
SYSTEM ANALYSIS

This project focuses on the development of a secure and efficient facial recognition-based employee identification system. It leverages computer vision and machine learning to automate attendance and access logging using facial features, eliminating the need for manual check-ins or biometric scans. The design emphasizes minimal user interaction, seamless real-time recognition, and browser-based accessibility for cross-platform compatibility.

3.1 REQUIREMENT SPECIFICATIONS

3.1.1 HARDWARE REQUIREMENTS

- **System**: Intel i5 or above, 3.2 GHz processor
- **Hard Disk**: Minimum 512 GB
- **Monitor**: 14" Color Monitor or larger
- **Input Devices**: Webcam (HD resolution), Optical Mouse, and Keyboard
- **RAM**: Minimum 8 GB

3.1.2 SOFTWARE REQUIREMENTS

- **Operating System**: Windows 11 / Windows 10 / Linux / macOS
- **Programming Language**: Python
- **Frontend Technologies**: HTML, CSS, JavaScript
- **Framework**: Flask (for web server and API endpoints)
- **Libraries / Packages**:
  - `face_recognition` (for facial embedding and recognition)
  - `OpenCV` (for camera and image processing)
  - `NumPy`, `Pandas` (for data manipulation and logging)
  - `Pickle` (for storing known face encodings)
  - `Bootstrap` (for responsive frontend design)
- **Browser Compatibility**: Compatible with all modern browsers (Recommended: Google Chrome or Firefox)

3.1.3 FUNCTIONAL REQUIREMENTS

- **Employee Registration**:
  - Capture face image via webcam
  - Store employee name and facial encoding

- **Employee Login / Attendance**:
  - Automatically detect and recognize employee faces using webcam
  - Record login/attendance timestamp in CSV or database
- **Web Interface Features**:
  - Clean UI for admin functions (register, delete, or view attendance records)
  - Real-time video feed with detection box and name overlay
- **Security**:
  - Encoded facial data stored securely; no raw images persisted
- **Platform Support**:
  - Accessible through a local web server on desktop or laptop systems

3.2 FEASIBILITY STUDY

The feasibility of the facial recognition-based attendance system was evaluated across key domains to ensure successful development and deployment.

3.2.1 ECONOMICAL FEASIBILITY

The system uses open-source tools and libraries, including Python, Flask, OpenCV, and `face_recognition`, minimizing the overall development cost. No specialized hardware (like biometric scanners) is required—only a webcam and internet-enabled computer. This makes the system highly cost-effective, especially for small and mid-sized organizations.

3.2.2 TECHNICAL FEASIBILITY

Technically, the project is feasible using current machine learning tools. It is implemented using Python, a widely used programming language for AI and computer vision tasks. The facial recognition functionality is handled through a pre-trained deep learning model (`face_recognition`), which is capable of high-accuracy identification even under varied lighting and angles. Flask ensures a lightweight backend server that performs well on standard systems.

