**Retrouver – Together We Come**

A Project Report

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

**BTech Information Technology**

Submitted by

Anushka Jain (2016716)

Muskan Gupta (2016780)

Khushi Rawal (2016767)

Name of Coordinator(s):

(Name, Designation, Organization)

Name of the Mentor:

Ms. Pooja Asopa

****

**Department of Mathematics & Computing**

**Banasthali Vidyapith**

**Banasthali - 304022**

**Session: 2022-23**

**Certificate**

### Certified that Muskan Gupta, Anushka Jain, Khushi Rawal has carried out the project work titled “Retrouver, Together We Come” from 17th Dec 2022 to 3rd May 2023 for the award of the BTech(Information Technology) from Banasthali Vidyapeeth under my supervision. The thesis embodies result of original work and studies carried out by Student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

Ms. Pooja Asopa

Designation:

Banasthali Vidyapeeth

13th April 2023

**Abstract**

The thought of a family member, a friend or someone else you care about going missing can be terrifying. This project aims to help find your loved ones using Face Recognition Technology. Therefore, the primary goal of our website hereby is to provide a platform that helps people in reconnecting with their loved ones with whom they have lost connection with. The users will create their own profiles adding some basic credentials such as name, age, their photo and other important details and similarly they will enter details of the person that they are trying to find. This information will be stored in the database and it matches the profile with all the other added profiles using face recognition and other parameters and find the one which has maximum similar features. Further we will be adding more functionalities to make our website more compatible and beneficial towards our goal of finding missing people.

**Acknowledgement**

In the accomplishment of completion of my project “**Retrouver-Together We Come”.**  I would like to convey my special gratitude to Ms. Pooja Asopa, of AIM & ACT Department of Banasthali Vidyapeeth.

Your valuable guidance and suggestions helped me in various phases of the completion of this project. I will always be thankful to you in this regard.

I am ensuring that this project was finished by us and not copied.

Banasthali Vidyapeeth

13th April 2023

Students of BTech (Information Technology):

Muskan Gupta (2016780)

Anushka Jain (2016716)

Khushi Rawal (2016767)

**TABLE OF CONTENT**

Pg.No

Chapter 1 INTRODUCTION 1

Chapter 2 SRS

Chapter 3 SDS

**Chapter 4** **CODINGS**

**Chapter 5 USER INTERFACES**

[REFERENCES](#_TOC_250000)

**Introduction:**

The purpose of this document is to describe the software requirement in developing the website “**Retrouver-Together We Come**” which will help people in reconnecting with their loved ones with whom they have lost connection with. This website will be mainly designed for lost people and their family members in order to establish communication on Internet.

**TABLE OF CONTENTS OF SRS**

**1**. **Introduction**

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms and Abbreviations

1.4 Overview

**2**. **General Description**

2.1 Product Perspective

2.1.1 Product Function

2.1.2 Hardware Interface

2.1.3 Software Interface

2.1.4 Communication Interface

2.2 User Characteristics

2.3 General Constraints

2.4 Technologies used

**3. Specific Requirements**

3.1 Functional Requirements

3.2 Non-Functional Requirements

3.2.1 AVAILABILITY

3.2.2 SECURITY

3.2.3 RELIABILITY

3.2.4 PORTABILITY

3.2.5 MAINTAINABILITY

# 1. Introduction

The purpose of this document is to describe the software requirement in developing the website “**Retrouver-Together We Come**” which will help people in reconnecting with their loved ones with whom they have lost connection with. This website will be mainly designed for lost people and their family members in order to establish communication on Internet.

## 1.1 Purpose

The primary goal of our website is to provide a platform that helps people in reconnecting with their loved ones with whom they have lost connection with. The users will create their own profiles adding some basic credentials such as name, age, their photo, and other important details and similarly they will enter details of the person that they are trying to find. This information will be stored in the database and it matches the profile with all the other added profiles using face recognition and other parameters and find the one which has maximum similar features. Further we will be adding more functionalities to make our website more compatible and beneficial towards our goal of finding missing people.

## 1.2 Scope

“**Retrouver-Together We Come**” is a web-based service which intends to reconnect lost people with their families. Our website will be capable of searching for an existing profile from the database which is similar to the lost person’s profile.

Following are the scopes of our project:

* The targeted groups of our project are the lost people and their families.
* The database will store all the information of the people who register on the website.
* The website uses face recognition model for comparing faces in the database to provide the closest match to the lost person’s image.

## 1.3 Definitions, Acronyms, and Abbreviations.

* **HTML:** Hypertext Markup Language is a markup language used to design static web pages.
* **HTTP**: Hypertext Transfer Protocol is a transaction oriented client/server protocol between web browser & a Web Server.
* **HTTPS:** Secure Hypertext Transfer Protocol is a HTTP over SSL (secure socket layer).
* **www:** World Wide Web
* **TCP/IP:** Transmission Control Protocol/Internet Protocol, the suite of communication protocols used to connect hosts on the Internet. TCP/IP uses several protocols, the two main ones being TCP and IP.
* **DB:** Database
* **RAM:** Random Access Memory
* **HDD:** Hard Disk Drive

## 1.4 Overview

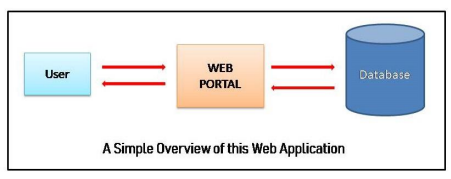
"Retrouver-"Together we come” is a website with advanced features that would help the lost people by accelerating the process of searching their family members using face recognition.  
  
It aims to allow users to search for their family members lost and helps to reconnect with them through using Face-Recognition Technique.  
   
The face recognition model in our system will try to find a match in the database. It is performed by comparing the face encodings of the uploaded image to the face encodings of the images in the database.

# 2. The Overall Description

## 

## 2.1 Product Perspective

This website is mainly aimed towards the people who might have lost connection with their family members. It will be user friendly and reliable to use.  
This system will consist of one web portal. The web portal will use the database to get data which will also add and modify data. Whole database connection will go over Internet.



### 2.1.1 Product Function

In this project, system will collect information from lost person’s family member regarding the person who needs to be found and apply various algorithms to find the profile similar to that person.

“**Retrouver-Together We Come”** will allow:

1. Login to the system through the login page of the website.
2. Create your profile.
3. Upload the lost person’s details.
4. Searching for a profile similar to the lost person’s profile.
5. Give feedback.
6. Sign Out from the website.

