

CHILD AGE EXTRACTION WITH FACE ON ABUSIVE WEBSITE BLOCKAGE AND FILTERING -PYTHON

Submitted in partial fulfilment of the required for the award of

Masters of Computer Science

by

THIRSHA ANGELINE R

(REGISTER NO 42734023)



DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A++” by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC I Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of THIRSHA ANGELINE R (42734023) who carried out the project entitled "CHILD AGE EXTRACTION WITH FACE ON ABUSIVE WEBSITE BLOCKAGE AND FILTERING" under our supervision from JANUARY 2024 to APRIL 2024.

Shamili V
INTERNAL GUIDE

Ms. SHAMILI.V, M.SC., Assistant professor

Rekha Chakravarthy
DEAN AND HEAD OF THE DEPARTMENT

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva voce examination held on

20/04/2024

SG-V
Internal Examiner

[Signature]
External Examiner

DECLARATION

I am, THIRSHA ANGELINE R (42734023) here by declare that the Project Report entitled "CHILD AGE EXTRACTION WITH FACE ON ABUSIVE WEBSITE BLOKAGE AND FILTERING" done by us under the guidance of MS.SHAMILI.V, M.SC., Assistant Professor, Department of Computer Science at SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfillment of the requirements for the award of Master of Science degree in Computer Scienc.

DATE: 20/04/2024

PLACE:CHENNAI



THIRSHA ANGELINE R

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. REKHA CHAKRAVARTHI M.E., Ph.D. Dean of Arts and Science** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sence of gratitude to my project guide **MS.SHAMILI.V M.SC.**, Assistant Professor, Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project.

THIRSHA ANGELINE R

ABSTRACT

Abusive websites is ruling over the world where people from child to old get worst because of using unwanted websites. Mostly children and young people use search engines to get some content but some malicious websites may distract their life. Thus this content will be a problematic in their life. Thus a solution needs to be enhanced for this internet technology to overcome this problem. In our proposed system, facial age classification based website blocking system is enhanced. Here Artificial Neural Network (ANN) with Multi Layer Perceptron (MLP) is used to classify the facial features with the child face. The MLP classification with age based system will train the certain websites with blockage assistance where the child age lesser than 18 will make an automatic blockage of website. The face feature extraction is done using HAAR cascade feature extraction in which the complete face feature are analyzed and detected. The detected face feature are then classified with the ANN used with MLP three layer working process. Thus the classification of this algorithm makes an easy identification of age at an accuracy of 86%. The system will give access to website only the person is within the trained age. Visual surveillance system is used to identify the facial extraction with Haar Cascade Feature extraction on using face detection. After the face detection age will be classified and access providence over website will be done. Thus the proposed solution helps children to be away from this malicious or abusive websites. Thus the implementation system reduces the child to see unwanted content in the Search Engines. Thus an smart system is developed with the proposed project.

CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	1
1.1	OVERVIEW OF PROJECT	2
1.2	EXISTING SYSTEM	4
2	LITERATURE SURVEY	4
3	PROPOSED SYSTEM	5
3.1	ENVIRONMENT REQUIREMENT	5
3.2	SYSTEM DESIGN	6
3.2.1	ARCHITECTURE DIAGRAM	6
3.2.2.	SEQUENCE DIAGRAM	7
3.2.3	CLASS DIAGRAM	8
3.2.4	DATA FLOW DIAGRAM	9
3.2.5	USECASE DIAGRAM	12
3.2.6	COLLABRATION DIAGRAM	13
4	MODULES	14
4.1	MODULES DESCRIPTION	
5	SOFTWARE DESCRIPTION	
5	SCREEN SHOT	
7	CONCLUSION	
8	SOURCE CODE	
9	REFERENCE	

LIST OF FIGURES

S.NO:	TITLE	PAGE NO
3.2.1	ARCHITECTURE DIAGRAM	6
3.2.2.	SEQUENCE DIAGRAM	7
3.2.3	CLASS DIAGRAM	8
3.2.4	DATA FLOW DIAGRAM	9
3.2.5	USECASE DIAGRAM	12
3.2.6	COLLABRATION DIAGRAM	13

1 INTRODUCTION

Developing methods for smart elimination or filtering of abusive websites based on face-based age classification introduces an innovative approach to safeguarding children's online experiences. By leveraging facial recognition technology, this method aims to accurately determine the age of users accessing websites, allowing for targeted filtering of content deemed inappropriate or harmful for minors. The introduction of such a system signifies a proactive step towards enhancing online safety measures for children, addressing the growing concern of exposure to abusive content on the internet. Traditional methods of content filtering often rely on keyword-based algorithms or manual moderation, which may not effectively adapt to the dynamic nature of online threats. In contrast, face-based age classification offers a more nuanced and personalized approach, enabling websites to tailor their content based on the perceived age of the user. This approach not only helps mitigate the risk of children accessing harmful material but also promotes a safer and more age-appropriate online environment. However, the implementation of face-based age classification also raises important ethical and privacy considerations. It necessitates robust measures to ensure the responsible use of facial recognition technology, including informed consent, data protection, and transparency in algorithmic decision-making. Overall, the introduction of smart elimination through face-based age classification represents a promising advancement in the ongoing efforts to promote child safety online. By combining technological innovation with ethical principles, this approach has the potential to significantly mitigate the risks associated with abusive website usage among children, fostering a healthier and more secure digital ecosystem.

1.2 EXISITING SYSTEM

In the existing system, almost everyone around the globe uses the internet. These internet users have different attributes like age, browsing behavior, gender, location, etc. that plays an important role for different business enterprises to target their users and also helps in providing the better user experience. Earlier researches demonstrate that the user browsing behavior is related to their basic attribute. The Internet is the basis for business for most of the enterprises. As a large number of user's use of the internet, their browsing pattern is required by these organizations and enterprises. They help to enhance the experience of various users. Along with this it also provides new and improved services in order to support the pre-existing customer as well as attracting new ones simultaneously. Also, behavior addressing provides the facility to the organizations to target the right users. Also, there are many websites which are not appropriate for the children below 18 years of age. Thus it helps in examining the complications in guessing the age of the user of the websites dependent on their browsing activities.

DISADVANTAGES

- The system will be completely a problematic one
- child also view unknown content from the browser

2 LITERATURE SURVEY

Title : A Segment-Based Drift Adaptation Method for Data Streams

Author : Yiliao Song, Jie Lu

Year : 2017

Data streams consist of infinite and fast evolving data instances which arrive in a sequential way[2],[3]. For a data stream, the newly arrived data Instances may exhibit a different pattern from the previous[4]. This phenomenon is called Concept drift ,where the terminology “concept” refers to the hidden patterns or the underlying distribution of data [5].

Title : Analyzing concept drift detection through sampling

Author : G. I. Webb, L. K. Lee, B. Goethals, and F. Petitjean

Year : 2018

Proposed Concept drift and shift are major issues that greatly affect the accuracy and reliability of many real-world applications of machine learning[6]. We propose a new data mining task, concept drift mapping—the description and analysis of instances of concept drift or shift. We argue that concept drift mapping is an essential prerequisite for tackling concept drift and shift. We propose tools for this purpose, arguing for the importance of quantitative descriptions of drift and shift in marginal distributions.

Title : Concept drift adaptation by exploiting historical knowledge

Author : Y. Sun, K. Tang, Z. Zhu, and X.

Year : 2018

Concept drift has often been tackled by ensemble methods, where models built in the past can be re-trained to attain new models for the current data. Two design questions need to be addressed in developing ensemble methods for incremental learning with concept drift, i.e., which historical (i.e., previously trained) models should be preserved and how to utilize them. A novel ensemble learning method, namely Diversity and Transfer based Ensemble Learning is proposed in this paper.

Title : A concept drift tolerant case base editing technique

Author : Guangquan Zhang a, Ramon Lopez de Mantara

Year : 2016

In Case-base Maintenance, case-base editing strategies to revise the case-base have proven to be effective instance selection approaches for handling concept drift. we present a two-stage case-base editing technique. In Stage 1, we propose a Noise-Enhanced Fast Context Switch algorithm, which targets the removal of noise in a dynamic environment, and in Stage 2, we develop an innovative Stepwise Redundancy Removal algorithm, which reduces the size of the case-base by eliminating redundancies while preserving the case-base coverage.

Title : Semi-supervised adaptive novel class detection and classification over data stream

Author : A. Haque, L. Khan, and M. Baron

Year : 2016

We propose an efficient semi-supervised framework in this paper which uses change detection on classifier confidence to detect concept drifts, and to determine chunk boundaries dynamically[11]. It also addresses concept evolution problem by detecting outliers having strong cohesion among themselves. Experiment results on bench mark and synthetic data sets show effectiveness of the proposed approach.

3 PROPOSED SYSTEM

The main issue is to detect the age of the users according to their search patterns. Earlier the problem of detecting the age is done using a regression model. The main focus was on designing the different linguistics writing and way of speaking is associated with the demographic feature. The solution is based on dividing the users into two different categories depending on their browsing behavior as well as profile. In this paper, the website users' age group is detected based on their browsing behavior, where the website viewed serves as the basis to predict the age of users. In the proposed solution, the information is taken from two different user groups, adults, and youngsters. Then it is analyzed to predict the age group of the user, they are divided into sessions then FP-growth algorithm is applied to it. It uses a pre-computed comparison of internet activities of the internet users. Based on the above method, the given solution results in giving a good accuracy.

3.1 ADVANTAGES

- The system identifies the age and block the adult websites
- Exact detection of age prediction is done

3.1 ENVIRONMENT REQUIREMENT

HARDWARE SPECIFICATION

Processor: Intel (R) Pentium (R)

