**“Sports Celebrity Image Classification”**

**A MINI PROJECT REPORT**

Submitted in partial fulfillment of the Requirement for the award of the degree of

**Bachelor of Engineering**

in

**Computer Science and Engineering**

By

Tirumala Sesha Sai Nishit (1608-20-733-033)

Under the guidance of

Mrs. B. Meenakshi

Assistant professor



**Department of Computer Science and Engineering**

**MATRUSRI ENGINEERING COLLEGE**

**SAIDABAD - 500059**

**July 2022**

**MATRUSRI ENGINEERING COLLEGE**

**SAIDABAD - 500059**



**Department of Computer Science and Engineering**

CERTIFICATE FROM PROJECT GUIDE

This is to certify that the project report entitled **“*Sports Celebrity Image Classification*”** submitted by Tirumala Sesha Sai Nishit bearing **H.T. No:1608-20-733-033,** in the partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science and Engineering** is a bonafide work carried by him/her**.**The results of the investigations enclosed in this report have been verified and found satisfactory.

Mrs. B. Meenakshi**, Dr. P. Vijaya Pal Reddy**

**(Assistant Professor) HOD**

**Dept. of CSE, MECS.**   **Dept. of CSE, MECS.**

**MATRUSRI ENGINEERING COLLEGE**

**SAIDABAD - 500059**



**Department of Computer Science and Engineering**

DECLARATION BY THE CANDIDATE

I, **Tirumala Sesha Sai Nishit** bearing **H.T. No:1608-20-733-033,** hereby certify that the mini project report entitled **“Sports Celebrity Image Classification”** is submitted in the partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science and Engineering.**

This is a Record of bonafide work carried out by us under the guidance of Mrs. B. Meenakshi, Assistant Professor, Matrusri Engineering College, Saidabad. The results embodied in this report have not been reproduced/copied from any source. The results embodied in this report have not been submitted to any other university or institute for the award of any other degree or diploma.

**Tirumala Sesha Sai Nishit**

**(1608-20-733-033)**

**Dept. of CSE, MECS.**

**ACKNOWLEDGEMENT**

I wish to take this opportunity to express my deep gratitude to all the people who have extended their cooperation in various ways during my mini project work. It’s my pleasure to acknowledge the help of all those individuals.

Firstly, I would like to thank **Dr. P. Vijaya Pal Reddy (HOD, DEPT OF CSE.)** for his encouragement and valuable guidance in bringing shape to dissertation.

I would like to thank my project guide, Mrs. B. Meenakshi and for her guidance and help throughout the development of this project work by providing me with required information and support. Without her guidance, cooperation, and encouragement, I couldn’t learn many new things during my mini project tenure.

**Tirumala Sesha Sai Nishit**

**(1608-20-733-033)**

**Dept. of CSE, MECS.**

**ABSTRACT**

The sports celebrity image classification and website project aims to develop a deep learning model that can accurately identify and classify sports celebrities in images and videos and build a website that allows users to upload images of celebrities and receive details about them. The deep learning model will be trained using a large dataset of labelled images of sports celebrities and fine-tuned to improve its accuracy on specific tasks, such as identifying a sports celebrity in a given image. The model will be used to power the website, allowing users to drag and drop an image of a celebrity and receive details about them. The project is restricted to only five classes, meaning the website will provide details about only five selected celebrities. This project's primary benefit is enhancing security measures in sports stadiums and other venues. Security personnel may need to identify and respond to potential threats quickly. The website can help identify known individuals, including sports celebrities, and alert security personnel to potential security risks. Sports journalists and media outlets can also use the project to quickly identify and classify sports celebrities in images and videos for their news stories and coverage. Moreover, the project can improve marketing campaigns that involve sports celebrities. Sports celebrities are often used in marketing and advertising campaigns to increase the effectiveness of these campaigns and improve the overall return on investment. With the website, marketers can identify the right celebrity for a given campaign and enhance the effectiveness of the campaign. The expected result of this project is an accurate and efficient image classification system that can be used in various contexts, including sports journalism, marketing, and fan engagement.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **LIST OF FIGURES** | **Ⅷ** |
| **ABSTRACT** | **Ⅴ** |
|  |  |
| **CHAPTER** | **PAGE NO.** |
| **1. INTRODUCTION** | 1 |
| 1.1. Objectives | 2 |
| 1.2. Existing System  1.2.1 Disadvantages | 2  2 |
| 1.3. Proposed System  1.3.1 Advantages | 3  3 |
| 1.4. Modules | 4 |
| 1.5. Module description | 5 |
| **2. LITERATURE SURVEY** |  |
| 2.1. Introduction | 7 |
| 2.2. Literature |  |
| 2.2.1. Face recognition system based on wavelet transform, histograms of oriented gradients and support vector machine. | 7 |
| 2.2.2.ImageNet classification with deep convolutional neural networks.  2.2.3 Pubface: Celebrity face identification based on deep learning. | 8  9 |
| **3. METHODOLOGY** | 11 |
| **4. UML DIAGRAMS** | 14 |
| 4.1. Use Case Diagram | 15 |
| 4.2. Activity Diagram | 16 |
| 4.3. Class Diagram | 17 |
| 4.4. Sequence Diagram | 17 |
| 4.5. Component Diagram  4.6. Deployment Diagram | 18  19 |
| **5. SAMPLE CODE** | 21 |
| **6. TECHNOLOGY STACK** | 40 |
| **7. SCREEN SHOTS** | 42 |
|  |  |
| **8. CONCLUSION** | 47 |
| **9. FUTURE ENHANCEMENTS** | 48 |
| **10. REFRENCES** | 49 |
|  |  |
|  |  |
| **LIST OF FIGURES**   |  |  |  | | --- | --- | --- | | **Figure No.** | **Figure name** | **Page No.** | | 4.1 | Use Case Diagram | 15 | | 4.2 | Activity Diagram | 16 | | 4.3 | Class Diagram | 17 | | 4.4 | Sequence Diagram | 18 | | 4.5 | Component Diagram | 19 | | 4.6 | Deployment Diagram | 20 | | 7.1 | Home Page of sports celebrity image classification website | 42 | | 7.2 | Classification of Messi with accuracy from user submitted image | 43 | | 7.3 | Classification of Virat Kohli with accuracy from user submitted image | 43 | | 7.4 | Classification of Jeremy Lin with accuracy from user submitted image | 44 | | 7.5 | Classification of PV Sindhu with accuracy from user submitted image | 45 | | 7.6 | Classification of Roger Federer with accuracy from user submitted image | 46 | |  |