### 2.1.2 Hardware Interfaces

. **Server Side:**

* **RAM:** Min 4GB
* **HDD:**10 GB
* **Processor:** Min 3GHz

**Client Side:**

* **RAM:** Min 512 MB
* **HDD:** 5 GB
* **Processor:** Min 1-2GHz

### 2.1.3 Software Interfaces

**Server Side:**

* **OS:** Windows Server 2000 or onwards
* **Web Server:** Flask Framework

**Client Side:**

* **OS:** Any Operating system that is compatible with the browser
* **Browser:** Any browser is compatible

### 2.1.4 Communications Interfaces

* Client on Internet will be using HTTP/HTTPS protocol.
* Client on Intranet will be using TCP/IP protocol.

## 2.2 User Characteristics

The users of “**Retrouver-Together We Come**” should be computer literate.

Users of this software can be categorized as following:

* LOST PERSON

These types of users are the ones who have lost connection with their families.

* LOST PERSON’S FAMILY

These types of users are the ones who want to reconnect with their loved ones.

* ADMINISTRATOR

These types of users are basically the administrator of whole website. They can change any data at any time without creating any conflict or any confusion for the rest of users.

## 2.3 Constraints

1. **Higher Order Language Functions:** The Visual Studio Code will be used for developing the web-pages and for the database information SQL Server 2008 will be used.
2. **Poor Image Quality:** If a clear image is not uploaded, it won’t be efficient in finding a similar profile.
3. **Criticality of the Website:** The server application will be available

24 X 7.

1. **Safety and Security Considerations:** The password and a valid username are the security issue.
2. Any substantial enhancement in website will require approval of the administrator.
3. Regulatory policies
4. Hardware limitation: For example-signal timing requirements.

**2.4Technologies Used**

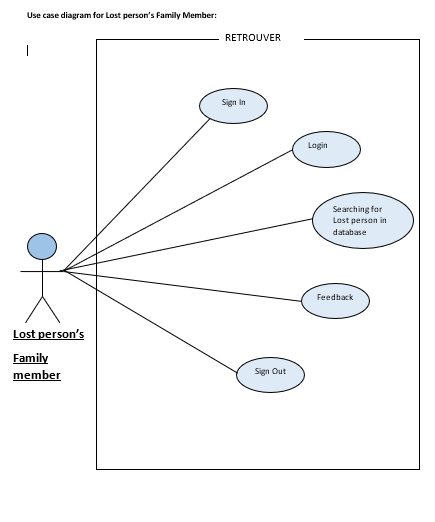
* **Front End:** HTML, CSS, JS
* **Back End:** MySQL DATABASE
* **Design Tool:** Microsoft Visual Studio Code
* **Web Server:** Flask Web Server

# 3. Specific Requirements

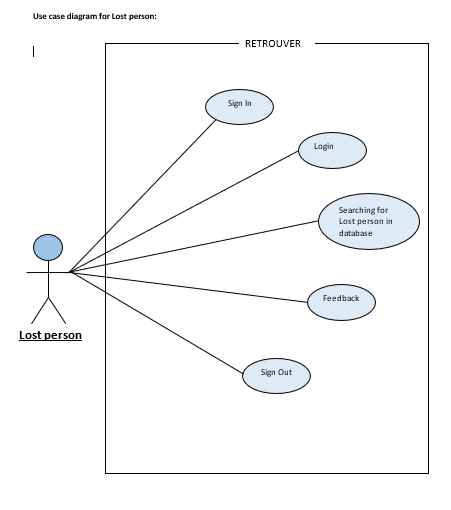
## 3.1 Functional

**Use-case diagram for the various users of this web application is:**

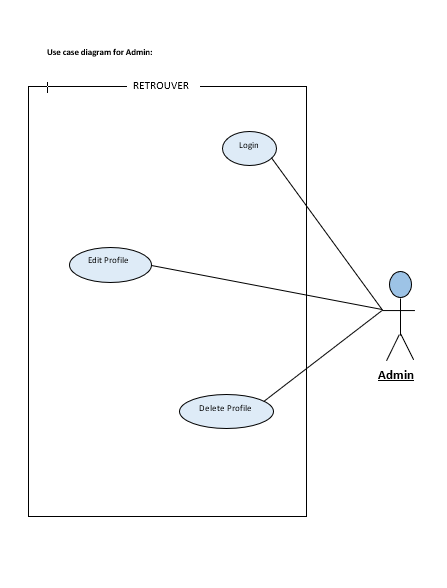
**Use case diagram for Lost Person’s Family Member:**



**Use case diagram for Lost Person:**



**Use case diagram for Admin:**



**Sign-Up**

| Use Case No | 1 |
| --- | --- |
| Use Case Name | Sign up |
| Actors: | Lost Person and Lost Person’s family member |
| Descriptions | This module helps the Lost People, their family member and administrator to create new accounts. |
| Input | First Name, Last Name, Gender, Contact number,  Date of Birth, Email address, Image, Username, Password |
| Normal Course Events | 1. User enter their username. 2. User enter their password. 3. Users click sign in button. 4. Users enter their first name and last name. 5. User enter their gender. 6. Users enter their contact number. 7. User enter their date of birth. 8. User insert their image. 9. User enter their email address. 10. Users click create profile button. 11. System connects to database. 12. A message appears which shows that the account is created. |
| Alternative Courses | 1. Same Username exists in database 2. Error message appears. 3. Continue with step 1 in the normal course   events. |
| 1. An error may occur during the database   operation.   1. System shows error messages. |
| Output | New user account will be created. |

**Table 1: Use Case of Sign-Up Function**

**Login**

| Use Case No | 2 |
| --- | --- |
| Use Case Name | Login |
| Actors: | Lost Person, Lost Person’s family member,  Administrator |
| Descriptions | This module helps the Lost People, their family member and administrator to login. |
| Pre-conditions | The user must be registered on the system. |
| Input | Username and password. |
| Normal Course Events | 1. Users enter their Username. 2. Users enter their password. 3. Users click login button. 4. System connects to database. 5. Homepage displayed. |
| Alternative Courses | 1. User enters incorrect user name and/or   password.   1. Error message appears. 2. Continue with step 1 in the normal course   events.   1. An error may occur during the database   operation.   1. System shows error messages. |
| Output | User homepage will be displayed. |

**Table 2: Use Case of Login Function**

**Logout**

| Use Case No. | 3 |
| --- | --- |
| Use Case Name | Logout |
| Actors: | Lost Person, Lost Person’s family member |
| Descriptions | This module helps the Lost people, their family member and administrator to logout. |
| Pre-conditions | The user must be logged in to the website. |
| Normal Course Events | 1. The registered users click on the Log Out button. 2. DB connection terminated. 3. The users Sign out successfully. 4. The website will be directed to login page. |
| Alternative Courses | 1. An error may occur during the database   operation.   1. System shows error messages. |
| Output | Website’s login page will be displayed. |