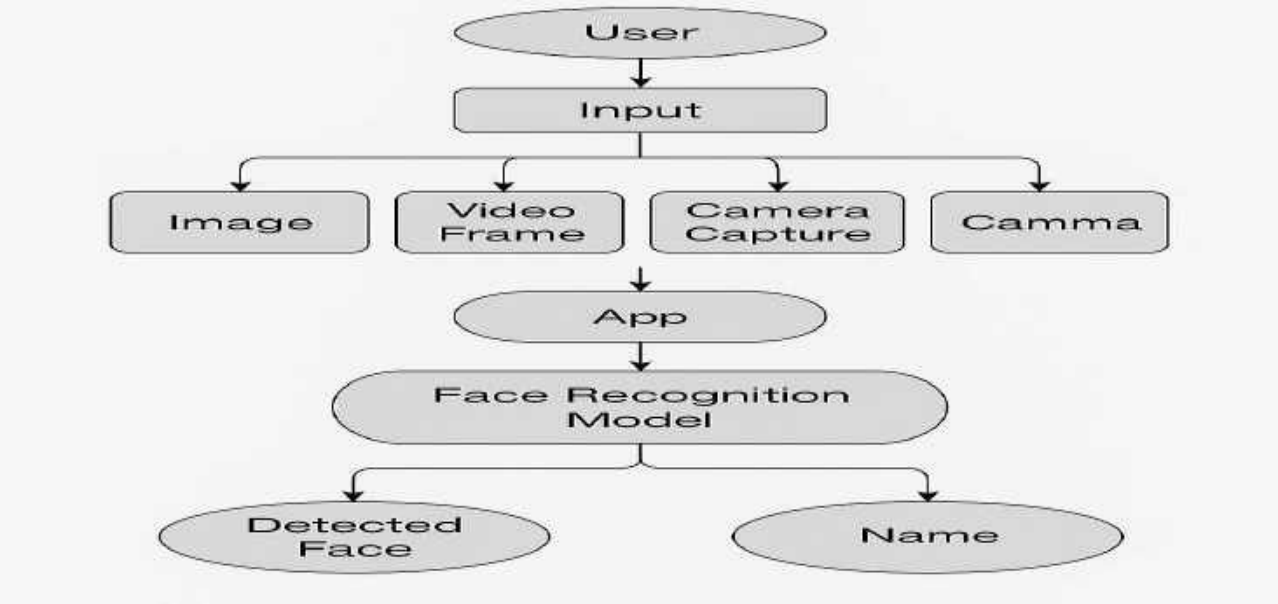
3.2.3 SOCIAL FEASIBILITY

The system requires minimal user training and is designed with simplicity and user-friendliness in mind. Since it removes the need for physical interaction (cards, pins, or fingerprints), employees experience faster and more hygienic attendance processes. Admins and HR personnel can easily manage records through the browser-based interface. This design encourages user acceptance and wide applicability across departments.

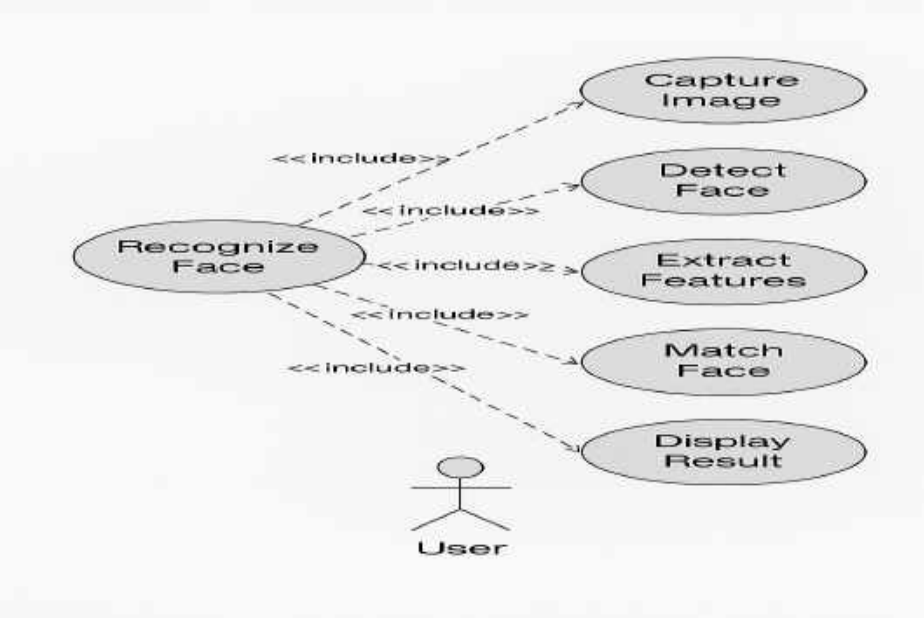
CHAPTER 4
SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM (DFD)

The Data Flow Diagram (DFD) represents how data moves through the facial recognition-based employee attendance system. The major steps involved are:



4.3 UML DIAGRAMS



4.3.1 USE CASE DIAGRAM

Actors:

- Employee
- Admin

Use Cases:

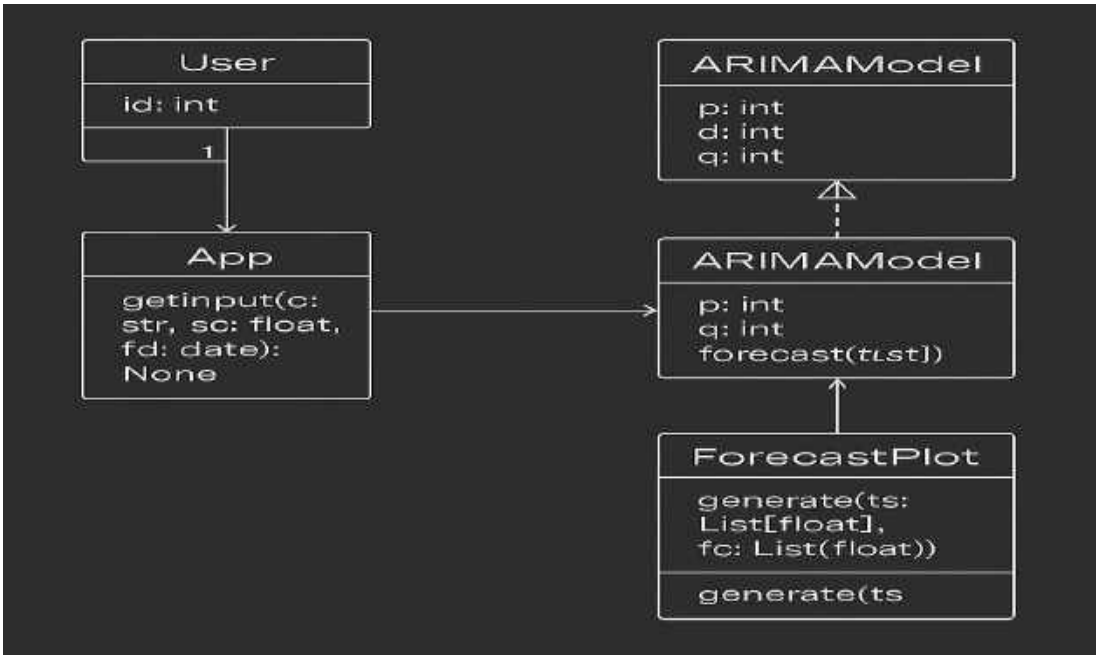
- **Employee**:
  - Show face to webcam
  - Get attendance marked
- **Admin**:
  - Add new employee face data
  - Delete employee data
  - View attendance logs

