**KARNATAK UNIVERSITY**

**DHARWAD**

**Janata Shikshana Samiti’s**

**Shri Manjunatheshwara Institute of UG & PG Studies Vidyagiri, Dharwad-580004.**

****

**A PROJECT REPORT ON**

**“IMAGE FORGERY DETECTION”**

**BACHELOR OF COMPUTER APPLICATION**

**OF**

**KARNATAK UNIVERSITY, DHARWAD**

**PROJECT GUIDED BY:**

**Miss. Ankita Shetty**

**Submitted By:**

**Bhavana Bhat Shwetha Hegde**

**BCA VI SEMESTER BCA VI SEMESTER**

**REG NO:20U11016 REGNO:20U11134**

**DEPARTMENT OF COMPUTER SCIENCE**

**2022-2023**

**Janata Shikshana Samiti’s**

**Shri Manjunatheshwara Institute of UG & PG Studies,**

**Vidyagiri, Dharwad. 580004**



**CERTIFICATE**

**This is to certify that Miss. Bhavana D. Bhat and Miss. Shwetha Hegde**

**hassatisfactorily complete project work entitled Image Forgery Detection**

**for the partial fulfilment of BCA prescribed by Karnataka University Dharwad during the academic year 2022-2023*.***

**Miss. Ankita Shetty Prof. Vivek M. Laxmeshwar Dr. Ajith Prasad**

**Project Guide H.O.D. Computer Department Principal**

**Examination Register No: Examiner’s 1. 20U11016 1. 2. 20U11134 2.**

**ACKNOWLEDGEMENT**

**The successful presentation of this project is acknowledgements of the immense support extended by JSS SMI UG & PG STUDIES, DHARWAD, which has provided us opportunity to fulfil the most cherished & desired way to reach our goal*.***

**We would like to express our heartfelt thanks to our President Shri Shri Vishwaprasanna Theertha Swamiji of Sri Pejavarmath, Udupi, Parama Pujya Dr. D. Veerendra Heggadeji, the Chairman of Janata Shikshana Samiti & Dharmadhikari of Shree Kshetra Dharmasthala.**

**We would like to express our sincere and hearty thanks of gratitude to our beloved Secretary of Janata Shikshana Samiti Dr. Ajith Prasad and our Head of the Computer Science Department Prof. Vivek M. Laxmeshwar who gave us the golden opportunity to do this wonderful project on the topic “IMAGE FORGERY DETECTION”, which also helped us in doing a lot of research and exposed to lot of new information that would help us in our mere future.**

**We would also take this opportunity to offer our sincere gratitude to our Project guide Miss. Ankita Shetty for her excellent support throughout the development of this project and providing the necessary information on our demand at all the times.**

**Bhavana Bhat & Shwetha Hegde**

DECLARATION

We, **Bhavana Bhat and Shwetha Hegde** students of Sixth Semester BCA, Department of Computer Science, JSS SHREE MANJUNATHESHWARA INSTITUTE OF UG & PG STUDIES, VIDYAGIRI, DHARWAD affiliated to Karnataka University, Dharwad hereby declare that the Project entitled **“IMAGE FORGERY DETECTION”** submitted in partial fulfilment of the course requirement for the award of degree in Bachelor of Computer Application, Karnataka University, Dharwad during the academic year 2022-2023. We have not submitted the matter embodied to any other university or institution for the award of any other degree.

**Date: Bhavana Bhat**

**Place: Dharwad Shwetha Hegde**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| SNO | TITLE | PAGE NO |
| 1 | Project Synopsis | 1-3 |
| 2 | Framework   * Platform * Python, Django, CSS,Jvascrip,MySQL & XAMPP? | 4-12 |
| 3 | Software Requirement Specification (SRS)   * System Requirements * Specification | 13-17 |
| 4 | Design of the Product   * + - * + DFD         + ER DIAGRAM         + Use-case diagram | 18-24 |
| 5 | Implementation | 25-26 |
| 6 | Database Table | 27-29 |
| 7 | Screenshots and Source code | 30-45 |
| 8 | System Testing & Resulting | 46-50 |
| 9 | Conclusion | 51 |
| 10 | Bibliography | 52 |

**1.PROJECT SYNOPSIS**

Image Forgery Detection

Following the explosion of social networking services, there has been a monumental increase in the volume of image data. Moreover, the development in image processing software such as Adobe Photoshop has given a rise to doctored images. Such doctored images can be used for malicious purposes such as spreading false information and inciting violence. This image forgery detection project allows users to detect even the slightest signs of forgery in an image.

**1.OBJECTIVE OF THE PROJECT:**

* Useful to detect doctored images or signs of forgery in photos.
* Easy to use.
* Maintains security by ensuring only Admin can see the results of the image analysis.