**1.INTRODUCTION**

Facial recognition technology has made significant strides in recent years, emerging as a powerful tool with diverse applications in both commercial and security contexts. By leveraging the distinct facial features of individuals, this technology enables large-scale face identification, revolutionizing various industries. It finds particular value in situations where pre-existing images of individuals are available, and a highly accurate recognition system is required for a fixed and substantial group of people. With this in mind, our project aims to develop an innovative website that harnesses the potential of facial recognition techniques to provide comprehensive information about celebrities. Through the integration of state-of-the-art machine learning algorithms and sophisticated computer vision techniques, our system will analyze and identify the facial features of celebrities, offering users instant access to relevant details, such as their name, occupation, and other pertinent information. Our project holds immense promise and utility for a wide range of users, including passionate celebrity enthusiasts, professionals in the entertainment industry, media agencies, and journalists. By simply uploading an image of the desired celebrity, users will be able to leverage our user-friendly website to swiftly process and analyze the image, instantly recognizing facial features and extracting valuable information. This streamlined approach eliminates the need for time-consuming manual searches and provides users with a centralized platform for comprehensive celebrity information. Moreover, the extensibility of our project allows for future integration of additional features and services, ensuring that users' needs and expectations are continuously met. Through this mini project report, we will delve into the intricate details of our methodology, discussing the underlying algorithms, techniques, and frameworks utilized in the development of our face recognition system. Additionally, we will explore the scope of the project, its limitations, and potential avenues for improvement. Finally, we will present our findings, showcasing the system's functionality, performance, and user feedback, providing valuable insights into the project's overall success and its potential to make a meaningful impact in the domain of celebrity recognition and information retrieval.

**1.1. Objective**

* The key objectives of this project are to develop a website that can recognise images of celebrities and provide information about them, and to improve the accuracy and efficiency of celebrity image recognition.
* The proposed system will use machine learning algorithms and computer vision techniques to identify celebrities based on their images.
* Specifically, the system will use support vector machines (SVM), wavelet transformation, and Haar cascade classifiers to recognize images of celebrities and extract relevant features.
* The main focus of the project is to provide a user-friendly interface that is easy to navigate and understand, making it accessible to a wide range of users.

## **1.2. Existing System**

* There are existing systems that use deep learning architectures such as convolutional neural networks (CNNs) to extract features from images and classify them into different categories based on the trained models.
* Some examples of such systems include the VGGFace model, the OpenFace model, and the FaceNet mode.

**1.2.1. Disadvantages:**

* Complex and time-consuming training process due to large dataset of celebrities
* High computational resources required for training and running models.
* Not easily customizable, requiring re-training from scratch for any changes or modifications.
* Not suitable for specific use cases with limited number of celebrities to classify.

**1.3. Proposed methodology**:

The proposed methodology for the system involves using a combination of machine learning algorithms (specifically support vector machines), wavelet transformation, and Haar cascade classifiers to identify celebrities based on their images.

**1.3.1. Advantages of Our Project**:

**1. Reduced Computational Resource Requirements**: Our project is designed to optimize computational resource utilization. By carefully selecting efficient algorithms and techniques, we ensure that the system operates effectively without requiring high-end hardware or significant computational power. This makes it more accessible and cost-effective for users with limited resources.

**2. Fast and Real-Time Processing**: The efficient implementation of algorithms and optimized resource usage allows our system to perform fast and real-time processing. This means that users can obtain near-instantaneous results when uploading images for celebrity identification, providing a seamless and efficient user experience.

**3. Scalability and Flexibility**: Our project is designed to be scalable and adaptable to varying requirements. It can handle a growing database of celebrities and accommodate future enhancements without compromising performance. This flexibility ensures that the system can evolve alongside changing needs and advancements in technology.

**4. Resource-Efficient Training Process**: The training process of our system is designed to be resource-efficient. By carefully curating and preprocessing the dataset, optimizing model architectures, and employing techniques like transfer learning, we minimize the computational resources required for training while still achieving high accuracy in celebrity recognition.

**5. Cost-Effective Solution**: With its reduced computational resource requirements, our project offers a cost-effective solution for celebrity image classification. Users can utilize the system without the need for expensive hardware or extensive computational resources, making it accessible to a wider range of individuals and organizations.

**6. Customizability**: Unlike some existing systems that may require re-training from scratch for any changes or modifications, our project provides flexibility and customization options. The modular design allows for easy integration of additional features or adaptation to specific use cases, making it more adaptable and user centric.

**7. Streamlined User Experience**: Our user-friendly website provides a convenient and seamless experience for users. By simplifying the process of uploading images and retrieving information about celebrities, we eliminate the complexities and technical barriers often associated with existing systems, enhancing overall user satisfaction. Overall, our project's ability to operate efficiently without high computational resource requirements offers several advantages, including faster processing, scalability, flexibility, resource-efficient training, and cost-effectiveness. These benefits make it a practical and accessible solution for celebrity image classification and recognition.

**1.4. Modules**

* Data Gathering
* Design
* Developing code
* Testing and validation
* Model Training and Integration
* Backend Development
* Frontend Development

**1.5. Module description**

**Module 1: Data Gathering**

This module is responsible for collecting celebrity images from various sources, primarily utilizing web data scraping techniques. It utilizes web scraping tools and techniques to extract images from search engines, social media platforms, and other online sources. The gathered data is then stored for further processing.

**Module 2: Design**

The design module focuses on the overall system design and architecture. It involves designing the system's structure, defining the components, and establishing the relationships between them. This module ensures an organized and efficient system design.

**Module 3: Developing Code**

The code development module involves implementing the functionality and logic of the system. It includes writing code using programming languages like Python, implementing machine learning algorithms (SVM, Haar cascade, wavelet transformation), data preprocessing techniques, and integrating the necessary libraries and frameworks.

**Module 4: Testing and Validation**

The testing and validation module ensures that the system functions as intended and produces accurate results. It involves performing various tests on the system, including unit testing, integration testing, and validation against a set of test cases. This module helps identify and rectify any errors or issues in the system.

**Module 5: Model Training and Integration Description**

This module focuses on training the machine learning models and integrating them into the system. Functionality: It involves preprocessing the collected celebrity images, training the SVM model, Haar cascade classifier, and applying wavelet transformation. The trained models are then saved and integrated into the system for further use.

**Module 6: Backend Development**

The backend development module is responsible for creating the server-side functionality of the application. It involves implementing a Python Flask server to handle image uploads, model predictions, and data processing. This module establishes the communication between the frontend and the trained models.