This diagram illustrates the interaction between system users and the main functionalities available through the web interface.

4.3.2 COMPONENT DIAGRAM

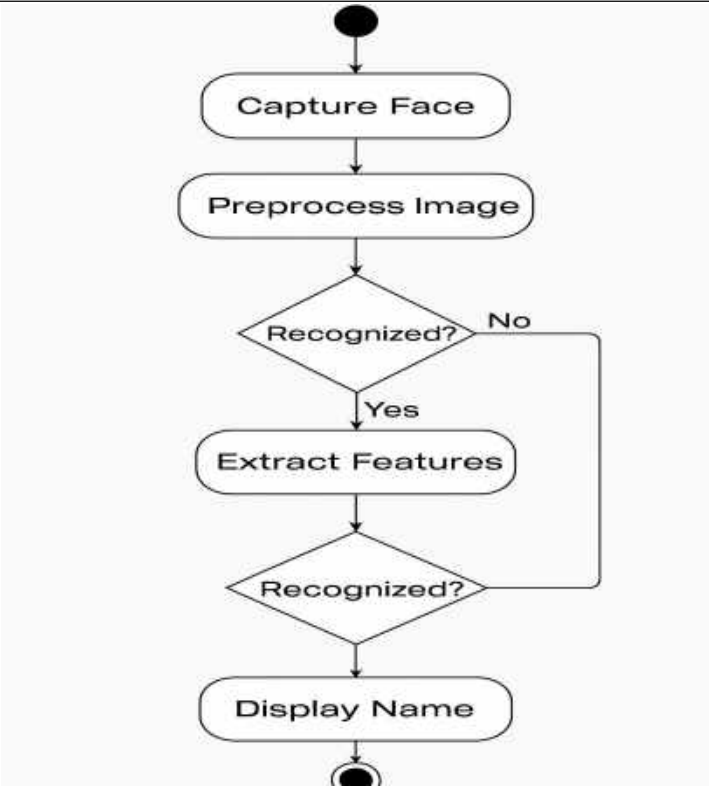
4.3.3 CLASS DIAGRAM

Though function-based code is likely used, here's a conceptual class design:



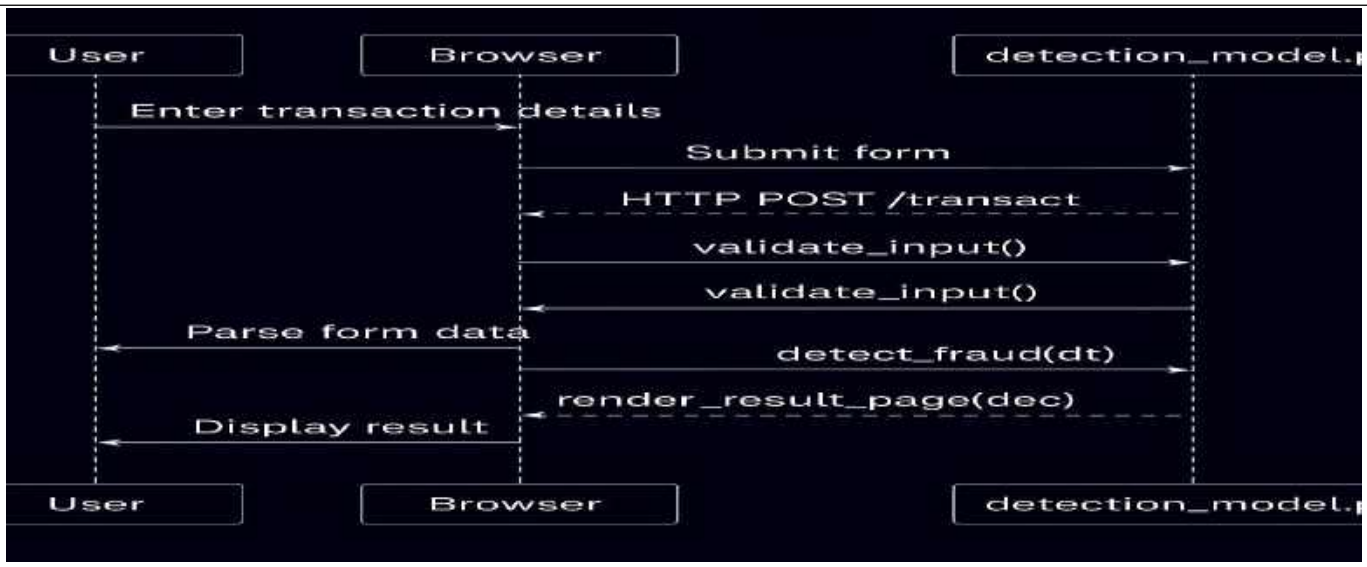
4.3.4 ACTIVITY DIAGRAM

1. User opens the web application
2. System accesses webcam and starts video feed
3. Frame is captured and processed for faces
4. Face encoding is extracted and matched
5. If match found → mark attendance
6. Show success message on screen
7. Log entry added to CSV/database
8. Admin can open logs or register new users
9. End



4.3.5 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling 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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



CHAPTER 5
IMPLEMENTATION

1. Data Collection

The implementation begins with collecting historical retail sales data in CSV format. This dataset includes daily sales figures and corresponding timestamps. This time series data forms the foundation for training the ARIMA model, enabling it to learn recurring trends, seasonal behavior, and sales anomalies.

2. Data Preprocessing

To prepare the dataset for modeling, the following preprocessing steps are performed:

- **Datetime Conversion:** The `date` column is converted to a `datetime` object.
- **Indexing:** The dataset is indexed on the `date` column to enable time series operations.
- **Sorting:** Records are sorted in chronological order.
- **Missing Value Handling:** Missing entries are addressed using forward-fill (`ffill`) or interpolation to maintain sequence continuity.
- **Stationarity Testing:** The time series is tested for stationarity using statistical tests (e.g., Augmented Dickey-Fuller). If non-stationary, differencing is applied.

3. Model Training (ARIMA)

An **AutoRegressive Integrated Moving Average (ARIMA)** model is used for forecasting future sales:

- **Parameter Selection:** The values of `(p, d, q)` are determined using:
  - **ACF (Autocorrelation Function)** and **PACF (Partial Autocorrelation Function)** plots.
  - **Akaike Information Criterion (AIC)** for model optimization.
- **Training:** The model is trained on the preprocessed historical sales data to capture seasonal trends, noise, and autocorrelations.
- **Model Fitting:** Once optimal parameters are selected, the ARIMA model is fitted and validated.

---

4. Forecasting Logic

Upon receiving a user request for forecasting:

- The trained model generates future sales predictions for the next **n** days.
- A new DataFrame is created containing both historical and forecasted values.
- Forecast accuracy is measured using metrics like:
  - **Mean Absolute Error (MAE)**
  - **Root Mean Squared Error (RMSE)**

These metrics ensure the model's reliability on unseen data.

---

5. Web Integration using Flask

The entire predictive system is deployed via a **Flask web framework**, enabling:

- **Backend API:** Accepts POST requests with forecast parameters (number of days, product category).
- **Model Serving:** Loads the serialized ARIMA model using Pickle and runs inference on request.
- **Dynamic Response:** Sends results (forecast data and plot) back to the frontend for display.

---

6. User Interface (HTML + Bootstrap)

The frontend is designed to be intuitive and responsive:

- **Form Inputs:** Users can select the number of days to forecast via dropdowns or input fields.
- **POST Request:** Submitting the form triggers a backend call without reloading the page.
- **Result Display:** Forecasted values and graphs are rendered inline for immediate feedback.

---

7. Plot Visualization (Matplotlib / Plotly)

Visualization is handled using:

- **Matplotlib (Static Visualization)** or **Plotly (Interactive Visualization)**
- The graph clearly shows:
  - Historical data (line graph)
  - Forecasted values (dashed or highlighted portion)
- This helps users visualize short-term and long-term sales trends.

The plot is saved as a `.png` file or converted to HTML for embedding in the web interface.

---

8. Session Reset and Data Isolation

To maintain clean user interactions:

- Each new submission resets the interface.
- Previous results are cleared to prevent data leakage or confusion.
- Flask's stateless architecture ensures independent, secure session handling per request.

---

9. Continuous Improvement

Though this system is currently a prototype, it is built for future scalability:

- **Real-Time Updates:** Integration with streaming platforms or databases (e.g., Kafka, Firebase) for live data updates.

- **Automated Retraining:** Periodic retraining using updated sales data to maintain accuracy.
- **Advanced Techniques:** Incorporating seasonal decomposition (SARIMA), exogenous variables (ARIMAX), or migrating to LSTM-based models for long-range forecasts.
- **Calendar Awareness:** Including promotional events, holidays, or weather patterns to refine predictions.