**2.INPUT OF THE PROJECT:**

* Admin Details
* User Details
* Original image
* Duplicate image

**3.OUTPUT OF THE PROJECT:**

* Result of image comparison

**4.PROCESS LOGIC:**

Input Process Output

Result of image comparison

Admin Details

User Details

Original image

Duplicate image

**5.TOOLS/PLATFORM,LANGUAGE TO BE USED:**

1.Hardware requirements:

* Processor –Core i3
* Hard Disk – 160 GB
* Memory – 1GB RAM
* Monitor

2.Software requirements:

* + Windows 7 or higher
  + Python
  + Django framework
  + MySQL database

**6.DURATION OF THE PROJECT:**

* 2 months

**7.MEMBERS OF THE PROJECT:**

* Bhavana Bhat - 20U11016
* Shwetha hegde - 20U11134

**8.LIMITATION OF THE PROJECT:**

* If image is not clear and has a threshold below 40 (in terms of color range and lightning) and higher that 255, then detection will not happen correctly.
* Black and white images must be first converted to color form, which increases the time taken for detection.

**9.SCOPE OR FUTURE APPLICATIONS:**

* Black and white images can be detected without converting them into color.
* Mobile applications can be developed.

**2.FRAMEWORK**

**1.Python:**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

* Python was developed by **Guido van Rossum** in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
* Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.
* At the time when he began implementing Python, Guido van Rossum was also reading the published scripts from "Monty Python's Flying Circus"). It occurred to him that he needed a name that was short, unique, and slightly mysterious, so he decided to call the language Python.
* Python is now maintained by a core development team at the institute, although **Guido van Rossum** still holds a vital role in directing its progress.
* Python 1.0 was released on **20 February, 1991.**
* Python 2.0 was released on **16 October 2000** and had many major new features, including a cycle detecting garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community- backed.
* Python 3.0 a major, backwards-incompatible release, was released on **3 December 2008** after a long period of testing. Many of its major features have been back ported to the backwards-compatible Python 2.6.x and 2.7.x version series.
* In January 2017 Google announced work on a Python 2.7 to go trans compiler, which The Register speculated was in response to Python 2.7's planned end-of-life.

**Python Features:**

Python's features include:

* **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
* **A broad standard library:** Python's bulk of the library is very portable and cross- platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases:** Python provides interfaces to all major commercial databases.
* **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX.
* **Scalable:** Python provides a better structure and support for large programs than shell scripting.

**Applications of Python:**

1. Systems Programming

2. GUIs

3. Internet Scripting

4. Component Integration

5. Database Programming

6. Rapid Prototyping

7. Numeric and Scientific Programming

**What Are Python’s Technical Strengths?**

1. It‘s Object-Oriented and Functional

2. It‘s Free

3. It‘s Portable

4. It‘s Powerful

5. It‘s Mixable

6. It‘s Relatively Easy to Use

7. It‘s Relatively Easy to Learn

**2.django**

Django is a Python-based web framework which allows you to quickly create web application without all of the installation or dependency problems that you normally will find with other frameworks. When you’re building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use.

#### Why django?

* Django is a rapid web development framework that can be used to develop fully fleshed web applications in a short period of time.
* It’s very easy to switch database in Django framework.
* It has built-in admin interface which makes easy to work with it.
* Django is fully functional framework that requires nothing else.
* It has thousands of additional packages available.

**3**.**CSS-Cascading Style Sheet:**

CSS stands for Cascading Style Sheets.CSS describes how HTML elements are to be displayed on screen, paper, or in other media.CSS saves a lot of work. It can control the layout of multiple web pages all at once.External stylesheets are stored in CSS files.

**SYNTAX:**

A CSS rule consists of a selector and a declaration block.



The selector points to the HTML element you want to style. The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon. Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

### Example

In this example all <p> elements will be center-aligned, with a red text color:

p {  
  color: red;  
  text-align: center;  
}

Here, p is a selector in css (it points to the html element you want to style: <p>). Color is a property, and red is the property value text-align is a property, and center is the property value.

There are three types of css which are given below:

1. Inline css
2. Internal css
3. External css

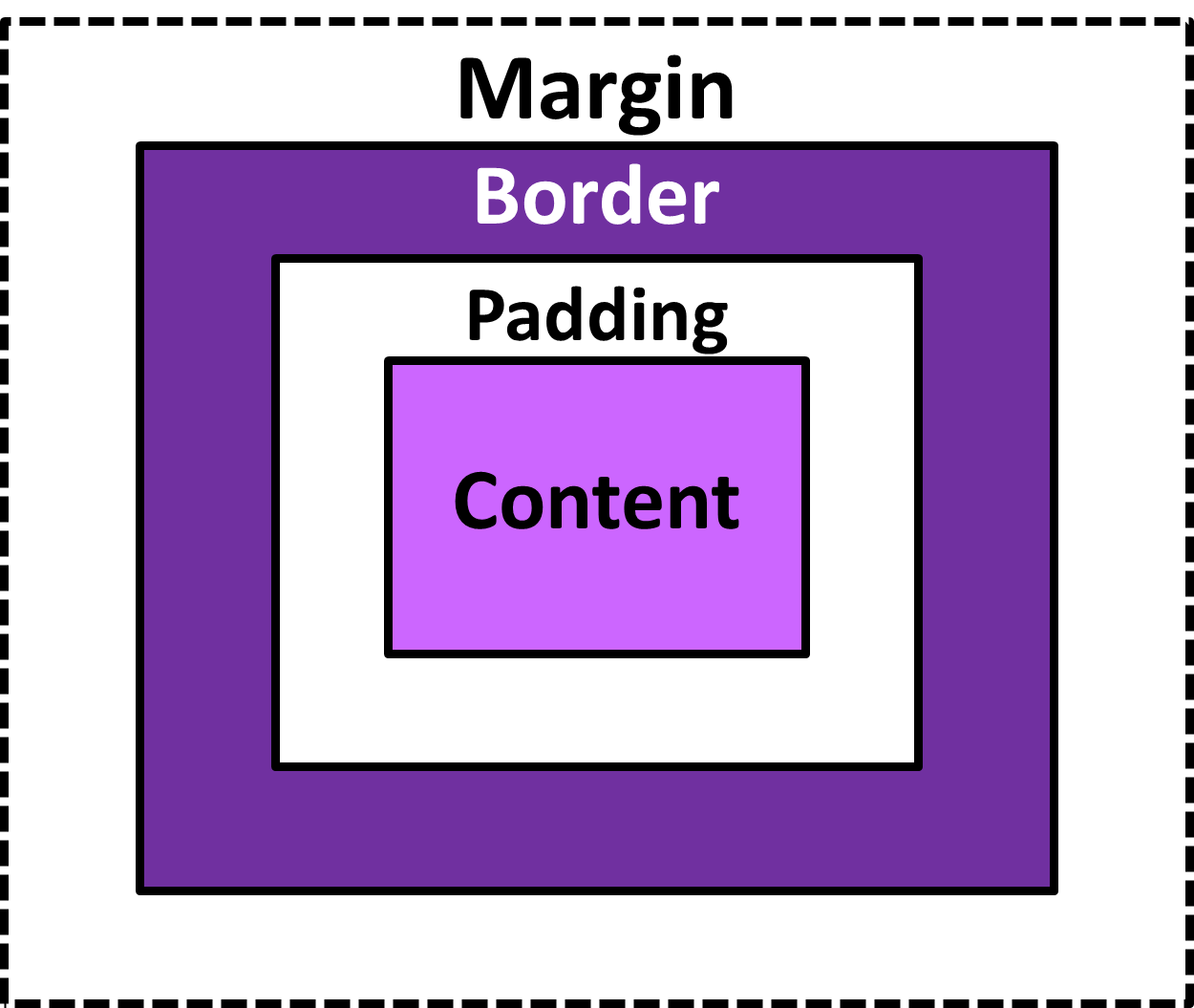
**Inline css:** Inline css contains the css property in the body section attached with element is known as inline css. This kind of style is specified within an html tag using the style attribute.

**External CSS:** External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, … etc). CSS property written in a separate file with .css extension and should be linked to the HTML document using **link** tag.

**CSS BOX MODEL:**

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

* In CSS, the term "box **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent model" is used when talking about design and layout.



**4.JAVASCRIPT:**

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

**HISTORY:**

In 1993, **Mosaic**, the first popular web browser, came into existence. In the **year 1994**, **Netscape** was founded by **Marc Andreessen**. He realized that the web needed to become more dynamic. Thus, a 'glue language' was believed to be provided to HTML to make web designing easy for designers and part-time programmers. Consequently, in 1995, the company recruited **Brendan Eich** intending to implement and embed Scheme programming language to the browser.

But, before Brendan could start, the company merged with **Sun Microsystems** for adding Java into its Navigator so that it could compete with Microsoft over the web technologies and platforms. Further, Netscape decided to give a similar name to the scripting language as Java's. It led to 'Javascript'. Finally, in May 1995, Marc Andreessen coined the first code of Javascript named '**Mocha**'. Later, the marketing team replaced the name with '**LiveScript**'. But, due to trademark reasons and certain other reasons, in December 1995, the language was finally renamed to 'JavaScript'. From then, JavaScript came into existence.

**APPLICATIONS:**

* JavaScript is used to create interactive websites. It is mainly used for:
* Client-side validation,
* Dynamic drop-down menus,
* Displaying date and time,
* Displaying pop-up windows and dialog
* Displaying clocks etc.

**Example:**

<script>

document.write("Hello JavaScript by JavaScript");

</script>

**5.MY SQL:**

MySQL is the world’s most popular open source database. According to [DB-Engines](https://db-engines.com/en/ranking), MySQL ranks as the second-most-popular database, behind [Oracle Database](https://www.oracle.com/database/). MySQL powers many of the most accessed applications, including Facebook, Twitter, Netflix, Uber, Airbnb, Shopify, and Booking.com.

Since MySQL is open source, it includes numerous features developed in close cooperation with users over more than 25 years. So it’s very likely that your favorite application or programming language is supported by MySQL Database.

