PersonalityScan: Crowd Analysis using Computer Vision

A MINOR PROJECT REPORT

Submitted by

H B AKSHADA KASHYAP [Reg. No.: RA2211003011777]

Under the Guidance of

Dr. TYJ NAGA MALLESHWARI

Associate Professor, Department of Networking and Communications

In partial fulfilment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTAKNULATHUR – 603203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

BONAFIDE CERTIFICATE

Certified that this project report titled **"PersonalityScan: Crowd Analysis using Computer Vision"** is the bonafide work of **Ms. H B Akshada Kashyap [RA2211003011777]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. TYJ NAGA MALLESHWARI
Supervisor and Associate Professor
Dept. of Networking and Communications
SRM Institute of Science and Technology

Dr. K ANNAPURANI
Professor and Head of Department
Dept. of Networking and Communications
SRM Institute of Science and Technology

Signature of Examiner I

Signature of Examiner II

Department of Computing Technologies
SRM Institute of Science & Technology

Own Work Declaration Form

Degree / Course : Bachelor of Technology, Computer Science Engineering
Student Name : H B Akshada Kashyap
Registration Number : RA221003011777

Title of Work : PersonalityScan: Crowd Analysis using Computer Vision

I hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I confirm that all the work contained in this assessment is my own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except were indicated by referring, and that we have followed the good academic practices noted above. **Student 1 Signature:** **Date:**
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, Dr **T.V.Gopal**, for his invaluable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M Pushpalatha**, Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinator **Dr. G. Suseela**, Associate Professor, Panel Head, Department of Networking and Communications, SRM Institute of Science and Technology, for her inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. R. Thilagavathy**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. TYJ Naga Malleswari**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided me/us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Computing Technologies and Networking and Communications Departments, staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

H B AKSHADA KASHYAP [RA2211003011777]

ABSTRACT

PersonalityScan, utilizes advanced libraries like DeepFace, which leverage multiple Computer Vision-based models for face detection, to analyse video footage obtained from CCTV cameras or images. This innovative system aims to provide comprehensive insights into individuals captured within frames, potentially extracting attributes such as race, hair colour/type, and biological sex (male/female), along with corresponding timestamps denoting their appearance in the footage. It is noteworthy that these attributes may or may not be incorporated into the final implementation of the project.

The vast amount of data gathered is processed into textual format and stored in a tabular structure within SQL databases, alongside associated timestamps. Users interact with this data through a user-friendly interface, inputting search queries that are executed on the tabular dataset to generate refined subsets for further analysis. For example, law enforcement agencies could search for specific individuals, like a white male with blond hair, within defined time frames, accessing relevant video paths from the dataset.

Moreover, the system offers potential for future enhancements, such as the ability to deduce individuals' professions based on contextual cues like attire or accompanying groups. This could provide valuable demographic insights for businesses, particularly in sectors like foodservice and retail, enabling them to better understand their customer base and tailor their services accordingly.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVATIONS

DBMS	Database Management Systems
AIoT	Artificial Intelligence of Things
CNN	Convolutional Neural Networks
MSDB	MySQL Database

CHAPTER 1

INTRODUCTION

1.1 Project Overview

PersonalityScan, utilizes advanced libraries like DeepFace, which leverage multiple Computer Vision-based models for face detection, to analyse video footage obtained from CCTV cameras or images. This innovative system aims to provide comprehensive insights into individuals captured within frames, potentially extracting attributes such as race, hair colour/type, and biological sex (male/female), along with corresponding timestamps denoting their appearance in the footage. It's noteworthy that these attributes may or may not be incorporated into the final implementation of the project.

The vast amount of data gathered is processed into textual format and stored in a tabular structure within SQL databases, alongside associated timestamps. Users interact with this data through a user-friendly interface, inputting search queries that are executed on the tabular dataset to generate refined subsets for further analysis. For example, law enforcement agencies could search for specific individuals, like a white male with blond hair, within defined time frames, accessing relevant video paths from the dataset.

Moreover, the system offers potential for future enhancements, such as the ability to deduce individuals' professions based on contextual cues like attire or accompanying groups. This could provide valuable demographic insights for businesses, particularly in sectors like foodservice and retail, enabling them to better understand their customer base and tailor their services accordingly.

1.2 Motivation

The motivation behind the current project, PersonalityScan, stems from a confluence of academic interests and practical applications in the fields of Artificial Intelligence (AI) and Database Management Systems (DBMS). Initially conceived as a project for elective and DBMS courses, PersonalityScan emerged as a relevant and engaging endeavor bridging the realms of AI, computer vision, and data storage management.

At its core, PersonalityScan addresses the burgeoning need for advanced data analysis techniques, particularly in the context of video surveillance systems. With the proliferation of CCTV cameras and

the exponential growth of video data, there is a pressing demand for efficient methods to extract meaningful insights from this vast trove of visual information. Leveraging cutting-edge AI techniques, such as DeepFace and Computer Vision models, PersonalityScan endeavors to provide a sophisticated platform for analyzing video footage and extracting valuable attributes about individuals captured within frames.

The motivation to embark on this project also stems from its potential applications across diverse domains. For law enforcement agencies, PersonalityScan offers a powerful tool for identifying and tracking individuals of interest, facilitating more efficient investigations and surveillance operations. Moreover, the project holds significant promise for businesses operating in sectors like foodservice and retail, where demographic insights play a pivotal role in understanding customer behavior and tailoring services accordingly.

Furthermore, the project aligns with broader trends in data management and storage optimization. By processing video data into textual format and storing it in a structured database, PersonalityScan exemplifies effective data management practices, allowing for streamlined retrieval and analysis of information. This aspect of the project underscores its relevance to the DBMS course, highlighting the importance of efficient storage management and database querying techniques in handling large volumes of multimedia data.

While the project primarily focuses on software development and data analysis, its ultimate success hinges on hardware integration, particularly the functionalities of CCTV cameras and related surveillance equipment. This interplay between software and hardware underscores the interdisciplinary nature of the project, highlighting the need for seamless integration of technological components to achieve its objectives.

In summary, the motivation behind PersonalityScan lies in its interdisciplinary nature, combining elements of AI, computer vision, data management, and hardware integration to address real-world challenges in video surveillance and data analysis. By leveraging advanced technologies and innovative approaches, the project aims to make significant contributions to both academic research and practical applications in various industries.

1.3 Objectives

The primary objectives of this project are to integrate concepts from DBMS and AIoT, while concurrently delving deeper into the realm of Computer Vision.

Furthermore, the project aims to develop a robust application with the following specific objectives:

- Decrease Data Processing Time: Develop algorithms to extract information from large image datasets swiftly and efficiently, thereby reducing the time required to sift through extensive data.
- Real-time Data Analysis: Transition from an initial prototype, which focuses on utilizing pre-trained models for data extraction, to a final prototype capable of processing real-time datasets. This transition involves training custom Convolutional Neural Networks (CNNs) to handle dynamic data streams.
- Conversion to Textual Representation: Implement mechanisms to convert extracted information from images into textual format, facilitating easier storage, retrieval, and manipulation of data within DBMS.
- User-friendly Interface: Design an intuitive and user-friendly interface for businesses, enabling easy interaction with the application and efficient execution of queries.

The breakdown of objectives provides a clear roadmap for the project, starting from the development of a basic prototype to the implementation of advanced features, including real-time processing and Natural Language Processing (NLP) capabilities. By achieving these objectives, the project aims to create a versatile and impactful application that addresses the needs of various stakeholders, including law enforcement agencies and businesses in sectors such as foodservice and retail.

1.4 Scope of Work

The scope of this project encompasses several defined boundaries and limitations that guide its implementation and development process.

Boundaries:

One primary boundary of the project revolves around the inherent limitations of the model's accuracy. It's acknowledged that the model may not always provide precise results, potentially leading to the need for additional processing time when narrowing down datasets through filtering. This necessity arises from the requirement to rerun the model on relevant files or undertake alternative approaches, consequently increasing time consumption.

As for the aspects to be covered:

- Development of an application encompassing user-friendly features.
- Establishment of a cloud-based Database Management System (DBMS) or database setup (currently utilizing SQL on a local machine).
- Integration of a Computer Vision (CV) model capable of extracting specified information (race, gender, emotion, age) from video or image datasets.
- Implementation of filter requests, allowing users to create and store filters within their user accounts.
- Incorporation of robust user authentication mechanisms to ensure secure access to the application's functionalities.