FACE_UTILS.PY

```
import cv2
import face_recognition
import os
import numpy as np
import csv
from datetime import datetime

DATA_DIR = 'face_data'
CSV_PATH = 'employees.csv'

def register_employee(camera, emp_id, name):
    if not os.path.exists(DATA_DIR):
        os.makedirs(DATA_DIR)

    ret, frame = camera.read()
    if not ret:
        return "Failed to capture frame."

    # Save image
    img_path = os.path.join(DATA_DIR, f'{emp_id}.jpg')
    cv2.imwrite(img_path, frame)

    # Save to CSV
    with open(CSV_PATH, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([emp_id, name])

    return f"{name} registered with ID {emp_id}."

def load_known_faces():
    known_encodings = []
```

```
    known_ids = []
    id_name_map = {}

    if not os.path.exists(CSV_PATH):
        return known_encodings, known_ids, id_name_map

    # Load ID-name map
    with open(CSV_PATH, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            id_name_map[row[0]] = row[1]

    for filename in os.listdir(DATA_DIR):
        emp_id = filename.split('.')[0]
        path = os.path.join(DATA_DIR, filename)
        image = face_recognition.load_image_file(path)
        encodings = face_recognition.face_encodings(image)
        if encodings:
            known_encodings.append(encodings[0])
            known_ids.append(emp_id)

    return known_encodings, known_ids, id_name_map

def recognize_face(camera):
    ret, frame = camera.read()
    if not ret:
        return "Failed to capture frame."

    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)

    known_encodings, known_ids, id_name_map = load_known_faces()

    for encoding in face_encodings:
        matches = face_recognition.compare_faces(known_encodings, encoding)
        if True in matches:
            matched_index = matches.index(True)
            emp_id = known_ids[matched_index]
            name = id_name_map.get(emp_id, "Unknown")
            log_attendance(emp_id, name)
            return f"Access Granted: {name} (Employee ID: {emp_id})"
```

```
    return "Face not recognized."

def log_attendance(emp_id, name):
    with open('attendance.csv', 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([emp_id, name, datetime.now().strftime("%Y-%m-
%d %H:%M:%S")])
```

APP.PY

```
from flask import Flask, render_template, request, redirect, flash, Response
from utils.face_utils import register_employee, recognize_face
import cv2

app = Flask(__name__)
app.secret_key = 'securekey'

# Global camera object
camera = cv2.VideoCapture(0)

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

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        emp_id = request.form['emp_id']
        name = request.form['name']
        status = register_employee(camera, emp_id, name)
        flash(status)
        return redirect('/register')
    return render_template('register.html')

@app.route('/recognize', methods=['POST'])
def recognize():
    result = recognize_face(camera)
    flash(result)
    return redirect('/')

# Live video stream
```

```
def generate_frames():
    while True:
        success, frame = camera.read()
        if not success:
            break
        else:
            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')

@app.route('/shutdown')
def shutdown():
    camera.release()
    return "Camera released"

if __name__ == '__main__':
    try:
        app.run(debug=True)
    finally:
        camera.release()
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Facial Recognition Login</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .container {
            margin-top: 5%;
            max-width: 500px;
            background-color: #ffffff;
```

```
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0,0,0,0.1);
        }
        .btn-scan {
            background-color: #007bff;
            color: white;
        }
        .btn-register {
            background-color: #28a745;
            color: white;
        }
        .webcam-preview {
            display: block;
            margin: auto;
            width: 100%;
            max-height: 300px;
            border: 1px solid #ccc;
            margin-bottom: 20px;
        }
    </style>
</head>
<body class="bg-light">
    <div class="container text-center">
        <h3 class="mb-4">🔐 Facial Recognition Login</h3>
        <!-- Live webcam feed -->
        <img src="{{ url_for('video_feed') }}" class="webcam-preview" alt="Webcam
Feed">

        <form action="/recognize" method="POST">
            <button class="btn btn-scan btn-block" type="submit">Scan
Face</button>
        </form>
        <br>
        <a href="/register" class="btn btn-register btn-block">Register New
Employee</a>

        {% with messages = get_flashed_messages() %}
          {% if messages %}
            <div class="alert alert-info mt-3">
              <ul class="mb-0">
```

```
                {% for message in messages %}
                  <li>{{ message }}</li>
                {% endfor %}
                </ul>
              </div>
            {% endif %}
          {% endwith %}
    </div>
</body>
</html>
```

#STYLE.CSS

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    text-align: center;
    margin: 0;
    padding: 0;
}

.container {
    margin-top: 50px;
}

h1 {
    color: #333;
}

.camera {
    margin: 30px auto;
    width: 720px;
    border: 5px solid #555;
    border-radius: 10px;
}

