

A
Major Project
On
**MISSING CHILD IDENTIFICATION SYSTEM USING DEEP
LEARNING AND MULTICLASS SVM**

(Submitted in partial fulfilment of the requirements for the award of Degree)

Bachelor of Technology
in
COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

By
NITHIN PRABHATH (207R1A6784)

Under the guidance of
SANJIB KUMAR NAYAK
(Assistant. Professor)



Department of Computer Science & Engineering (Data Science)

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

**Accredited by NBA & NAAC with 'A' Grade
Approved by AICTE, New Delhi and JNTU, Hyderabad.**

2020-2024

Department of Computer Science & Engineering (Data Science)



CERTIFICATE

This is to certify that the project entitled “**MISSING CHILD IDENTIFICATION SYSTEM USING DEEP LEARNING AND MULTICLASS SVM**” being submitted by **NITHIN PRABHATH (207R1A6784)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering (Data Science) to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Sanjib Kumar Nayak

(Assistant. Professor)

Internal Guide

Dr. K. Srinivas

Professor & HoD

External Examiner

Submitted for viva voice Examination held on: _____

ACKNOWLEDGMENT

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I take this opportunity to express my profound gratitude and deep regard to my guide “**Mr. Sanjib Kumar Nayak**”, his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry me a long way in the journey of life on which we are about to embark.

I also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Mrs. V. Sandya, Mr. A. Veerender and Mrs. K. Sudha Pavani** for their cordial support, valuable information and guidance, which helped me in completing this task through various stages.

I am obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. I also express my sincere gratitude to **Sri. Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. I am grateful for their constant support and help.

Finally, I would like to take this opportunity to thank my family for their constant encouragement, without which this assignment would not be completed. I sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

NITHIN PRABHATH (207R1A6784)

ABSTRACT

In India a countless number of children are reported missing every year. Among the missing child cases a large percentage of children remain untraced. This paper presents a novel use of deep learning methodology for identifying the reported missing child from the photos of multitude of children available, with the help of face recognition. The public can upload photographs of suspicious child into a common portal with landmarks and remarks. The photo will be automatically compared with the registered photos of the missing child from the repository. Classification of the input child image is performed and photo with best match will be selected from the database of missing children. For this, a deep learning model is trained to correctly identify the missing child from the missing child image database provided, using the facial image uploaded by the public. The Convolutional Neural Network (CNN), a highly effective deep learning technique for image based applications is adopted here for face recognition. Face descriptors are extracted from the images using a pre- trained CNN model VGG-Face deep architecture. Compared with normal deep learning applications, our algorithm uses convolution network only as a high level feature extractor and the child recognition is done by the trained SVM classifier. Choosing the best performing CNN model for face recognition, VGG-Face and proper training of it results in a deep learning model invariant to noise, illumination, contrast, occlusion, image pose and age of the child and it outperforms earlier methods in face recognition based missing child identification. The classification performance achieved for child identification system is 99.41%. It was evaluated on 43 Child.

TABLE OF CONTENTS

CERTIFICATES	ii
ACKNOWLEDGEMENT	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	x
LIST OF SCREENSHOTS	xi
LIST OF ABBREVIATIONS	xii
1. INTRODUCTION	1-2
1.1 PROJECT INTRODUCTION	1
1.1 SCOPE	2
1.2 PROJECT PURPOSE	2
1.3 PROJECT FEATURE	2
2. SYSTEM ANALYSIS	3-8
2.1 PROBLEM DEFINITION	3
2.2 EXISTING SYSTEM	3
2.2.1 LIMITATIONS OF EXISTING SYSTEM	3
2.3 PROPOSED SYSTEM	3
2.4 FEASIBILITY STUDY	4
2.4.1 ECONOMICAL FEASIBILITY	4
2.4.2 TECHNICAL FEASIBILITY	4

2.4.3 SOCIAL FEASIBILITY	4
2.5 SOFTWARE AND HARDWARE REQUIREMENTS	5
2.5.1 HARDWARE REQUIREMENTS	5
2.5.2 SOFTWARE REQUIREMENTS	5
2.6 PROGRAMMING LANGUAGE(S) USED	5
2.6.1 PYTHON	5
2.6.2 SQL	6
2.6.3 DJANGO	7
3. ARCHITECTURE	9 - 13
3.1 PROJECT ARCHITECTURE	9
3.2 USE CASE DIAGRAM	10
3.3 CLASS DIAGRAM	11
3.4 SEQUENCE DIAGRAM	12
3.5 ACTIVITY DIAGRAM	13
4. IMPLEMENTATION	
4.1 SAMPLE CODE	14 - 26
5. SCREENSHOTS	27 - 31
5.1 INTRODUCTION	27
6. TESTING	32
6.1 INTRODUCTION TO TESTING	32
6.2 TYPES OF TESTING	32
6.2.1 UNIT TESTING	32
6.2.2 INTEGRATION TESTING	32
6.2.3 FUNCTION TESTING	32
6.3 TEST CASES	33
6.3.1 CLASSIFICATION	33
7. CONCLUSION & FUTURE SCOPE	34
7.1 PROJECT CONCLUSION	34
7.2 FUTURE SCOPE	34

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
Table 6.1	Test cases of missing child	41

LIST OF FIGURES

FIGURE NO	DESCRIPTION	Page no
Fig 3.1	Architecture Diagram	12
Fig 3.2	Use Case Diagram	15
Fig 3.3	Class Diagram	16
Fig 3.4	Sequence Diagram	16
Fig 3.5	Sequence Diagram	17

LIST OF SCREENSHOTS

SCREENSHOT NUMBER	DESCRIPTION	PAGENO
Screenshot 5.1	Start DJANGO and run Browser	34
Screenshot 5.2	Public Upload Suspected Child	35
Screenshot 5.3	Click On Submit Button	36
Screenshot 5.4	Can See Child Not Found In Missing DB	36
Screenshot 5.5	Enter Username, Password as 'Admin' and 'Admin'	37
Screenshot 5.6	View Public Upload Missing Child Status	37
Screenshot 5.7	Officials Can See All Details And Then Take Action	38

LIST OF ABBREVIATIONS

I.	ML	Machine Learning
II.	SVM	Support Vector Machine
III.	SQL	Structured Query Language
IV.	DT	Decision Tree

Chapter 1

INTRODUCTION

1.1 Introduction

The Missing Child Identification System (MCIS) integrates deep learning and multiclass SVM to improve the accuracy and efficiency of locating missing children. It employs a facial recognition system based on CNNs to identify children from images, even accounting for changes in appearance over time. The multiclass SVM classifier prioritizes potential matches, aiding law enforcement in allocating resources effectively. By harnessing these technologies, MCIS aims to expedite search and rescue operations, ultimately reuniting missing children with their families and preventing exploitation.

1.2 Project Scope

To develop a system that leverages advanced machine learning techniques to accurately identify missing children. By collecting a dataset of images and utilizing deep learning models like CNNs, we can train the system to recognize patterns and features in the images. Additionally, incorporating multiclass SVM further enhances the classification accuracy. This project combines the power of deep learning and multiclass SVM to address the critical issue of missing child identification, contributing to the important cause of reuniting missing children with their families.

1.3 Project Purpose

The purpose of the project on "Missing Child Identification using Deep Learning and Multiclass SVM" is to develop a system that can effectively identify missing children by leveraging advanced machine learning techniques. By utilizing deep learning models, such as convolutional neural networks (CNNs), and incorporating multiclass SVM, the system aims to accurately classify and match images of missing children with available data. The ultimate goal is to contribute to the important cause of reuniting missing children with their families by providing a powerful and efficient identification tool.