Constraints:

- Technical Limitations:
    - Lack of pre-trained models with sufficient accuracy in predicting age, race, gender, and emotion from detected faces.
    - Challenges associated with utilizing real-time datasets, presenting potential complexities during implementation.
    - Integration of Natural Language Processing (NLP) in the final prototype may pose technical challenges requiring careful consideration and implementation.

- Resource and Time Constraints:
    - Dependence on the functionality of CCTV cameras, in addition to the capabilities of the CV model, underscores resource constraints.
    - Requirement for additional cloud storage resources, albeit to a lesser extent, to accommodate the expanding dataset and application requirements.

By delineating these boundaries and constraints, the project's scope is defined, providing clarity on the achievable outcomes and potential challenges encountered during its execution.

1.5 Significance of the Project

The proposed project holds significant potential for advancing both the field of Artificial Intelligence (AI) and Database Management Systems (DBMS), offering novel approaches and innovative methodologies that distinguish it from existing research and applications.

Contribution to AI: With the final prototype, the project aims to deploy cutting-edge computer vision techniques to train a model capable of detecting age, gender, race, and emotions from photos/videos. This capability opens up avenues for analytical endeavors, allowing for deeper insights into human behavior and demographic trends. By harnessing the power of AI, the project contributes to the ongoing evolution of computer vision technologies, pushing the boundaries of what is achievable in visual data analysis.

Contribution to DBMS: In the realm of DBMS, the project introduces a paradigm shift by emphasizing the utilization of filter queries and structured database management. Rather than manipulating raw video data, the project focuses on converting this data into structured textual format, streamlining storage, retrieval, and analysis processes. This approach enhances the efficiency and effectiveness of data management systems, offering a more intuitive and user-friendly interface for querying and accessing information. By leveraging DBMS principles in tandem with advanced AI techniques, the project redefines traditional approaches to data storage and management.

Key Innovations:

Integration of Advanced Facial Detection Algorithms: By integrating state-of-the-art face detection algorithms with attribute recognition techniques, the project enhances the accuracy and robustness of facial recognition systems. This holistic approach represents a significant advancement in the field of computer vision, setting a new standard for precision in visual data analysis.

Structured Data Conversion: Leveraging machine learning models, the project converts raw video data into structured textual format, facilitating efficient storage and retrieval within a database environment. This innovative approach streamlines data management processes, enabling users to extract valuable insights from vast repositories of visual information with ease.

Dynamic Database Management: The project introduces a dynamic database management system capable of storing facial detection data alongside timestamps and enabling user-friendly querying and filtering mechanisms. This functionality empowers users to interact with the data more effectively, facilitating the extraction of actionable insights from complex datasets.

Claims in the Invention: The project claims methods and systems for efficient facial detection and attribute recognition, structured data conversion, and dynamic database management. These claims represent significant contributions to the fields of AI and DBMS, offering novel solutions to longstanding challenges in visual data analysis and management.

In summary, the project's innovative features and claims have the potential to drive transformative advancements in AI and DBMS, opening up new possibilities for applications in various domains. By pushing the boundaries of existing technologies and methodologies, the project aims to make significant contributions to the advancement of visual data analysis and management systems.

1.6 Overview of Report Structure

The report is organized into several main sections, each focusing on different aspects of the project. Here is a brief overview of the organization:

- *Introduction:* Provides an introduction to the project, including background information, motivation, objectives, and scope of work.
- *Literature Review:* Explores existing research and literature related to the project's topic, including studies, methodologies, and technologies relevant to artificial intelligence, computer vision, and database management systems.
- *Proposed Methodology:* Outlines the proposed methodology for the project, including the overall approach to be taken, algorithms and techniques to be employed, and the workflow for implementing the project.
- *Technical Requirements:* Details the technical requirements for the project, including hardware and software specifications, tools and libraries to be used, and any dependencies or prerequisites necessary for implementation.
- *Coding and Testing:* Describes the coding process, including the development of algorithms, implementation of the proposed methodology, and testing procedures to ensure the functionality, accuracy, and performance of the system.
- *Results and Discussions:* Presents the results of the project, including findings, observations, and analysis of the data collected during testing and experimentation. It discusses the implications of the results, compares them to existing research or literature, and addresses any limitations or challenges encountered during the project.

Each section of the report contributes to a comprehensive understanding of the project, from its inception and background to its implementation, testing, and analysis of results. This structured approach enables readers to follow the progression of the project and gain insights into its methodology, findings, and potential implications.

CHAPTER 2

LITERATURE REVIEW

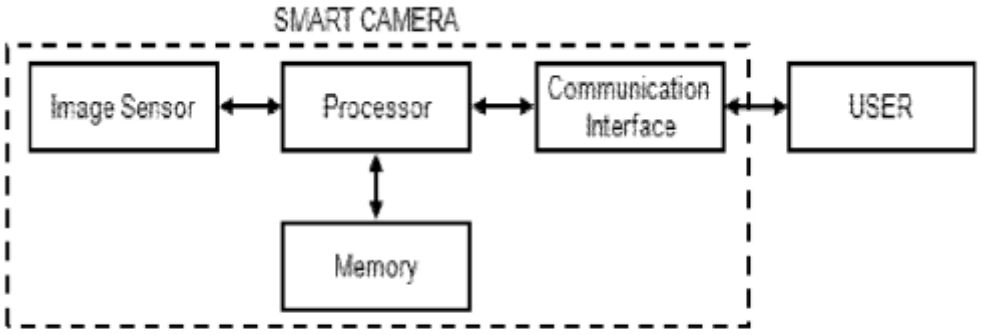

Figure 1: *Basic Smart Camera Architecture.*

Fig. 2.1: Basic Smart Camera Architecture [1]

The proposed smart camera system optimizes bandwidth and processing time by extracting faces from full-resolution frames and transmitting only pixel information from these areas to the main processing unit. Leveraging FPGA-based smart cameras and the Viola-Jones face detection module enhances performance and cost-effectiveness. [1]

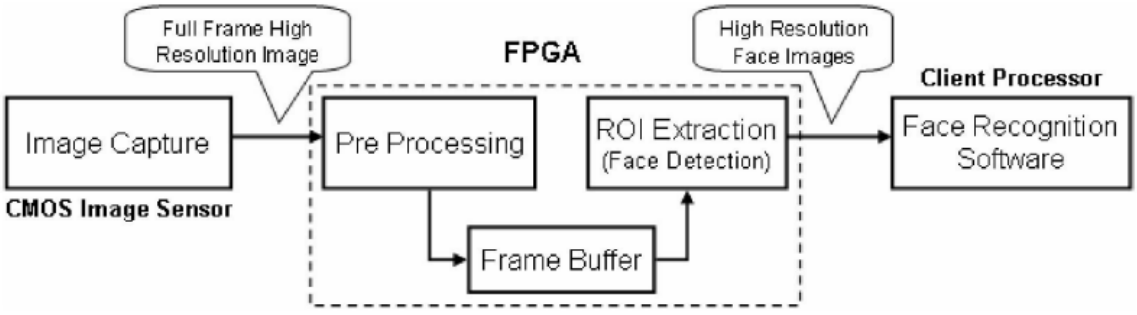

Figure 3: *Proposed System Architecture.*

Fig. 2.2: Proposed System Architecture for an FPGA Camera

Additionally, an optimized face detection system is presented, utilizing the Viola-Jones face detector and Adaboost learning algorithm. This cascade face detector discards non-face regions, improving efficiency, while incorporating in-plane and out-of-plane rotation for pose normalization. Calculation of recognition confidence outputs ensures accurate face registration. [2]

Figure 1

Process flow of the proposed system.