**Table 3: Use Case of Logout Function**

**Feedback**

| Use Case No | 5 |
| --- | --- |
| Use Case Name | Feedback |
| Actors: | Lost Person, Lost Person’s family member |
| Descriptions | This module helps the lost people, their family member and administrator to give their feedback. |
| Pre-conditions | The user must be logged into the system. |
| Input | Username, Feedback, Suggestions |
| Normal Course Events | 1. Users enter their username. 2. Users click on their feedback. 3. User enter their suggestions. |
| Alternative Courses | 1. User enters incorrect username. 2. Error message appears. 3. Continue with step 1 in the normal course   events.   1. An error may occur during the database   operation.  System shows error messages |
| Output | “Feedback submitted” message is displayed. |

**Table 5: Use Case of Feedback Function**

* **Profile Creation**

| Use Case No | 6 |
| --- | --- |
| Use Case Name | Lost Person’s Profile Creation |
| Actors: | Lost Person, Lost Person’s family member |
| Descriptions | This module helps the Lost people, their family member and administrator to create lost person’s profile. |
| Input | Information of the lost person |
| Normal Course Events | 1. Users enter lost person’s first name. 2. Users enter lost person’s last name. 3. Users enter lost person’s date of birth. 4. Users enter lost person’s gender. 5. User enters other required Information. 6. Users click create profile button. 7. System connects to database. 8. A message appears which shows Profile created. |
| Alternative Courses | 1. An error may occur during the database   operation.   1. System shows error messages. |
| Output | Lost person’s profile will be created. |

**Table 6: Use Case of Profile creation Function**

* 1. **Nonfunctional Requirements (Software System Attributes)**

**3.2.1 AVAILABILITY**

The availability of this web-site is up to the Internet connection of the client. Since this is client-server related web-site shall be attainable all the time. User should have an account to enter the system; if user does not have an account, then user can only see the information which will be displayed on the homepage of the web-site.

.

**3.2.2 RELIABILITY**

A backup file is maintained so that in case of system crash, the data will not be affected.

**3.2.3 PORTABILITY**

The system is developed using Flask which provides a framework for developing web-based applications.

.

**3.2.4 MAINTAINABILITY**

This website will follow the modular structure so it will be easy to maintain.

**TABLE OF CONTENTS OF SDS**

1. **Introduction**

1.1 Purpose

1.2 Scope

1.3 Terms, Definitions, Acronyms and Abbreviations

1.3.1 Definitions, Acronyms, and Abbreviations

1.3.2 Acronyms and Abbreviations

2. **System Architectural Design**

2.1 Detailed Description of Components

2.1.1 Component and Processing Detail

2.1.2 Structure and Relationship

3. **Data Design**

3.1 Database Description

3.2 E-R Diagram

4. **User Interface Design**

4.1 Layout

## 5. References

# 1. Introduction

## 1.1 Purpose

The purpose of SDS document includes an architectural design of “Retrouver-Together We Come.” It includes data design, detail design and interface design. It provides a description of the design of the website fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to build.

## 1.2 Scope

“**Retrouver-Together We Come**” is a web-based service which intends to reconnect lost people with their families. Our website will be capable of searching for an existing profile from the database which is similar to the lost person’s profile.

## 1.3 Terms, Definitions, Acronyms and Abbreviations

## 1.3.1 Definitions, Acronyms, and Abbreviations.

● **E-R diagram**-It stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases.

● **Activity Diagram**- Activity Diagram illustrate the dynamic nature of a system by modelling the flow of control form activity to activity.

● **Sequence Diagram**- UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

**● Use Case Diagram**- A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

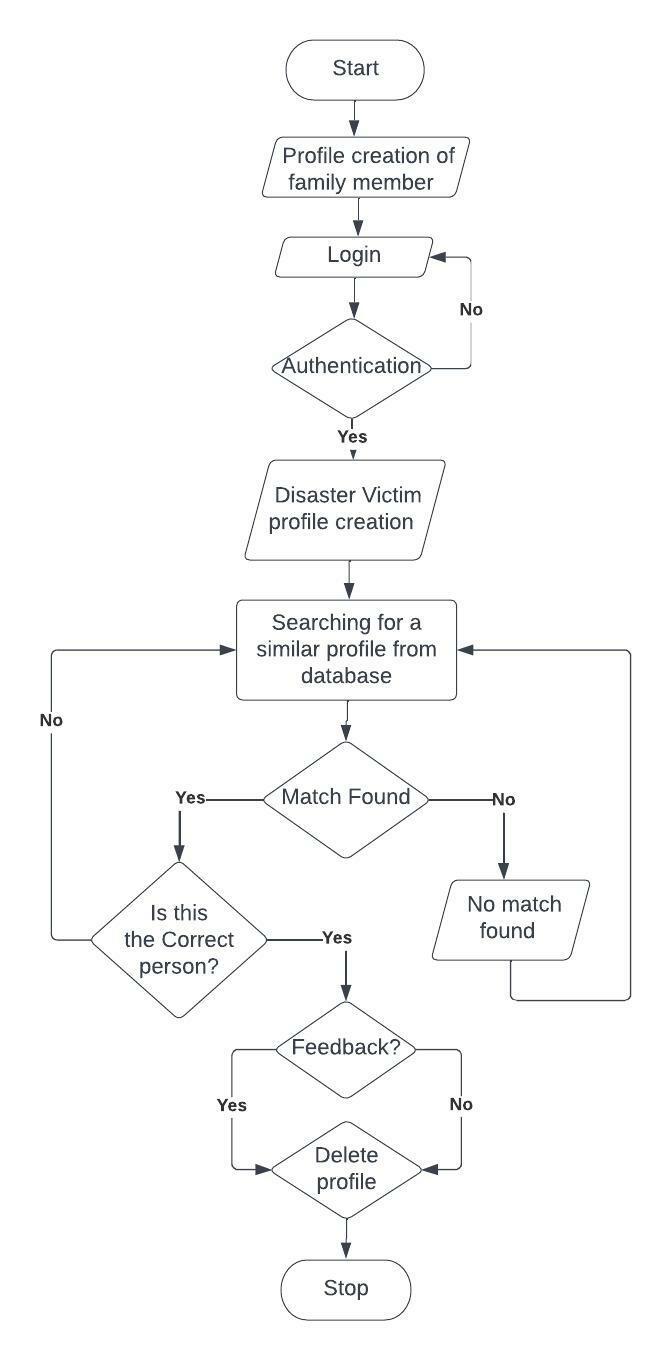
## 1.3.2 Acronyms and Abbreviations.