1.4 Project Features

By incorporating features like data preprocessing, deep learning models, multiclass SVM, transfer learning, ensemble methods, and evaluation metrics, we can create a robust and accurate missing child identification system. These techniques will help improve the system's performance and increase the chances of reuniting missing children with their families.

Chapter 2

SYSTEM ANALYSIS

2.1 Problem Definition

The problem we aim to solve is developing a missing child identification system using Deep Learning and Multiclass SVM. This system will use advanced techniques to accurately classify and match images of missing children. By combining deep learning and multiclass SVM, we can improve the efficiency and effectiveness of the identification process. The goal is to create a powerful tool that aids in the recovery of missing children.

2.2 Existing System

Mostly missing child cases are reported to the police. The child missing from one region may be found in another region or another state, for various reasons. So even if a child is found, it is difficult to identify him/her from the reported missing cases. A framework and methodology for developing an assistive tool for tracing missing child is described in this paper. An idea for maintaining a virtual space is proposed, such that the recent photographs of children given by parents at the time of reporting missing cases is saved in a repository.

2.2.1 Limitations of Existing System

- A deeply disturbing fact about India's missing children is that while on an average 174 children go missing every day, half of them remain untraced. .
- There is no system to identify the photographs of children with different lighting conditions, noises and also images at different ages of children.

2.3 Proposed System

Whenever public uploads photo of a suspected child, the system generates template vector of the facial features from the uploaded photo. If a matching is found in the repository, the system displays the most matched photo and pushes a message to the concerned Officer portal or SMSs the alert message of matching child. Similarly, the Officer can check for any matching with the database at any time using the proposed system.

2.3.1 Advantages of Proposed System

- The proposed system is comparatively an easy, inexpensive and reliable method compared to other biometrics like finger print and iris recognition systems.

- Features extracted using a CNN network for getting facial representations gives better performance in face recognition than handcrafted features.

2.4 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 Economic Feasibility

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

2.4.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources.

2.4.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.5 Hardware & Software Requirements

2.5.1 Software Requirements:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- Python
- Django
- MySQL
- MySQLclient
- WampServer 2.4

2.5.2 Hardware Requirements:

For developing the application the following are the Hardware Requirements:

- Processor : Pentium IV or higher
- RAM : 256 MB
- Space on Hard Disk : minimum 512MB

2.6 PROGRAMMING LANGUAGE(S) USED

2.6.1 PYTHON

Python was created in 1980s by Guido van Rossum. During his research at the National Research Institute for Mathematics and Computer Science in the netherlands, he created Python – a super easy programming language in terms of reading and usage. The first ever version was released in the year 1991 which had only a few built- in data types and basic functionality.

Later, when it gained popularity among scientists for numerical computations and data analysis, in 1994, Python 1.0 was released with extra features like map, lambda, and filter functions. After that adding new functionalities and releasing newer versions of Python came into fashion.

- Python 1.5 released in 1997
- Python 2.0 released in 2000
- Python 3.0 in 2008 brought newer functionalities

The latest version of Python, Python 3.11 was released in 2022.

Newer functionalities being added to Python makes it more beneficial for developers

and improved its performance. In recent years, Python has gained a lot of popularity and is a highly demanding programming language. It has spread its demand in various fields which includes machine learning, artificial intelligence, data analysis, web development, and many more giving you a high-paying job.

2.6.1.2 Features of Python

Python has plenty of features that make it the most demanding and popular. Let's read about a few of the best features that Python has:

- Easy to read and understand
- Interpreted language
- Object-oriented programming language
- Free and open-source
- Versatile and Extensible
- Multi-platform
- Hundreds of libraries and frameworks
- Flexible, supports GUI
- Dynamically typed
- Huge and active community

2.6.2 SQL

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.

SQL is not a database system, but it is a query language.

Suppose you want to perform the queries of SQL language on the stored data in the database. You are required to install any database management system in your systems, for example, Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, etc.

2.6.2.1 Some SQL Commands

The SQL commands help in creating and managing the database. The most common SQL commands which are highly used are mentioned below:

- CREATE command
- UPDATE command
- DELETE command

- SELECT command
- DROP command

2.6.2.2 Advantages of SQL

SQL provides various advantages which make it more popular in the field of data science. It is a perfect query language which allows data professionals and users to communicate with the database. Following are the best advantages or benefits of Structured Query Language:

1. No programming needed

SQL does not require a large number of coding lines for managing the database systems. We can easily access and maintain the database by using simple SQL syntactical rules. These simple rules make the SQL user-friendly.

2. High-Speed Query Processing

A large amount of data is accessed quickly and efficiently from the database by using SQL queries. Insertion, deletion, and updation operations on data are also performed in less time.

3. Standardized Language

SQL follows the long-established standards of ISO and ANSI, which offer a uniform platform across the globe to all its users.

2.6.3 DJANGO

Django is a Python framework that makes it easier to create web sites using Python. Django takes care of the difficult stuff so that you can concentrate on building your web applications.

Django emphasizes reusability of components, also referred to as DRY (Don't Repeat Yourself), and comes with ready-to-use features like login system, database connection and CRUD operations (Create Read Update Delete).

Django follows the MVT design pattern (Model View Template).

- Model - The data you want to present, usually data from a database.
- View - A request handler that returns the relevant template and content - based on the request from the user.
- Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data

2.6.3.1 Django History

Django was invented by Lawrence Journal-World in 2003, to meet the short deadlines in the newspaper and at the same time meeting the demands of experienced web developers. Initial release to the public was in July 2005. Latest version of Django is 4.0.3 (2022).

Chapter 3

ARCHITECTURE

3.1 Project Architecture

The missing child identification system architecture encompasses data collection and preprocessing, where a dataset of images is gathered and standardized. Feature extraction is then performed using a pre-trained deep learning model to extract relevant features. These features are utilized to train both a deep learning classifier for binary classification (missing child or non-missing child) and a multiclass SVM classifier to identify specific missing children. The system is integrated into a user-friendly interface for image upload and identification. Upon deployment, the system undergoes evaluation and monitoring for performance and security, ensuring both accuracy and privacy of the data.