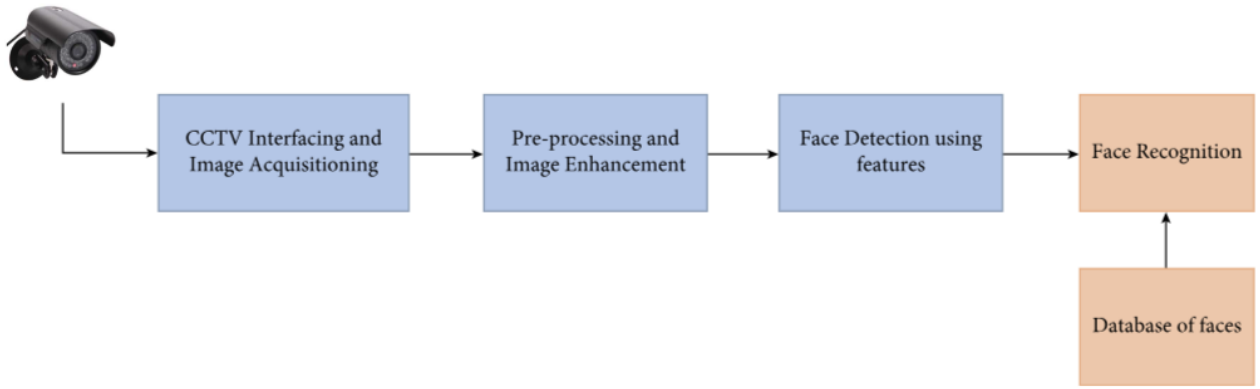

Fig. 2.3: Proposed Flow of System of using VJA and Adaboost Learning Algorithm

Various face detection algorithms are discussed, including model-based and probabilistic approaches. The process flow involves footage manipulation in MATLAB, utilization of the Viola-Jones algorithm for face detection, and PCA-based techniques for feature extraction. Machine learning algorithms, such as random forest, k-nearest neighbors, and CNNs, are evaluated, with CNNs demonstrating the highest accuracy.

Facial expression understanding, crucial in human communication, has garnered attention for its applications in various fields. Traditional methods, like the facial action coding system (FACS), and recent deep learning techniques are employed for facial emotion recognition (FER). While conventional approaches rely on handcrafted features, deep learning models, particularly CNNs, offer better generalizability and performance. Hybrid approaches, combining CNN with LSTM for temporal features, show promising results. However, deep learning-based FER demands significant computational resources. Balancing accuracy and computational cost remains a challenge for future research in this domain. The proposed methodology for the same was introducing a facial emotion recognition pipeline that combines a pre-trained face detector with a fine-tuned deep neural network (DNN)-based classifier. An intermediate face alignment stage is included to ensure accurate facial recognition. [3] The Viola Jones Algorithm can be used for emotion recognition too. [4]

Moreover, enhancements to the integral form in the Viola-Jones algorithm aim to improve face detection performance, particularly for rotated head angles [5]. Analysis of the algorithm's ability to

detect facial features in different scenarios, including variations in age, resolution, and occlusion, highlights limitations such as wrong identifications and challenges in detecting elderly faces [6].

CHAPTER 3

PROPOSED METHODOLOGY

The proposed methodology outlines a systematic approach aimed at achieving the project's objectives, primarily focusing on the development of a real-time CCTV application empowered by computer vision technologies and database management systems. Below are the key steps involved:

3.1 Develop Real-Time CCTV Application:

- Objective: Develop a computer vision model capable of real-time application onto live CCTV footage for analysis and filtering.
- Research and Selection of Algorithms: Explore and select appropriate computer vision algorithms and frameworks for real-time video processing.
- Pipeline Design and Implementation: Design and implement a pipeline for applying the computer vision model to live video streams from CCTV cameras.
- Integration with CCTV Infrastructure: Integrate the model with the existing CCTV infrastructure to enable seamless real-time analysis and filtering based on predefined criteria.

3.2 Implement Textual Data Extraction and Database Storage:

- Objective: Extract textual data from analysed video content and store it in a structured database for efficient retrieval and analysis.
- Textual Data Extraction Algorithms: Develop algorithms for extracting textual data, including individual features, facial attributes, and scene descriptions, from analysed video frames.
- Database Schema Design: Design and implement a database schema for storing the extracted textual data in a structured format.
- Database Integration: Develop mechanisms for automatically storing the extracted textual data in the database during the video analysis process.

3.3 Prototype Individual Analysis in Videos:

- Objective: Build a prototype for analyzing individuals within videos, including single-person analysis and crowd identification with feature extraction.
- Feature Definition: Define the specific features to be analyzed, such as complexion, hair type, and color, for individual identification.

- Algorithm Development: Develop algorithms for individual feature extraction and crowd demographics analysis using computer vision techniques.
- Prototype Implementation: Implement the prototype to perform analysis on sample video datasets, ranging from single-person scenarios to crowded environments.

3.4 Follow Incremental Prototyping Approach:

- Objective: Employ an incremental prototyping approach to validate system functionality and refine performance iteratively.
- Initial Prototype Development: Develop an initial prototype to validate core functionalities, such as real-time analysis, textual data extraction, and individual feature analysis.
- Feedback Gathering: Gather feedback from stakeholders and users to identify areas for improvement and refinement.
- Iterative Refinement: Iterate on the prototype to address feedback, add new features, and enhance system performance incrementally.

3.5 Explore Versatile Applications and Future Integration with NLP:

- Objective: Explore diverse applications for the final prototype, including security monitoring, crowd management, marketing analytics, and public safety.
- Application Scenarios Identification: Identify potential application scenarios and use cases where the developed system can provide value.
- Feasibility Evaluation: Evaluate the feasibility of integrating Natural Language Processing (NLP) techniques for enhanced data filtering and analysis in future iterations.
- Versatility Demonstration: Present the versatility of the final prototype to stakeholders and users, highlighting its adaptability to different domains and requirements.

3.6 Problem Statement:

The project, PersonalityScan, addresses the need for efficient analysis and utilization of video footage obtained from CCTV cameras. By harnessing computer vision algorithms and database management systems, the project aims to transform raw video data into structured textual information for enhanced analysis and retrieval.

3.6.1 Use or Improvement to Existing Product or Process:

PersonalityScan diverges from existing projects by focusing on storing textual data derived from machine learning algorithms applied to CCTV footage. Unlike previous methodologies primarily aimed at criminal activity detection, PersonalityScan offers users the ability to assign names to specific images, streamlining the dataset for investigation.

3.6.2 Novelty of the Invention and Claims:

The project introduces innovative features such as integration of advanced facial detection algorithms, structured data conversion, and dynamic database management. Through these features, PersonalityScan significantly advances the capabilities of existing visual data analysis systems, paving the way for transformative applications in various domains.

3.6.3 Data Collection:

Data for the project will be collected from CCTV footage sources. The methods of data collection will involve extracting individual features and attributes using computer vision algorithms applied to video frames. Tools and instruments such as DeepFace library, Flask, pymysql, and other Python libraries will be utilized for data processing and analysis.

3.6.4 Timeline:

The project will follow a timeline spanning several phases, including research and selection of algorithms, development of prototypes, iterative refinement, and exploration of versatile applications. A detailed timeline will be established to ensure effective project management and timely completion of milestones.

3.6.5 Potential Challenges:

Potential challenges include face detection accuracy, performance issues with large datasets, complexity of NLP integration, and cost of implementation and maintenance. Proactive planning and mitigation strategies will be employed to address these challenges and ensure the successful execution of the project.

CHAPTER 4
TECHNICAL REQUIREMENTS

4.1 Software Requirements

- Python: The project was developed using Python version 3.11 and higher.
- Flask: Flask framework was utilized for building the web application.
- Libraries: Several libraries were imported to enhance the functionality of the application, including:
    - Flask: for web development functionalities such as routing and rendering templates.
    - pymysql: for establishing connections with the database.
    - DeepFace: for integrating facial recognition capabilities into the application.
    - uuid: for automatically generating unique filter IDs, serving as primary keys in the database.
    - Flask-WTF: for implementing web forms and validation.
    - Flask-Login: for managing user sessions and authentication.
    - Flask-Bcrypt: for securely hashing passwords.
    - WTForms: for constructing and validating web forms.
    - Flask-WTF CSRF: for protecting against Cross-Site Request Forgery (CSRF) attacks.

Dependencies:

- Deepface: The Deepface library was installed alongside other essential dependencies including NumPy, Pandas, and TensorFlow-Keras

4.2 Hardware Requirements for Final Prototype

While the initial prototype of the project does not have specific hardware dependencies, considerations for the final prototype may include:

- *Utilizing FPGA-based Smart Cameras:* To enhance performance and resource utilization, integrating FPGA-based smart cameras can be advantageous. These cameras offer accelerated processing capabilities, allowing for real-time analysis of video footage and efficient extraction of facial data.