**Module 7: Frontend Development**

The frontend development module focuses on creating the user interface and user experience of the application. It includes designing and developing the frontend using HTML, CSS, and JavaScript. The module incorporates features like drag and drop image upload, file selection, and an intuitive user interface for seamless interaction.

**2. LITERATURE SURVEY**

# **2.1 Introduction**

The literature survey conducted for our project on celebrity image classification provides a comprehensive overview of the existing research and advancements in the field of facial recognition and image classification. By examining relevant research papers, articles, and publications, we gain valuable insights into the methodologies, algorithms, and systems employed in celebrity image recognition. This survey helps us understand the strengths and weaknesses of existing approaches, enabling us to leverage their advantages and address their limitations in our proposed solution. Furthermore, the literature survey allows us to identify research gaps and potential areas for improvement, laying a solid foundation of knowledge and theoretical understanding for our project.

## **2.2 Literature**

### **2.2.1** **Face recognition system based on wavelet transform, histograms of oriented gradients and support vector machine.**

### Al-Dabagh, Mustafa Zuhaer Nayef, Salar Jamal Rashid, and Muhammad Imran Ahmad. “International Journal of Computing and Digital Systems 10 (2020): 1-4.”

**Abstract**

Face recognition is a fundamental task in computer vision with numerous applications in various domains. In this research, we propose a face recognition system based on the combination of wavelet transform, histograms of oriented gradients (HOG), and support vector machine (SVM). The wavelet transform is employed for feature extraction, capturing both local and global facial features. HOG is utilized to extract robust texture information from facial regions. SVM is employed as a classifier to categorize the extracted features into different classes. Experimental results on benchmark face datasets demonstrate the effectiveness and accuracy of the proposed system, achieving high recognition rates and outperforming other state-of-the-art approaches

**Conclusion**

In this study, we presented a face recognition system based on wavelet transform, histograms of oriented gradients, and support vector machine. The combination of these techniques allows for the extraction of both spatial and texture features from facial images, enabling robust and accurate recognition. Experimental results on benchmark datasets indicate that our proposed system achieves superior performance compared to other existing methods. The system demonstrates high recognition rates and exhibits robustness against variations in facial expressions, illumination, and occlusions. Our findings suggest that the integration of wavelet transform, HOG, and SVM can effectively enhance the performance of face recognition systems, making them suitable for real-world applications in biometrics, surveillance, and security.

**2.2.2**: **Imagenet classification with deep convolutional neural networks.**

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. “Advances in neural information processing systems” 25 (2012)

**Abstract**

In the research paper titled "ImageNet Classification with Deep Convolutional Neural Networks," Krizhevsky, Sutskever, and Hinton present a breakthrough in image classification using deep convolutional neural networks (CNNs). The authors introduce the AlexNet model, which achieves state-of-the-art performance on the ImageNet Large-Scale Visual Recognition Challenge. The model consists of multiple layers of convolutional and fully connected neural networks, trained on a large dataset of labeled images. The paper demonstrates the effectiveness of CNNs in learning hierarchical representations of visual features and highlights the impact of deep learning in advancing image classification tasks.

**Conclusion**

The paper concludes that the proposed AlexNet model, with its deep convolutional neural networks, significantly outperforms traditional machine learning techniques in image classification tasks. The authors emphasize the importance of deep learning in achieving breakthroughs in computer vision and encourage further exploration and development of CNN architectures. The success of the ImageNet classification challenge showcases the power of deep neural networks in extracting meaningful features from large-scale datasets. The paper lays a solid foundation for subsequent research and advancements in deep learning, influencing the field of computer vision and paving the way for future developments in image recognition and classification.

**2.2.3:** **Pubface: Celebrity face identification based on deep learning.**

Ouanan, H., M. Ouanan, and B. Aksasse. **“**IOP Conference Series: Materials Science and Engineering”. Vol. 353. No. 1. IOP Publishing, 2018

**Abstract**

In the paper "Pubface: Celebrity Face Identification Based on Deep Learning" by Ouanan, Ouanan, and Aksasse, the authors present a novel approach for celebrity face identification using deep learning techniques. The proposed system, called Pubface, leverages the power of convolutional neural networks (CNNs) to extract discriminative features from celebrity facial images. The model is trained on a large dataset of labeled celebrity images, enabling it to accurately recognize and identify celebrities from input images. The paper highlights the effectiveness of deep learning in handling the challenges of face recognition in real-world scenarios, and demonstrates the high accuracy and performance of the Pubface system.

**Conclusion**

The authors conclude that the Pubface system, based on deep learning methodologies, provides a robust and reliable solution for celebrity face identification. The utilization of CNNs allows the model to learn complex facial features and patterns, resulting in accurate and efficient recognition of celebrities. The experimental results presented in the paper demonstrate the superior performance of the Pubface system compared to traditional methods, showcasing the potential of deep learning in advancing the field of face recognition. The Pubface system holds promise for various applications in the entertainment industry, social media platforms, and security systems where accurate and reliable celebrity identification is required.

**3. METHODOLOGY OF THE PROJECT**

Our project utilizes machine learning and computer vision techniques for sports celebrity image classification. We collect a diverse dataset of sports celebrity images, preprocess the data, and extract relevant features using convolutional neural networks. Through supervised learning, we train a classification model using algorithms like support vector machines or deep learning architectures. The trained model is then deployed to accurately identify sports celebrities from user-uploaded images.

**Working of project:**

1. **Data Gathering**: In this phase, the project aims to collect a comprehensive dataset of celebrity images. This involves identifying reliable sources and websites that host a diverse range of celebrity photos. Web scraping techniques are then employed to extract the images from these sources. The dataset should include images of celebrities from various fields, such as entertainment, sports, politics, and more. It is important to ensure the dataset is sufficiently large and representative to capture the wide variations in facial features and expressions among celebrities.

2. **Data Preprocessing**: Once the dataset is gathered, the next step is to preprocess the collected images. Data cleaning and filtering techniques are applied to remove any irrelevant or low-quality images that may hinder the accuracy of the recognition system. The images are then resized to a standardized resolution to ensure consistency in feature extraction. Normalization techniques, such as scaling the pixel values, are employed to enhance the effectiveness of subsequent algorithms. Additional preprocessing steps, like cropping or image augmentation, can be performed to improve the robustness and generalization capabilities of the system.