Speed: 1.6 GHz and Above

RAM: 4 GB and Above

Hard Disk: 120 GB

SOFTWARE SPECIFICATION

Operating system: Windows 7 / 8 / 8.1 / 10

Coding Language: PYTHON

IDE: PyCharm

Database: MySQL

3.2 SYSTEM DESIGN

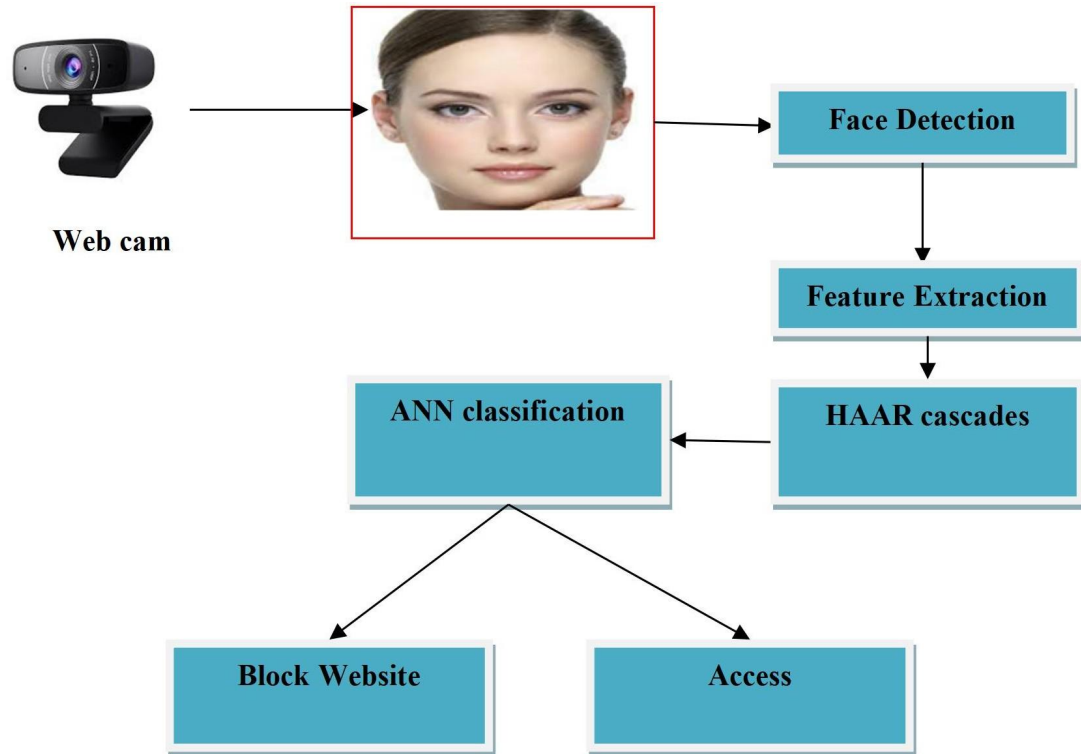


FIG 3.2.1 ARCHITECTURE DIAGRAM

SEQUENCE DIAGRAM

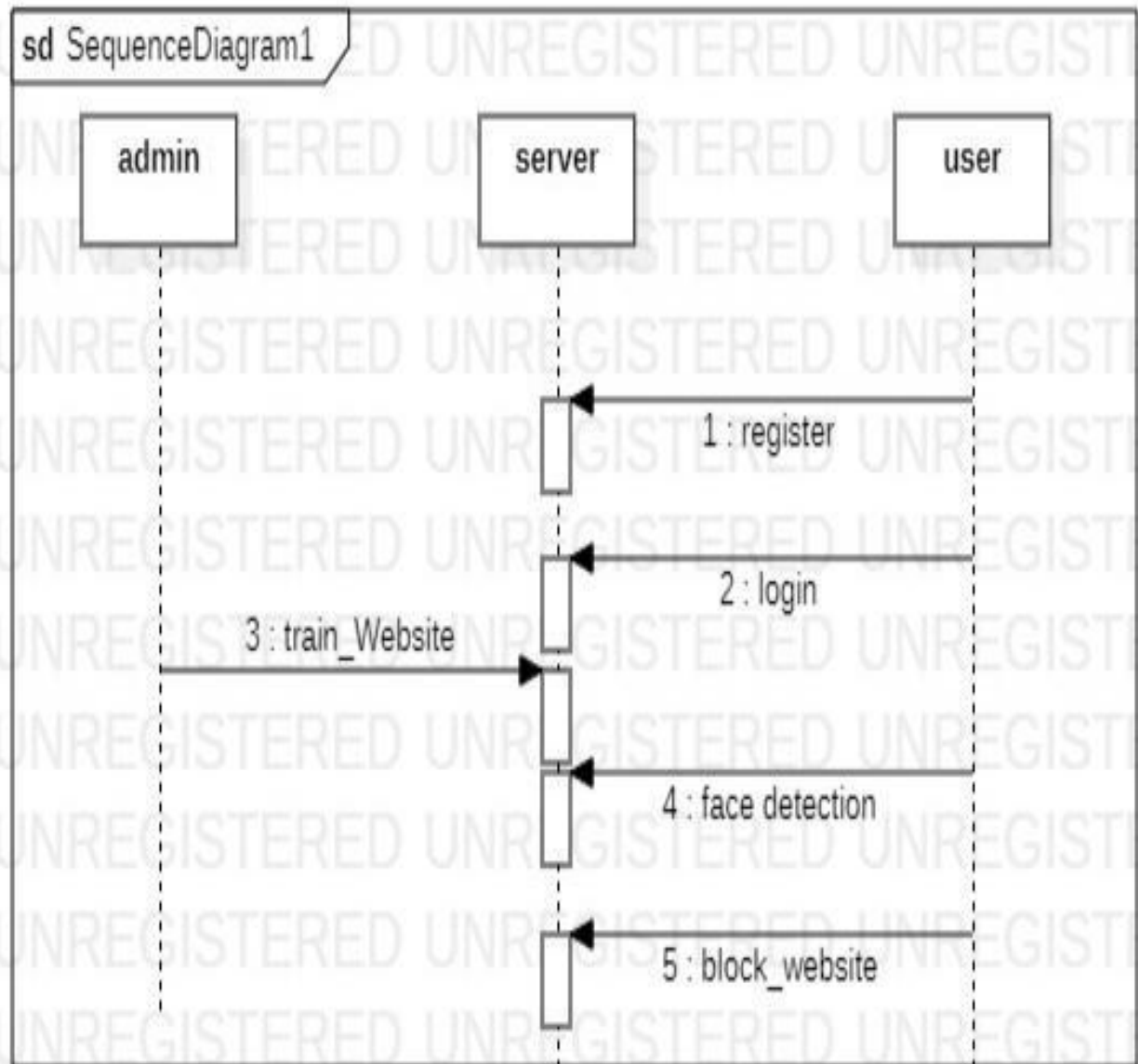


FIG 3.2.2 .SEQUENCE DIAGRAM

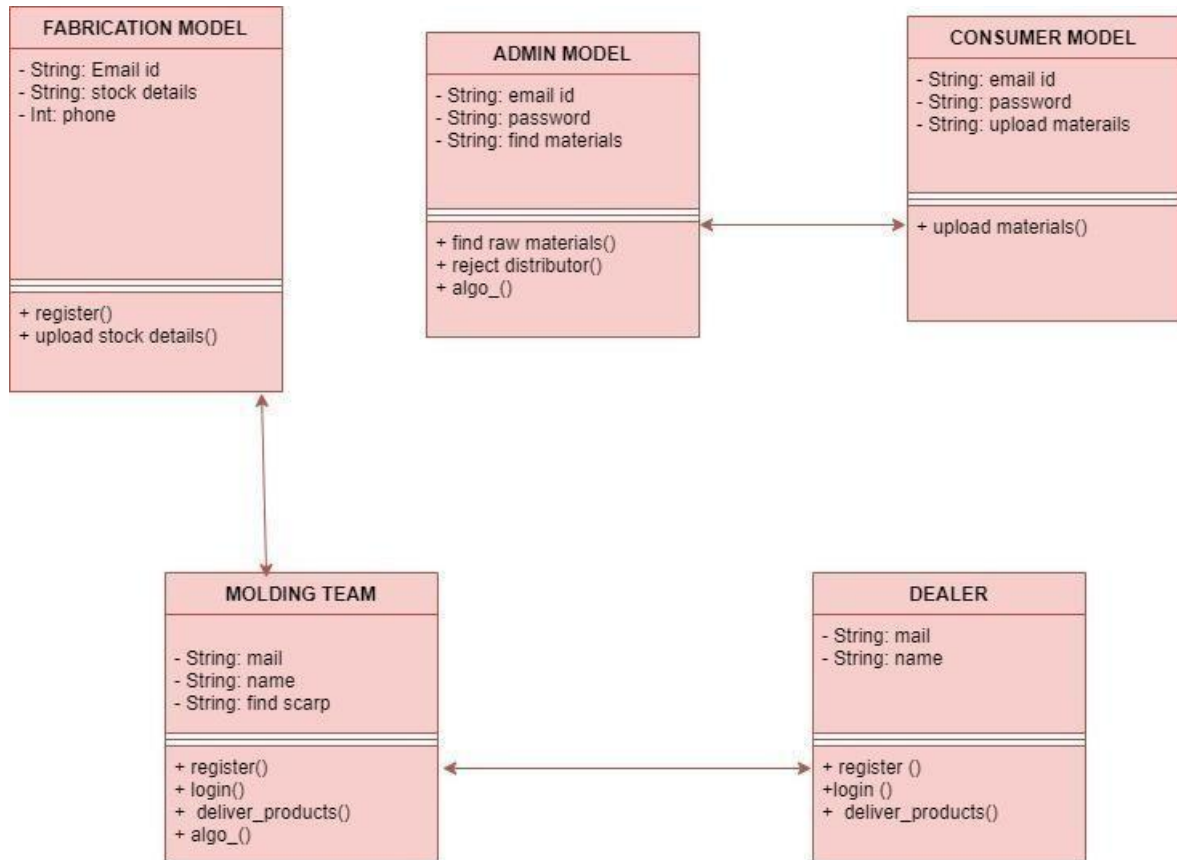


FIG 3.2.3 CLASS DIAGRAM

DATA FLOW DIAGRAM

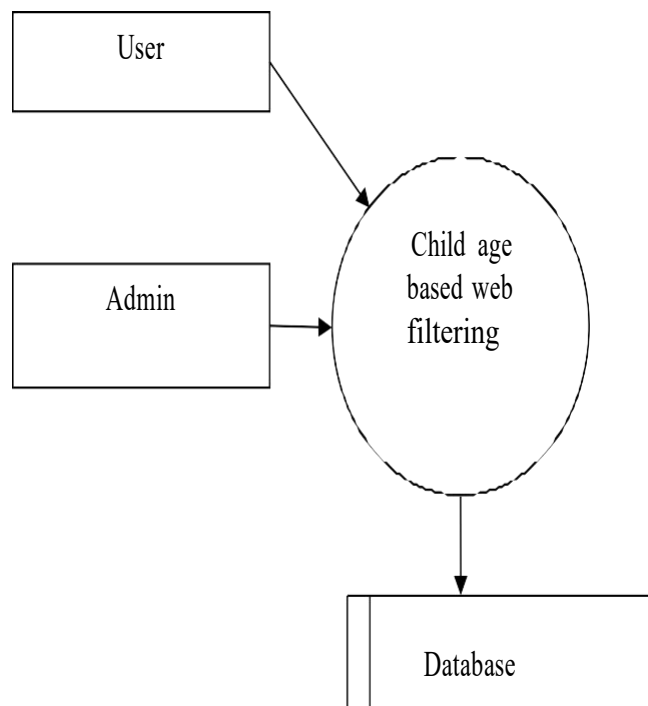
A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

Symbol	Description
	An entity . A source of data or a destination for data.
	A process or task that is performed by the system.
	A data store , a place where data is held between processes.