**How to pronounce “MySQL” :**

“My ess-cue-el” is the “official” way to pronounce “MySQL,” but pronouncing it “my sequel” is common too.

**What is SQL?:**

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

**What is RDBMS?**

* RDBMS stands for Relational Database Management System.
* RDBMS is a program used to maintain a relational database.
* RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.
* RDBMS uses SQL queries to access the data in the database.

## What Can SQL do?

* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database

## SQL is …

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

However, to be compliant with the ANSI standard, they all support at least the major commands

(such as**SELECT, UPDATE, DELETE, INSERT, WHERE**) in a similar manner.

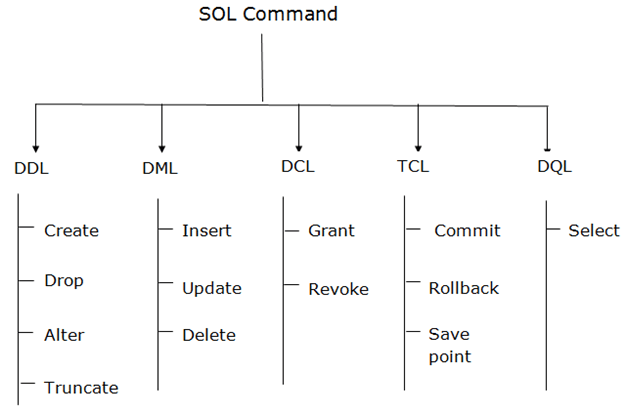
**SQL Commands:**

SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.

SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

**Types of SQL Commands:**

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



### ****Applications of SQL****

Here, we will see some applications of SQL that render it so valuable in a data-driven world where governing massive databases is the criterion of the day.

* SQL is used as aData Definition Language (DDL) which means you can autonomously make a database, characterize its structure, use it and afterwards dispose of it when you are finished with it. It is additionally conveyed as a **Data Control Language (DCL)** which determines how you can ensure your information base against debasement and misuse.
* SQL is used as aData Manipulation Language (DML)which implies you can use it for keeping up a previously existing database. Hence, it is an incredible language for entering information, changing information, and separating information with respect to a database.
* It is broadly used as aClient or Server languageto interface the front-end with the back-end consequently supporting the customer or worker architecture. Likewise, it can be used in the three-level design of a customer, an application worker, and a database which characterizes the Internet architecture.

**3.SOFTWARE REQUIREMENT SPECIFICATION**

**Introduction:**

SRS is the official statement of what is required by the system developers. It includes both user requirements for the system and detailed specification of the system requirements. This document is used while designing the proposed system and can also be used in the future if the system is to be enhanced.

**Purpose:**

This SRS document presents a detailed description of the **“IMAGE FORGERY DETECTION”.** This is client based website, that defines the functional and non-functional requirements of the labour website and its different functionalities. This system allows the user to register as the labour and it also allow the user to search and book the required labours.

**Conventions**:

* Main Section Title: Font: Times New Roman, Bold: Size 20
* Sub Section Title: Font: Times New Roman, Bold: Size 16
* Other Text Matter: Font: Times New Roman, Bold: Size 13

**Scope:**

This document is intended for providing an abstract overview of the system and general overview of the entire project.

The scope of the document:

* Gives the platform for admin to upload original image and compare those images with user uploaded image.
* User can upload images and view the comparision result.

**References:**

* https://www.w3schools.com/
* https://www.youtube.com/
* https://stackoverflow.com/
* https://openai.com/

**General Description:**

This section will give an overview of the whole application. The explanation of the application will be in its context to show the application interacts with other systems and introduce the basic functionality of it. It will also describe all the constraints and assumptions for the application.

**Product perspective:**

* This project represents the initial version of the Image Forgery Detection.
* Admin uploads the orginal images.
* Allows registered user to upload the image to be compared with original.
* Admin compares the original image and user upladed image.
* Comparion result will be displayed for both user and admin.

**Product functionality:**

Image forgery detection must support following conditions:

**Registration:**

* User must register to this website to create his account.

**Login:**

After registration login through email and password

* User can upload image, can edit his/her password, can see comparison result, can give feedback.

**Forgot password:**

* If the user forgot his/her password, his password will be sent to mail by authenticating him.

**Image upload:**

* There should be an option for user to upload image to be compared.

**View result:**

* Result of image comparision must be visible to user.

**User characteristics:**

**User:**

* User can upload image, can edit his/her password, can see comparison result, can give feedback.

**Admin:**

* Admin can upload original images and he can compare it with the user uploaded images.
* Admin can view users and view feedback given by the user.

**Specific requirements:**

**Functional requirements:**

**a) Home page:**

* in this page there will be 2 options.
* One is user registration.
* Another on is user login.

**b) Register:**

* If the user want to creat an accout in this website he must register to this website.
* Name of the user, address , phone number, email, and password are required.

**c) Login:**

* This section is mandatory for both user and admin.
* Email and password must match.

**d) Forgot password:**

* If the user forgot his password, he has to click on the forgot password link and he has to enter his email then he got his password through email.