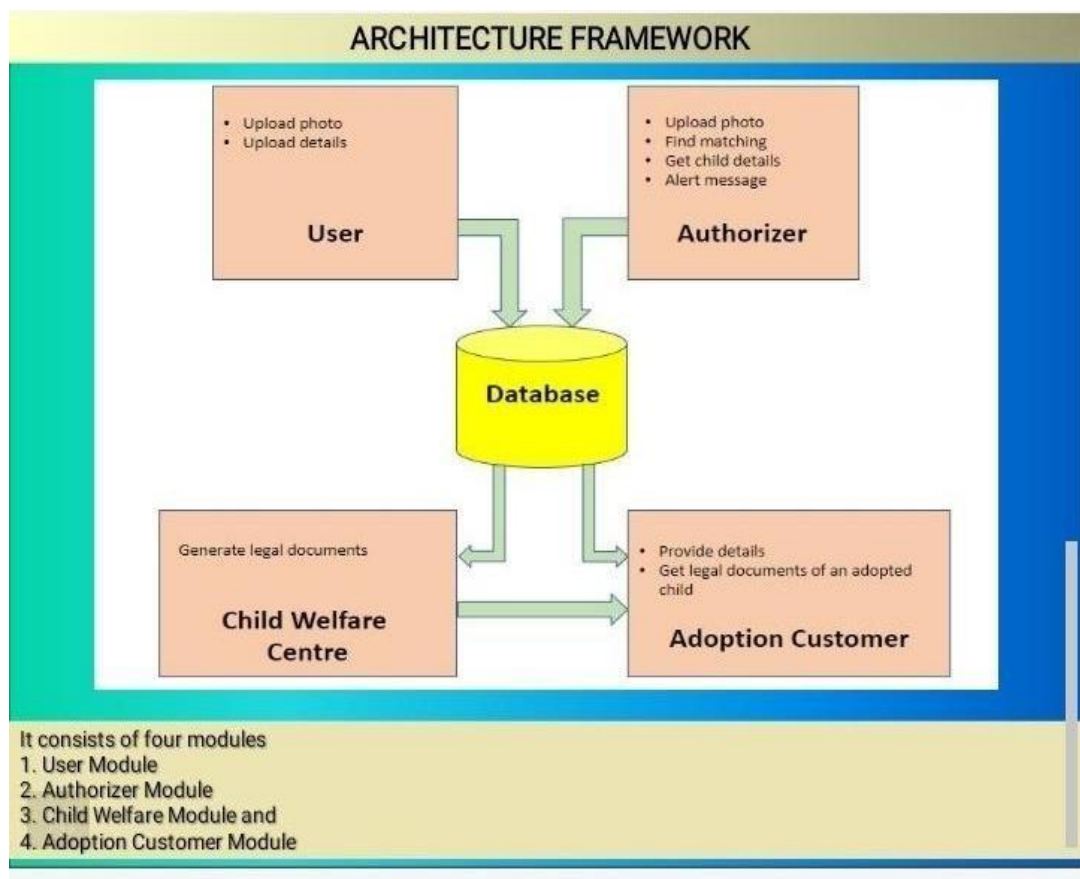


Figure 3.1: Architecture Diagram

3.2 Use Case Diagram:

The primary actor in this scenario would be the user (such as law enforcement or volunteers), and the system would include the DL and SVM algorithms, as well as any necessary data sources or databases. The use cases could include steps like inputting the child's information, processing the data through DL and SVM, and generating results or alerts. Creating a use case diagram requires more detailed information about the specific steps and actors involved.

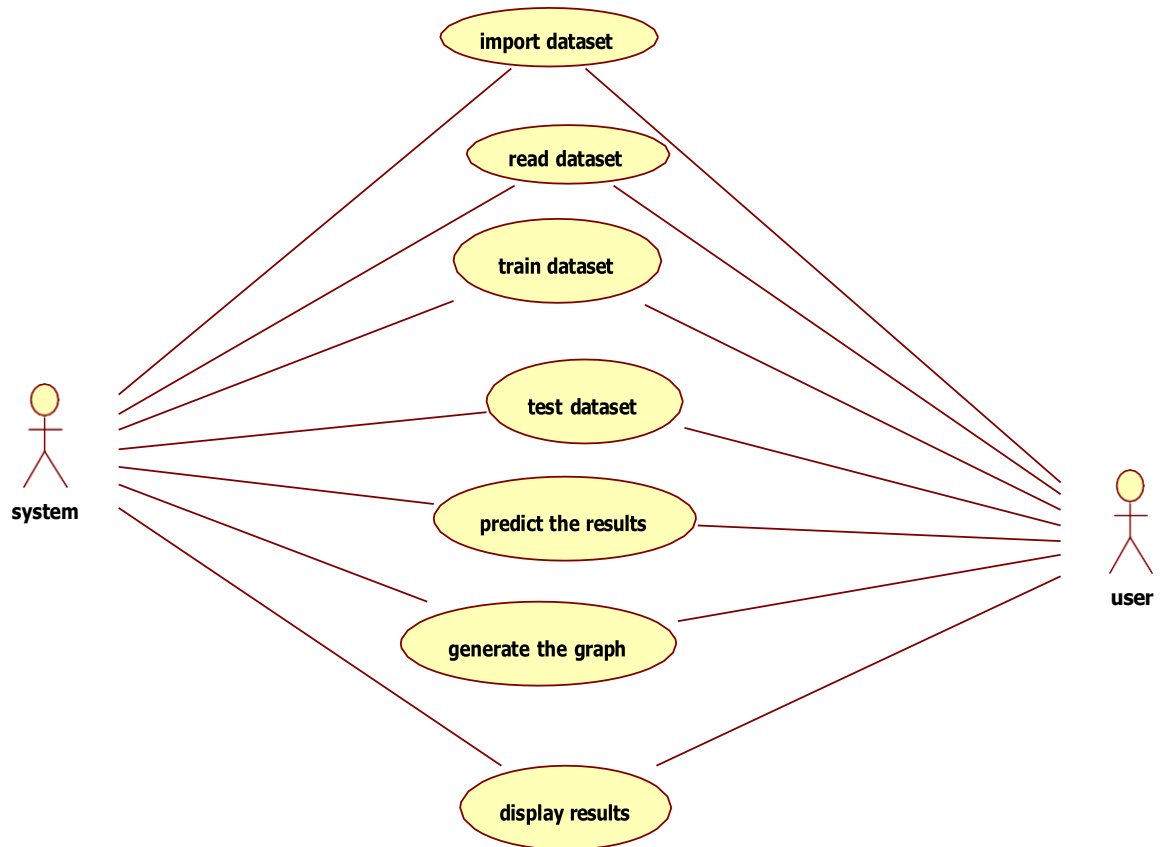


Figure 3.1: Use Case Diagram

3.3 Class Diagram

In this system, we can identify several key classes. First, we have the "MissingChild" class, which represents an individual missing child. This class would have attributes like name, age, gender, and a photo. Next, we have the "DLModel" class, which encapsulates the deep learning model used for image recognition. This class would have methods for training the model, loading pre-trained weights, and predicting the likelihood of a match between the missing child's photo and a given image. Then, we have the "SVMModel" class, which represents the multiclass SVM model used for classification.

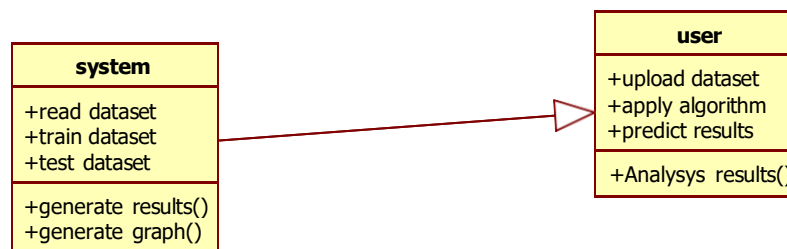


Figure 3.3: Class Diagram

class diagram is a visual representation of the structure and relationships of the classes in a system. In the context of missing child identification using Deep Learning and Multiclass SVM, the class diagram would show the different classes involved in the system and how they are related. It would include classes like "MissingChild", "DeepLearningModel", "MulticlassSVM", and possibly others depending on the specific implementation. The relationships between these classes would illustrate how they interact with each other to identify missing children. class diagram is a visual representation of the structure and relationships of the classes in a system. In the context of missing child identification using Deep Learning and Multiclass SVM, the class diagram would show the different classes involved in the system and how they are related. It would include classes like "MissingChild", "DeepLearningModel", "MulticlassSVM", and possibly others depending on the specific implementation. The relationships between these classes would illustrate how they interact with each other to identify missing children.

3.4 Sequence diagram

First, the user (such as a law enforcement officer or a volunteer) interacts with the system by providing the information and photo of a missing child. The user initiates the identification process by submitting the data.

The system receives the input and passes it to the DL model for image recognition. The DL model analyzes the photo and generates a feature vector representation.

Next, the system sends the feature vector to the multiclass SVM model. The SVM model processes the data and predicts the probability of the missing child belonging to each class. The system receives the classification results from the SVM model and presents them to the user through the user interface. The user interface displays the probabilities for each class, indicating the likelihood of a match.