FIG 3.2.4 DATA FLOW DIAGRAM

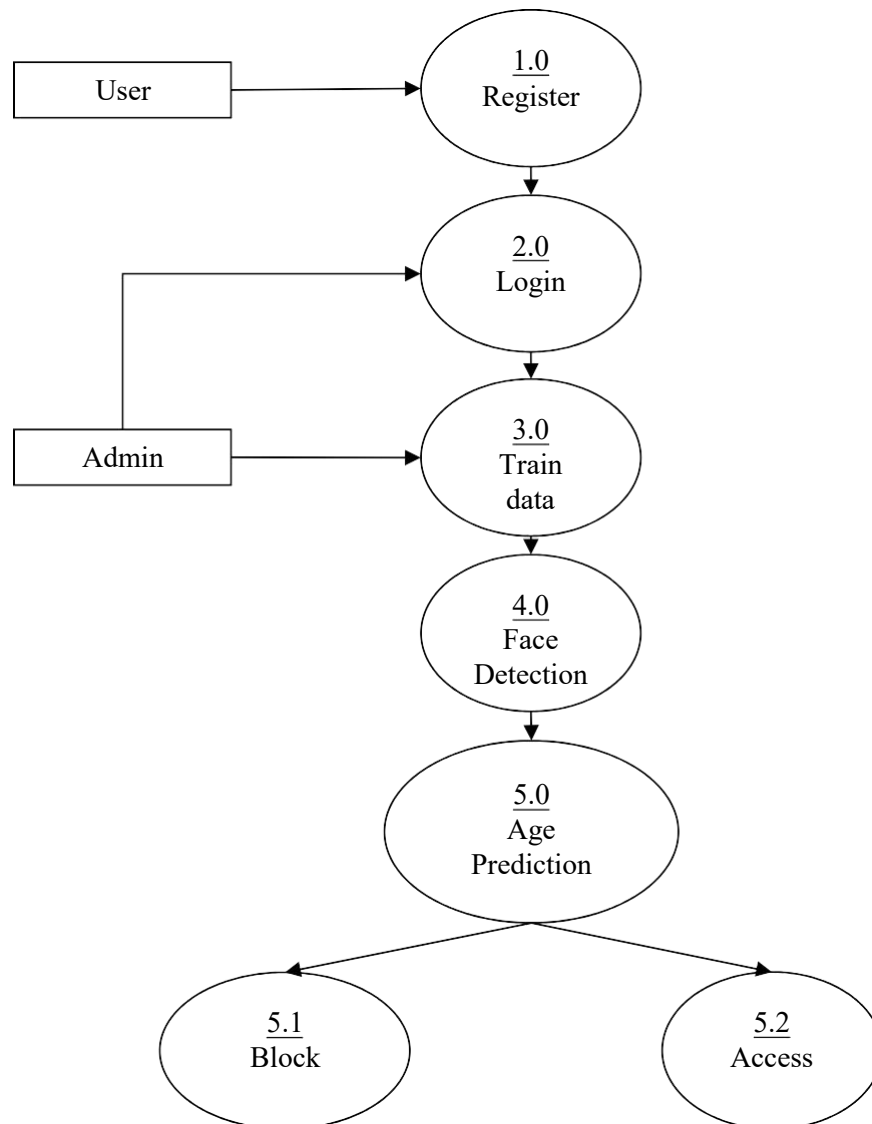
LEVEL 0

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.



LEVEL 1

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub – processes. A level 1 data flow diagram (DFD) is more detailed than a level 0 DFD but not as detailed as a level 2 DFD. It breaks down the main processes into sub processes that can then be analyzed and improved on a more intimate level.



DATA DICTIONARY

Field	Description	Example
Id	id	1
Keywords	keywords	kill
Name	name	kavi
Contact	contact	8557441223
Email	email	kavi@gmail.com
Address	address	trichy
Password	password	123
Image	image	photo.jpg
Mac	mac	56664668
Dob	dob	15.02.2000
Latitude	latitude	18465455
Longitude	longitude	17656568
Sender	sender	kavi
Receiver	receiver	janvi
Subject	subject	this is message
Message	message	hi this is important

FIG 3.2 DATAFLOW DIAGRAM

ER DIAGRAM

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system.

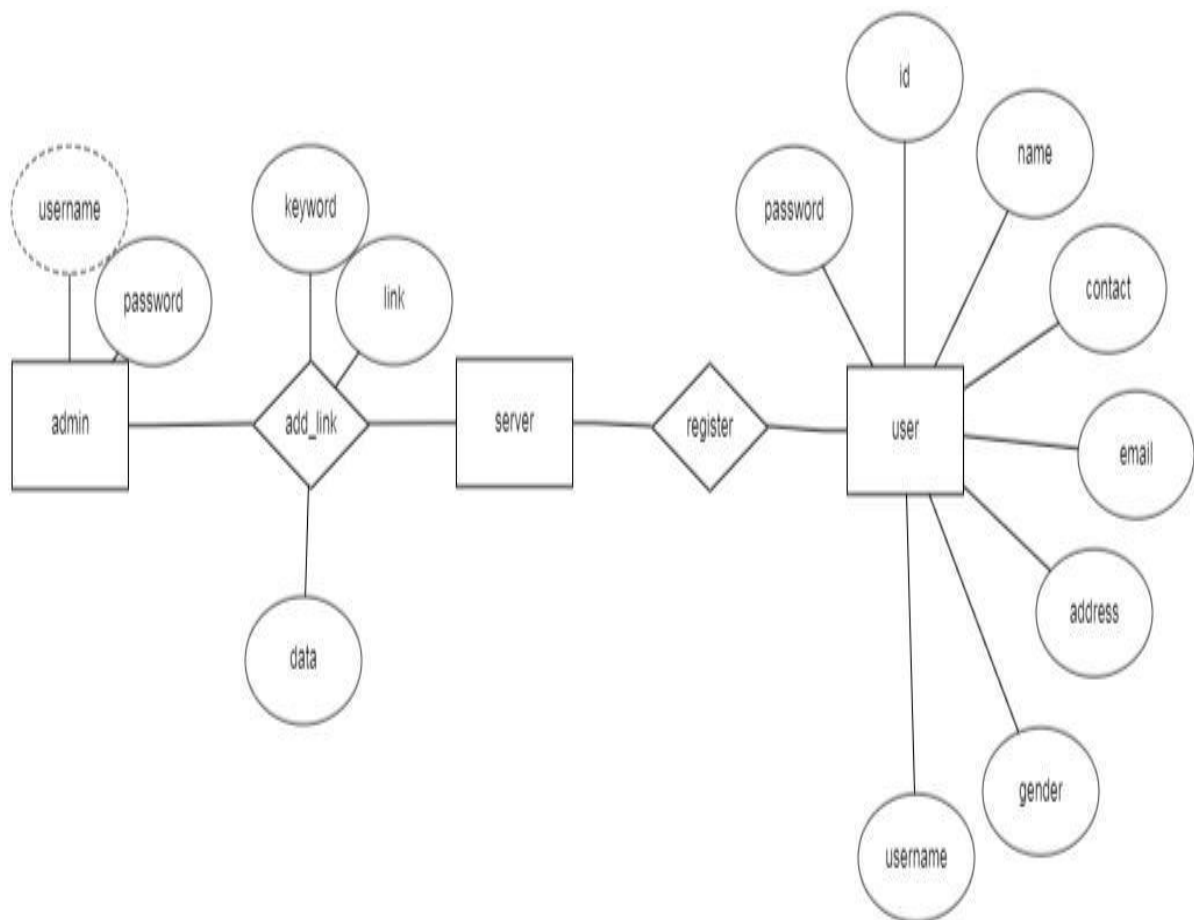


FIG 3.2. ER DIAGRAM

USE CASE DIAGRAM

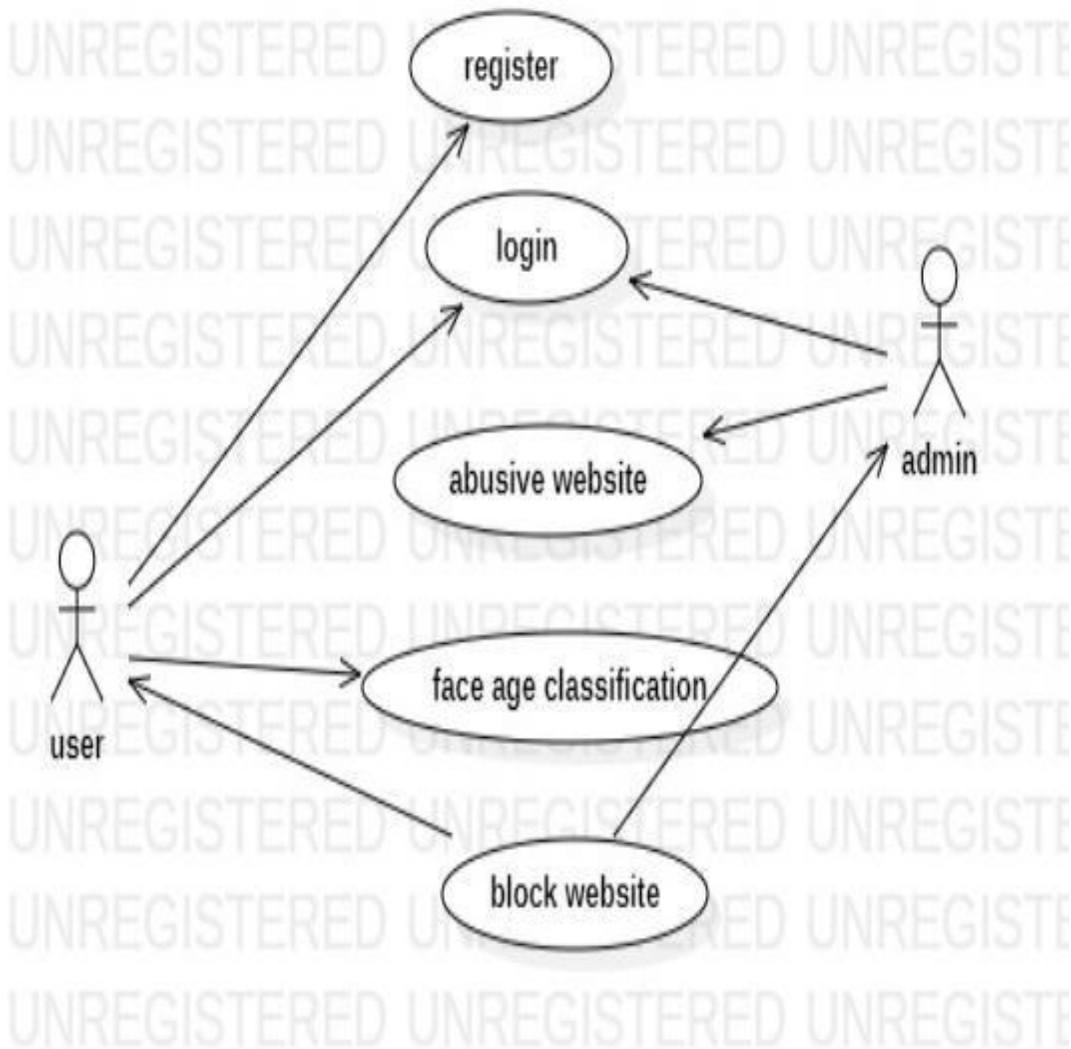


FIG 3.2.5 USE CASE DIAGRAM

5 SOFTWARE DESCRIPTION

5.1 INTRODUCTION TO PYTHON

About Python

- Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatics in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989; Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator for Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coughlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project. Guido van Rossum has since then withdrawn his nomination for the 2020 Steering council.
- Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.
- Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.
- Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.
- Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

Key points

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python Syntax compared to other programming languages