**e) Admin Home:**

* The Admin can upload original image.
* Admin can compare user image with original image.
* Admin can view users details.
* Admin can view feedback given by the users.

**g) User Home:**

* User can upload images.
* User can view image comparision result.
* User can change his password.
* User can give feedback.

**h)Logout:**

User and Admin can logout from page. Session will end.

**Other non-functional requirement:**

**a) Performance Requirements**

* Should run on 500MHz, 64bit machine.
* 90% of responses should be within 3 seconds.

**b) Security Requirements:**

* This application requires a Name and password to verify the identification of the user.

**External interface requirements:**

**a) User Interfaces:**

* Most user-friendly interface has been designed for the main page. The user can search for the available labour and book them.

**b) Hardware Interfaces:**

* The system does not require any additional hardware interfaces, so the user need not focus on the hardware apart from the standard hardware.

**c) Software Interfaces:**

* XAAMP server, internet browser installed on the server machine.

**4.DESIGN OF THE SYSTEM**

**Introduction:**

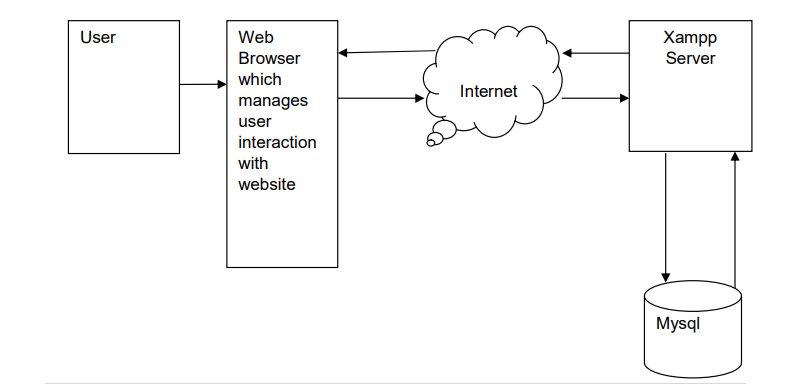
The purpose of the decision phase is to plan a solution of the problem specified by the requirements document. This phase is the first step in moving the problem domain to the solution domain. It involves the process, in which conceiving, planning and carrying out the plan generating the necessary report, In other words, the design phase act as a bridge between SRS and implementation phase. The design of the system is perhaps the most critical factor affecting the quality of the software, and as a major impact on the later phase, particularly the testing and maintenance.

**Software Design:**

Design is the key phase of any project. It is the first step in moving from the problem domain to the solution domain. The input to the design phase is the specifications of the system to be designed. The output of the top-level design is the architectural design, or the system design for the software system to be built. A design should be very clear, verifiable, complete, traceable, efficient and simple.

**Architecture Design**:

The architecture design defines the relationship among major structural element of the program. Architecture diagram shows the relationship between different components of system. This diagram helps to understand the overall concept of system.



**Logical design:**

The graphical representation of system data and how the process transforms the data is known as Data Flow Diagram. It shows the logical flow of the data. The logical design describes the detailed specification for the system, describing its features, an effective communication and the user interface requirements.

The logical design of proposed system should include the following.

* External system structure.
* Relationship between all the activities.
* The physical construction and all the activities.
* Global data.
* Control flow.
* Derived program structure.

**Design Principles:**

Basic design principles that enable the software engineer to navigate the design process are:

* The design process should not suffer from “Tunnel vision”.
* The design should be traceable to analysis model.
* The design should not reinvent the wheel.
* The design should minimize the intellectual distance between the software and the problem, as it exists in the real world.
* The design should exhibit uniformity and integrity.
* The design should be structured to accommodate changes.
* The design is not coding and coding is not design.
* The design should be reviewed to minimize the conceptual errors.

**Data Flow Diagram:**

The data flow diagram (DFD) is one of the important modelling tools. It shows the user of the data pictorially. DFD represents the flow of the data between different transformations and processes in the systems. The data flow diagram shows logical flow of the data. It represents the functional dependencies within a system. It shows output values in a computation or derived from input values. It is a simple pictorial representation or model for system behaviour. It specifies, “What is to be done but not how is to be done”. It describes the logical structure of the system. It relates data information to various processes of the system. It follows top-down approach.

**Data Flow Diagram Notations:**

**Data Flow:**

* It may be from file-to-file or file of attributes.
* There may be either an input data flow or output data flow.

**Functional process:**



* The process is nothing but the transformation of data.
* It starts with the subject and has the verb followed by the subject.

**Data store:**

****

* It includes file, data base and repository.