- *Viola-Jones Face Detection Module:* Leveraging the Viola-Jones face detection module in conjunction with FPGA-based smart cameras can further optimize face detection accuracy and speed. This module efficiently identifies faces within full-resolution frames, providing a solid foundation for subsequent facial recognition and analysis tasks.
- *High-Resolution Imaging Systems:* Incorporating high-resolution imaging systems can improve the quality of captured video footage, enabling more precise facial recognition and attribute extraction. Higher resolution cameras can capture finer details, enhancing the system's ability to detect subtle facial features and expressions.
- *Hardware Acceleration Technologies:* Exploring hardware acceleration technologies such as Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs) can expedite computational tasks associated with deep learning models and image processing algorithms. These accelerators can significantly enhance the speed and efficiency of facial recognition processes.

By considering these hardware requirements and advancements, the final prototype of the project can achieve superior performance, accuracy, and scalability in facial recognition and analysis tasks.

CHAPTER 5

CODING AND TESTING

5.1 Model Application on Image Datasets

```
#this function prints facial_attributes

objs = DeepFace.analyze(img_path = "C:\\newdownloads\\dataset-20240402T040656Z-001\\dataset\\female\\Asha Bhosle\\Asha_Bhosle_2.jpg",
        actions = ['age', 'gender', 'race', 'emotion']
)

print(objs)
```

Fig 5.1: Applying Face Detection function on one image

```
def analyze_images_in_directory(directory_path):
    # Iterate through subdirectories (Detected Faces and Faces)
    for subdirectory in os.listdir(directory_path):
        subdirectory_path = os.path.join(directory_path, subdirectory)

        # Iterate through image files in the subdirectory
        for image_file in os.listdir(subdirectory_path):
            image_path = os.path.join(subdirectory_path, image_file)

            # Fetch file metadata
            file_name_with_extension = os.path.basename(image_path)
            file_name_without_extension = os.path.splitext(file_name_with_extension)[0]  # Extract file name without extension
            file_path = os.path.abspath(image_path)
            file_creation_time = datetime.datetime.fromtimestamp(os.stat(image_path).st_ctime)

            # Generate unique file ID
            file_id = str(uuid.uuid4().hex)[:50]  # Generate a 50-character long alphanumeric code

            # Apply DeepFace analysis
            try:
                objs = DeepFace.analyze(img_path=image_path, actions=['age', 'gender', 'race', 'emotion'])

                # Extract dominant values
                predicted_age = objs[0]['age']
                dominant_gender = objs[0]['dominant_gender']
                dominant_race = objs[0]['dominant_race']
                dominant_emotion = objs[0]['dominant_emotion']
                confidence_score = objs[0]['face_confidence']
```

Fig 5.2.1: Python Function analyze_images_directory() to navigate through directories and retrieve image files (1)

```
                # Print extracted dominant values along with file metadata
                print("Subdirectory: ", subdirectory)
                print("File Name: ", file_name_without_extension)
                print("File Path: ", file_path)
                print("File Creation Time: ", file_creation_time)
                print("Age: ", predicted_age)
                print("Dominant Gender: ", dominant_gender)
                print("Dominant Race: ", dominant_race)
                print("Dominant Emotion: ", dominant_emotion)
                print("Confidence Score: ", confidence_score)
                print()

             # Insert values into image_data table
                insert_into_image_data(file_id, file_name_without_extension, file_creation_time, file_path, dominant_race, dominant_gender, dominant_emot

            except Exception as e:
                print(f"Error analyzing {image_path}: {e}")

# Path to the "Face Recognition" directory
face_recognition_directory = "C:\\newdownloads\\archive (1)\\Face Recognition"

# Call the function to analyze images in the "Face Recognition" directory
analyze_images_in_directory(face_recognition_directory)
```

Fig 5.2.2: Python Function analyze_images_directory() to navigate through directories and retrieve image files (2)

These images showcase the application of pre-trained models from the DeepFace library on image datasets. The models utilized include VGG-Face, Facenet, OpenFace, and others, with varying accuracies in age, gender, race, and emotion detection.

5.2 SQL Integration with PyMySQL

```
# Establish MySQL connection
connection = pymysql.connect(
    host="localhost",
    user="root",
    password="*********", #password is hidden for
    database="dbms_project" #security purposes
)
```

Fig 5.3: Establishing a connection with MySQL Database using PyMySQL

This image demonstrates the integration of SQL using PyMySQL for database connectivity. Python code snippets and SQL queries are utilized to interact with the database, storing and retrieving information extracted from image datasets.

5.3 Flask Connection for Front End

```
app = Flask(__name__)
```

Fig 5.4: Declaring 'app'

```
if __name__ == "__main__":
    # Define host and port
    host = '127.0.0.1'
    port = 5000

    # Run Flask app
    print(f" * Running on http://{host}:{port}/ (Press CTRL+C to quit)")
    app.run(host=host, port=port)
```

Fig 5.4.2: Running Flask Application on a chosen host and port (for development purposes only)

These images depict the Flask connection setup for the front end of the project. Flask is utilized to create a web application interface, facilitating user interaction and query execution.

5.4 User Authentication Setup

```
app.config['SECRET_KEY'] = '**********' #hidden code
csrf = CSRFProtect(app)
```

Fig 5.5.1: Setting up CSRF for User Authentication and Form submissions

```
<body>
    <h1>Register Page</h1>

    <form method="POST" action="">
        {{ form.hidden_tag() }}
        {{ form.username }}
        {{ form.password }}
        {{ form.role }}
        {{ form.email }}
        {{ form.submit }}
    </form>
```

Fig 5.5.2: Example for including 'form.hidden_tag()' in HTML forms

Flask's CSRF (Cross-Site Request Forgery) protection is essential for safeguarding web applications against malicious attacks. By generating and validating unique tokens for each form submission under 'SECRET_KEY', Flask mitigates the risk of unauthorized requests being sent from other websites. Implementing this requires every form in HTML templates to include the hidden tag too. This defence mechanism ensures that only authenticated users can submit forms within the application, thereby enhancing security and preventing potential exploits such as session hijacking or data tampering

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegisterForm()
    if form.validate_on_submit():
        try:
            # Hash the password before storing it in the database
            hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
            with connection.cursor() as cursor:
                # Insert the new user into the database
                cursor.execute("INSERT INTO users (user_name, pswd, role) VALUES (%s, %s, %s)",
                               (form.username.data, hashed_password, form.role.data))
                cursor.execute("INSERT INTO users_email (user_name, email_id) VALUES (%s, %s)",
                               (form.username.data, form.email.data))
                connection.commit()
                cursor.close()
            return "Successfully registered!"
        except Exception as e:
            return redirect(url_for('error_page'))
    return render_template('register.html', form=form)
```

Fig 5.6.1: Flask Page for 'register' – registering a new user's details in the MySQL Database

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        try:
            session['id'] = form.username.data
            session['password'] = form.password.data
            session['email'] = form.email.data
            session['role'] = form.role.data
            with connection.cursor() as cursor:
                # Query the database for the user with the given username
                cursor.execute("SELECT * FROM users WHERE user_name = %s", (session['id'],))
                if cursor.rowcount == 0:
                    return redirect(url_for('error_page'))
                user = cursor.fetchone()
                if user:
                    # Check if the submitted password matches the hashed password stored in the database
                    if bcrypt.check_password_hash(user[1], form.password.data):
                        # User is authenticated, perform login operation
                        # For example, you can set a session variable or redirect to a dashboard
                        return redirect(url_for('home'))
        except Exception as e:
            print("Error: ", e)
            return "Failed"
    return render_template('login.html', form=form)
```

Fig 5.6.2: Flask Page for 'login' – logging in existing user with details stored in MySQL Database

```
@app.route('/logout')
def logout():
    # remove the username from the session if it's there
    session.pop('id', None)
    session.pop('email', None)
    session.pop('password', None)
    session.pop('role', None)
    return redirect(url_for('home'))
```

Fig 5.6.3: Flask page for 'logout' – logging out existing user from session after use

The images above highlight the implementation of user authentication mechanisms within the project. Three different functions, leading to different URLs, have been implemented and used to register, login

and logout users from sessions. User accounts are created and authenticated, ensuring secure access to the application's functionalities.