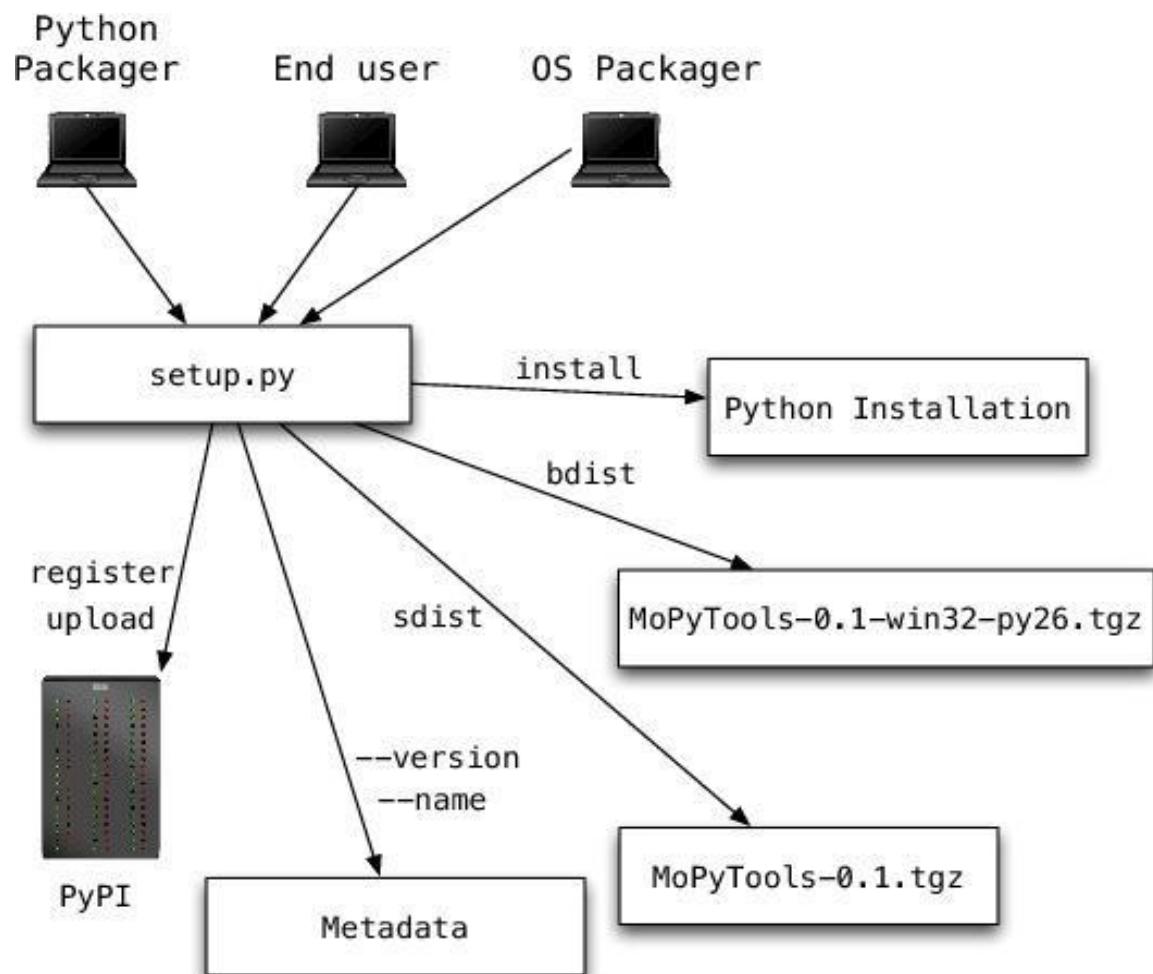
- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Importance of python to the Internet

Python is a general-purpose language — sometimes referred to as utilitarian — which is designed to be simple to read and write. The point that it's not a complex language is important. The designers placed less of an emphasis on conventional syntax, which makes it easier to work with, even for non-programmers or developers. Furthermore, because it's considered truly universal and used to meet various development needs, it's a language that offers a lot of options to programmers in general. If they begin working with Python for one job or career, they can easily jump to another, even if it's in an unrelated industry. The language is used for system operations, web development, server and administrative tools, deployment, scientific modelling and much more. But, surprisingly, many developers don't pick up Python as their primary language. Because it's so easy to use and learn, they choose it as a second or third language. This may be another reason why it's so popular among developers. Plus, it just so happens that one of the biggest tech companies in the world —

Google — uses the language for a number of their applications. They even have a developer portal devoted to Python, with free classes offered including exercises, lecture videos and more. In addition, the rise in the use of the Django framework for web development and a decline in popularity of PHP has also contributed to Python's success, but, ultimately, it's the perfect storm — just the right amount of developer and official support, as well as demand.

Design of python



PyPy's Python Interpreter is written in Python and implements the full Python language. This interpreter very closely emulates the behaviour of Python. It contains the following key components:

- a bytecode compiler responsible for producing Python code objects from the source code of a user application;
- a bytecode evaluator responsible for interpreting Python code objects;
- a standard object space, responsible for creating and manipulating the Python objects seen by the application.

The bytecode compiler is the pre-processing phase that produces a compact bytecode format via a chain of flexible passes (tokenizer, lexer, parser, abstract syntax tree builder, bytecode generator). The bytecode evaluator interprets this bytecode. It does most of its work by delegating all actual manipulations of user objects to the object space. The latter can be thought of as the library of built-in types. It defines the implementation of the user objects, like integers and lists, as well as the operations between them, like addition or truth-value-testing. This division between bytecode evaluator and object space gives a lot of flexibility. One can plug in different object spaces to get different or enriched behaviours of the Python objects.

Simple

Python was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer. Learning Python will oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

Object oriented

Python was not designed to be source-code compatible with any other language. This allowed the Python team the freedom to design with a blank state. One outcome of this was a clean, usable, pragmatic approach to objects. The object model in Python is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

Robust

The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Python. Python is a strictly typed language; it checks your code at compile time and runtime.

Python virtually eliminates the problems of memory management and deallocation, which is completely automatic. In a well-written Python program, all run-time errors can and should be managed by your program.

USER AUTHORIZATION

One way to perform session tracking is to leverage the information that comes with User authorization. When a web server restricts access to some of its resources to only those clients that log in using a recognized username and password. After the client logs in, the username is available to a servlet through `getRemoteUser()`.

When use the username to track the session. Once a user has logged in, the browser remembers her user name and resends the name and password as the user views new pages on the site. A servlet can identify the user through her username and they're by Track her session.

The biggest advantage of using user authorization to perform session tracking is that it's easy to implement. Simply tell the protect a set of pages, and use `getRemoteUser()` to identify each client. Another advantage is that the technique works even when the user accesses your site from or exits her browser before coming back.

The biggest disadvantage of user authorization is that it requires each user to register for an account and then log in each time the starts visiting your site. Most users will tolerate registering and logging in as a necessary evil when they are accessing sensitive information, but it's all overkill for simple session tracking. Another problem with user authorization is that a user cannot simultaneously maintain more than one session at the same site.

Hidden Form Fields

One way to support anonymous session tracking is to use hidden form fields. As the name implies, these are fields added to an HTML form that are not displayed in the client's browser, they are sent back to the server when the form that contains them is submitted.

In a sense, hidden form fields define constant variables for a form. For a servlet receiving a submitted form, there is no difference between a hidden field and a visible field.

As more information is associated with a client's session. It can become burdensome to pass it all using hidden form fields. In these situations, it's possible to pass on just a unique session ID that identifies a particular client session.

That session ID can be associated with complete information about its session that is stored on the server.

The advantage of hidden form fields is their ubiquity and support for anonymity. Hidden fields are supported in all the popular browsers, they demand no special server requirements, and they can be used with clients that haven't registered or logged in.

The major disadvantage with this technique, however, is that it works only for a sequence of dynamically generated forms, the technique breaks down immediately with static documents, emailed documents, bookmarked documents and browser shutdowns.

URL Rewriting

URL rewriting is another way to support anonymous session tracking. With URL rewriting every local URL the user might click on is dynamically modified. Or rewritten, to include extra information. The extra information can be in the form of extra path information, added parameters, or some custom, server-specific URL change. Due to the limited space available in rewriting a URL, the extra information is usually limited to a unique session.

Using extra path information works on all servers, and it works as a target for forms that use both the Get and Post methods. It does not work well if the servlet has to use the extra path information as true path information.

The advantages and disadvantages of URL rewriting closely match those of hidden form fields. The major difference is that URL rewriting works for all dynamically created

documents, such as the Help servlet, not just forms. With the right server support, custom URL rewriting can even work for static documents.

Persistent Cookies

A fourth technique to perform session tracking involves persistent cookies. A cookie is a bit of information. Sent by a web server to a browser that can later be read back from that browser. When a browser receives a cookie, it saves the cookie and there after sending the cookie back to the server each time it accesses a page on that server, subject to certain rules. Because a cookie's value can uniquely identify a client, cookies are often used for session tracking.

Persistent cookies offer an elegant, efficient, easy way to implement session tracking. Cookies provide as automatic an introduction for each request, as we could hope for. For each request, a cookie can automatically provide a client's session ID or perhaps a list of clients' performance. The ability to customize cookies gives them extra power and versatility.

The biggest problem with cookies is that browsers don't always accept cookies sometimes this is because the browser doesn't support cookies. More often it's because the browser doesn't support cookies. More often it's because the user has specifically configured the browser to refuse cookies.

5.2 Django

Django sometimes stylized as **django** is a Python-based free and open-source web framework that follows the model-template-views (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an American independent organization established as non-profit. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models. Despite having its own nomenclature, such as naming the callable objects generating the HTTP responses "views",^[9] the core Django framework can be seen as an MVC architecture.^[10] It consists of an object-relational mapper (ORM) that mediates between data models (defined as Python

classes) and a relational database ("**Model**"), a system for processing HTTP requests with a web templating system ("**View**"), and a regular-expression-based URL dispatcher ("**Controller**").

Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

The microframework Flask is based on the Poccoo projects, Werkzeug and Jinja.

Jinja

Jinja, also by Ronacher, is a template engine for the Python programming language and is licensed under a BSD License. Similar to the Django web framework, it handles templates in a sandbox.

6 MODULES DISCRIPTION

1.Account access:

Normally on using website interface the concern user need to make a registration system. Mostly in the existing system DOB can be added as a fake system where the verification will not be done. As a personnel access the Gmail account creation can be done with this implementation system.

2.Camera Opening:

The system can be done with a video based access when the account creation is done with Search Engine (SE) access maintenance system. The camera verification is done for the face and age base classification system.

3.Face Detection:

Applications for face detection are numerous and include computer vision communication, robotics, intelligent human-machine interfaces, virtual reality, videophone systems, and automatic access control systems. Finding every image region with a face, regardless of lighting conditions, is the aim of face detection. Five categories can be used to group these techniques: Rapid object detection employing a boosted cascade of simple features, appearance-based methods, template matching methods, knowledge (rules)-based approaches, and feature invariant ways.

4.Training Process

Training process is the account creation system where the user will make a account generation over detection of face and recognition system.