**Actor/source/sink:**

****

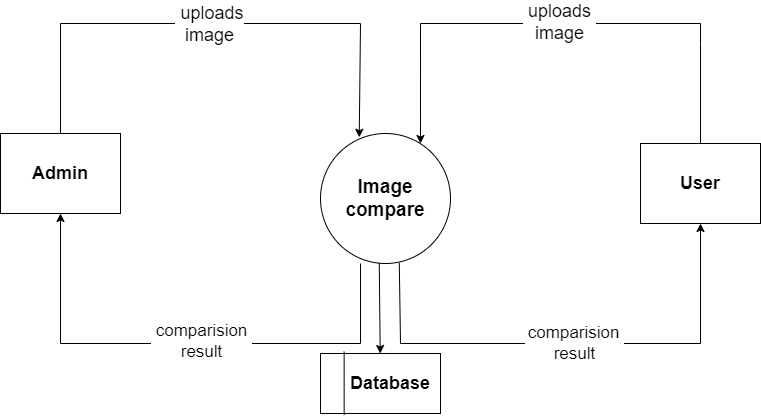
* The files which are outside the system and used by the process or processes of the system.

**Objectives:**

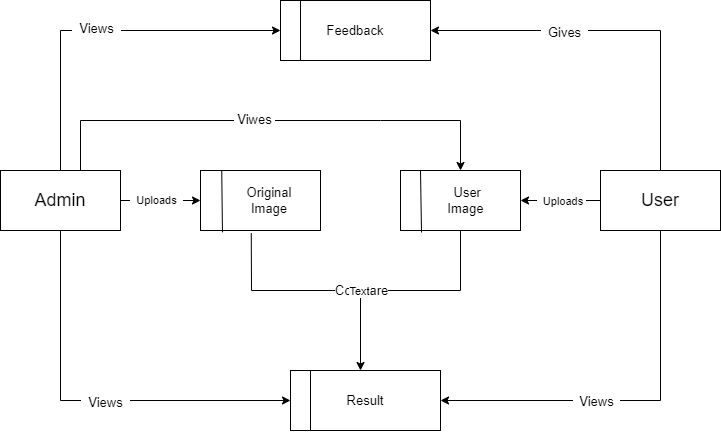
* To graphically document boundaries of a system.
* To provide hierarchy breakdown of the system.
* To show movement of information between a system and its environment.
* To document information flows within the system.