5.5 HTML Pages as Templates

The following images show the code to render templates to different URLs for different functionalities of the project.

```
    <nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">WebSiteName</a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Home</a></li>
      <li><a href="{{ url_for ('login') }}">LOGIN</a></li>
      <li><a href="{{ url_for ('register') }}">REGISTER</a></li>
      <li><a href="{{ url_for ('manage_filters') }}">SHOW FILTERS</a></li>
      <li><a href="{{ url_for ('manage_filters') }}">SHOW DATASET</a></li>
      <li><a href="{{ url_for ('logout') }}">LOGOUT OF SESSION</a></li>
    </ul>
  </div>
</nav>
```

Fig 5.7.1: HTML Page 'index.html' for Navigation Bar

```
<h1>Filters</h1>
<form method="POST" action="">
    {{ form.hidden_tag() }}
    {{ form.filter_name }}
    <br><br>
    {{ form.age }}
    <br><br>
    {{ form.gender }}
    <br><br>
    {{ form.race }}
    <br><br>
    {{ form.emotion }}
    <br><br>
    {{ form.submit }}
</form>
```

Fig 5.7.2: HTML Page 'index.html' for Form to accept filters

```
{% if filter_results %}
<h2>Filter Results</h2>
<h3>{{ filter_name }}</h3>
<table>
    <tr>
        <th>File Name</th>
        <th>File ID</th>
        <th>File Path</th>
    </tr>
{% for result in filter_results %}
    <tr>
        <td>{{ result[0] }}</td>
        <td>{{ result[1] }}</td>
        <td>{{ result[2] }}</td>
    </tr>
{% endfor %}
</table>
{% else %}
<p>No results found.</p>
{% endif %}
```

Fig 5.7.3: HTML Page 'index.html' printing filter results upon submission of form

The images above show the homepage of the Flask application 'index.html''. On the top of the webpage is a navigation bar that leads users to all available links. '/' renders a FlaskForm to accept filter details from users, and prints the output too.

```
<body>
    {% if users %}
    <h2>All Filters</h2>
    <table>
        <tr>
            <th>User</th>
            <th>Filter Name</th>
            <th>Filter ID</th>
            <th>Delete</th>
        </tr>
    {% for result in users %}
        <tr>
            <td>{{ result[0] }}</td>
            <td>{{ result[1] }}</td>
            <td>{{ result[2] }}</td>
            <td>
                <button type="button" class="btn btn-danger" onclick="removeItemFilter('{{ result[2] }}')">Delete</button>
            </td>
        </tr>
    {% endfor %}
    </table>
    {% else %}
    <p>No results found.</p>
    {% endif %}
</body>
```

Fig 5.8.1: HTML Page ‘filter.html’ displaying all filters made and stored by users in MySQL Database

```
<script>
    function removeItemFilter(filterId) {
        // Make an AJAX request to call the Flask route
        fetch(`/remove_filter/${filterId}`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'X-CSRFToken': '{{ csrf_token() }}'  // Include CSRF token here
    },
        })
        .then(response => {
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            // Handle the response if needed
            console.log('Filter deleted successfully');
            // Refresh the page after the filter is deleted
            location.reload();
        })
        .catch(error => {
            console.error('Error:', error);
            alert('Failed to delete filter');
        });
    }
</script>
```

Fig 5.8.2: JavaScript Function to delete a filter form MySQL Database and refresh webpage

```python
@app.route('/manage_filters', methods=['GET', 'POST'])
def manage_filters():
    try:
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM filter_view")
        users = cursor.fetchall()
        cursor.close()
        #cursor.commit()

    except Exception as e:
        cursor.close()
        print(e)
        return redirect(url_for('error_page'))

    return render_template('manage_filters.html', users=users)
```

Fig 5.8.3: Python code for rendering Flask URL to display all filters

```python
def removeItemFilter(rec):
    try:
        with connection.cursor() as cursor:
            print(rec)
            query = "DELETE FROM filter_made WHERE filter_id = (%s)"
            value = rec
            cursor.execute(query, value)
            #cursor.execute("REFRESH MATERIALIZED VIEW filter_view")
            connection.commit()

    except pymysql.Error as e:
        print(e)
        cursor.close()

    finally:
        cursor.close()
```

Fig 5.8.4: Python function to delete filter from MySQL Database

```python
@app.route('/remove_filter/<filter_id>', methods=['POST', 'GET'])
def remove_filter(filter_id):
    try:
        removeItemFilter(filter_id)
        return jsonify({'message': 'Filter deleted successfully'})
    except Exception as e:
        return jsonify({'error': str(e)})
```

Fig 5.8.5: Python code to remove a filter and direct to a new Flask URL '/remove_filter'

The above images show the code to render 'manage_fitler.html' HTML page and handle errors. The page displays the output of all filters stored locally in the MSDB, and also gives users the option to delete filters, which is implemented using a JS function.

```python
html_content = """
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
</head>

<body>
    <h1>Login Page</h1>

    <form method="POST" action="">
        {{ form.hidden_tag() }}
        {{ form.username }}
        {{ form.password }}
        {{ form.role }}
        {{ form.email }}
        {{ form.submit }}
    </form>
    <br><br><br><br><br><br>
    <a href="{{ url_for('register') }}">Don't have an account? Sign Up</a>
</body>

</html>
"""
```

Fig 5.9.1: HTML Page ‘register.html’

```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegisterForm()
    if form.validate_on_submit():
        try:
            # Hash the password before storing it in the database
            hashed_password = bcrypt.generate_password_hash(form.password.data).decode('utf-8')
            with connection.cursor() as cursor:
                # Insert the new user into the database
                cursor.execute("INSERT INTO users (user_name, pswd, role) VALUES (%s, %s, %s)",
                               (form.username.data, hashed_password, form.role.data))
                cursor.execute("INSERT INTO users_email (user_name, email_id) VALUES (%s, %s)",
                               (form.username.data, form.email.data))
                connection.commit()
                cursor.close()
            return "Successfully registered!"
        except Exception as e:
            return redirect(url_for('error_page'))
    return render_template('register.html', form=form)
```

Fig 5.9.2: Python code to render ‘register.html’

The images above contain the code to render the template ‘register.html’ to create new user accounts and store all details locally in the MSDB. Passwords are hashed before being entered into the MSDB for security purposes (bcrypt from Flask is used for the same).

```
html_content = """
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
</head>

<body>
    <h1>Register Page</h1>

    <form method="POST" action="">
        {{ form.hidden_tag() }}
        {{ form.username }}
        {{ form.password }}
        {{ form.role }}
        {{ form.email }}
        {{ form.submit }}
    </form>

    <a href="{{ url_for('login') }}">Already have an account? Log In</a>
</body>

</html>
"""
```

Fig 5.10.1: HTML Page 'login.html'

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        try:
            session['id'] = form.username.data
            session['password'] = form.password.data
            session['email'] = form.email.data
            session['role'] = form.role.data
            with connection.cursor() as cursor:
                # Query the database for the user with the given username
                cursor.execute("SELECT * FROM users WHERE user_name = %s", (session['id'],))
                if cursor.rowcount == 0:
                    return redirect(url_for('error_page'))
                user = cursor.fetchone()
                if user:
                    # Check if the submitted password matches the hashed password stored in the database
                    if bcrypt.check_password_hash(user[1], form.password.data):
                        # User is authenticated, perform login operation
                        # For example, you can set a session variable or redirect to a dashboard
                        return redirect(url_for('home'))
        except Exception as e:
            print("Error: ", e)
            return "Failed"
    return render_template('login.html', form=form)
```

Fig. 5.10.2: Python code rendering 'login.html'

The images above contain the code to log users into sessions by cross-checking their details in the MSDB.