8 SAMPLE CODE

```
import pymysql
class master_flask_code:
    def __init__(self):
        self.user = 'root'
        self.password = ""
        self.host = 'localhost'
        self.database = 'python_age'
    def find_max_id(self,table):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
        cursor = conn.cursor()
        cursor.execute("SELECT id FROM "+table)
        data = cursor.fetchall()
        maxin = len(data)
        if maxin == 0:
            maxin = 1
        else:
            maxin += 1
        return maxin
    def insert_query(self,qry):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
        cursor = conn.cursor()
        result=cursor.execute(qry)
        conn.commit()
        conn.close()
        return result
    def select_login(self,qry):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
        cursor = conn.cursor()
        cursor.execute(qry)
        data = cursor.fetchall()
```

```

        check = len(data)
        if check == 0:
            return 'no'
        else:
            return 'yes'

    def select_single_colum(self, table, column):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
        qry1 = ("select " + column + " from " + table)
        cursor = conn.cursor()
        cursor.execute(qry1)
        data = cursor.fetchall()
        return data

    def select_entire_colum(self, table, column):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
        qry1=("select * from "+table+" where symptoms LIKE '%"+column+"%")
        cursor = conn.cursor()
        cursor.execute(qry1)
        data = cursor.fetchall()
        return data

    def select_direct_query(self, qry):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
        cursor = conn.cursor()
        cursor.execute(qry)
        data = cursor.fetchall()
        return data

import time
from tkinter import Tk, messagebox, ttk
from tkinter import *
from tkinter.messagebox import askyesno
import cv2

```

```

import pymysql
from PIL import Image, ImageTk
import numpy as np
import os
import sample

# import speech_recognition as sr
from PIL import ImageTk
import ar_master
store_data=sample.demo
mm= ar_master.master_flask_code()
class tk_master:
    global uid
    user=""
    def __init__(self):
        self.master='ar_master'
        self.title ='Age'
        self.titlec ='AGE'
        self.background_color ='#2F4F4F'
        # self.text_color ='#c0c0c0'
        self.text_color ='#FFF'
        self.background_image='images/background_hd1.jpg'
    def get_title(self):
        return self.title
    def get_titlec(self):
        return self.titlec
    def get_background_color(self):
        return self.background_color
    def get_text_color(self):
        return self.text_color
    def get_background_image(self):
        return self.background_image

    def set_window_design(self):

```

```

root = Tk()
w = 780
h = 500
ws = root.winfo_screenwidth()
hs = root.winfo_screenheight()
x = (ws / 2) - (w / 2)
y = (hs / 2) - (h / 2)
root.geometry('%dx%d+%d+%d' % (w, h, x, y))
image = Image.open("images/background_hd1.jpg")
resized = image.resize((w, h), Image.Resampling.LANCZOS)
image2 = ImageTk.PhotoImage(resized)
canvas = Canvas(root, width=200, height=300)
canvas.pack(fill="both", expand=True)
canvas.create_image(0, 0, image=image2, anchor=NW)
canvas.create_text(390, 20, text=self.title, font=("Times New Roman", 24), fill=self.text_color)
def admin_login():
    tt = tk_master()
    tt.admin_login()
def user_login():
    tt = tk_master()
    tt.user_login()

b1 = Button(canvas, text="ADMIN", command=admin_login, font=('times', 15, 'bold '),width=20)
canvas.create_window(400, 150, window=b1)
b1 = Button(canvas, text="USER", command=user_login, font=('times', 15, 'bold '),width=20)
canvas.create_window(400, 250, window=b1)

root.mainloop()
# def help_desk_login(self):
#     help_desk_login =Toplevel()
#     help_desk_login.attributes('-topmost', 'true')
#     get_data=tk_master()
#     w = 780
#     h = 500

```

```

# ws = help_desk_login.winfo_screenwidth()
# hs = help_desk_login.winfo_screenheight()
# x = (ws / 2) - (w / 2)
# y = (hs / 2) - (h / 2)
# help_desk_login.geometry('%dx%d+%d+%d' % (w, h, x, y))
# image = Image.open('images/background_hd1.png')
# img = image.resize((w, h))
# my_img = ImageTk.PhotoImage(img)
# help_desk_login.resizable(False, False)
# canvas1 = Canvas(help_desk_login, width=200, height=300)
# canvas1.create_image(0, 0, image=my_img, anchor=NW)
# canvas1.pack(fill="both", expand=True)
#         canvas1.create_text(390, 20, text=get_data.get_title(), font=("Times New Roman", 30),
fill=get_data.get_text_color())
#     ##
#     admin_id2 = canvas1.create_text(390, 100, text="HELP DESK LOGIN", font=("Times New
Roman", 24), fill=get_data.get_text_color())
#     admin_id2 = canvas1.create_text(300, 200, text="Username", font=("Times New Roman", 24),
fill=get_data.get_text_color())
#     admin_id2 = canvas1.create_text(300, 300, text="Password", font=("Times New Roman", 24),
fill=get_data.get_text_color())
#     e1 = Entry(canvas1, font=('times', 15, ' bold '))
#     canvas1.create_window(470, 200, window=e1)
#     e2 = Entry(canvas1, font=('times', 15, ' bold '), show="*")
#     canvas1.create_window(470, 300, window=e2)
#     def exit_program():
#         a=e1.get()
#         b=e2.get()
#         if (a == ""):
#             messagebox.showinfo(title="Alert", message="Enter Username", parent=help_desk_login)
#         elif (b == ""):
#             messagebox.showinfo(title="Alert", message="Enter Password", parent=help_desk_login)
#         else:
#             data=mm.select_direct_query("select * from help_desk_details where username='"+str(a)+"'")

```

```

and password="" + str(b) + "")
#         if len(data)>0:
#             messagebox.showinfo("Result","Login Success", parent=help_desk_login)
#             help_desk_login.destroy()
#             tt = tk_master()
#             tt.help_desk_home()
#         else:
#             messagebox.showinfo("Result","Login Failed", parent=help_desk_login)
#     b1=Button(canvas1, text="Login", command=exit_program, font=('times', 15, 'bold'))
#     canvas1.create_window(470, 400, window=b1)
#     help_desk_login.mainloop()
def admin_login(self):
    admin_login_root = Toplevel()
    admin_login_root.attributes('-topmost', 'true')
    get_data=tk_master()
    w = 780
    h = 500
    ws = admin_login_root.winfo_screenwidth()
    hs = admin_login_root.winfo_screenheight()
    x = (ws / 2) - (w / 2)
    y = (hs / 2) - (h / 2)
    admin_login_root.geometry("%dx%d+%d+%d" % (w, h, x, y))
    image = Image.open('images/background_hd1.jpg')
    img = image.resize((w, h))
    my_img = ImageTk.PhotoImage(img)
    admin_login_root.resizable(False, False)
    canvas1 = Canvas(admin_login_root, width=200, height=300)
    canvas1.create_image(0, 0, image=my_img, anchor=NW)
    canvas1.pack(fill="both", expand=True)
    canvas1.create_text(390, 20, text=get_data.get_title(), font=("Times New Roman", 30),
fill=get_data.get_text_color())
    ##
    admin_id2 = canvas1.create_text(390, 100, text="ADMIN LOGIN", font=("Times New Roman", 24),
fill=get_data.get_text_color())

```

```

admin_id2 = canvas1.create_text(300, 200, text="Username", font=("Times New Roman", 24),
fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 300, text="Password", font=("Times New Roman", 24),
fill=get_data.get_text_color())
e1 = Entry(canvas1,font=('times', 15, ' bold '))
canvas1.create_window(470, 200, window=e1)
e2 = Entry(canvas1, font=('times', 15, ' bold '),show="*")
canvas1.create_window(470, 300, window=e2)
def exit_program():
    a=e1.get()
    b=e2.get()
    if(a == ""):
        messagebox.showinfo(title="Alert", message="Enter Username", parent=admin_login_root)
    elif(b == ""):
        messagebox.showinfo(title="Alert", message="Enter Password", parent=admin_login_root)
    elif((a=="admin")and(b=="admin")):
        messagebox.showinfo("Result","Login Success", parent=admin_login_root)
        admin_login_root.destroy()
        tt = tk_master()
        tt.admin_home()
    else:
        messagebox.showinfo("Result","Login Failed", parent=admin_login_root)
b1=Button(canvas1, text="Login", command=exit_program, font=('times', 15, ' bold '))
canvas1.create_window(470, 400, window=b1)
admin_login_root.mainloop()
def admin_home(self):
    admin_home_root = Toplevel()
    admin_home_root.attributes('-topmost', 'true')
    get_data = tk_master()
    w = 780
    h = 500
    ws = admin_home_root.winfo_screenwidth()
    hs = admin_home_root.winfo_screenheight()
    x = (ws / 2) - (w / 2)

```

```

y = (hs / 2) - (h / 2)
admin_home_root.geometry('%dx%d+%d+%d' % (w, h, x, y))
image = Image.open('images/background_hd1.jpg')
img = image.resize((w, h))
my_img = ImageTk.PhotoImage(img)
admin_home_root.resizable(False, False)
canvas1 = Canvas(admin_home_root, width=200, height=300)
canvas1.create_image(0, 0, image=my_img, anchor=NW)
canvas1.pack(fill="both", expand=True)
canvas1.create_text(390, 20, text=get_data.get_title(), font=("Times New Roman",
24),fill=get_data.get_text_color())
##
admin_id2 = canvas1.create_text(390, 80, text="ADMIN HOME", font=("Times New Roman",
24),fill=get_data.get_text_color())
def add_help_desk():
    add_help_desk_root = Toplevel()
    add_help_desk_root.attributes('-topmost', 'true')
    get_data = tk_master()
    w = 780
    h = 500
    ws = add_help_desk_root.winfo_screenwidth()
    hs = add_help_desk_root.winfo_screenheight()
    x = (ws / 2) - (w / 2)
    y = (hs / 2) - (h / 2)
    add_help_desk_root.geometry('%dx%d+%d+%d' % (w, h, x, y))
    image = Image.open('images/background_hd1.jpg')
    img = image.resize((w, h))
    my_img = ImageTk.PhotoImage(img)
    add_help_desk_root.resizable(False, False)
    canvas1 = Canvas(add_help_desk_root, width=200, height=300)
    canvas1.create_image(0, 0, image=my_img, anchor=NW)
    canvas1.pack(fill="both", expand=True)
    # canvas1.create_text(390, 20, text=get_data.get_title(), font=("Times New Roman", 24),
    #
        fill=get_data.get_text_color())

```

```

##
admin_id2 = canvas1.create_text(390, 20, text="ADD HELP DESK", font=("Times New Roman",
24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 70, text="NAME", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 120, text="FATHER NAME", font=("Times New Roman",
24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 170, text="CONTACT", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 220, text="EMAIL", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 270, text="ADDRESS", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 320, text="PINCODE", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 370, text="USERNAME", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 420, text="PASSWORD", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())

e1 = Entry(canvas1, font=('times', 15, 'bold '))
canvas1.create_window(550, 70, window=e1)
e2 = Entry(canvas1, font=('times', 15, 'bold '))
canvas1.create_window(550, 120, window=e2)
e3 = Entry(canvas1, font=('times', 15, 'bold '))
canvas1.create_window(550, 170, window=e3)
e4 = Entry(canvas1, font=('times', 15, 'bold '))
canvas1.create_window(550, 220, window=e4)
e5 = Entry(canvas1, font=('times', 15, 'bold '))
canvas1.create_window(550, 270, window=e5)
e6 = Entry(canvas1, font=('times', 15, 'bold '))
canvas1.create_window(550, 320, window=e6)