**Context level diagram (zero level DFD):**

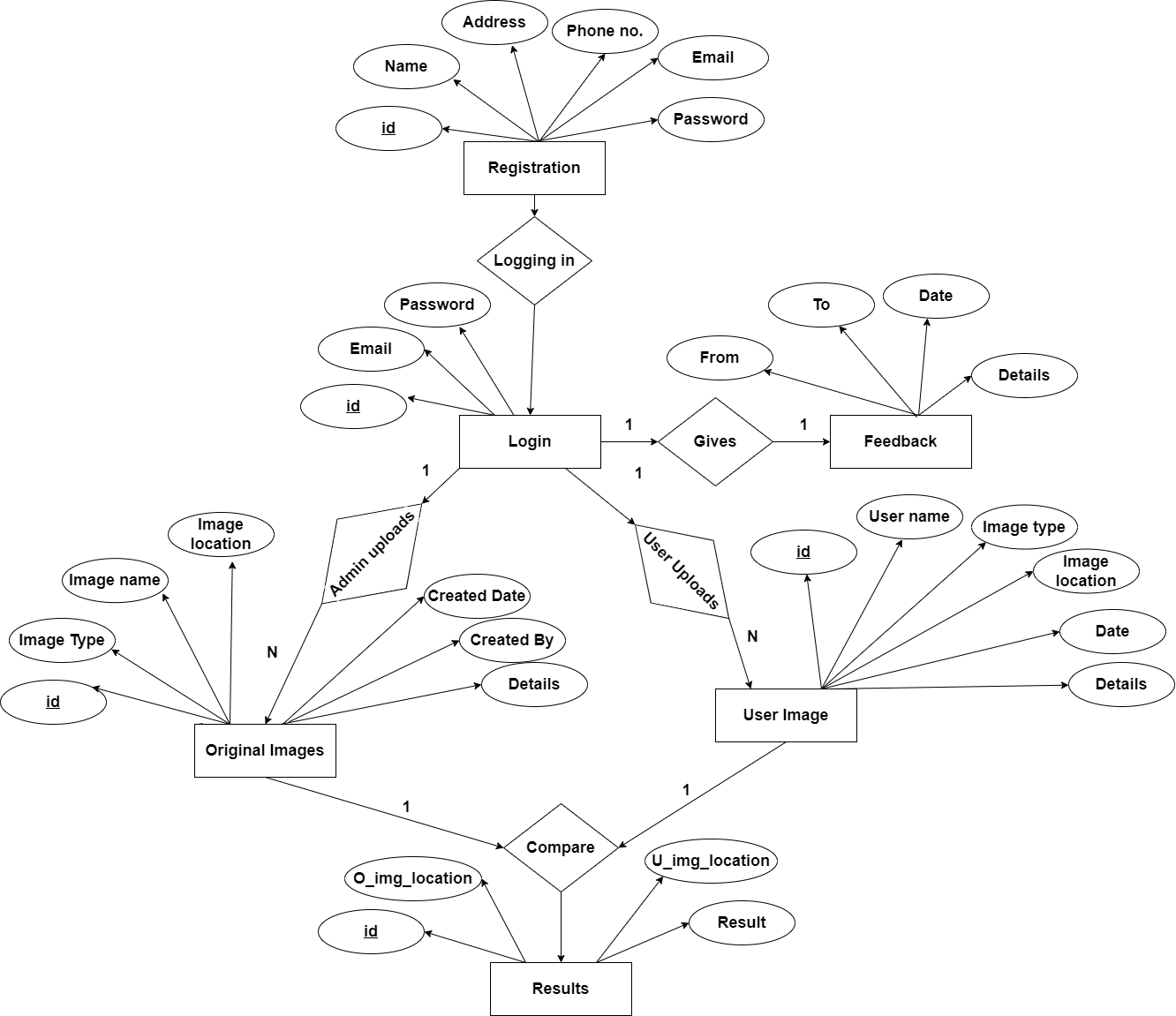
****



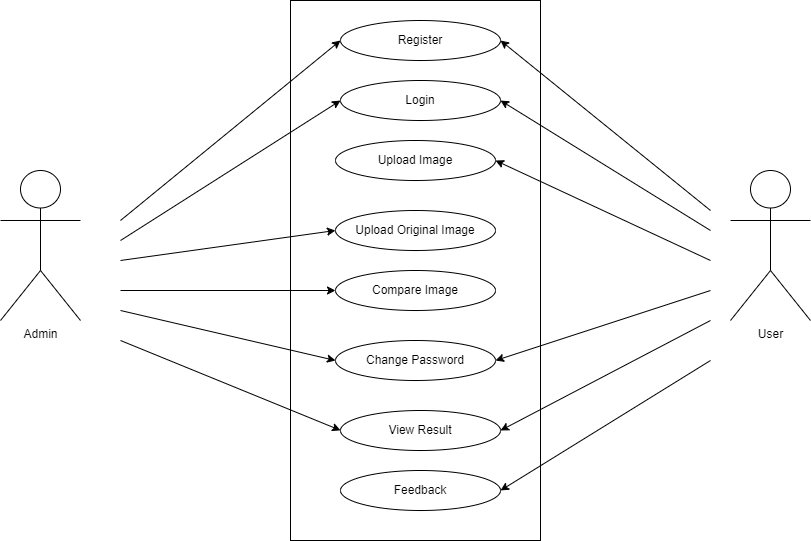
**Context level diagram (one level DFD):**

****

**Entity relationship diagram:**

****

**Use-Case Diagram:**



**5.IMPLEMENTATION**

**Introduction:**

Implementation is the process of converting a new revised system design into operation. The objective is to put the new revised system, which has been tested into operation while holding costs, risks and personal irritation to the minimum. A critical aspect of the implementation process is to ensure that there will be no description in the function of the organization. The best methods for going control while implementation is that, any new system would be to use well planned test files for testing all new programs. Another factor to be convinced in the implementation phase in the acquisition of the hardware and software. Once the software is developed for the system and testing is carried out, it is the process of making the newly designed system fully operational and consistent in performance

**Example to display message using HTML page:**

<!DOCTYPE HTML >

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<title>Untitled Document</title>

</head>

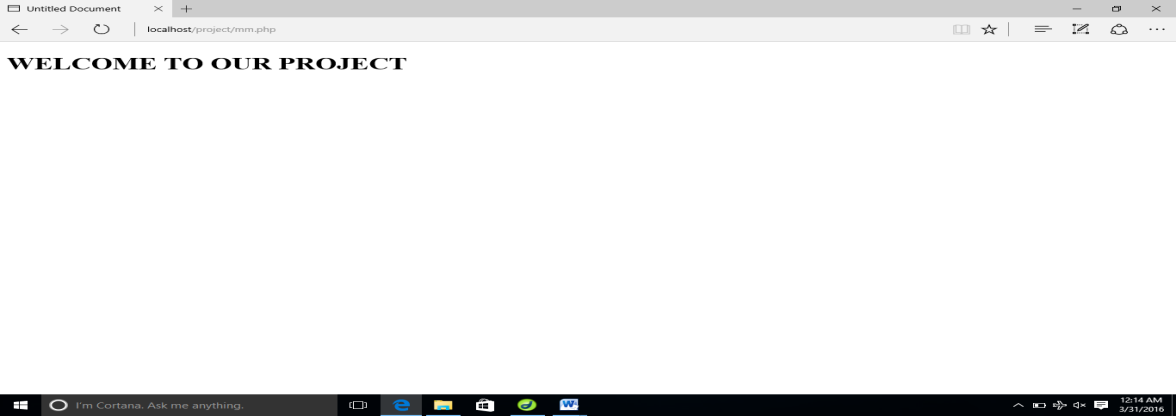
<body>

<h1>WELCOME TO OUR PROJECT</h1>

</body>

</html>

**Output:**



**Database Evolution:**

SQL was invented in the year 1960’s by E. F. Cod of IBM in order to increase data integrity and reduce repetitive data. RDBMS did not appear until the late 70’s when Sybase and Oracle introduced systems.