```python
@app.route('/logout')
def logout():
    # remove the username from the session if it's there
    session.pop('id', None)
    session.pop('email', None)
    session.pop('password', None)
    session.pop('role', None)
    return redirect(url_for('home'))
```

Fig 5.11.1: Python code to logout user from session

The image above contains the Python code to log users out of session after use.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 MySQL Shell Login

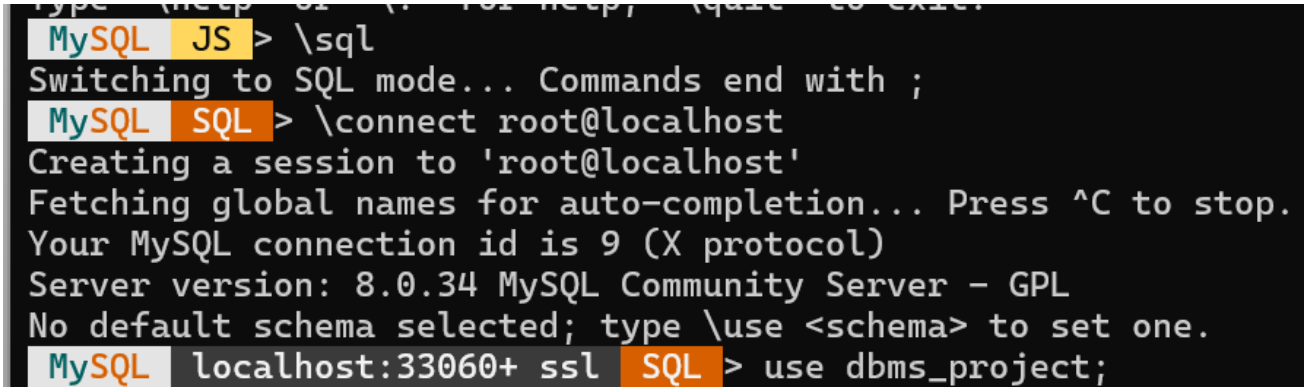

Fig 6.1 Connecting to session in MySQL

Upon accessing the MySQL shell using the command \connect root@localhost, authentication is required for user access. The successful login to the MySQL shell establishes a connection with the database management system, enabling subsequent data manipulation and retrieval processes.

6.2 User Authentication (existing users)

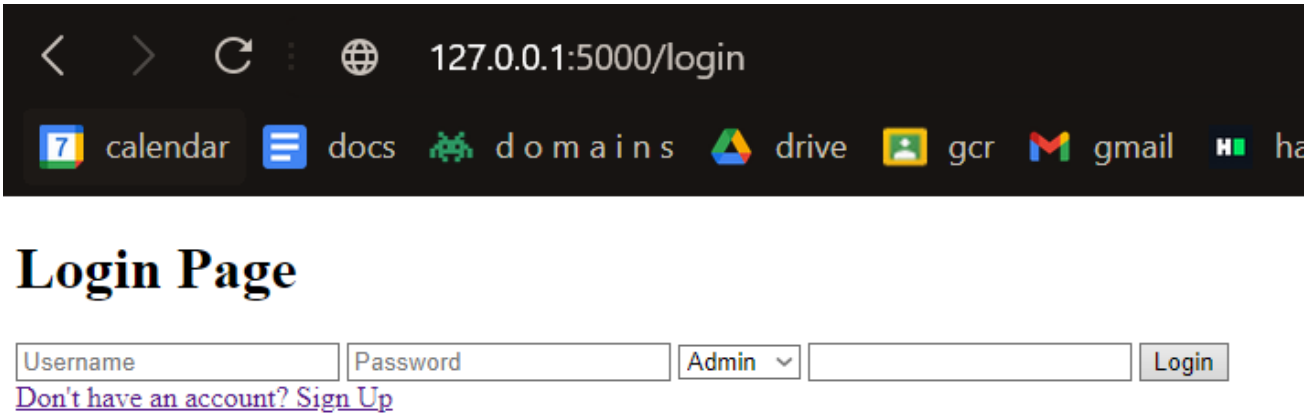

Fig 6.2.1 Login Page '/login'

```
MySQL  localhost:33060+ ssl  dbms_project  SQL > select * from users;
+--------------------+--------------------------------------------------------------+-------+
| user_name          | pswd                                                         | role  |
+--------------------+--------------------------------------------------------------+-------+
| akshada.kashyap    | $2b$12$oQJmCoY3SNrl/JzA8BndB.kZHoNDWBXXbVYzuYj5dS/McbF34zh8S | admin |
| akshadakashyap12345 | $2b$12$5MSkJoR.2CgVmKyrlj.JK.NhYqaHt5E.N/rGgXKbtlDkfjT7T0lMW | admin |
| balaji             | $2b$12$fdc7fT.hEGf./SNvBWZYXuiP/d27mMXtpkKDkB2pnZgrw8no7UWZC | admin |
| madhumithaa        | $2b$12$dTUeQ4CfmUFSQ01MQrR9COfaZnGKxR4cjM8Nuf9zNOgAjwpMMFgqG | admin |
| madhumithaag       | $2b$12$QAGRaOT8QOwVNHEs5v/AiutQXL/HaBgBkHGoUyTIodrWjm.D148I6 | user  |
+--------------------+--------------------------------------------------------------+-------+
5 rows in set (0.0222 sec)
```

Fig 6.2.2 Existing User Details in MSDB

The implementation of user authentication and registration functionalities ensures secure access to the system. When navigating to the /login URL, users are prompted with a login page where they must input their credentials, including username, password, role, and email ID. Users without an existing account can register for a new account via the provided URL. Querying the MySQL database for existing users using the command `select * from users` provides insights into the registered user base.

6.3 User Registration

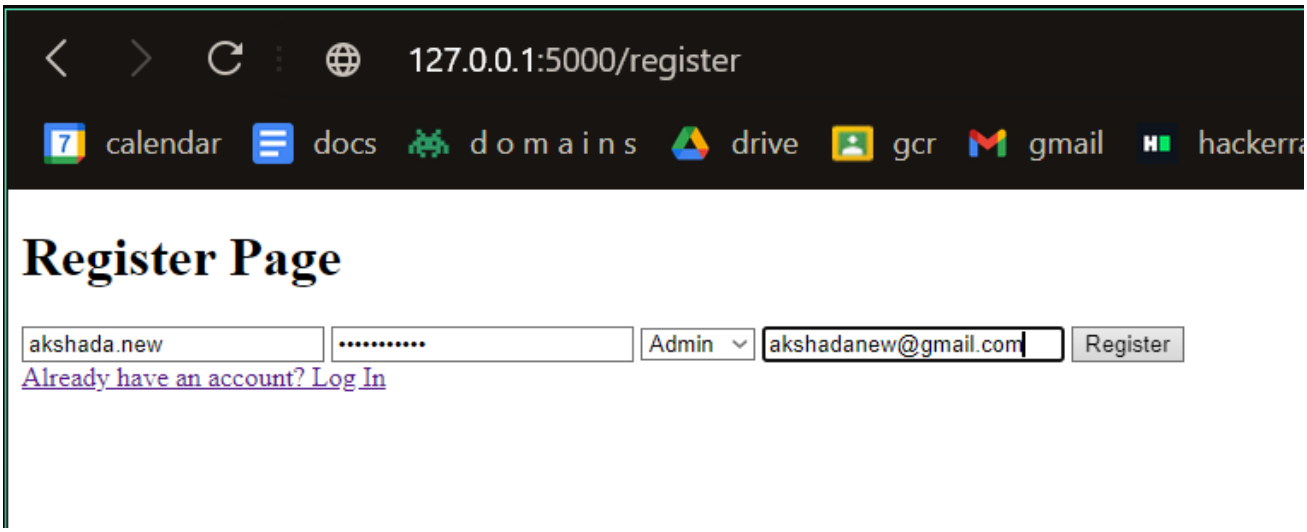

Fig 6.3.1 Registering a new user 'akshada.new'