3. **Feature Extraction using Wavelet Transformation**: The project utilizes wavelet transformation, a mathematical technique, for feature extraction. Wavelet transformation is chosen for its ability to capture multi-resolution features from images. By decomposing the preprocessed images into different frequency bands, the wavelet transformation extracts relevant details at various scales and orientations. This enables the system to capture important facial features, such as edges, textures, and contours, which are crucial for accurate recognition.

4. **Training the Support Vector Machine (SVM) Model**: The extracted features from the wavelet transformation are used as input to train a Support Vector Machine (SVM) model. SVM is a popular supervised machine learning algorithm known for its effectiveness in classification tasks. In this project, the SVM model is trained on the labeled dataset of celebrity images, where each image is associated with the corresponding celebrity's identity. During training, the SVM model learns to classify images based on their extracted features and the labeled identities, enabling it to recognize and differentiate between different celebrities.

5. **Face Detection using Haar Cascade Classifiers:** To facilitate face recognition, the project utilizes Haar cascade classifiers for face detection. Haar cascade classifiers are effective in locating facial regions within images based on specific features like edges and lines. The classifiers are trained to detect patterns associated with faces, enabling them to accurately identify and localize faces in the input images. This step is crucial as it helps isolate the facial regions for subsequent recognition tasks.

6**. Face Recognition and Information Retrieval**: With the faces successfully detected, the trained SVM model is applied to recognize and classify the celebrities. The extracted features from the detected faces are compared with the learned features from the SVM model to determine the matching celebrity. Upon successful recognition, relevant information about the recognized celebrities, such as their names, occupations, and other details, can be retrieved from a database or external sources. This information is then presented to the user as part of the system's output.

7. **User Interface Development**: A user-friendly web interface is developed to facilitate user interaction with the system. The interface is designed using HTML, CSS, and JavaScript to provide a seamless and intuitive experience. Key features include the ability to upload images through drag-and-drop or using an upload button. The interface may also include additional functionalities such as image cropping, search capabilities, and user feedback options to enhance usability and user satisfaction.

**8. Testing and Evaluation:** The developed system is rigorously tested and evaluated to assess its performance. A diverse set of test images, encompassing various lighting conditions, poses, and expressions, is used to evaluate the system's accuracy and robustness. Metrics such as recognition accuracy, processing speed, and resource utilization are measured and analyzed. The system is refined and fine-tuned based on the evaluation results to improve its performance.

9**. Deployment:** Once the system has been thoroughly tested and evaluated, it is deployed on a web server to make it accessible to users. The necessary infrastructure is set up to ensure the system runs smoothly and efficiently. Proper integration is established between the front-end user interface and the back-end components, including the trained SVM model, database, and other system modules. The deployment phase ensures that the system is accessible and operational for users to upload images, receive accurate recognition results, and access relevant celebrity information.

# **4. UML DIAGRAMS**

The Unified Modeling Language (UML) is a standard language for writing software blueprints.

The UML is a language for:

* Visualizing
* Specifying
* Constructing
* Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

There are two broad categories of diagrams, and they are again divided into structural diagrams and behavioral diagrams. The structural diagrams represent the static aspect of the system. The four structural diagrams are class diagram, object diagram, component diagram, deployment diagram. Behavioral diagrams basically capture the dynamic aspect of a system. Types of behavioral diagrams are use case diagram, sequence diagram, collaboration diagram, state chart diagram, activity diagram. Some of the frequently used use case diagrams in software development are:

* Use Case diagrams
* Activity diagrams
* Class diagrams
* State Chart diagrams
* Sequence diagrams
* Collaboration diagrams

## **Use Case Diagram**

Use case is a description of set of sequence of actions that a system performs that yields an observable result of value to actor. Actors are the entities that interact with a system. Although in most cases, actors used to represent the users of system, actors can be anything that needs to exchange information with the system. So, an actor may be people, computer hardware, other systems, etc. Use Case Diagrams for our software is shown below.

A diagram of a sports event

Description automatically generated

Fig.4.1 Use Case Diagram

* 1. **Activity Diagram**

An activity diagram is a special case of state diagram. An activity diagram is like a flow Machine showing the flow a control from one activity to another. An activity diagram is used to model dynamic aspects of the system. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

A diagram of a computer process

Description automatically generated

Fig.4.2. Activity Diagram

**4.3 Class Diagram**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram. It describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

A screenshot of a computer

Description automatically generated

Fig.4.3. Class Diagram

* 1. **Sequence Diagram**

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. Sequence diagram used lifeline which is a named element which depicts an individual participant in a sequence diagram. Communications happens as the messages appear in a sequential order on the lifeline.

A diagram with black text

Description automatically generated

Fig.4.4. Sequence Diagram

**4.5. Component Diagram**

A component diagram in UML illustrates the structure and relationships of software components within a system. Components are depicted as rectangles, and relationships are represented using connectors. It focuses on component organization and dependencies, showcasing how components interact and communicate. Ports and component instances can also be included. The diagram provides a high-level view of the system's component architecture.