Based on the results, the user can take appropriate actions, such as contacting relevant authorities or further investigating potential matches.

Please note that the sequence diagram would include more detailed interactions and messages between the components involved. This high-level description provides an overview of the process.

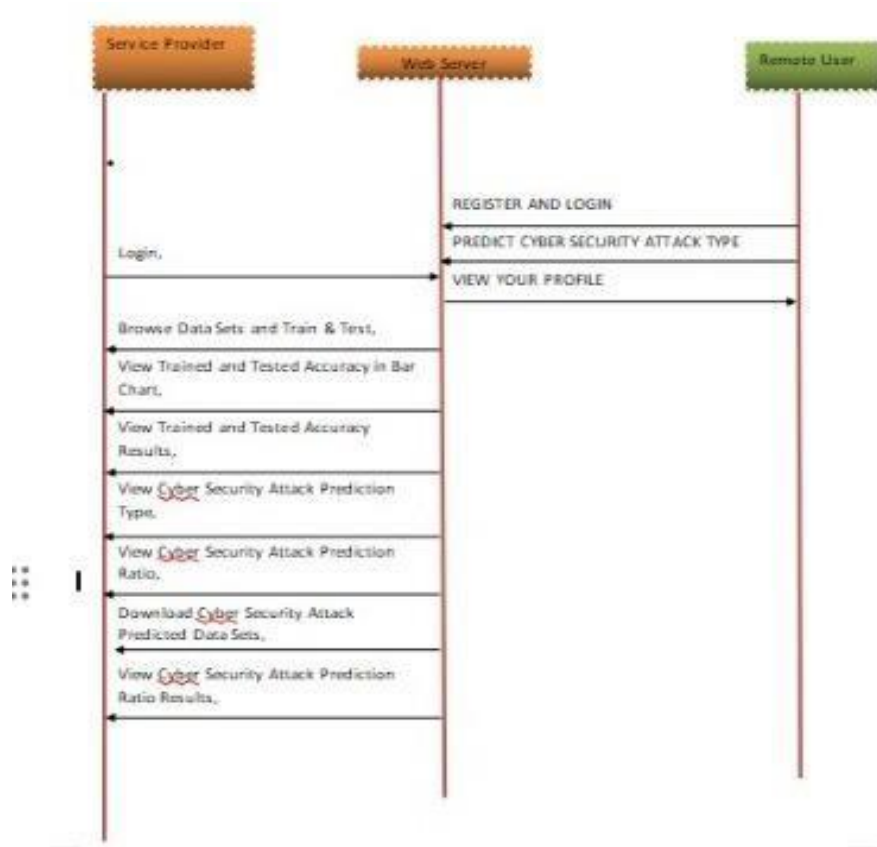


Figure 3.4: Sequence Diagram.

3.5 Activity Diagram:

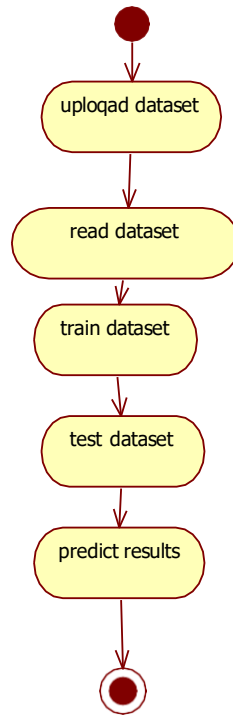


Figure 3.5: Activity Diagram

An activity diagram is a visual representation that shows the flow of activities or actions within a system. It helps to illustrate the steps involved in a particular process or workflow. For example, let's consider the activity diagram for a simple online shopping process. It might include activities like "Browse Products", "Add to Cart", "Proceed to Checkout", "Enter Shipping Information", "Select Payment Method", "Place Order", and "Receive Confirmation". Each activity would be represented by a rectangular box, and arrows would connect them to show the flow of control. Decision points or branches in the process would be represented by diamond-shaped boxes. For instance, the diagram would show that the first activity is to browse products. Once the user finds a desired item, they would add it to their cart. Then, they would proceed to the checkout process, where they enter their shipping information and select a payment method. After confirming the details, they would place the order and receive a confirmation. This is just a simplified description of an activity diagram. The actual diagram would provide a more detailed and visual representation of the steps and decision points involved in the online shopping process.

Chapter 4

IMPLEMENTATION

4.1 ALGORITHMS USED

4.1.1 SVM Algorithm

In classification tasks a discriminant machine learning technique aims at finding, based on an independent and identically distributed (iid) training dataset, a discriminant function that can correctly predict labels for newly acquired instances. Unlike generative machine learning approaches, which require computations of conditional probability distributions, a discriminant classification function takes a data point x and assigns it to one of the different classes that are a part of the classification task. Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space.

SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to genetic algorithms (GAs) or perceptrons, both of which are widely used for classification in machine learning. For perceptrons, solutions are highly dependent on the initialization and termination criteria. For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptrons is only to minimize error during training, which will translate into several hyperplanes' meeting this requirement.

4.1.2 Random Forest Algorithm

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

4.2 Sample Code

```
from django.shortcuts import render

from django.template import
RequestContext import pymysql
from django.http import
HttpResponse from django.conf
import settings
from django.core.files.storage import
FileSystemStorage import datetime
import
os
import
cv2
import numpy as np

from keras.utils.np_utils import
to_categorical from keras.layers import
MaxPooling2D
from keras.layers import Dense, Dropout, Activation,
Flatten from keras.layers import Convolution2D
from keras.models import Sequential
from keras.models import import
```

```

model_from_json import datetime

global
index
index =
0
global
missing_child_classifie
r global cascPath
    global faceCascade

if request.method == 'GET':
    return render(request, 'Login.html', {})

def WelfareLogin(request):
    if request.method == 'GET':
        return render(request, 'WelfareLogin.html', {})

def
    WelfareLoginAction(reque
st): if request.method ==
'POST':
    username = request.POST.get('t1', False)
    password = request.POST.get('t2', False)
if username == 'welfare' and password == 'welfare':
    context= {'data': 'welcome '+username}
        return render(request, 'WelfareScreen.html', context)
    else:
        context= {'data': 'login failed'}
    return render(request, 'WelfareLogin.html', context)

def ParentRegister(request):

```

```

if request.method == 'GET':
    return render(request, 'ParentRegister.html', {})

def Upload(request):
    if request.method == 'GET':
        return render(request, 'Upload.html', {})

def ParentLogin(request):
    if request.method == 'GET':
        return render(request, 'ParentLogin.html', {})
def
    ParentLoginAction(reque
st): if request.method ==
'POST':
    username = request.POST.get('t1',
False)          password =
request.POST.get('t2', False) index =
0

    con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root',
database = 'MissingChildDB',charset='utf8')
    with con:
        cur = con.cursor()
        cur.execute("select * FROM parentsignup")
        rows = cur.fetchall()
        for row in rows:
            if row[0] == username and password == row[1]:
                index = 1
                break
            k if index
            == 1:
                file = open('session.txt','w')
                file.write(username)

```

```

file.close()
context= {'data':'welcome '+username}
    return    render(request,  'ParentScreen.html',
context) else:
context= {'data':'login failed'}
return render(request, 'ParentLogin.html', context)

def ChildDetails(request):
if request.method == 'GET':
    return render(request, 'ChildDetails.html', { })
def
AdoptionRules(request):
if request.method ==
'GET':
return render(request, 'AdoptionRules.html', { })
def
checkImage(name):
    index = 0
    con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password =
'root', database = 'MissingChildDB',charset='utf8')
    with con:
        cur = con.cursor()
        cur.execute("select childname FROM adoption")
        rows = cur.fetchall()
        for row in rows:
            if row[0] == name: index = 1
            break
            return index
def getDetails(name): parent = "
age = " occupation = " contact = "
email = " address = "
con = pymysql.connect(host='127.0.0.1',port = 3308,
    user = 'root', password = 'root',
    database = 'MissingChildDB',charset='utf8')