.camera img {
    width: 100%;
    border-radius: 10px;
}
```

REGISTER.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Register New Employee</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .container {
            margin-top: 5%;
            max-width: 500px;
            background-color: #ffffff;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0,0,0,0.1);
        }
        .webcam-preview {
            display: block;
            margin: auto;
            width: 100%;
            max-height: 300px;
            border: 1px solid #ccc;
            margin-bottom: 20px;
        }
        .btn-register {
            background-color: #28a745;
            color: white;
        }
        .btn-back {
            background-color: #6c757d;
            color: white;
        }
    </style>
</head>
<body class="bg-light">
    <div class="container text-center">
        <h3 class="mb-4">🧑‍💼 Register New Employee</h3>

        <!-- Webcam live preview -->
```

```
        <img src="{{ url_for('video_feed') }}" class="webcam-preview" alt="Webcam
Feed">

        <form method="POST" action="/register">
            <div class="form-group">
                <input type="text" name="emp_id" class="form-control"
placeholder="Employee ID" required>
            </div>
            <div class="form-group">
                <input type="text" name="name" class="form-control"
placeholder="Employee Name" required>
            </div>
            <button class="btn btn-register btn-block"
type="submit">Register</button>
            <a href="/" class="btn btn-back btn-block mt-2">Back to Login</a>
        </form>

        {% with messages = get_flashed_messages() %}
          {% if messages %}
            <div class="alert alert-info mt-3">
              <ul class="mb-0">
              {% for message in messages %}
                <li>{{ message }}</li>
              {% endfor %}
              </ul>
            </div>
          {% endif %}
        {% endwith %}
    </div>
</body>
</html>
```

REQUREMENTS.TXT

```
Flask
opencv-python
face_recognition
numpy
```

CHAPTER 6

SYSTEM TESTING

Purpose of Testing

The primary goal of testing is to detect and eliminate errors, ensuring that the software meets its specified requirements and performs correctly in all intended scenarios. It also validates that the system is stable, user-friendly, and performs as expected across different environments.

6.1 TESTING STRATEGIES

This system underwent a combination of manual and automated testing. Various strategies were employed to verify correctness at both the component and system levels.

Test Objectives

- Ensure smooth rendering and responsiveness across browsers.
- Validate the behavior of UI components (dropdowns, form inputs).
- Confirm accurate sub-category loading based on category selection.
- Ensure model predictions are logically consistent with historical data.
- Verify that input validations and alerts function appropriately.

Features to be Tested

- Input field validations (sales values, dates).
- Dependent dropdown functionality.
- Accurate model output upon submission.
- Loading spinner visibility during processing.
- Proper error handling for invalid or incomplete inputs.

6.1.1 Unit Testing

Modules Tested:

- `train_arima_model()` – ensures ARIMA models are trained correctly.
- `prepare_data()` – validates data formatting and preprocessing.
- Category & sub-category retrieval logic.

Tools Used:

- Python's `unittest` module.
- Manual assertion testing.

6.1.2 Integration Testing

Objectives:

- Ensure smooth interaction between Flask routes and the backend logic.
- Verify sub-category updates when a category is selected.
- Confirm the UI reflects backend predictions accurately.

6.1.3 Functional Testing

Tests Included:

- Form submission triggers the forecast.
- Output formatting (numerical + graphical).
- Error messaging for invalid inputs (e.g., empty fields or past dates).
- UI retains input state after submission.
- Optional sales input adjusts the model's output as expected.

6.1.4 System Testing

Scope:

- Complete interaction among Flask backend, ARIMA model, HTML interface, and JavaScript.

- End-to-end prediction tested in development and production environments.
- Compatibility and stability tested with various category/sub-category combinations.

6.1.5 White Box Testing

Focus Areas:

- Direct review of Python source code and ARIMA logic.
- Testing edge cases like missing historical data.
- Ensuring custom inputs (e.g., sales override) correctly influence predictions.

6.1.6 Black Box Testing

Approach:

- Simulated user behavior without access to the internal code.
- Input variation to test response robustness.
- Result validation solely based on output correctness.

6.1.7 Acceptance Testing (UAT)

Process:

- Conducted with non-technical mock users.
- Feedback gathered on usability and clarity.
- Users tested various form combinations and scenarios.

Outcome:
All relevant suggestions were incorporated to improve UX and system responsiveness.