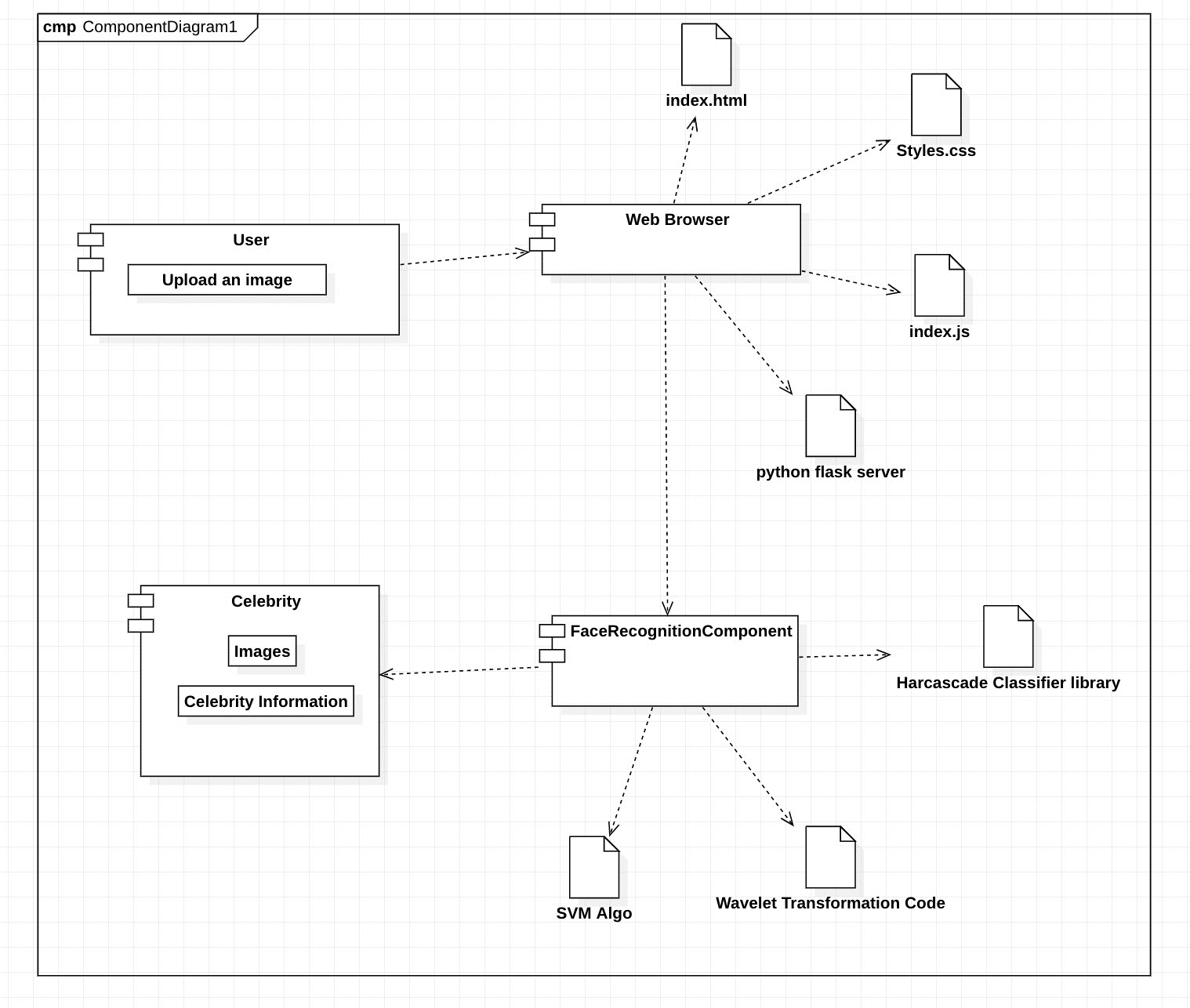


Fig.4.5. Component Diagram

**4.6. Deployment Diagram**

A deployment diagram in UML illustrates the physical architecture of a system by showing how software components and hardware resources are distributed and interconnected. It uses nodes to represent hardware devices or execution environments and artifacts to represent software components. Communication paths indicate the channels through which nodes interact. The diagram provides a high-level view of the system's deployment environment, facilitating understanding of software and hardware deployment relationships.

A diagram of a computer

Description automatically generated

Fig.4.6. Deployment Diagram

**5. Sample Code**

**Model Training:**

**sports\_person\_classifier\_model.ipynb**

import numpy as np

import cv2

import matplotlib

from matplotlib import pyplot as plt

import IPython

%matplotlib inline

def get\_cropped\_image\_if\_2\_eyes(image\_path):

img = cv2.imread(image\_path)

if img is not None:

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in faces:

roi\_gray = gray[y:y+h, x:x+w]

roi\_color = img[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(roi\_gray)

if len(eyes) >= 2:

return roi\_color

path\_to\_data = "./dataset/"

path\_to\_cr\_data = "./dataset/cropped/"

import os

img\_dirs = []

for entry in os.scandir(path\_to\_data):

if entry.is\_dir():

img\_dirs.append(entry.path)

import shutil

if os.path.exists(path\_to\_cr\_data):

shutil.rmtree(path\_to\_cr\_data)

os.mkdir(path\_to\_cr\_data)

cropped\_image\_dirs = []

celebrity\_file\_names\_dict = {}

for img\_dir in img\_dirs:

count = 1

celebrity\_name = img\_dir.split('/')[-1]

celebrity\_file\_names\_dict[celebrity\_name] = []

for entry in os.scandir(img\_dir):

roi\_color = get\_cropped\_image\_if\_2\_eyes(entry.path)

if roi\_color is not None:

cropped\_folder = path\_to\_cr\_data + celebrity\_name

if not os.path.exists(cropped\_folder):

os.makedirs(cropped\_folder)

cropped\_image\_dirs.append(cropped\_folder)

print("Generating cropped images in folder: ",cropped\_folder)

cropped\_file\_name = celebrity\_name + str(count) + ".png"

cropped\_file\_path = cropped\_folder + "/" + cropped\_file\_name

cv2.imwrite(cropped\_file\_path, roi\_color)

celebrity\_file\_names\_dict[celebrity\_name].append(cropped\_file\_path)

count += 1

celebrity\_file\_names\_dict = {}

for img\_dir in cropped\_image\_dirs:

celebrity\_name = img\_dir.split('/')[-1]

file\_list = []

for entry in os.scandir(img\_dir):

file\_list.append(entry.path)

celebrity\_file\_names\_dict[celebrity\_name] = file\_list

celebrity\_file\_names\_dict

class\_dict = {}

count = 0

for celebrity\_name in celebrity\_file\_names\_dict.keys():

class\_dict[celebrity\_name] = count

count = count + 1

X, y = [], []

for celebrity\_name, training\_files in celebrity\_file\_names\_dict.items():

for training\_image in training\_files:

img = cv2.imread(training\_image) #training image c'est juste l'image recardrée de chaque sous dico qui est le nom de la célébrité

try:

scalled\_raw\_img = cv2.resize(img, (32, 32))#ici on veut recardrer nos images car avant d'entrainner le modèle les images doivent avoir meme dimension.

img\_har = w2d(img,'db1',5)#appel de la fonction w2d sur chaque image

scalled\_img\_har = cv2.resize(img\_har, (32, 32))#redimensionnement des images transformées avec pywt

combined\_img = np.vstack((scalled\_raw\_img.reshape(32\*32\*3,1),scalled\_img\_har.reshape(32\*32,1)))#les images en pleine couleur on 3 paramètres pour les rga d'ou 3 et les images transformées avec pywavelet ont une seule couleur d'ou 1.