```

```

with con:

cur = con.cursor()

cur.execute("select * FROM parentsignup")

rows = cur.fetchall()

for row in rows:

    if row[0] == name:
        parent = row[2]
        age = row[3]
        occupation = row[4]
        contact = row[5]
        email = row[6]
        address = row[7]

        break

    return parent, age, occupation, contact, email, address
def ViewAdoption(request):

    if request.method == 'GET':

        output = '<table border=1 align=center>'

        output+='<tr><th>Parent
        Name</th><th>Parent age</th><th>Occupation</th><th>Contact
        No</th><th>Email ID</th><th>Address</th><th>Child Name</td></tr>'

        color = '<font size="" color="black">'

        con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root',
        database = 'MissingChildDB',charset='utf8')

        with con:

            cur = con.cursor()

            cur.execute("select      *      FROM
            adoption") rows = cur.fetchall()

            for row in
            rows: user =

            row[0] child

            = row[1]

            parent, age, occupation, contact, email, address = getDetails(user)

            output+= '<tr><td>'+color+parent+'</td>'

```

```

output+='<td>'+color+age+'</td>'
output+='<td>'+color+occupation+'</td>'
output+='<td>'+color+contact+'</td>'
output+='<td>'+color+email+'</td>'

output+='<td>'+color+address+'</td>'
output+='<td>'+color+child+'</td></tr>'
output+='</table><br><br><br><br><br>'
context= {'data':output}
return render(request, 'ViewAdoption.html', context)

```

```
def AdoptAction(request):
```

```
if request.method == 'GET':
```

```
name = request.GET.get('name',
```

```
False) user = "
```

```
with open("session.txt", "r") as
```

```
file: for line in file:
```

```
user = line.strip('\n')
```

```
file.close
```

```
now = datetime.datetime.now()
```

```
current_time = now.strftime("%Y-%m-%d %H:%M:%S")
```

```
db_connection = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root',
password = 'root', database = 'MissingChildDB',charset='utf8')
```

```
db_cursor = db_connection.cursor()
```

```
query= "INSERT INTO adoption(username,childname,adoption_date)
VALUES('"+user+"','"+name+"','"+str(current_time)+"')"
```

```
db_cursor.execute(query)
```

```
db_connection.commit()
```

```
print(db_cursor.rowcount, "Record
```

```
Inserted") output = "
```

```
parent, age, occupation, contact, email, address =
```

```
getDetails(user) output = '<table border=1 align=center>'
```

```

        output+= '<tr><th>Parent
        Name</th><th>Parentage</th><th>
        Occupation</th><th>
Contact No</th><th>Email ID</th><th>Address</th><th>
Child Name</td></tr>'

        color = '<font size="" color="black">'
        output+= '<tr><td>'+color+parent+'</td>'
        output+= '<td>'+color+age+'</td>'
        output+= '<td>'+color+occupation+'</td>'
        output+= '<td>'+color+contact+'</td>'
        output+= '<td>'+color+email+'</td>'
        output+= '<td>'+color+address+'</td>'
        output+= '<td>'+color+name+'</td></tr></table><br><br><br><br><br>'

        context= {'data':output}

        return render(request, 'Certificate.html', context)

    def
ChildDetailsAction(request)
: if request.method ==
'POST':
    age = request.POST.get('t1',
False) colour = request.POST.get('t2',
False) user = "
    index = 0
    with open("session.txt", "r") as
file: for line in file:
        user = line.strip('\n')
file.close

    con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root',
database = 'MissingChildDB',charset='utf8')

    with con:
        cur = con.cursor()

        cur.execute("select child_age,child_color FROM parentsignup where
username='"+user+"'")

```



```

        rows =
cur.fetchall() for row in
rows:
    if row[0] == age and row[1] == colour: index
    = 1
    if index == 1:
        imgs = ['002A03.JPG','049A10.JPG','053A03.JPG','053A04.JPG','053A06.JPG']
        output = '<table border=1 align=center>' output+='<tr><th>Username</th><th>Child
        Image</th><th>Adopt Child</th></tr>' color = '<font size="" color="black">'
        for i in range(len(imgs)):
            if checkImage(imgs[i]) == 0:
                output+='<tr><td>'+color+user+'</td><td><img          src=/static/testImages/'+imgs[i]+'
                width=200 height=200></img></td>'
                output+='<td><a          href=\'\AdoptAction?name='+imgs[i]+'\'><font          size=3
                color=black>Click Here</font></a></td></tr>'
                output+='</table><br/><br/><br/><br/>'
                context= {'data':output}
                return render(request, 'ViewImages.html', context)
            else:
                context= {'data':"child details mismatch"}
                return render(request, 'ParentScreen.html', context)

def OfficialLogin(request):
    if request.method == 'POST':
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        if username == 'admin' and password == 'admin':
            context= {'data':'welcome '+username}
            return render(request, 'OfficialScreen.html', context)
        else:
            context= {'data':'login failed'}
            return render(request, 'Login.html', context)
def ViewUpload(request):

```

```

if request.method == 'GET':

    strdata = '<table border=1 align=center width=100%><tr><th>Upload Person  
Name</th><th>Child Name</th><th>Contact No</th><th>Found  
Location</th><th>Child Image <th>Uploaded Date</th><th>Status</th></tr><tr>'

    con = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password = 'root',  
database = 'MissingChildDB',charset='utf8')

    with con:

        cur = con.cursor()

        cur.execute("select * FROM missing")

        rows = cur.fetchall()

        for row in rows:

            strdata+='<td>'+row[0]+'</td><td>'+str(row[1])+'</td><td>'+row[2]+'</td><td>'+row[3]+'</td><td><img src=/static/photo/'+row[4]+' width=200 height=200></img></td><td>'

            strdata+=str(row[5])+'</td><td>'+str(row[6])+'</td></tr>'

        context= {'data':strdata}

    return render(request, 'ViewUpload.html', context)

def

UploadAction(reques

t): global index

global

missing_child_classifier

global cascPath

global faceCascade

if request.method == 'POST' and request.FILES['t5']:

    output = "

    person_name = request.POST.get('t1', False)

    child_name = request.POST.get('t2', False)

    contact_no = request.POST.get('t3', False)

    location = request.POST.get('t4', False)

```

```

myfile = request.FILES['t5']

fs = FileSystemStorage()

filename =
fs.save('C:/Python/MissingChilds/MissingChildApp/static/photo/'+child_name+'.png',
myfile) #if index == 0:

cascPath = "haarcascade_frontalface_default.xml"

faceCascade = cv2.CascadeClassifier(cascPath)

#index = 1

option = 0;

frame = cv2.imread(filename)

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(gray,1.3,5)

print("Found {0} faces!".format(len(faces)))

img = "

status = 'Child not found in missing database'

if len(faces) > 0:

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

        img = frame[y:y + h, x:x + w]

        option = 1

if option == 1:

    with open('model/model.json', "r") as json_file:

        loaded_model_json = json_file.read()

        missing_child_classifier = model_from_json(loaded_model_json)

        missing_child_classifier.load_weights("model/model_weights.h5")

        missing_child_classifier._make_predict_function()

        img = cv2.resize(img, (64,64))

        im2arr = np.array(img)

        im2arr = im2arr.reshape(1,64,64,3)

        img = np.asarray(im2arr)

        img = img.astype('float32')

        img = img/255