● SDS: Software Design Specifications

● SRS: Software Requirements Specification

**2. System Architectural Design**

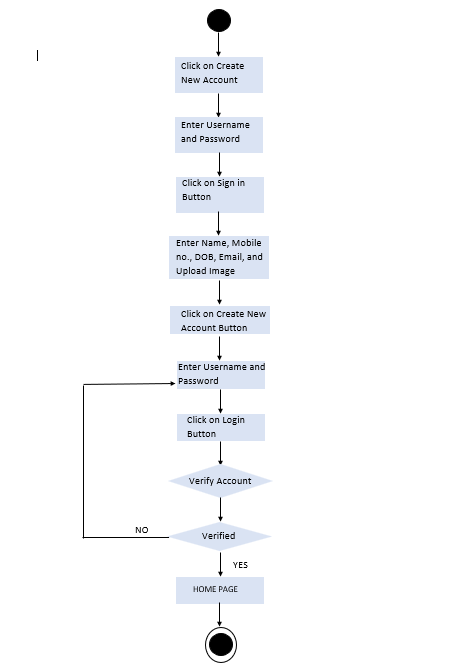
**2.1 Component and processing detail**



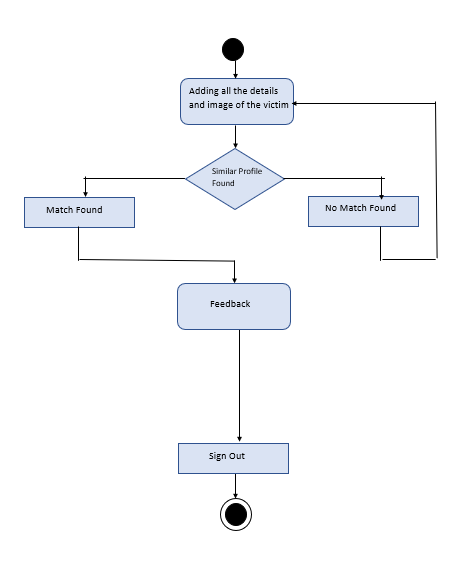
**2.2 Structure and relationships**

**ACTIVITY DIAGRAMS:**

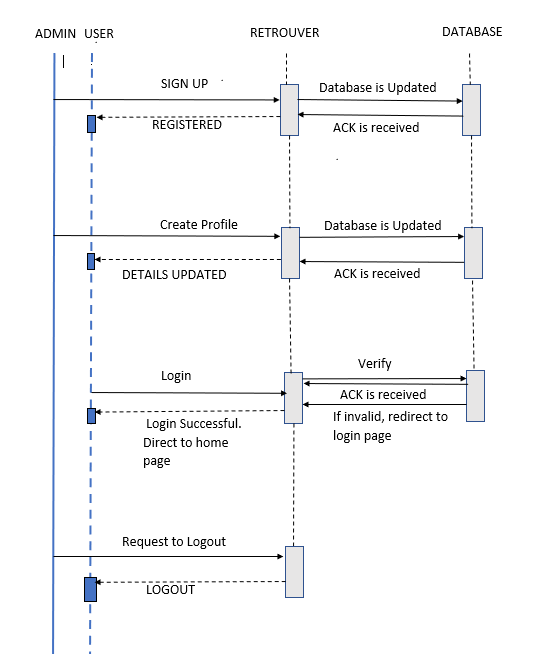
* **Login/Profile Creation**



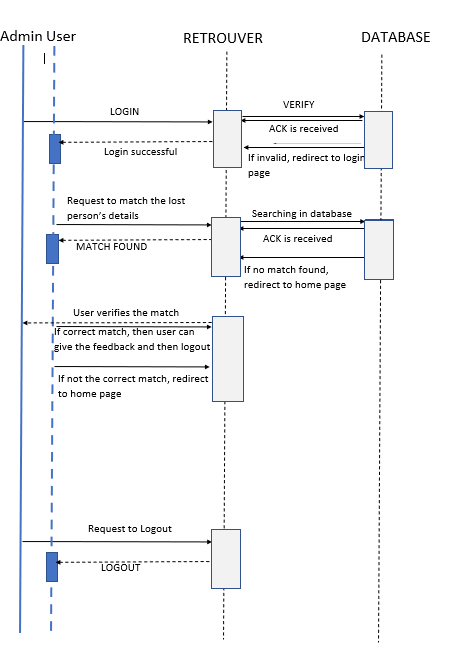
* **Profile Searching**



**SEQUENCE DIAGRAM FOR REGISTRATION AND PROFILE CREATION**



**SEQUENCE DIAGRAM FOR PROFILE MATCHING**



**3.Data Design**

**3.1 Database Description**

* **User\_details**

| **Fields** | **Data-Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| Username | Varchar(50) | Username of the user | Primary Key |
| Password | Varchar(50) | Password of the user | Not Null |
| User\_fname | Varchar(50) | First name of user | Null |
| User\_Lname | Varchar(50) | Last name of user | Null |
| User\_Gender | Varchar(10) | Gender of user | Null |
| User\_Contact | Varchar(14) | Contact no. of user | Null |
| User\_DOB | date | Date of birth of user | Null |
| User\_Email | Varchar(255) | Email address of user | Null |

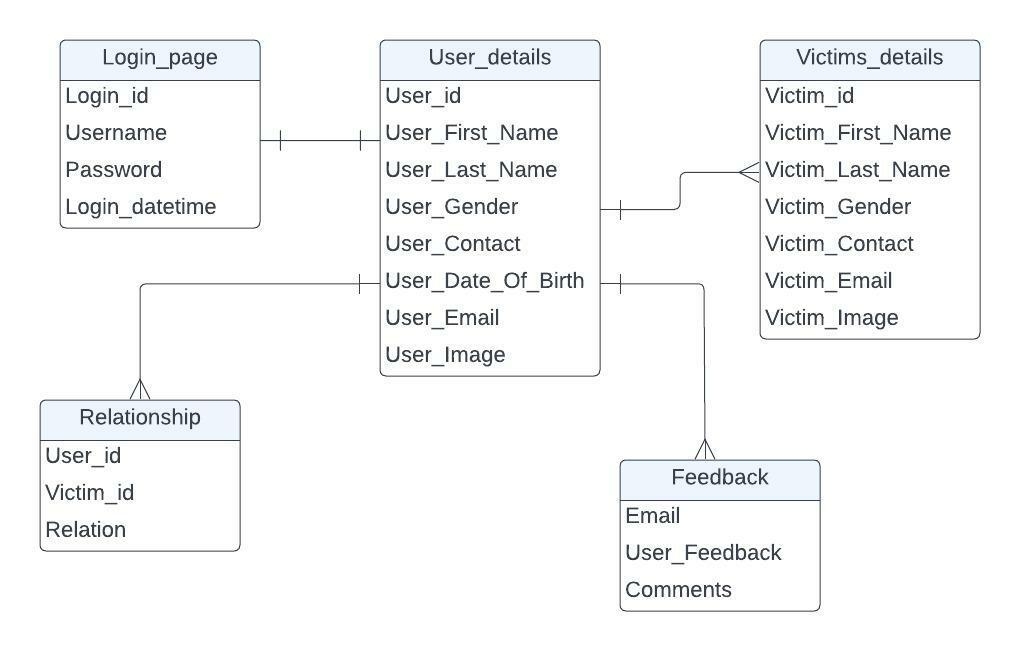
* **Victim\_details**

| **Fields** | **Data-Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| Username | Varchar(50) | Username of user | Foreign Key |
| Victim\_Fname | Varchar(50) | First name of victim | Null |
| Victim\_Lname | Varchar(50) | Last name of victim | Null |
| Victim\_Contact | Varchar(14) | Contact of victim | Null |
| Victim\_DOB | date | Date of birth of victim | Null |
| Victim\_Email | Varchar(255) | Email address of victim | Null |
| Victim\_Gender | Varchar(10) | Gender of victim | null |