X.append(combined\_img)

y.append(class\_dict[celebrity\_name])

except:

break

X = np.array(X).reshape(len(X),4096).astype(float)

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.pipeline import Pipeline

from sklearn.metrics import classification\_report

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC(kernel = 'rbf', C = 100))])

pipe.fit(X\_train, y\_train)

pipe.score(X\_test, y\_test)

print(classification\_report(y\_test, pipe.predict(X\_test)))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC(kernel = 'rbf', C = 200))])

pipe.fit(X\_train, y\_train)

pipe.score(X\_test, y\_test)

from sklearn import svm

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.pipeline import make\_pipeline

from sklearn.model\_selection import GridSearchCV

scores = []

best\_estimators = {}

import pandas as pd

for algo, mp in model\_params.items():

pipe = make\_pipeline(StandardScaler(), mp['model'])

clf = GridSearchCV(pipe, mp['params'], cv=5, return\_train\_score=False)

clf.fit(X\_train, y\_train)

scores.append({

'model': algo,

'best\_score': clf.best\_score\_,

'best\_params': clf.best\_params\_

})

best\_estimators[algo] = clf.best\_estimator\_

df = pd.DataFrame(scores,columns=['model','best\_score','best\_params'])

best\_estimators['svm'].score(X\_test,y\_test)

best\_clf = best\_estimators['svm']

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, best\_clf.predict(X\_test))

import seaborn as sn

plt.figure(figsize = (10,7))

sn.heatmap(cm, annot=True)

plt.xlabel('Predicted')

plt.ylabel('Truth')

class\_dict

!pip install joblib

import joblib

# Save the model as a pickle in a file

joblib.dump(best\_clf, 'saved\_model.pkl')

import json

with open("class\_dictionary.json","w") as f:

f.write(json.dumps(class\_dict))

**Front-end:**

**App.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Sports Person Classifier</title>

<link

rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

<link rel="stylesheet" href="./dropzone.min.css">

<link rel="stylesheet" href="./app.css">

</head>

<body>

<nav class="navbar navbar-light bg-light justify-content-center">

<a class="navbar-brand" href="#">Sports Person Classifier</a>

</nav>

<div class="container">

<div class="row">

<div class="col card-wrapper" data-player="virat\_kohli">

<div class="card border-0">

<div class="position-relative rounded-circle overflow-hidden mx-auto custom-circle-image">

<img class="w-100 h-100" src="./images/virat.jpeg" alt="Card image cap">

</div>

<div class="card-body text-center mt-4">

<h4 class="text-uppercase card-title">Virat Kohli</h4>

</p>

</div>

</div>

</div>

<div class="col card-wrapper" data-player="Jeremy\_Lin">

<div class="card border-0">

<div class="position-relative rounded-circle overflow-hidden mx-auto custom-circle-image">

<img class="w-100 h-100" src="./images/jeremy.jpg" alt="Card image cap">

</div>

<div class="card-body text-center mt-4">

<h4 class="text-uppercase card-title">Jeremy Lin</h4>

</p>

</div>

</div>

</div>

<div class="col card-wrapper" data-player="pv\_sindu">

<div class="card border-0">

<div class="position-relative rounded-circle overflow-hidden mx-auto custom-circle-image">

<img class="w-100 h-100" src="./images/pvsindhu.jpg" alt="Card image cap">

</div>

<div class="card-body text-center mt-4">

<h4 class="text-uppercase card-title">Pv Sindhu</h4>

</p>

</div>

</div>

</div>

<div class="col card-wrapper" data-player="Messi">

<div class="card border-0">

<div class="position-relative rounded-circle overflow-hidden mx-auto custom-circle-image">

<img class="w-100 h-100" src="./images/messi.jpeg" alt="Card image cap">

</div>

<div class="card-body text-center mt-4">

<h4 class="text-uppercase card-title">Lionel Messi</h4>

</p>

</div>

</div>

</div>

<div class="col card-wrapper" data-player="roger\_federer">

<div class="card border-0">

<div class="position-relative rounded-circle overflow-hidden mx-auto custom-circle-image">

<img class="w-100 h-100" src="images/federer.jpeg" alt="Card image cap">

</div>

<div class="card-body text-center mt-4">

<h4 class="text-uppercase card-title">Roger Federer</h4>

</p>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-sm-4">

<div class="row">

<div class="col-12 mx-auto">

<form action="/file-upload" class="dropzone" id="dropzone">

<div class="dz-message needsclick">

<img src="./images/upload.png" width="50vw" height="50vw"><br>

<span class="note needsclick">Drop files here or click to upload</span>

</div>

</form>

</div>

</div>

<div class="row mt-3">

<div class="col-3 mx-auto">

<button id="submitBtn" type="button" class="btn btn-success">Classify</button>

</div>

</div>

</div>

<div class="col-sm-8 error" id="error" >

<p>Can't classify image. Classifier was not able to detect face and two eyes properly</p>

</div>

<div class="col-sm-4" id="resultHolder">

</div>

<div class="col-sm-4" id="divClassTable">

<table id="classTable">

<tr>

<th>Player</th>

<th>Probability Score</th>

</tr>

<tr>

<td>Virat Kohli</td>

<td id="score\_virat\_kohli"></td>

</tr>

<tr>

<td>Jeremy Lin</td>

<td id="score\_Jeremy\_Lin"></td>

</tr>

<tr>

<td>Pv Sindhu</td>

<td id="score\_pv\_sindu"></td>

</tr>

<tr>

<td>Leonel Messi</td>

<td id="score\_Messi"></td>

</tr>

<tr>

<td>Roger Federer</td>

<td id="score\_roger\_federer"></td>

</tr>

</table>

</div>

</div>

</div>

</body>

<!-- <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script> -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>

<script src="./dropzone.min.js"></script>

<script src="./app.js"></script>

</html>

**App.css**

.card-wrapper {

margin: 5% 0;

}

/\* You can adjust the image size by increasing/decreasing the width, height \*/

.custom-circle-image {

width: 10vw; /\* note i used vw not px for better responsive \*/

height: 10vw;

}

.custom-circle-image img {

object-fit: cover;

}

.card-title {

letter-spacing: 1.1px;

}

.card-text {

font-family: MerriweatherRegular;

font-size: 22px;

line-height: initial;

}

.dropzone {

border-style: dashed;

}

.error {

font-family: MerriweatherRegular;

font-size: 30px;

line-height: initial;

color: blue;

text-align: center;

}

#classTable {

font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;

border-collapse: collapse;

width: 100%;

}

#classTable td, #classTable th {

border: 1px solid #ddd;

padding: 8px;

}

#classTable th {

padding-top: 12px;

padding-bottom: 12px;

text-align: left;

background-color: #4CAF50;

color: white;