```

```

admin_id2 = canvas1.create_text(300, 170, text="CONTACT", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())
admin_id2 = canvas1.create_text(300, 220, text="EMAIL", font=("Times New Roman", 24),
                                fill=get_data.get_text_color())

conn      =      pymysql.connect(user=self.user,      password=self.password,      host=self.host,
database=self.database)
cursor = conn.cursor()
cursor.execute(qry)
data = cursor.fetchall()
return data

```

6.2 SAMPLE SCREENSHOT

Fig -6.2.1- SCREEN SHOTS

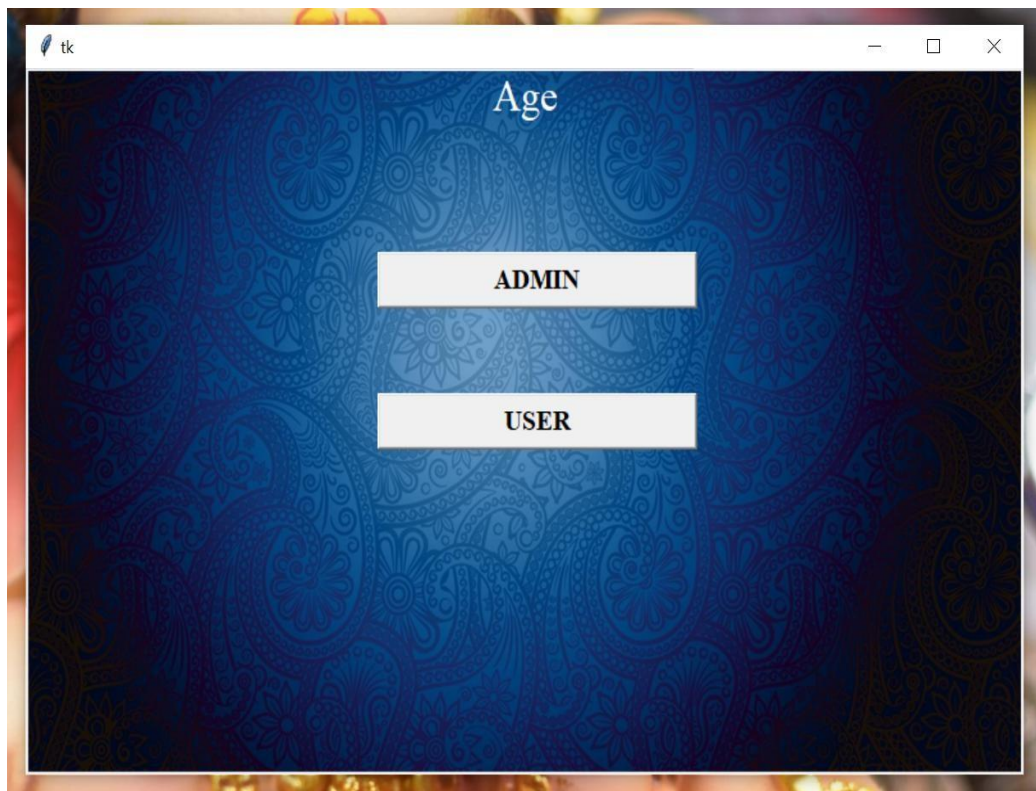
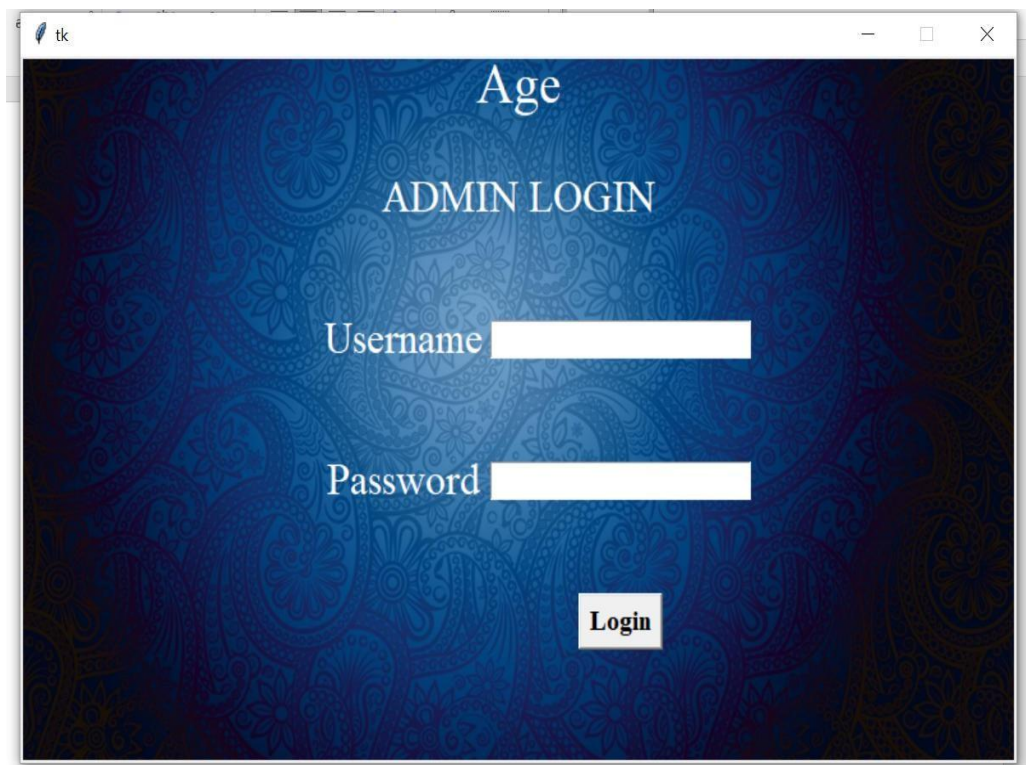
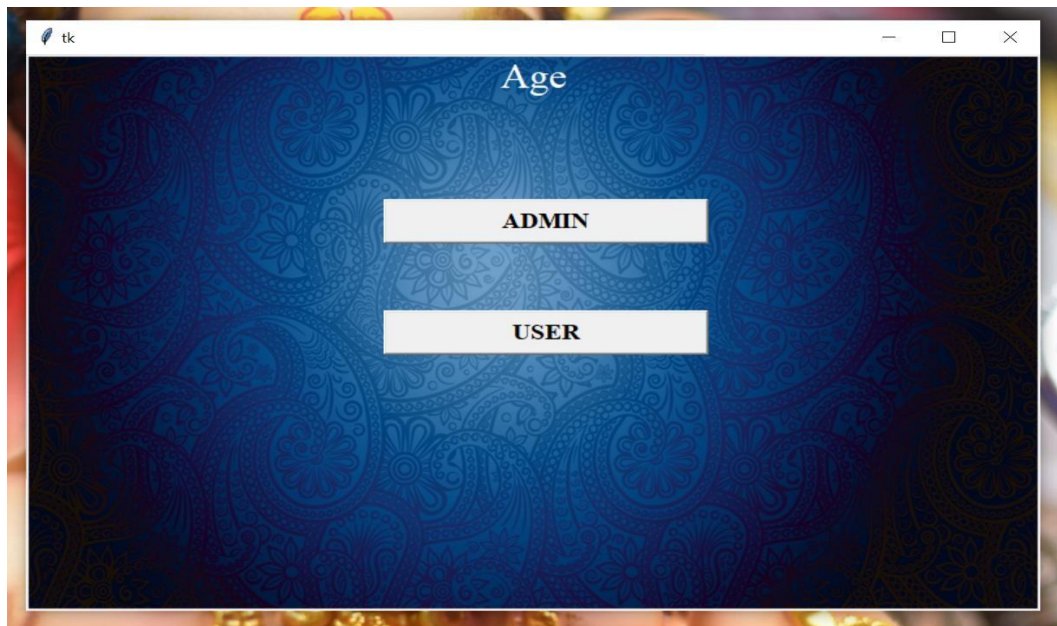