* **Feedback**

| **Fields** | **Data-Type** | **Description** | **Constraints** |
| --- | --- | --- | --- |
| Username | Varchar(50) | Username of user | Foreign Key |
| User\_feedback | Varchar(20) | Feedback of user | Null |
| Comments | Varchar(100) | Comments of user | Null |

**3.2 E-R Diagram (Crow-Notation)**

**

Appy.py

from flask import Flask, flash, render\_template, request, redirect, url\_for, session

from flask\_mysqldb import MySQL

from werkzeug.utils import secure\_filename

import MySQLdb.cursors

import re

import os

import cv2

import numpy as np

import face\_recognition

app = Flask(\_\_name\_\_)

app.config["IMAGE\_UPLOADS"] = "C:/Users/hp/OneDrive/Desktop/Retrouver/static/user\_images"

app.config["IMAGE\_UPLOADV"] = "C:/Users/hp/OneDrive/Desktop/Retrouver/static/victim\_images"

app.secret\_key = 'xyzsdfg'

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'root'

app.config['MYSQL\_PASSWORD'] = ''

app.config['MYSQL\_DB'] = 'retrouver'

mysql = MySQL(app)

**@app.route**('/')

**@app.route**('/login', methods =['GET', 'POST'])

def **login**():

    mesage = ''

    if request.method == 'POST' and 'uname' in request.form and 'password' in request.form:

        uname = request.form['uname']

        password = request.form['password']

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        cursor.execute('SELECT \* FROM user\_details WHERE Username = % s AND Password = % s', (uname, password, ))

        user = cursor.fetchone()

        if user:

            session['loggedin'] = True

            session['username'] = uname

            mesage = 'Logged in successfully !'

            return redirect(url\_for('victim'))

        else:

            mesage = 'Please enter correct email / password !'

    return render\_template('login.html', mesage = mesage)

**@app.route**('/logout')

def **logout**():

    session.pop('loggedin', None)

    session.pop('userid', None)

    session.pop('email', None)

    return redirect(url\_for('login'))

**@app.route**('/register', methods=['GET', 'POST'])

def **register**():

    message = ''

    if request.method == 'POST' and 'name' in request.form and 'password' in request.form:

        username = request.form['name']

        password = request.form['password']

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        cursor.execute('SELECT \* FROM user\_details WHERE Username = %s', (username,))

        account = cursor.fetchone()

        if account:

            message = 'Account already exists!'

        elif not username or not password:

            message = 'Please fill out the form!'

        else:

            cursor.execute('INSERT INTO user\_details VALUES (%s, %s, NULL, NULL, NULL, NULL, NULL, NULL)', (username, password))

            mysql.connection.commit()

            message = 'You have successfully registered!'

            session['username'] = username *# Store the username in the session*

            return redirect(url\_for('user')) *# Redirect to the user.html template*

    elif request.method == 'POST':

        message = 'Please fill out the form!'

    return render\_template('register.html', message=message)

**@app.route**('/user', methods=['GET', 'POST'])

def **user**():

    message = ''

    username = None

    if 'username' in session:

        username = session['username']

    if request.method == 'POST' and 'fname' in request.form and 'lname' in request.form:

        image = request.files['file']

        if image.filename == '':

**print**("Filename is invalid")

            return redirect(request.url)

        username = session['username']

        basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

        filename = secure\_filename(username + '.jpg') *# Adding username to the filename and setting the extension to '.jpg'*

        image.save(os.path.join(basedir, app.config["IMAGE\_UPLOADS"], filename))

        fname = request.form['fname']

        lname = request.form['lname']

        phone = request.form['phone']

        email = request.form['email']

        gender = request.form['gender']

        dob = request.form['dob']

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        cursor.execute('SELECT \* FROM user\_details WHERE Username = %s', (username,))

        account = cursor.fetchone()

        if not lname or not fname or not phone or not email or not gender:

            message = 'Please fill out the form!'

        else:

            cursor.execute('UPDATE user\_details SET User\_fname = %s, User\_lname = %s, User\_gender = %s, User\_contact = %s, User\_DOB = %s, User\_email = %s WHERE username = %s', (fname, lname, gender, phone, dob, email, username, ))

            mysql.connection.commit()

            message = 'You have created wuhuuu!'

            return redirect(url\_for('victim'))

    elif request.method == 'POST':

        message = 'Please fill out the form!'

    return render\_template('user.html', message=message, username=username)

**@app.route**('/victim', methods=['GET', 'POST'])

def **victim**():

    message = ''

    filename = ''

    username = None

    if 'username' in session:

        username = session['username']

    idname=""

    if request.method == 'POST' :

        image = request.files['file']

        if image.filename == '':

**print**("Filename is invalid")

            return redirect(request.url)

        username = session['username']

        basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

        filename = secure\_filename(username + '\_'+image.filename) *# Adding username to the filename and setting the extension to '.jpg'*

        image.save(os.path.join(basedir, app.config["IMAGE\_UPLOADV"], filename))

        session['filename'] = filename

        fname = request.form['fname']

        lname = request.form['lname']

        phone = request.form['phone']

        email = request.form['email']

        gender = request.form['gender']

        dob = request.form['dob']

        path='static/user\_images'

        images=[]

        classNames=[]

        myList=os.listdir(path)

*#print(myList)*

        for cl in myList:

            curImg=cv2.imread(f'{path}/{cl}')

            images.append(curImg)

            classNames.append(os.path.splitext(cl)[0])

*#print(classNames)*

*#print(images)*

        def **findEncodings**(images):

            encodeList=[]

            for img in images:

                img=cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

                faceEncodings = face\_recognition.face\_encodings(img)

*#print(faceEncodings)*

                if **len**(faceEncodings) > 0:

                    encode= faceEncodings[0]

                    encodeList.append(encode)

            return encodeList

        encodeListKnown=findEncodings(images)

*#print("encoding completed")*

        a="C:/Users/hp/OneDrive/Desktop/Retrouver/static/victim\_images/"

        c=a+filename

        imgTest=cv2.imread(c)

        facelocTest= face\_recognition.face\_locations(imgTest)[0]

        encodeTest= face\_recognition.face\_encodings(imgTest)[0]

        matches=face\_recognition.compare\_faces(encodeListKnown,encodeTest)

        faceDis= face\_recognition.face\_distance(encodeListKnown,encodeTest)

*#print(faceDis)*

        matchIndex=np.argmin(faceDis)

        if matches[matchIndex]:

            idname=classNames[matchIndex].upper()

            session['idname'] = idname

        else:

            message = 'No match found!'