```

```

preds      =      missing_child_classifier.predict(img)
if(np.amax(preds) > 0.60):
    status = 'Child found in missing database' now
= datetime.datetime.now()
    current_time  =  now.strftime("%Y-%m-%d  %H:%M:%S")
filename = os.path.basename(filename)

    db_connection = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root', password
= 'root', database = 'MissingChildDB',charset='utf8')

    db_cursor = db_connection.cursor()

    query          =          "INSERT          INTO
missing(person_name,child_name,contact_no,location,image,upload_date,status)
VALUES('"+person_name+"','"+child_name+"','"+contact_no+"','"+location+"','"+filena
me+" ','"+str(current_time)+"','"+status+"')"
```

```

    db_cursor.execute(query)
db_connection.commit()
print(db_cursor.rowcount, "Record Inserted")

    context=  {'data': 'Thank  you  for  uploading.
'+status}  return  render(request, 'Upload.html',
context)

def ParentRegisterAction(request):
    if      request.method      ==      'POST'      and
request.FILES['t9']: output = "
    username  =  request.POST.get('t1',
False)      password      =
request.POST.get('t2', False) parent =
request.POST.get('t3', False) age =
request.POST.get('t4', False)
    occupation  =  request.POST.get('t5',
False) contact = request.POST.get('t6',
False) email  =  request.POST.get('t7',
False) address = request.POST.get('t8',
False) child_age =

```

```

request.POST.get('t10', False)
child_color = request.POST.get('t11',
False)

myfile = request.FILES['t9']
filenames =
request.FILES['t9'].name fs =
FileSystemStorage()

filename =
fs.save('C:/Python/MissingChlds/MissingChildApp/static/documents/'+filenames,
myfile)

db_connection = pymysql.connect(host='127.0.0.1',port = 3308,user = 'root',
password = 'root', database = 'MissingChildDB',charset='utf8')

db_cursor = db_connection.cursor()

query="INSERT INTO
parents signup(username,password,name,age,occupation,contactno,email,address,filen
ame,child_age,child_color)
VALUES('"+username+"','"+password+"','"+parent+"','"+age+"','"+occupation+"','"+con
tact+
"','"+email+"','"+address+"','"+filenames+"','"+child_age+"','"+child_color+"')"

db_cursor.execute(query)

db_connection.commit()

print(db_cursor.rowcount, "Record Inserted")

context= {'data':'Signup process completed'}

return render(request, 'ParentRegister.html', context)

```

Chapter 5

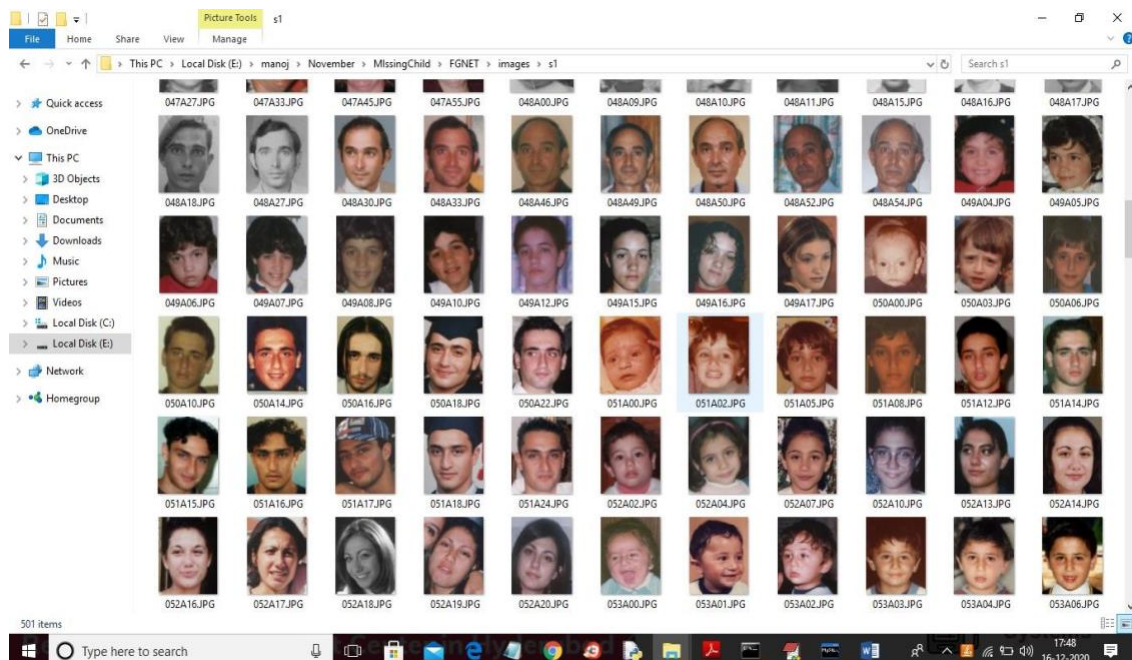
SCREENSHOTS

5.1 Introduction

Missing Child Identification System using Deep Learning and Multiclass SVM is describing concept to identify missing children by using Deep Learning and Multiclass SVM classifier and to implement this project author has used below modules

- 1) Using public dataset of missing children's called FGNET is used to train deep learning CNN prediction model. After training model whenever public upload any suspected child image then this model will check in trained model to detect whether this child is in missing database or not. This detected result will store in database and whenever want official persons will login and see that detection result.
- 2) SVM Multiclass classifier use to extract face features from images based on age and other facial features and then this detected face will input to CNN model to predict whether this face child exists in image database or not.

First we used below dataset to train deep learning CNN model



Screenshot 5.1 : Start DJANGO and run browser

To run project, follow below steps

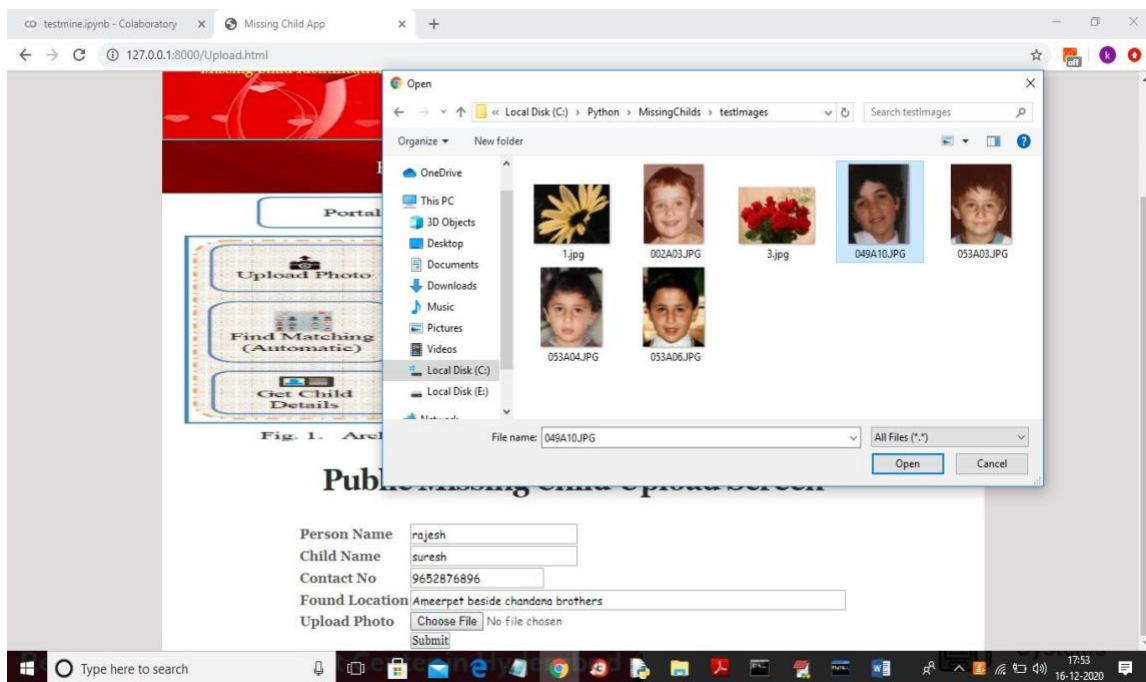
1. First create database in MYSQL by copying content from 'DB.txt' file and paste in MYQL
2. Install python, DJANGO and MYSQL software
3. Create 'Python' folder in C directory and put 'MissingChilds' folder in it
4. start DJANGO server and run in browser to get first