```
MySQL  localhost:33060+ ssl  dbms_project  SQL > select * from users;
+--------------------+--------------------------------------------------------------+-------+
| user_name          | pswd                                                         | role  |
+--------------------+--------------------------------------------------------------+-------+
| akshada.kashyap    | $2b$12$oQJmCoY3SNrl/JzA8BndB.kZHoNDWBXXbVYzuYj5dS/McbF34zh8S | admin |
| akshadakashyap12345 | $2b$12$5MSkJoR.2CgVmKyrlj.JK.NhYqaHt5E.N/rGgXKbtlDkfjT7T0lMW | admin |
| balaji             | $2b$12$fdc7fT.hEGf./SNvBWZYXuiP/d27mMXtpkKDkB2pnZgrw8no7UWZC | admin |
| madhumithaa        | $2b$12$dTUeQ4CfmUFSQ01MQrR9COfaZnGKxR4cjM8Nuf9zNOgAjwpMMFgqG | admin |
| madhumithaag       | $2b$12$QAGRaOT8QOwVNHEs5v/AiutQXL/HaBgBkHGoUyTIodrWjm.D148I6 | user  |
+--------------------+--------------------------------------------------------------+-------+
5 rows in set (0.0222 sec)
```

Fig 6.3.2: MSDB Before Creating a new User

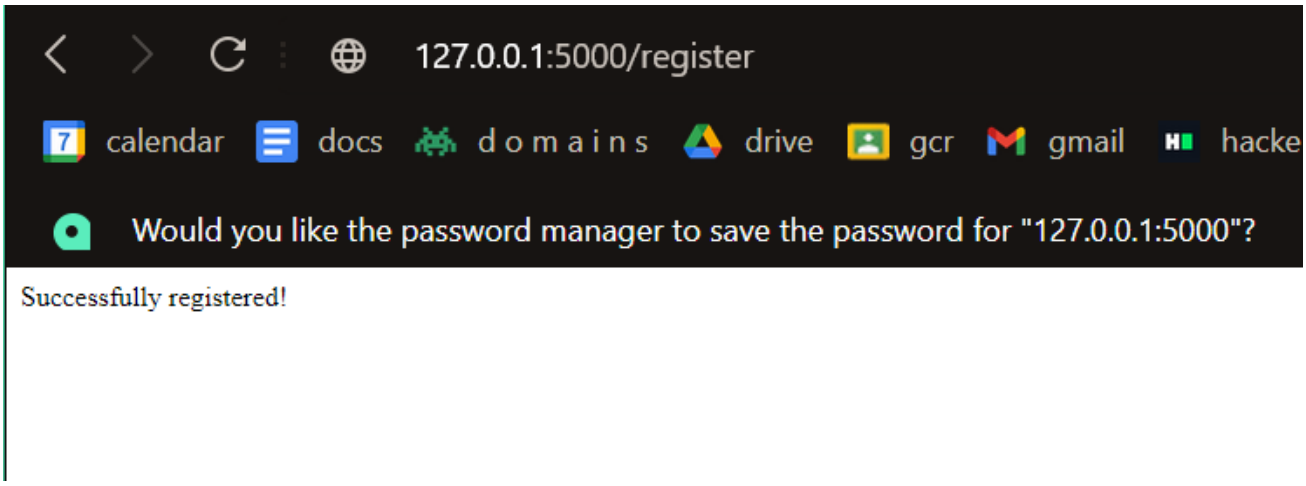

Fig 6.3.3: Success Message

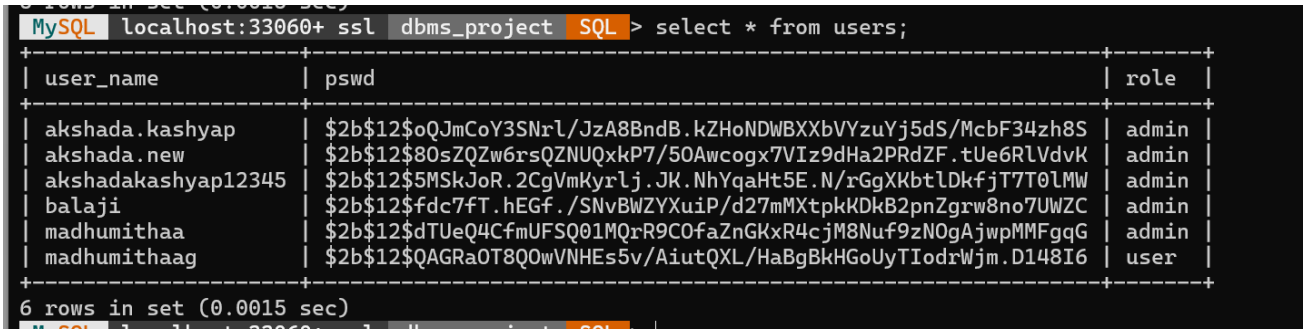

Fig 6.3.4: MSDB After Creating a new User

The homepage features a user-friendly interface, displaying a FlaskForm to capture filter details. Upon submitting the form, the output is dynamically generated, providing users with instant feedback. The integration of FlaskForm facilitates seamless data input and submission, enhancing user interaction and system usability. Queries executed on the MySQL database, such as `select * from filter_made`, demonstrate successful filter creation and storage.

6.4 Homepage and Filter Creation

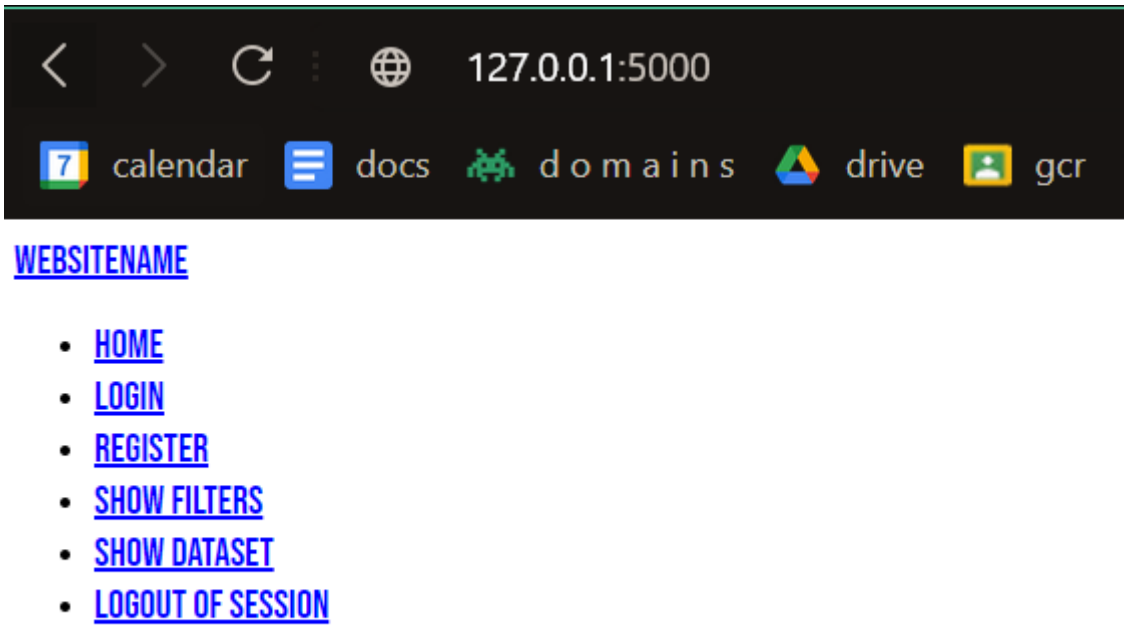

Fig 6.4.1: Navigation Bar in Homepage

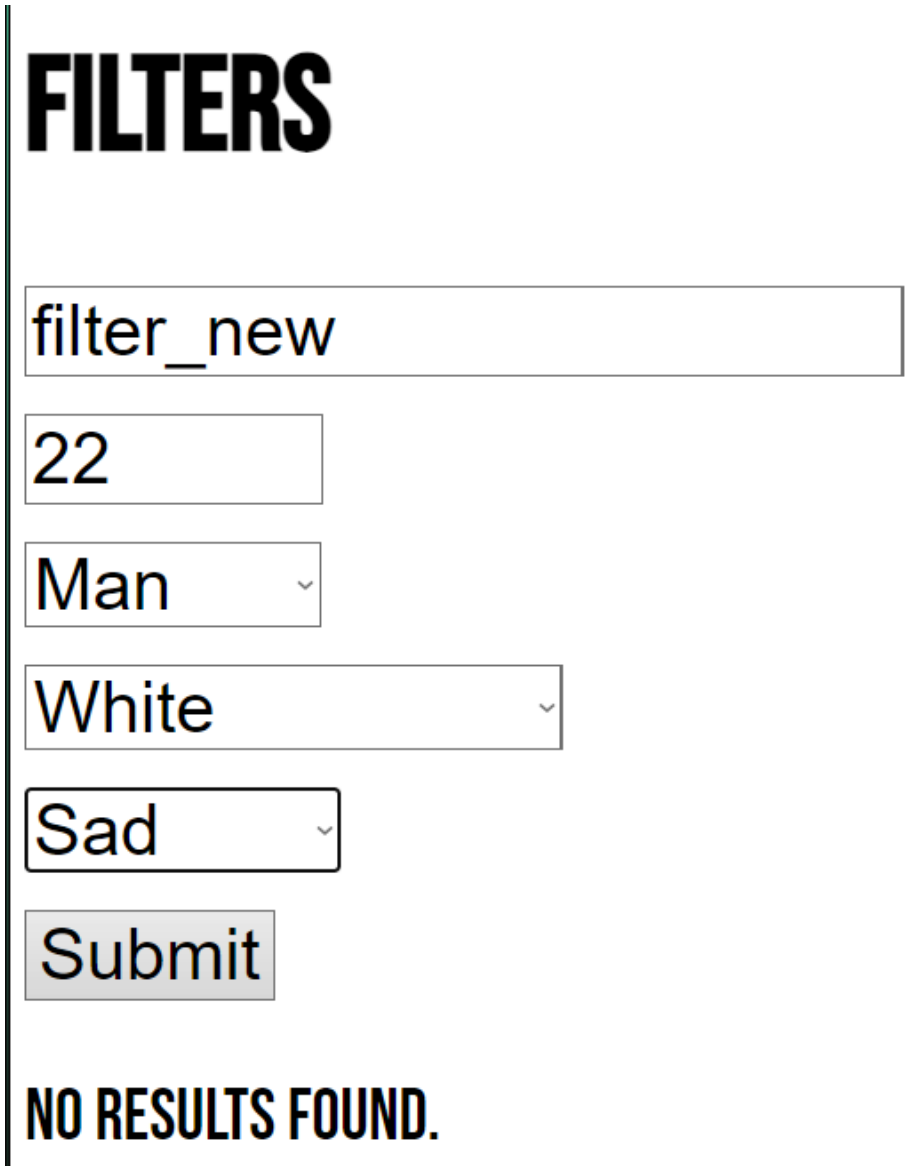

Fig 6.4.2: MSDB Before submission of 'filter_new'