        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

        if not lname or not fname or not phone or not email or not gender:

            message = 'Please fill out the form!'

        else:

            cursor.execute('INSERT INTO victim\_details VALUES (%s, %s, %s, %s, %s, %s, %s)', (username, fname, lname, phone, dob, email, gender, ))

            mysql.connection.commit()

            message = 'You have created wuhuuu!'

            return redirect(url\_for('match'))

    elif request.method == 'POST':

        message = 'Please fill out the form!'

    return render\_template('victim.html', message=message, username=username, filename=filename ,idname=idname)

**@app.route**('/feedback', methods=['GET', 'POST'])

def **feedback**():

    message = ''

    if request.method == 'POST' and 'name' in request.form and 'password' in request.form:

*# Code to update user details*

        pass

*# Access the username from the session*

    username = session['username']

    return render\_template('feedback.html', message=message, username=username)

**@app.route**('/match', methods =['GET', 'POST'])

def **match**():

    mesage = ''

    filename = None

    idname = None

    fname = None

    lname = None

    dob = None

    email = None

    contact = None

    gender = None

    if 'idname' in session:

        idname = session['idname']

    if 'filename' in session:

        filename = session['filename']

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

    cursor.execute('SELECT \* FROM user\_details WHERE Username = % s', (idname, ))

    user = cursor.fetchone()

    if user:

        fname = user['User\_fname']

        lname = user['User\_lname']

        gender = user['User\_gender']

        contact = user['User\_contact']

        dob = user['User\_DOB']

        email = user['User\_email']

**print**(idname)

        return render\_template('match.html', mesage=mesage, fname=fname, lname=lname, dob=dob, email=email, contact=contact, gender=gender, filename=filename ,idname=idname)

    else:

        mesage = 'Please enter correct email / password !'

    return render\_template('match.html', mesage=mesage, fname=fname, lname=lname, dob=dob, email=email, contact=contact, gender=gender, filename=filename, idname=idname)

**@app.route**('/signup')

def **signup**():

    return render\_template('register.html')

**@app.route**('/display/<filename>')

def **display\_image**(filename):

    return redirect(url\_for('static',filename = "/user\_images/" + filename), code=301)

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

    app.run(debug=True,port=8000)

Victim.html

<!DOCTYPE *html*>

<html *lang*="en" *dir*="ltr">

   <head>

      <meta *charset*="utf-8">

      <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge" />

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

      <title>Retrouver-Together We Come</title>

      <style>

        \*{

          margin: 0;

          padding: 0;

          list-style: none;

          text-decoration: none;

          box-sizing: border-box;

        font-family: "Poppins", sans-serif;

        }

        body{

          font-family: sans-serif;

          background:**linear-gradient**(120deg,#2980b9, #8e44ad);

          user-select: none;

        }

        .mg{

          min-height: 100vh;

        display: flex;

        align-items: left;

        justify-content: center;

        padding: 20px;

        background-color: red ;

        scroll-behavior: auto;

        }

        nav .logo{

          color: white;

          font-size: 33px;

          font-weight: bold;

          line-height: 70px;

          padding-left: 110px;

        }

        nav{

          height: 70px;

          background: #3B2667;

          box-shadow: 0 3px 15px **rgba**(0,0,0,.4);

        }

        nav ul{

          float: right;

          margin-right: 30px;

        }

        nav ul li{

          display: inline-block;

        }

        nav ul li a{

          color: white;

          display: block;

          padding: 0 15px;

          line-height: 70px;

          font-size: 20px;

          background: #3B2667;

          transition: .5s;

        }

        nav ul li a:hover,

        nav ul li a.active{

          color: #23dbdb;

        }

        nav ul ul{

          position: absolute;

          top: 85px;

          border-top: 3px solid #23dbdb;

          opacity: 0;

          visibility: hidden;

        }

        nav ul li:hover > ul{

          top: 70px;

          opacity: 1;

          visibility: visible;

          transition: .3s linear;

        }

        nav ul ul li{

          width: 150px;

          display: list-item;

          position: relative;

          border: 1px solid #042331;

          border-top: none;

        }

        nav ul ul li a{

          line-height: 50px;

        }

        nav ul ul ul{

          border-top: none;

        }

        nav ul ul ul li{

          position: relative;

          top: -70px;

          left: 150px;

        }

        nav ul ul li a i{

          margin-left: 45px;

        }

        .container {

         position: absolute;

         left:24%;

          max-width: 700px;

          width: 100%;

          background: #fff;

          padding: 25px;

          border-radius: 8px;

          box-shadow: 0 0 15px **rgba**(0, 0, 0, 0.1);

          align-items: left;

        }

        .container header {

          font-size: 1.5rem;

          color: #333;

          font-weight: 500;

          text-align: center;

        }

        .container .form {

          margin-top: 30px;

        }

        .form .input-box {

          width: 100%;

          margin-top: 20px;

        }

        .input-box label {

          color: #333;

        }

        .form :where(.input-box input, .select-box) {

          position: relative;

          height: 50px;

          width: 100%;

          outline: none;

          font-size: 1rem;

          color: #707070;

          margin-top: 8px;

          border: 1px solid #ddd;

          border-radius: 6px;

          padding: 0 15px;

        }

        .input-box input:focus {

          box-shadow: 0 1px 0 **rgba**(0, 0, 0, 0.1);

        }

        .form .column {

          display: flex;

          column-gap: 15px;

        }

        .form .gender-box {

          margin-top: 20px;

        }

        .gender-box h3 {

          color: #333;

          font-size: 1rem;

          font-weight: 400;

          margin-bottom: 8px;

        }

        .form :where(.gender-option, .gender) {

          display: flex;

          align-items: center;

          column-gap: 50px;

          flex-wrap: wrap;

        }

        .form .gender {

          column-gap: 5px;

        }

        .gender input {

          accent-color: **rgb**(130, 106, 251);

        }

        .form :where(.gender input, .gender label) {

          cursor: pointer;

        }

        .gender label {

          color: #707070;

        }

        .address :where(input, .select-box) {

          margin-top: 15px;