Fig 6.2.2-ADMIN LOGIN



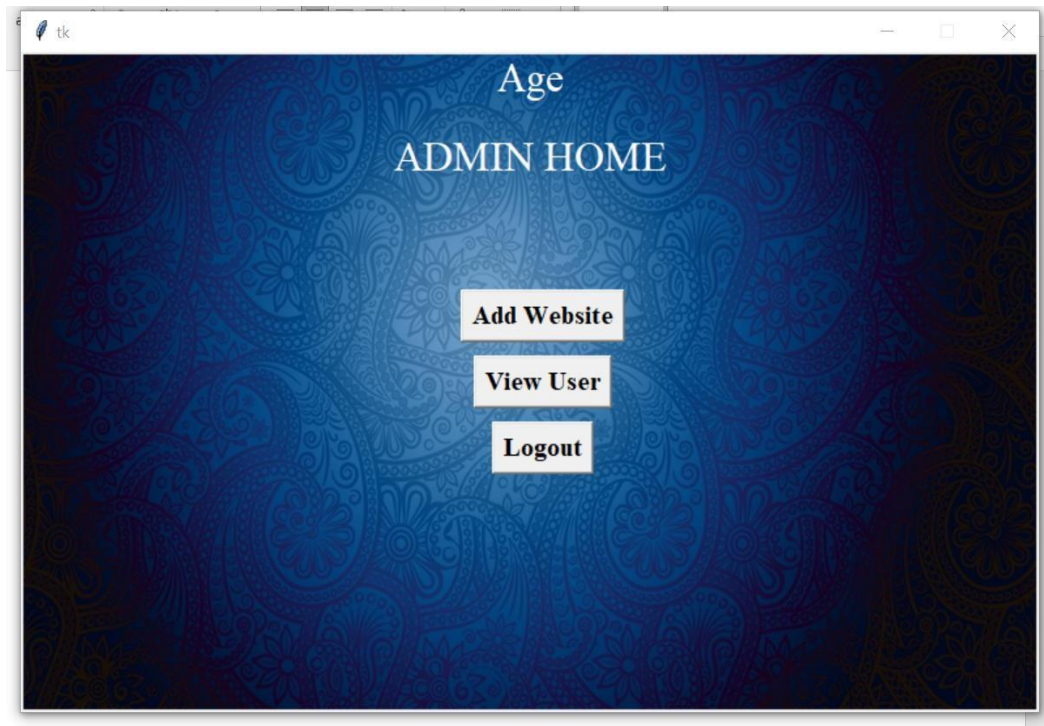


Fig 6.2.3- ADD WEBSITE

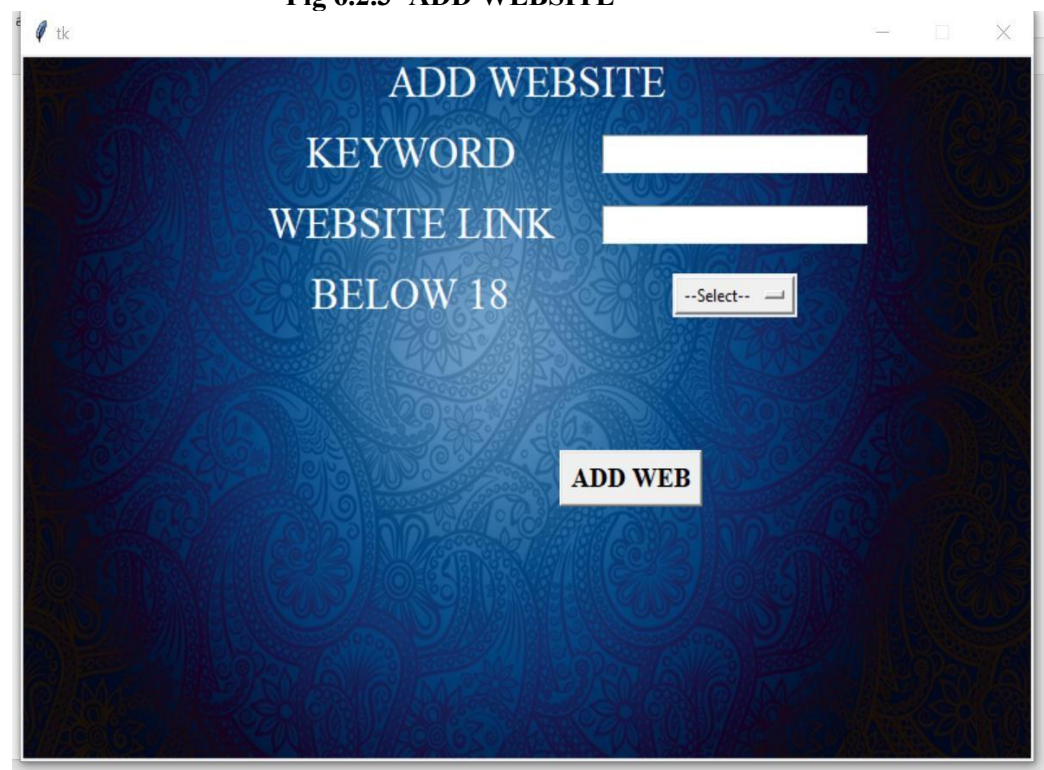


Fig 6.2.4 USER DETAILS

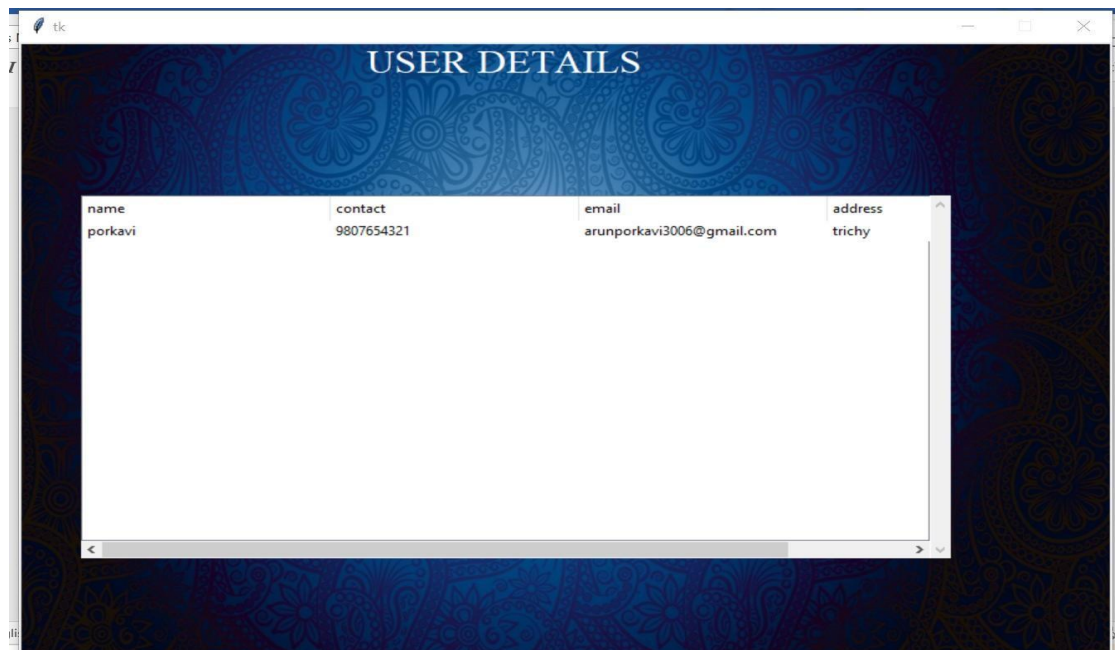


Fig 6.2.5- LOGIN

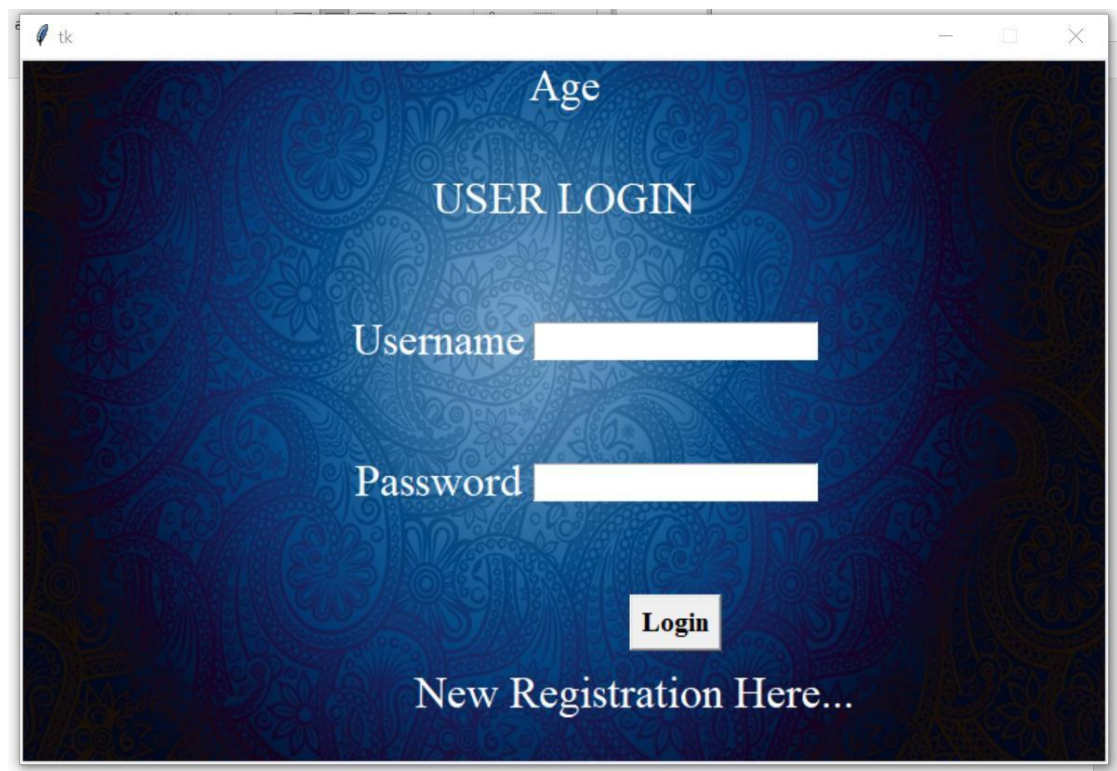
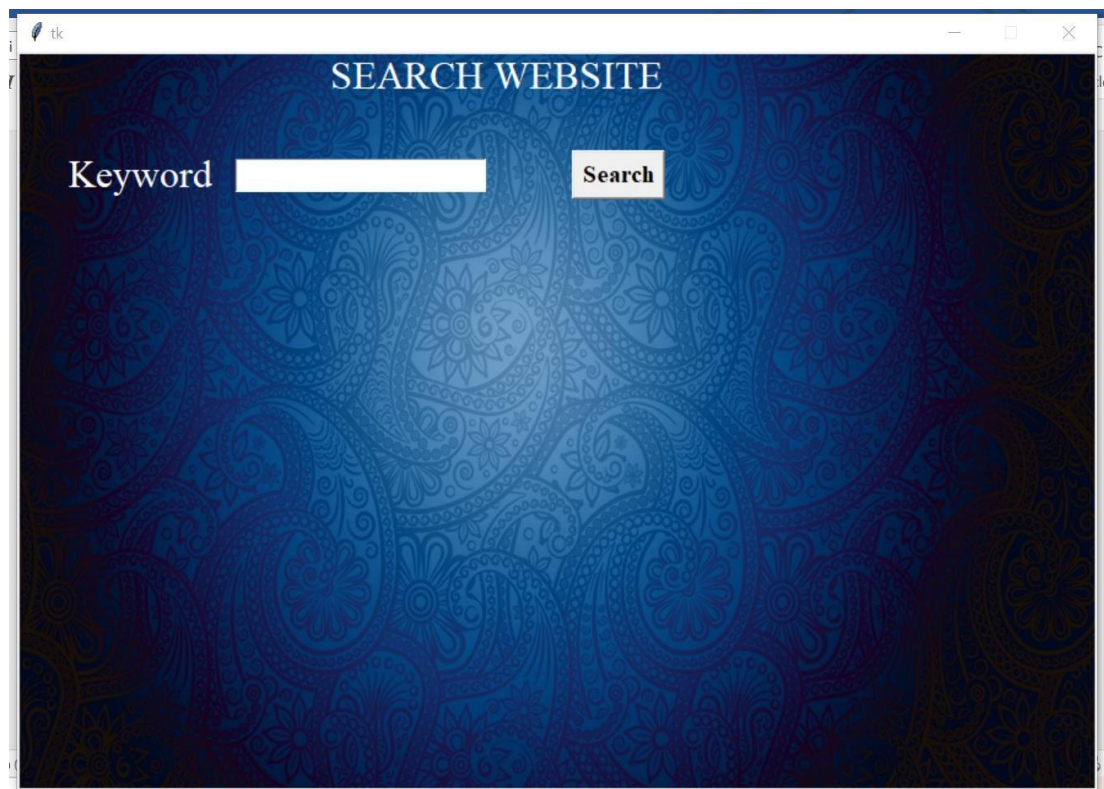
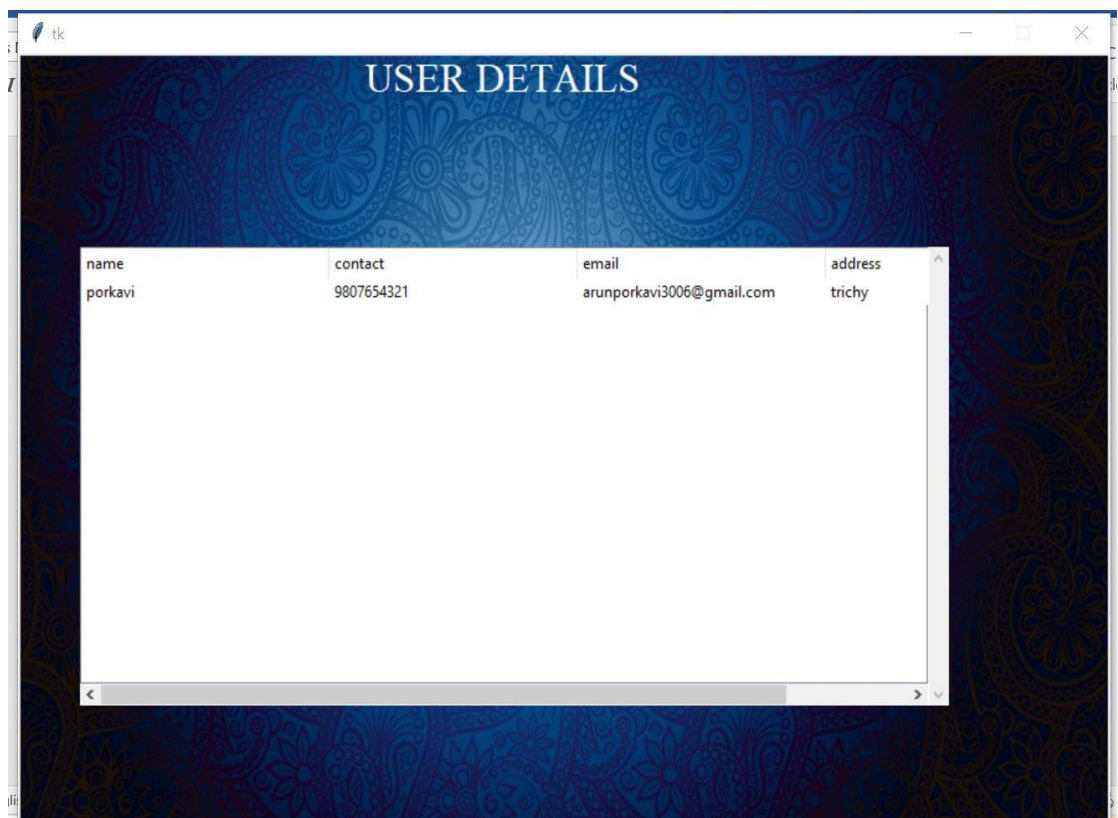