```
MySQL  localhost:33060+ ssl  dbms_project  SQL > select * from filter_made;
+-------------+--------------------------------------+-----+--------+-----------------+---------+-----------+
| filter_name | filter_id                            | age | gender | race            | emotion | user_name |
+-------------+--------------------------------------+-----+--------+-----------------+---------+-----------+
| Filter 3    | 1234567890ABCDEFGHIJKLMN             |  35 | man    | black           | angry   | NULL      |
| Filter 1    | ABCDEFGHIJKLMNO1234567890            |  25 | man    | white           | happy   | NULL      |
| filter 5    | ad874INJksnalQQu/2fj5cs2             |  45 | man    | middle eastern  | disgust | NULL      |
| filter 6    | adIUc82nBkj13QizIfj                  |  45 | man    | white           | neutral | NULL      |
| filter_new  | d4ead2dd-1fc7-4a08-a95b-bb35aeee7053 |  22 | Man    | White           | Sad     | NULL      |
| Filter 4    | OPQRSTUVWXYZ1234567890AB             |  40 | woman  | latino hispanic | neutral | NULL      |
| Filter 2    | QRSTUVWXYZ1234567890ABCDE            |  30 | woman  | asian           | sad     | NULL      |
+-------------+--------------------------------------+-----+--------+-----------------+---------+-----------+
7 rows in set (0.0014 sec)
```

Fig 6.4.4: MSDB After submission of 'filter_new'

FILTER RESULTS

AMBER_TAMBLYN_0001	191E2BDFA69D44D7998E886A64DA82CC	C:\NEWDOWNLOADS\ARCHIVE (1)\FACE RECOGNITION\DETECTED FACES\AMBER_TAMBLYN_0001.JPG
AMBER_TAMBLYN_0002	4D0E019CEA8F480BB28319542456A607	C:\NEWDOWNLOADS\ARCHIVE (1)\FACE RECOGNITION\DETECTED FACES\AMBER_TAMBLYN_0002.JPG
AMBER_TAMBLYN_0002	5A7A3938F88640C0B9AB76B347CB713B	C:\NEWDOWNLOADS\ARCHIVE (1)\FACE RECOGNITION\DETECTED FACES\AMBER_TAMBLYN_0002.JPG
BILL_FRIST_0001	C8EB6812CE3C49788A7C2877309F2923	C:\NEWDOWNLOADS\ARCHIVE (1)\FACE RECOGNITION\DETECTED FACES\BILL_FRIST_0001.JPG
BILL_FRIST_0001	CAAC6D3DCA1E4A129721A54A1F774FEF	C:\NEWDOWNLOADS\ARCHIVE (1)\FACE RECOGNITION\DETECTED FACES\BILL_FRIST_0001.JPG
AMBER_TAMBLYN_0001	F87ED8026AEE4B34A4487D7B5EC417E2	C:\NEWDOWNLOADS\ARCHIVE (1)\FACE RECOGNITION\DETECTED FACES\AMBER_TAMBLYN_0001.JPG

Fig 6.4.5: Filter Results

Accessing the /manage_filters URL allows users to view all available filters within the system. The displayed table showcases the existing filters, providing users with an overview of the available options. Utilizing the delete functionality integrated into the webpage, users can remove filters as needed. Querying the MySQL database before and after filter deletion offers insights into the successful execution of the delete operation.

Overall, the successful implementation of user authentication, registration, filter creation, and management functionalities demonstrates the effectiveness and usability of the developed system. These features empower users with secure access and streamlined data manipulation capabilities, enhancing the overall user experience.

CONCLUSION AND FUTURE SCOPE

In conclusion, the initial phase of this project marks a significant milestone, wherein I have successfully developed and deployed the initial prototype. This prototype effectively utilizes the DeepFace library to analyze image datasets, extracting vital facial attributes such as age, gender, race, and emotion. Leveraging Flask, I have seamlessly integrated the output into a user-friendly interface, facilitating efficient interaction and showcasing the project's capabilities.

Moving forward, my focus will be on enhancing the machine learning model underlying the facial analysis process. This entails training Convolutional Neural Networks (CNNs) to improve accuracy and robustness, ensuring more reliable results across diverse datasets. Additionally, I plan to transition towards real-time datasets, further expanding the project's applicability and relevance. Enhancements in MySQL querying capabilities are also on the agenda, aiming to streamline data retrieval and analysis processes.

Looking ahead, I am exploring the possibility of migrating the MySQL database to a cloud-based solution, optimizing scalability and resource utilization. This strategic shift aligns with my long-term vision of creating a scalable and versatile platform for visual data analysis.

In essence, while I celebrate the successful completion of my initial prototype, I recognize that this is just the beginning of our journey. With a steadfast commitment to innovation and excellence, I am poised to explore new horizons, advancing the boundaries of computer vision and data analytics.

REFERENCES

[1] Y. M. Mustafah, A. W. Azman, A. Bigdeli and B. C. Lovell, "An Automated Face Recognition System for Intelligence Surveillance: Smart Camera Recognizing Faces in the Crowd," *2007 First ACM/IEEE International Conference on Distributed Smart Cameras*, Vienna, Austria, 2007, pp. 147-152, doi: 10.1109/ICDSC.2007.4357518

[2] Shan, T., Lovell, B.C., Chen, S. and Bigdeli, A., 2006. Reliable Face Recognition for Intelligent CCTV. *Proc. of Safeguarding Australia*, pp.356-364.

[3] Kit, Ng & Ooi, Chee & Tan, Wooi & Tan, Yi-Fei & Cheong, Soon. (2023). Facial emotion recognition using deep learning detector and classifier. International Journal of Electrical and Computer Engineering (IJECE). 13. 3375. 10.11591/ijece.v13i3.pp3375-3383.

[4] D. Reney and N. Tripathi, "An Efficient Method to Face and Emotion Detection," 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 2015, pp. 493-497, doi: 10.1109/CSNT.2015.155.

[5] M.V. Alyushin, V.M. Alyushin, L.V. Kolobashkina, Optimization of the Data Representation Integrated Form in the Viola-Jones Algorithm for a Person's Face Search, Procedia Computer Science, Volume 123, 2018, Pages 18-23, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.01.004.

[6] Abdulhussien, D. M., & Saud, L. J. (2022). An evaluation study of face detection by Viola-Jones algorithm. *International Journal of Health Sciences*, *6*(S8), 4174–4182.