        }

        .select-box select {

          height: 100%;

          width: 100%;

          outline: none;

          border: none;

          color: #707070;

          font-size: 1rem;

        }

        .form button {

          height: 55px;

          width: 100%;

          color: #fff;

          font-size: 1rem;

          font-weight: 400;

          margin-top: 30px;

          border: none;

          cursor: pointer;

          transition: all 0.2s ease;

          background: **rgb**(130, 106, 251);

        }

        .form button:hover {

          background: **rgb**(88, 56, 250);

        }

*/\*Responsive\*/*

        @media screen and (max-width: 500px) {

          .form .column {

            flex-wrap: wrap;

          }

          .form :where(.gender-option, .gender) {

            row-gap: 15px;

          }

        }

        .wrapper {

          max-width: 550px;

          top:100px;

          height: 400px;

          background:beige;

          align-items: center;

          justify-content: center;

          position: relative;

          border-radius: 12px;

        }

        .wrapper i.button {

          position: absolute;

          top: 50%;

          transform: **translateY**(-50%);

          height: 36px;

          width: 36px;

          background-color: #343f4f;

          border-radius: 50%;

          text-align: center;

          line-height: 36px;

          color: beige;

          font-size: 15px;

          transition: all 0.3s linear;

          z-index: 100;

          cursor: pointer;

        }

        i.button:active {

          transform: **scale**(0.94) **translateY**(-50%);

        }

        i#prev {

          left: 25px;

        }

        i#next {

          right: 25px;

        }

        .image-container {

          height: 320px;

          max-width: 500px;

          width: 100%;

          overflow: hidden;

        }

        .image-container .carousel {

          display: flex;

          height: 100%;

          width: 100%;

          transition: all 0.4s ease;

        }

        .carousel img {

          height: 100%;

          width: 100%;

          border-radius: 18px;

          border: 10px solid beige;

          object-fit: cover;

        }

       .mk{

          display: right;

          width:600px;

          background:beige;

        }

        section{

          display: flex;

          background:**linear-gradient**(120deg,#2980b9, #8e44ad);

        }

        .bg{

          background:**linear-gradient**(120deg,#2980b9, #8e44ad);

        }

      </style>

      <link *rel*="stylesheet" *href*="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"/>

      <link *rel*="stylesheet" *href*="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.1/css/all.min.css" />

      <script *src*="C:\Users\hp\OneDrive\Desktop\flask\static\js\slider.js" *defer*></script>

   </head>

   <body>

      <nav>

         <label *class*="logo">Retrouver-Together We Come</label>

         <ul>

            <li><a *class*="active" *href*="#">Home</a></li>

            <li>

               <a *href*="#">Profile

               </a>

            </li>

            <li>

               <a *href*="#">About Us

               </a>

            </li>

            <li><a *href*="Feed.html">Feedback</a></li>

         </ul>

      </nav>

      <section>

           <div *class*="bg">

      <div *class*="container">

        <header>Victim Profile</header>

        <h1>Welcome {{ username }}!</h1>

        <form  *action*="{{ url\_for('victim') }}"  *class*="form" *method*="post" *enctype*="multipart/form-data" >

          {% if mesage is defined and mesage %}

          <div *class*="alert alert-warning">{{ mesage }}</div>

          {% endif %}

*<!-- <div class="input-box">*

*<label>First Name</label>*

*<input type="text" placeholder="Enter full name" required />*

*</div> -->*

          <div *class*="column">

              <div *class*="input-box">

                <label>First Name</label>

                <input *type*="text" *id*="fname" *name*="fname" *placeholder*="Enter first name" *required* />

              </div>

              <div *class*="input-box">

                <label>Last Name</label>

                <input *type*="text" *class*="form-control" *id*="lname" *name*="lname" *placeholder*="Enter last name" *required* />

              </div>

            </div>

          <div *class*="column">

            <div *class*="input-box">

              <label>Phone Number</label>

              <input *type*="number" *class*="form-control" *id*="phone" *name*="phone" *placeholder*="Enter phone number" *required* />

            </div>

            <div *class*="input-box">

              <label>Birth Date</label>

              <input *type*="date" *name*="dob" *id*="dob" *placeholder*="Enter birth date" *required* />

            </div>

          </div>

          <div *class*="input-box">

              <label>Email Address</label>

              <input *type*="text" *name*="email" *id*="email" *placeholder*="Enter email address" *required* />

            </div>

          <div *class*="gender-box">

            <h3>Gender</h3>

            <div *class*="gender-option">

              <div *class*="gender">

                <input *type*="radio" *class*="form-control" *id*="check-male" *name*="gender" *value*="male" *checked* />

                <label *for*="check-male">Male</label>

              </div>

              <div *class*="gender">

                <input *type*="radio" *class*="form-control" *id*="check-female" *name*="gender" *value*="female"/>

                <label *for*="check-female">Female</label>

              </div>

              <div *class*="gender">

                <input *type*="radio" *class*="form-control" *id*="check-other" *name*="gender" *value*="other"/>

                <label *for*="check-other">Other</label>

              </div>

              <div *class*="input-box">

                <label>Image</label>

                <input *type*="file" *name*="file" *class*="form-control" *autocomplete*="off" *required* />