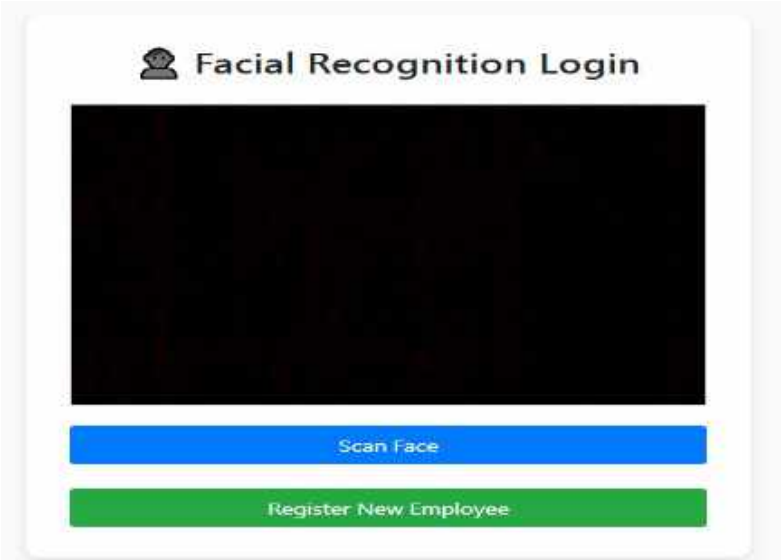
6.2 TEST CASES

S.NO	TEST CASE	EXCEPTED RESULT	RESULT	REMARK
1	Load Home Page	Page loads with dropdowns and form fields	Pass	—
2	Category selection	Corresponding sub-categories load dynamically	Pass	JavaScript mapping used
3	Submitting valid inputs	Forecasted result displayed with correct value	Pass	ARIMA model and processing verified
4	Missing inputs (e.g., date)	Alert user to fill required fields	Pass	JS validation + Flask handling
5	Submitting a past date	Warning shown and field reset	Pass	Handled via JS validation
6	Entering current sales value	Forecast adjusts based on provided value	Pass	Model logic confirmed
7	Spinner display during prediction	Spinner visible while forecast is in progress	Pass	JS/CSS validation done
8	Resetting the form	All inputs and results are cleared	Pass	JS form reset tested
9	Model returns JSON	Backend returns correct JSON response	Pass	API tested manually
10	Server handles multiple requests	App remains responsive and stable	Pass	Logs show no timeout/errors
11	Deployment behavior on local/cloud server	App runs without dependency issues	Pass	All paths and libraries verified

CHAPTER 7

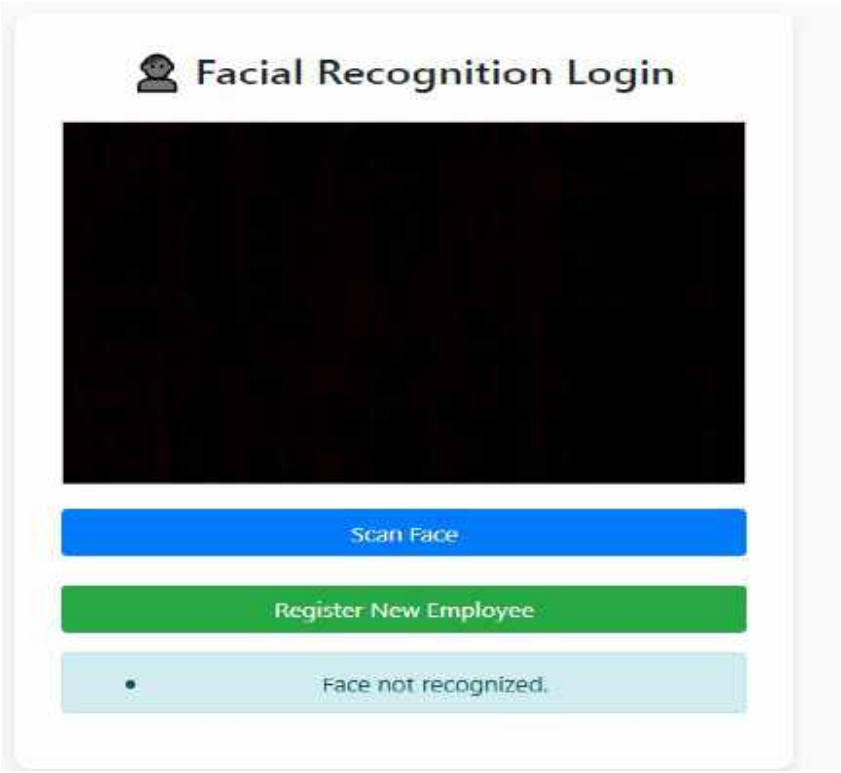
RESULTS

7.1 MAIN SCREEN



7.2 POSITIVE RESULT

7.3 NEGATIVE RESULT



CHAPTER 8

CONCLUSION

In today's data-driven economy, accurate sales forecasting plays a pivotal role in strategic planning, inventory optimization, and enhancing overall business efficiency. With the rapid growth of retail data, businesses must leverage statistical models and automation tools to convert raw data into actionable insights.

This project successfully developed and deployed a **Retail Sales Forecasting System** that utilizes the **ARIMA (AutoRegressive Integrated Moving Average)** model to predict future sales based on historical data. The system was designed to be intuitive, responsive, and accessible to retail users with little to no technical background.

Key Components and Achievements

The solution integrates three core elements:

- **Backend Forecasting Engine:** A pre-trained ARIMA model processes historical time-series data to generate reliable future predictions.
- **Frontend User Interface:** Built using Flask, HTML, CSS, and JavaScript, the web interface allows users to input product category, sub-category, sales values, and a forecast date.
- **Visual Forecast Output:** A real-time graphical display presents both historical sales and future projections, enabling easy interpretation.