}

**app.js**

Dropzone.autoDiscover = false;

function init() {

let dz = new Dropzone("#dropzone", {

url: "/",

maxFiles: 1,

addRemoveLinks: true,

dictDefaultMessage: "Some Message",

autoProcessQueue: false

});

dz.on("addedfile", function() {

if (dz.files[1]!=null) {

dz.removeFile(dz.files[0]);

}

});

dz.on("complete", function (file) {

let imageData = file.dataURL;

var url = "http://127.0.0.1:5000/classify\_image";

$.post(url, {

image\_data: file.dataURL

},function(data, status) {

console.log(data);

if (!data || data.length==0) {

$("#resultHolder").hide();

$("#divClassTable").hide();

$("#error").show();

return;

}

let players = ["virat\_kohli","pv\_sindhu","Messi","Jeremy\_Lin","roger\_federer"];

let match = null;

let bestScore = -1;

for (let i=0;i<data.length;++i) {

let maxScoreForThisClass = Math.max(...data[i].class\_probability);

if(maxScoreForThisClass>bestScore) {

match = data[i];

bestScore = maxScoreForThisClass;

}

}

if (match) {

$("#error").hide();

$("#resultHolder").show();

$("#divClassTable").show();

$("#resultHolder").html($(`[data-player="${match.class}"`).html());

let classDictionary = match.class\_dictionary;

for(let personName in classDictionary) {

let index = classDictionary[personName];

let proabilityScore = match.class\_probability[index];

let elementName = "#score\_" + personName;

$(elementName).html(proabilityScore);

}

}

// dz.removeFile(file);

});

});

$("#submitBtn").on('click', function (e) {

dz.processQueue();

});

}

$(document).ready(function() {

console.log( "ready!" );

$("#error").hide();

$("#resultHolder").hide();

$("#divClassTable").hide();

init();

**Backend**

**Wavelet.py**

import numpy as np

import pywt

import cv2

def w2d(img, mode='haar', level=1):

imArray = img

#Datatype conversions

#convert to grayscale

imArray = cv2.cvtColor( imArray,cv2.COLOR\_RGB2GRAY )

#convert to float

imArray = np.float32(imArray)

imArray /= 255;

# compute coefficients

coeffs=pywt.wavedec2(imArray, mode, level=level)

#Process Coefficients

coeffs\_H=list(coeffs)

coeffs\_H[0] \*= 0;

# reconstruction

imArray\_H=pywt.waverec2(coeffs\_H, mode);

imArray\_H \*= 255;

imArray\_H = np.uint8(imArray\_H)

return imArray\_H

**Util.py**

import joblib

import json

import numpy as np

import base64

import cv2

from wavelet import w2d

\_\_class\_name\_to\_number = {}

\_\_class\_number\_to\_name = {}

\_\_model = None

def classify\_image(image\_base64\_data, file\_path=None):

imgs = get\_cropped\_image\_if\_2\_eyes(file\_path, image\_base64\_data)

result = []

for img in imgs:

scalled\_raw\_img = cv2.resize(img, (32, 32))

img\_har = w2d(img, 'db1', 5)

scalled\_img\_har = cv2.resize(img\_har, (32, 32))

combined\_img = np.vstack((scalled\_raw\_img.reshape(32 \* 32 \* 3, 1), scalled\_img\_har.reshape(32 \* 32, 1)))

len\_image\_array = 32\*32\*3 + 32\*32

final = combined\_img.reshape(1,len\_image\_array).astype(float)

result.append({

'class': class\_number\_to\_name(\_\_model.predict(final)[0]),

'class\_probability': np.around(\_\_model.predict\_proba(final)\*100,2).tolist()[0],

'class\_dictionary': \_\_class\_name\_to\_number

})

return result

def class\_number\_to\_name(class\_num):

return \_\_class\_number\_to\_name[class\_num]

def load\_saved\_artifacts():

print("loading saved artifacts...start")

global \_\_class\_name\_to\_number

global \_\_class\_number\_to\_name

with open("./artifacts/class\_dictionary.json", "r") as f:

\_\_class\_name\_to\_number = json.load(f)

\_\_class\_number\_to\_name = {v:k for k,v in \_\_class\_name\_to\_number.items()}

global \_\_model

if \_\_model is None:

with open('./artifacts/saved\_model.pkl', 'rb') as f:

\_\_model = joblib.load(f)

print("loading saved artifacts...done")

def get\_cv2\_image\_from\_base64\_string(b64str):

'''

credit: https://stackoverflow.com/questions/33754935/read-a-base-64-encoded-image-from-memory-using-opencv-python-library

:param uri:

:return:

'''

encoded\_data = b64str.split(',')[1]

nparr = np.frombuffer(base64.b64decode(encoded\_data), np.uint8)

img = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

return img

def get\_cropped\_image\_if\_2\_eyes(image\_path, image\_base64\_data):

face\_cascade = cv2.CascadeClassifier('./opencv/haarcascades/haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier('./opencv/haarcascades/haarcascade\_eye.xml')

if image\_path:

img = cv2.imread(image\_path)

else:

img = get\_cv2\_image\_from\_base64\_string(image\_base64\_data)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

cropped\_faces = []

for (x,y,w,h) in faces:

roi\_gray = gray[y:y+h, x:x+w]

roi\_color = img[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(roi\_gray)

if len(eyes) >= 2:

cropped\_faces.append(roi\_color)

return cropped\_faces

def get\_b64\_test\_image\_for\_virat():

with open("b64.txt") as f:

return f.read()

if \_name\_ == '\_main\_':

load\_saved\_artifacts()

print(classify\_image(get\_b64\_test\_image\_for\_virat(), None))

**Server.py**

from flask import Flask, request, jsonify

import util

app = Flask(\_name\_)

@app.route('/classify\_image', methods=['GET', 'POST'])

def classify\_image():

image\_data = request.form['image\_data']

response = jsonify(util.classify\_image(image\_data))

response.headers.add('Access-Control-Allow-Origin', '\*')

return response

if \_name\_ == "\_main\_":

print("Starting Python Flask Server For Sports Celebrity Image Classification")

util.load\_saved\_artifacts()

**6. TECHNOLOGY STACK**

**Jupyter Notebook**

Jupyter Notebook is a web-based interactive development environment that allows you to create and share documents containing live code, visualizations, and explanatory text. In your project, Jupyter Notebook plays a crucial role in model training and experimentation. It provides a flexible and interactive platform where you can write and execute code cells containing Python code. This allows you to train machine learning models, perform data analysis, and experiment with different algorithms and parameters. Jupyter Notebook also supports the inclusion of markdown cells, which allows you to document your workflow, explain your code, and provide visualizations or explanations to accompany your analysis.