              </div>

            </div>

          </div>

          <button *type*="submit" *class*="btn btn-primary" >Search</button>

*<!-- <button>Submit</button> -->*

        </form>

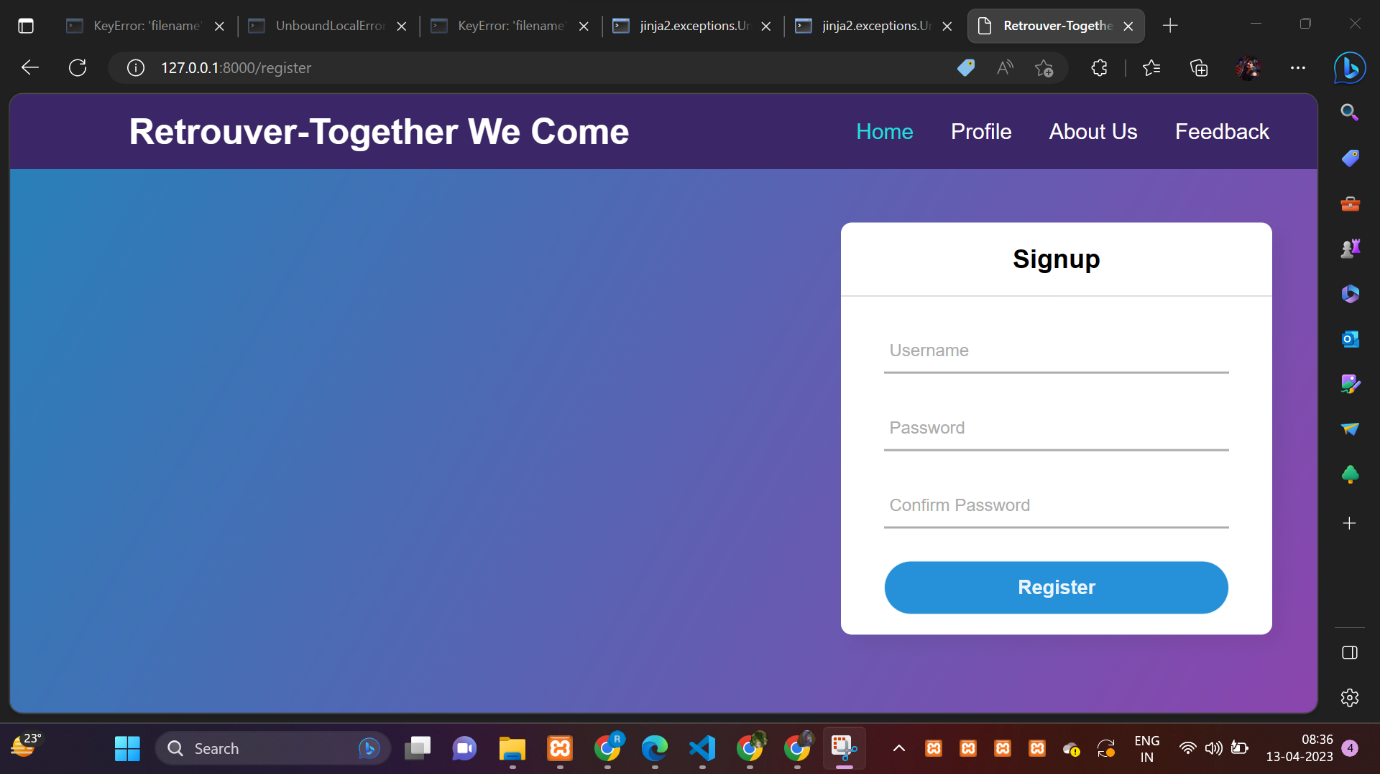
    </div></div></section>

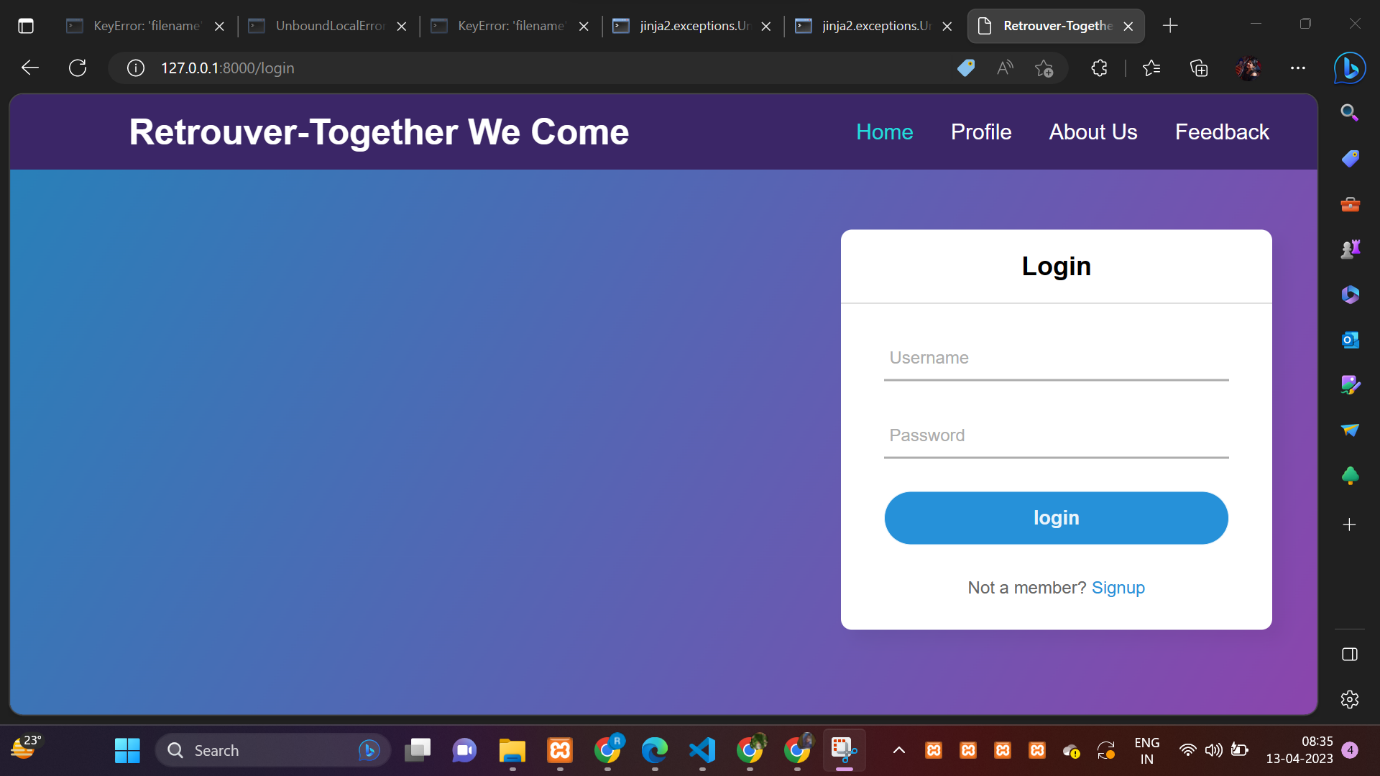
   </body>

</html>

**Layout**

* **Sign Up page**



* **Login page**

### User Profile Page

### 

### Lost Person Details Page

### 

### Feedback Page

### 

### About Us Page

### 

## *5.References*

1. Pressman Roger S., Software Engineering “A Practitioner’s Approach”

Fifth Edition, McGraw-Hill Publication, 2000.

1. Navathe Shamkant B.*,* Fundamentals of Database Systems,

Fifth Edition, Pearson Publication*.*

1. IEEE STD 830-1998, IEEE Recommended Practice for Software Requirement

|  |
| --- |

|  |
| --- |

1. <https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/opencv-csharp#here_are_some_external_uses_of_opencv>
2. <https://pyimagesearch.com/2021/05/10/opencv-eigenfaces-for-face-recognition/>
3. <https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html>
4. <https://www.researchgate.net/figure/Class-diagram-of-the-entities-that-compose-a-recognition-system-FaceLab-provides-some_fig2_228966022>
5. <https://itsourcecode.com/uml/sequence-diagram-for-face-recognition-attendance-system-uml/>
6. <https://www.guru99.com/er-diagram-tutorial-dbms.html>
7. <https://www.mygreatlearning.com/blog/face-recognition/>