page SCREEN SHOTS



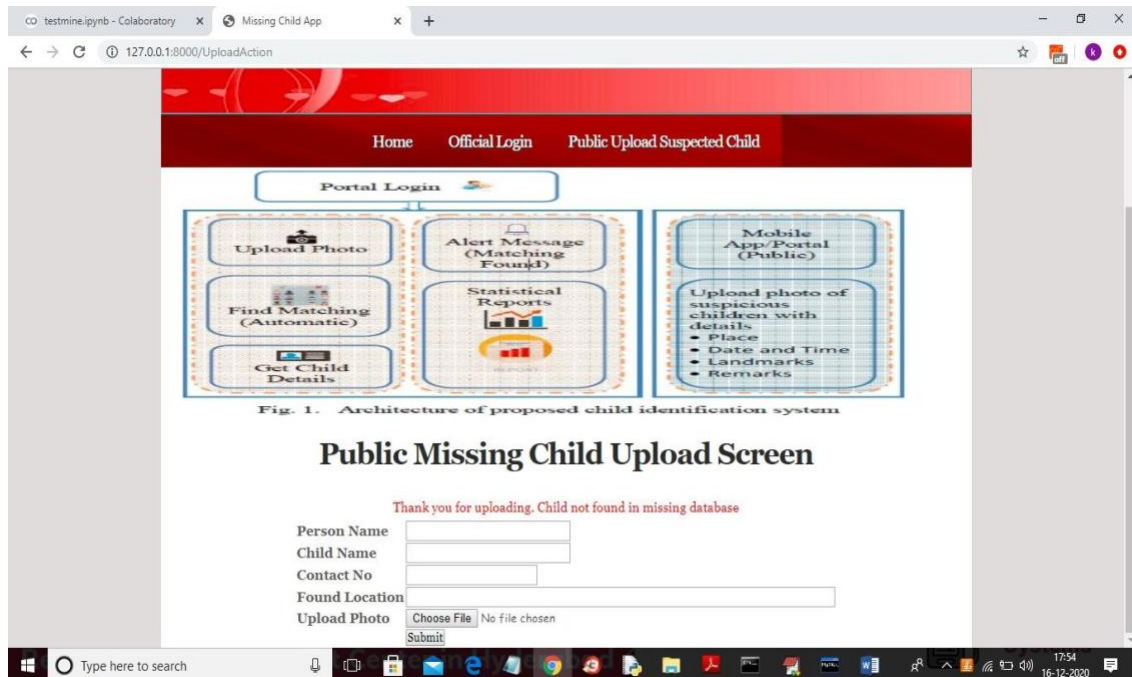
Screenshot 5.2: Public Upload Suspected Child

In above screen public can click on 'Public Upload Suspected Child' link to get below page and to add missing child details



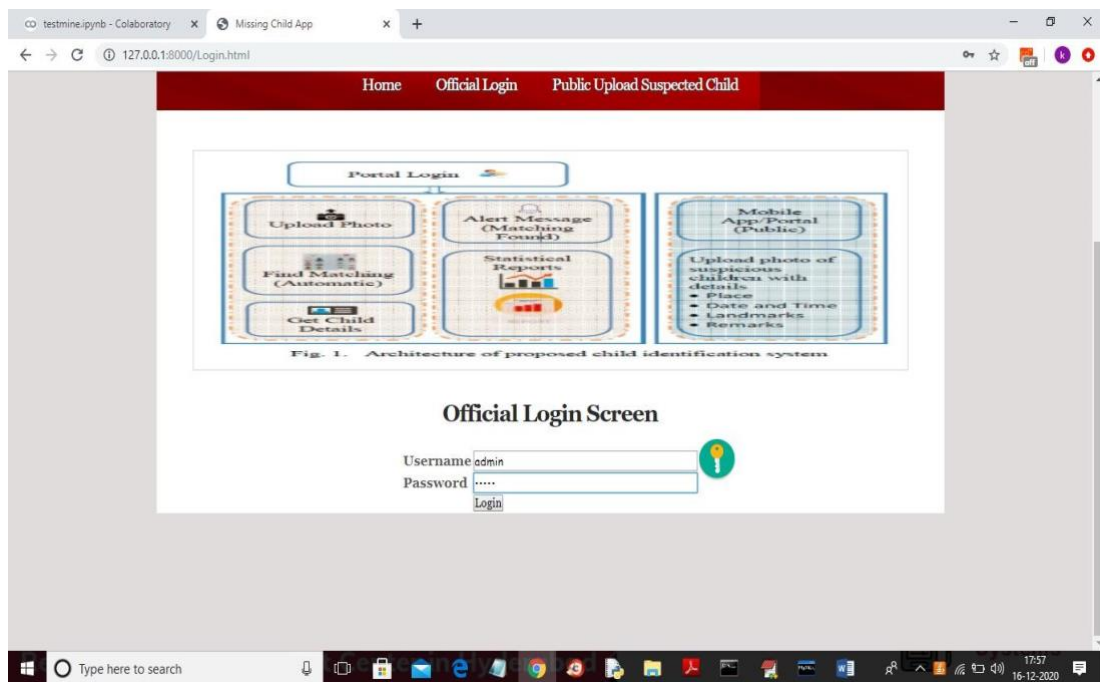
Screenshot 5.3 : click on Submit button

In above screen public will enter suspected child details and then upload photo and then click on ‘Submit’ button and to get below result