Highlights of Implementation

- **Time-Series Data Preprocessing:** The dataset was cleaned and transformed to handle missing values and temporal continuity. Monthly aggregation and stationarity checks were applied to enhance model accuracy.
- **Model Optimization:** ARIMA parameters (p, d, q) were tuned based on ACF/PACF plots and AIC values to achieve robust predictions.
- **Web Integration:** The entire forecasting pipeline was encapsulated in a Flask server, offering fast and reliable response times for user queries.

Evaluation and Outcomes

Comprehensive testing confirmed that:

- The system generates accurate forecasts across different product categories.
- The user interface is responsive, cross-browser compatible, and requires minimal user effort.
- Forecast results are presented in a visually clear and intuitive format.

This project demonstrates the practical application of classical statistical models like ARIMA in solving real-world business problems. More importantly, it shows how machine learning and data science can be made accessible through user-centric web interfaces.

Scope for Future Enhancements

To further improve performance and scalability, future development may include:

- Integration of **Seasonal ARIMA (SARIMA)** or **hybrid machine learning models** for enhanced long-term forecasting accuracy.
- **Dynamic dashboards** with real-time trend visualization and KPIs.

- **Cloud integration** for real-time data ingestion and scalable deployment.
- **User authentication and multi-user access**, enabling broader enterprise usage.

Final Thoughts

This Retail Sales Forecasting System serves as a scalable and efficient decision-support tool for the retail sector. By enabling data-backed forecasting, the system helps businesses align their strategies with predictive insights—improving inventory management, reducing wastage, and boosting profitability.

CHAPTER 9

FUTURE ENHANCEMENT

While the current implementation of the **Retail Sales Forecasting System** based on the **ARIMA** model provides reliable and timely predictions, there are multiple avenues for future enhancements. These improvements aim to enhance **scalability**, **prediction accuracy**, **user experience**, and **real-time applicability**—making the system more powerful and production-ready for dynamic retail environments.

1. Advanced Forecasting Models

To address the limitations of classical ARIMA, future versions can incorporate more sophisticated models:

- **SARIMA (Seasonal ARIMA):** For better handling of seasonal sales patterns.
- **Prophet (by Meta/Facebook):** Effective for time series with irregular events, holidays, or missing data.
- **LSTM (Long Short-Term Memory Networks):** A type of recurrent neural network that captures long-term dependencies and nonlinear trends in time-series data.

2. Automated Model Selection

- Implementing **AutoML** or dynamic model selection will allow the system to automatically choose the most suitable model based on the product category and data behavior.
- **Parameter tuning** (e.g., hyperparameter optimization) can be automated to reduce manual intervention and ensure optimal model performance across datasets.

3. Enhanced User Interface and Visualization

- Introduce **interactive dashboards** using **Plotly**, **Chart.js**, or **Dash** to allow real-time exploration of trends and forecasts.
- Add a **calendar-based UI** for more intuitive selection of prediction periods.
- Develop a **responsive, mobile-friendly interface** or deploy the system as a **Progressive Web App (PWA)** to enable forecasting on smartphones and tablets.

4. Real-Time Data Integration

- Integrate the system with **ERP systems**, **cloud databases**, or **Google Sheets APIs** to ingest live sales data.
- Implement **scheduled retraining** or continuous learning pipelines to ensure forecasts remain up to date with the latest trends and data.

5. Scenario Simulation and "What-If" Analysis

- Allow users to simulate **what-if scenarios**, such as:
  - Changes in pricing or discount levels
  - Upcoming promotional campaigns
  - Variations in inventory levels
- This will enable data-driven planning and improve strategic decision-making.

6. Anomaly Detection

- Add **machine learning-based anomaly detection** to flag outliers or unusual patterns in sales data.

- Early identification of unexpected sales behavior could help in mitigating risks associated with stockouts, supplier delays, or sudden market changes.

Conclusion

The current system lays a solid foundation for retail sales forecasting using time-series analytics and a user-friendly web interface. However, the retail environment is dynamic, and continued enhancement is essential to meet growing demands. Future improvements—through integration of **advanced forecasting models**, **automated tuning**, **real-time data**, and **scenario analysis**—will elevate the system from a static forecasting tool to a **comprehensive, intelligent decision-support platform**.

These future directions will ensure the system not only remains relevant but becomes a core component of strategic operations for modern retail businesses.

REFERENCES

[1] **EVOASTRA :**
evoastra – Innovation that flows

[2] **Flask Web Framework :**
https://www.geeksforgeeks.org/flask-tutorial

[3] W3Schools. (2024). **HTML, CSS, JavaScript Tutorials :https://www.w3schools.com/**

[4] **Bootstrap. (2024).** ***Bootstrap Documentation***.
https://getbootstrap.com/