Fig 6.2.6- SEARCH WEBSITE



A screenshot of a Tkinter window titled "SEARCH WEBSITE". The window has a dark blue background with a complex, repeating paisley pattern. At the top center, the title "SEARCH WEBSITE" is displayed in a white, serif font. Below the title, on the left, is the label "Keyword" in a white, serif font. To the right of the label is a white rectangular input field. Further to the right is a white rectangular button with the text "Search" in a black, sans-serif font. The window's title bar at the top shows the Tkinter logo and the text "tk:".



A screenshot of a Tkinter window titled "USER DETAILS". The window has the same dark blue paisley background as the first window. At the top center, the title "USER DETAILS" is displayed in a white, serif font. Below the title, there is a white rectangular area containing a table of user information. The table has four columns: "name", "contact", "email", and "address". The first row of data contains the values "porkavi", "9807654321", "arunporkavi3006@gmail.com", and "trichy". The table is enclosed in a white border with a scroll bar on the right side. The window's title bar at the top shows the Tkinter logo and the text "tk:".

name	contact	email	address
porkavi	9807654321	arunporkavi3006@gmail.com	trichy

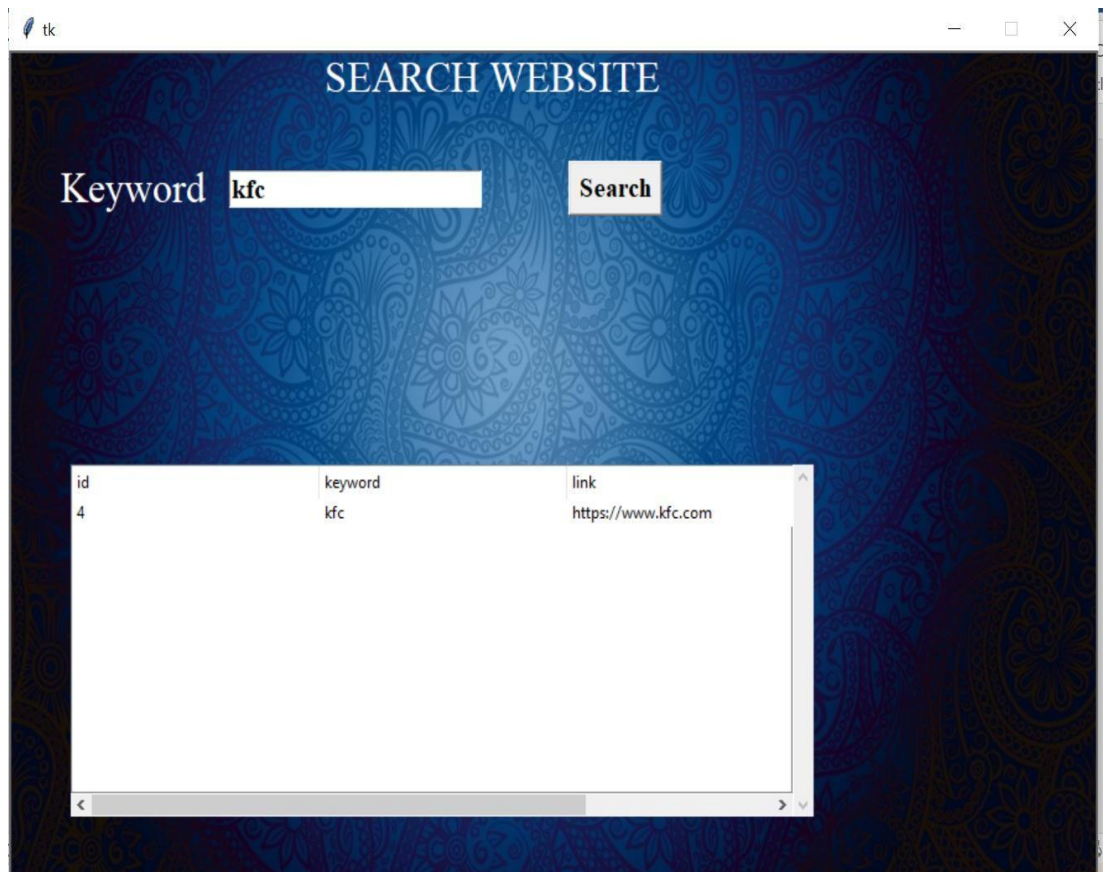


Fig -6.2.6-NEW USER REGISTRATION

USER REGISTRATION

Name

Contact

Email

Address

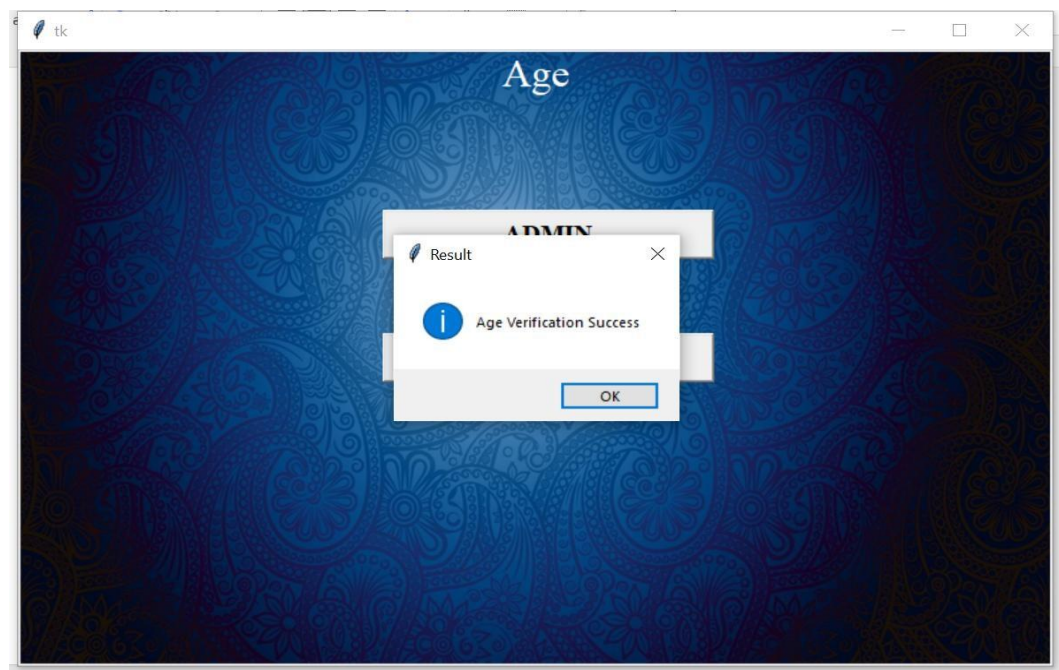
Username

Password

Register



Fig6.2.7-USER AGE VERIFICATION



CHAPTER 9

CONCLUSION AND FUTURE WORK

This project has executed that there are both effective and ineffective components in the proposed age based filtering system intended to restrict and prohibit access to content that could be harmful to children. The system's inefficiency is mostly explained by how simple it is to get around the present methodology. A higher level of efficacy can be attained by having the system support multiple face kinds. With the filtering tools at hand, the only feasible solution is to ban websites using the proposed strategy. For this, we also created a tool. Using the data sets, we conducted experiments to test the tool. Based on the experimental findings, we deduce that the tool is well-suited for informative content filtering and offers a high degree of precision in distinguishing between informative and non-informative web page content. The primary elements of the suggested system are facial and skin detection methods. To determine if an image contains enough human skin pixels to be categorized as pornography or not, we used the percentage of pixels that contain skin material. To get around the conventional filtering methods, we computed the face skin % in our implementation and then deducted it from the overall estimated skin percentage.

CHAPTER -10

REFERENCE

- [1]. H Weber, a. cruz rodrigues, and a. mateus, "feeling and mood in design thinking," design doctoral conference'16: transversality - proceedings of the ddc third conference, no. july, pp. 65– 72, 2023.
- [2]. g. guo, "human l age estimation and sex characterization," studies in computational intelligence, vol. 409, pp. 101– 131, 2023.
- [3]. r. hurley, s. prusty, h. soroush, r. j. dividers, j. albrecht, e. cecchet, b. n. levine, m. liberatore, b. lynn, and j. wolak, "estimation and investigation of kid sex entertainment dealing on p2p systems," in proceedings of the 22nd worldwide gathering on world wide web. acm, 2022, pp. 631– 642.
- [4] j. farina, m. scanlon, n.- a. le-khac, and m.- t. kechadi, "outline of the forensic investigation of cloud services," in tenth international conference on availability, reliability and security (ares 2023). toulouse, france: ieee, 08 2023, pp. 556– 565.
- [5]. m. aiken, m. moran, and m. j. berry, "kid misuse material and the web: cyberpsychology of online tyke related sex irritating," in 29th gathering of the interpol specialist group on crimes against children, lyons, france, september, 2023, pp. 5– 7.
- [6]. b. jones, s. pleno, and m. wilkinson, "the utilization of irregular testing in examinations including tyke misuse material," digital investigation, vol. 9, pp. s99– s107, 2023.
- [7]. t. krone, "a typology of online kid erotic entertainment irritating," trends and issues in wrongdoing and criminal equity, no. 279, pp. 1– 6, 2023.
- [8]. z. erkin, m. franz, j. guajardo, s. katzenbeisser, i. lagendijk, and t. toft, "security saving face acknowledgment," in proceedings of the ninth international symposium on privacy enhancing technologies, ser. pets '09. berlin, heidelberg: springer-verlag, 2019, pp.