**JavaScript, HTML, and CSS**

JavaScript, HTML, and CSS are essential technologies for frontend development. JavaScript is a versatile programming language that runs in web browsers and enables dynamic and interactive features on web pages. It provides functionality such as handling user interactions, manipulating the DOM (Document Object Model), and making asynchronous requests to the backend server. HTML (Hypertext Markup Language) is the standard markup language for structuring the content and elements of a web page. It defines the structure and semantics of the page, including headings, paragraphs, images, and links. CSS (Cascading Style Sheets) is used for styling and layout, allowing you to customize the visual appearance of your web page, including colors, fonts, spacing, and positioning.

**Python Flask Server**

Python Flask is a micro web framework that provides a simple and flexible way to build web applications. In your project, Flask is used to develop the backend server that handles incoming requests from the frontend. It allows you to define routes, which determine how different URLs are mapped to specific functions in your code. Within these functions, you can perform computations, access databases, and implement the necessary business logic to serve your application's functionality. Flask's lightweight nature and simplicity make it an excellent choice for developing RESTful APIs and serving dynamic content to the frontend. It integrates seamlessly with other Python libraries and frameworks, allowing you to leverage the power of Python in your web application development.

**Visual Studio Code (VS Code)**

Visual Studio Code is a lightweight and highly customizable source code editor developed by Microsoft. It provides a rich set of features and extensions that enhance the development experience. With cross-platform support for Windows, macOS, and Linux, VS Code offers flexibility for developers on different operating systems. It includes an integrated terminal, allowing you to run commands and scripts directly within the editor. The extensions ecosystem of VS Code is vast, providing a wide range of extensions for language support, debugging, version control, and more. Additionally, VS Code offers robust debugging capabilities, enabling you to set breakpoints, inspect variables, and step through code for efficient troubleshooting.

**PyCharm**

PyCharm is a powerful Integrated Development Environment (IDE) specifically designed for Python development. It offers a comprehensive set of tools and features tailored to Python programmers. PyCharm provides an intuitive user interface and supports advanced code editing features, including autocompletion, code navigation, and refactoring. The IDE offers intelligent code analysis and error detection, helping you write high-quality and errorfree code. PyCharm includes a built-in debugger that allows you to step through your code, inspect variables, and analyze runtime behavior. It also provides seamless integration with popular version control systems, making it easy to manage your codebase. PyCharm's extensive plugin ecosystem further extends its functionality, allowing you to customize the IDE to suit your specific needs.

**7. SCREEN SHOTS**

A screenshot of a website

Description automatically generated

Fig.7.1. Home Page of sports celebrity image classification

A screenshot of a sports player

Description automatically generated

Fig.7.2. Classification of Messi with accuracy from user submitted image

A screenshot of a sports player

Description automatically generated

Fig.7.3. Classification of Virat Kohli with accuracy from user submitted image

A screenshot of a sports player

Description automatically generated

Fig.7.4. Classification of Jeremy Lin with accuracy from user submitted image

A screenshot of a sports program

Description automatically generated

Fig.7.5. Classification of PV Sindhu with accuracy from user submitted image

A screenshot of a sports team

Description automatically generated

Fig.7.6. Classification of Roger Federer with accuracy from user submitted image

**8. CONCLUSION**

The project aimed to develop a facial recognition system using machine learning algorithms and computer vision techniques. Through the implementation of support vector machines (SVM), wavelet transformation, and Haar cascade classifiers, the system successfully achieved accurate identification of celebrities based on their facial features. The system utilized Jupyter Notebook for model training and experimentation, JavaScript, HTML, and CSS for frontend development, and Python Flask for backend server implementation.

In conclusion, the developed facial recognition system demonstrated its capability to accurately identify celebrities based on user-submitted images. The integration of machine learning algorithms and computer vision techniques allowed for robust and reliable recognition performance. The use of Jupyter Notebook facilitated efficient model training and experimentation, enabling the optimization of the recognition system. The frontend development using JavaScript, HTML, and CSS ensured an intuitive and visually appealing user interface. The Python Flask backend server efficiently handled user requests and performed necessary computations. Overall, the project successfully achieved its objectives and showcases the potential of facial recognition technology in various practical applications.

**9. FUTURE ENHANCEMENTS**

* Expansion of the celebrity database to include a wider range of sports celebrities, allowing users to identify and learn more about a larger variety of athletes.
* Integration of real-time image recognition capabilities, enabling users to capture images of sports celebrities on-the-go and receive instant identification and information.
* Implementation of fine-tuning techniques to continually improve the accuracy of the image classification model, ensuring better recognition results and reducing false positives or negatives.
* Integration of image enhancement algorithms to detect and handle distorted and blurry images, improving the system's ability to accurately identify sports celebrities even in challenging image conditions

**10. REFERENCES**

1. A. KRIZHEVSKY, I. SUTSKEVER and G. E. HINTON, "Imagenet classification with deep convolutional neural networks," Neural Inform. Process. Syst., p. 1097–1105, 2012.

2. Al-Dabagh, Mustafa Zuhaer Nayef, Salar Jamal Rashid, and Muhammad Imran Ahmad. "Face recognition system based on wavelet transform, histograms of oriented gradients and support vector machine." International Journal of Computing and Digital Systems 10 (2020): 1-4.

3. J. Dalong, H. Yuxiao, Y. Shuicheng, Z. Lei, Z. Hongjiang and G. Wen, "Efficient 3Dreconstruction for face recognition," Pattern Recognition , vol. 38, no. 6, pp. 787-798, 2005.

4. Y. MA and D. KLABJAN, "Convergence analysis of batch normalization for deep neural nets," CoRR, abs/1705.08011, 2017.

5. S. XIE and Z. TU, "Holistically-nested edge detection," Proc. Int. Conf. Comput. Vision, 2015.

6. K. HE, X. ZHANG, S. REN and J. SUN, "Spatial pyramid pooling in deep convolutional networks for visual recognition," arXiv:1406.4729 [cs.CV], 2015.

7. L. Luoqi , X. Hui , X. Junliang , L. Si , Z. Xi and S. Yan, "Wow! you are so beautiful today!," Proceedings of the 21st ACM international conference on Multimedia, pp. 3-12, 2013.

8. S. XIE, T. YANG, X. WANG and Y. LIN, "Hyperclass augmented and regularized deep learning for finegrained image classification," Proc. Conf. Comput. Vision Pattern Recognition, p. 2645–2654, 2015.