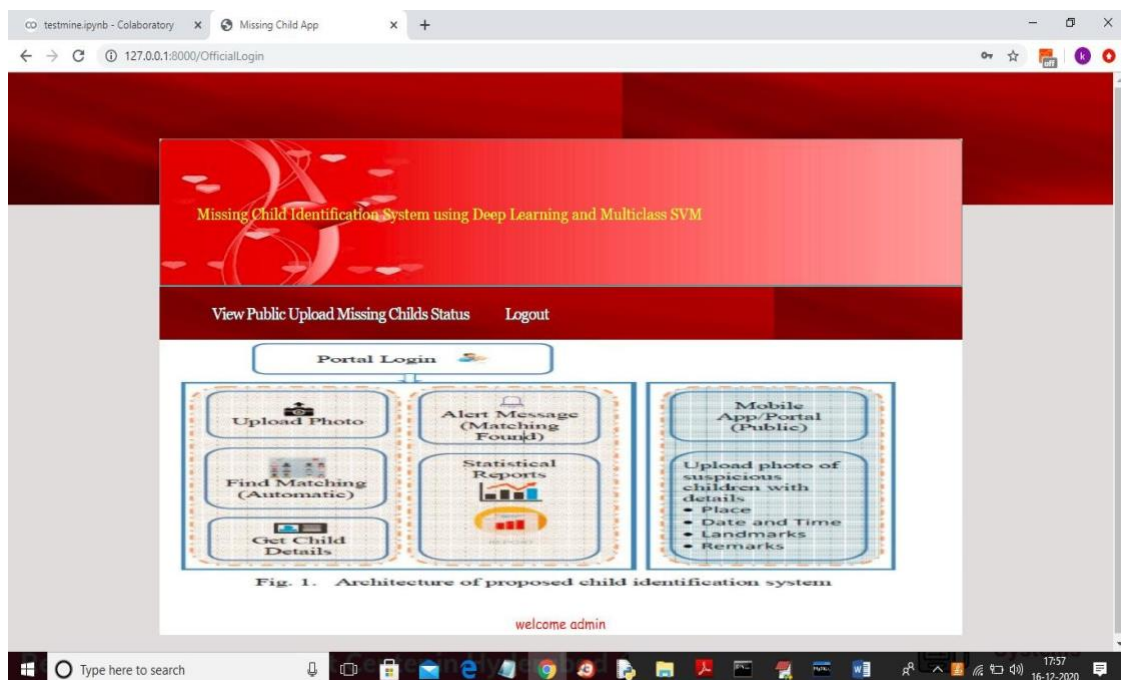
Screenshot 5.4: can see child not found in missing DB

In above screen we can see child not found in missing DB and we can try with other image



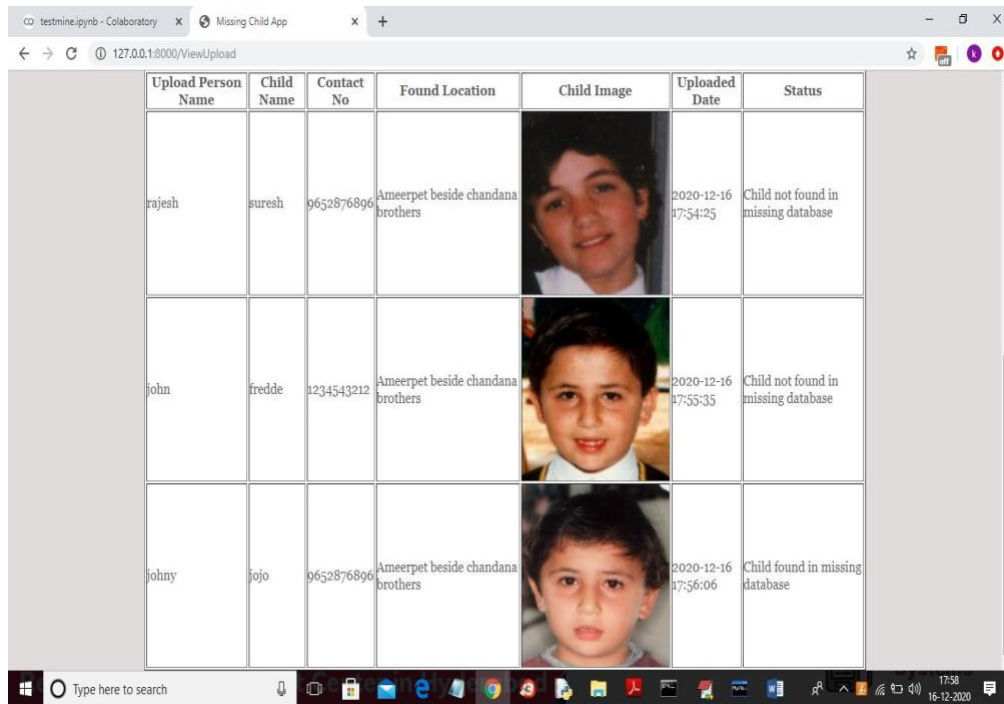
Screenshot 5.5: Enter username and password as 'admin' and 'admin'




In above screen admin can login by entering username and password as 'admin' and 'admin' and after clicking on 'Login' button will get below screen



Screenshot 5.6: View Public Upload Missing Childs Status

In above screen official can click on 'View Public Upload Missing Childs Status' link to view all uploads and its result done by public



Upload Person Name	Child Name	Contact No	Found Location	Child Image	Uploaded Date	Status
rajesh	suresh	9652876896	Ameerpet beside chandana brothers		2020-12-16 17:54:25	Child not found in missing database
john	freddie	9234543212	Ameerpet beside chandana brothers		2020-12-16 17:55:35	Child not found in missing database
johny	jojo	9652876896	Ameerpet beside chandana brothers		2020-12-16 17:56:06	Child found in missing database

Screenshot 5.7: Officials can see all details and then take action to find that child

In above screen officials can see all details and then take action to find that child.

Chapter 6

TESTING

6.1 Introduction To Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner.

6.2 Types of Testing

6.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive.

6.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases.

6.3 Test Cases of Missing Child

Table 6.1: Test Cases of missing child

S.NO	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	User Register	If User registration successfully.	Pass	If already user email exist then it fails.
2.	User Login	If Username and password is correct then it will getting valid page.	Pass	Un Register Users will not logged in.
3.	User View User	Show our dataset	Pass	If Data set Not Available fail.
4.	View Fast History Results	The Four Alarm Score Should be Displayed.	Pass	The Four Alarm Score Not Displaying fail
5.	User Prediction	Display Review with true results	Pass	Results not True Fail
6.	Show Detection process	Display Detection process	Pass	Results Not True Fail
7.	Show Eye Blink Process	Display Eye Blink Process	Pass	If Results not Displayed Fail.
8.	Admin login	Admin can login with his login credential. If success he get his home page	Pass	Invalid login details will not allowed here
9.	Admin can activate the register users	Admin can activate the register user id	Pass	If user id not found then it won't login.
10.	Results	For our Four models the accuracy and F1 Score	Pass	If Accuracy And F1 Score Not Displayed fail

Chapter 7

CONCLUSION & FUTURE SCOPE

7.1 Conclusion

By combining multiclass SVM and Deep Learning techniques, we can develop a powerful system that can accurately classify and identify missing children based on their images. Deep learning allows for the extraction of meaningful features from images, while Multiclass SVM enables effective classification into different categories. However, it's important to consider the advantages and disadvantages of each technique and optimize their performance based on your specific requirements. Overall, this approach has the potential to significantly aid in the identification and recovery of missing children.

7.2 Future scope

A missing child identification system is proposed, which combines the powerful CNN based deep learning. Approach for feature extraction and support vector machine classifier for classification of different child categories. This system is evaluated with the deep learning model which is trained with feature representations of children faces. By discarding of the VGG-Face model and extracting CNN image features to train a multi class SVM, it was possible to achieve superior performance. Performance of the proposed system is tested using the photographs of children with different lighting conditions, noises and also images at different ages of children. The classification achieved a higher accuracy of 99.41% which shows that the proposed methodology of face recognition could be used for reliable.

REFERENCES

- [1]. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, 521(7553):436–444, 2015.
- [2]. O. Deniz, G. Bueno, J. Salido, and F. D. la Torre, "Face recognition using histograms of oriented gradients", *Pattern Recognition Letters*, 32(12):1598–1603, 2011.
- [3]. C. Geng and X. Jiang, "Face recognition using sift features", *IEEE International Conference on Image Processing (ICIP)*, 2009.
- [4]. Rohit Sate, Vishnu prasad Poojary, John Abraham, Shilpa Wakode , "Missing child identification using face recognition system", *International Journal of Advanced Engineering and Innovative Technology (IJAEIT)*, Volume 3 Issue 1 July - August 2016