SQL server was originally a Sybase product. Microsoft bought the product outright from Sybase and by version 7.0, the version prior to 2000 all the code had been rewritten by Microsoft’s programming.

**Features of SQL:**

* The entire SQL has been divided into 4 major categories

1. Data Manipulation Language.
2. Data Definition Language.
3. Transaction control language.
4. Data Control Lnaguage.

* It is simple English like language and uses simple commands such as SELECT, CREATE, DROP etc.
* It is not having conditional loops, variables and most of the commands are single line commands.
* To implement application logics, SQL has got extension language popularly called as PL/SQL (Procedural language of SQL).
* One of the key features of sql server is the XML support. XML has grown to be standard technology for organizations that share data on the web.

**Security:**

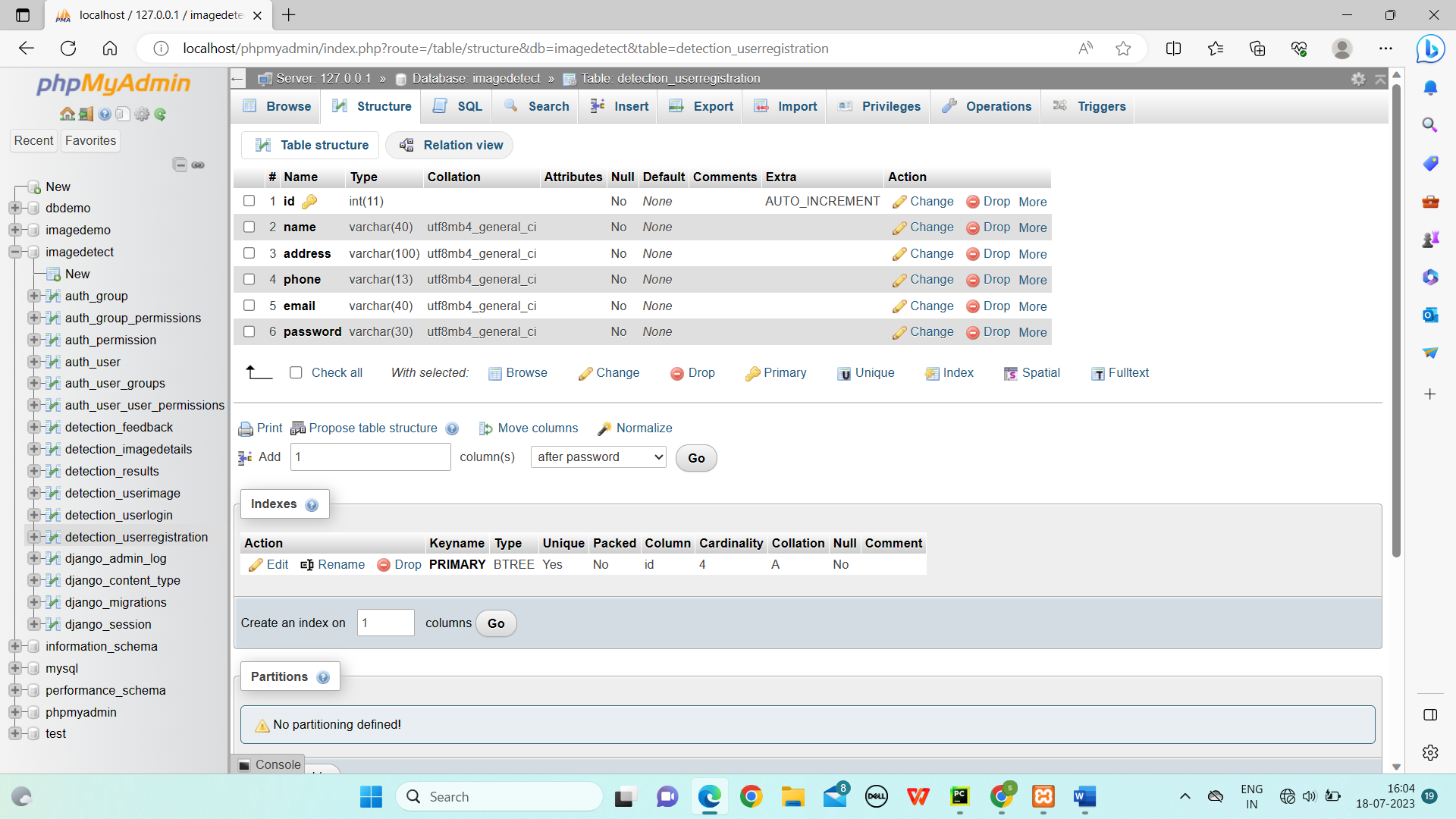
Views are basically used as a part of security, means in many organizations end user will never be given original tables and all data entry will be done with the help of views only. But the database administrator will be able to see everything because all the operations done by the different users will come to the same table.

**6.DATABASE TABLES**

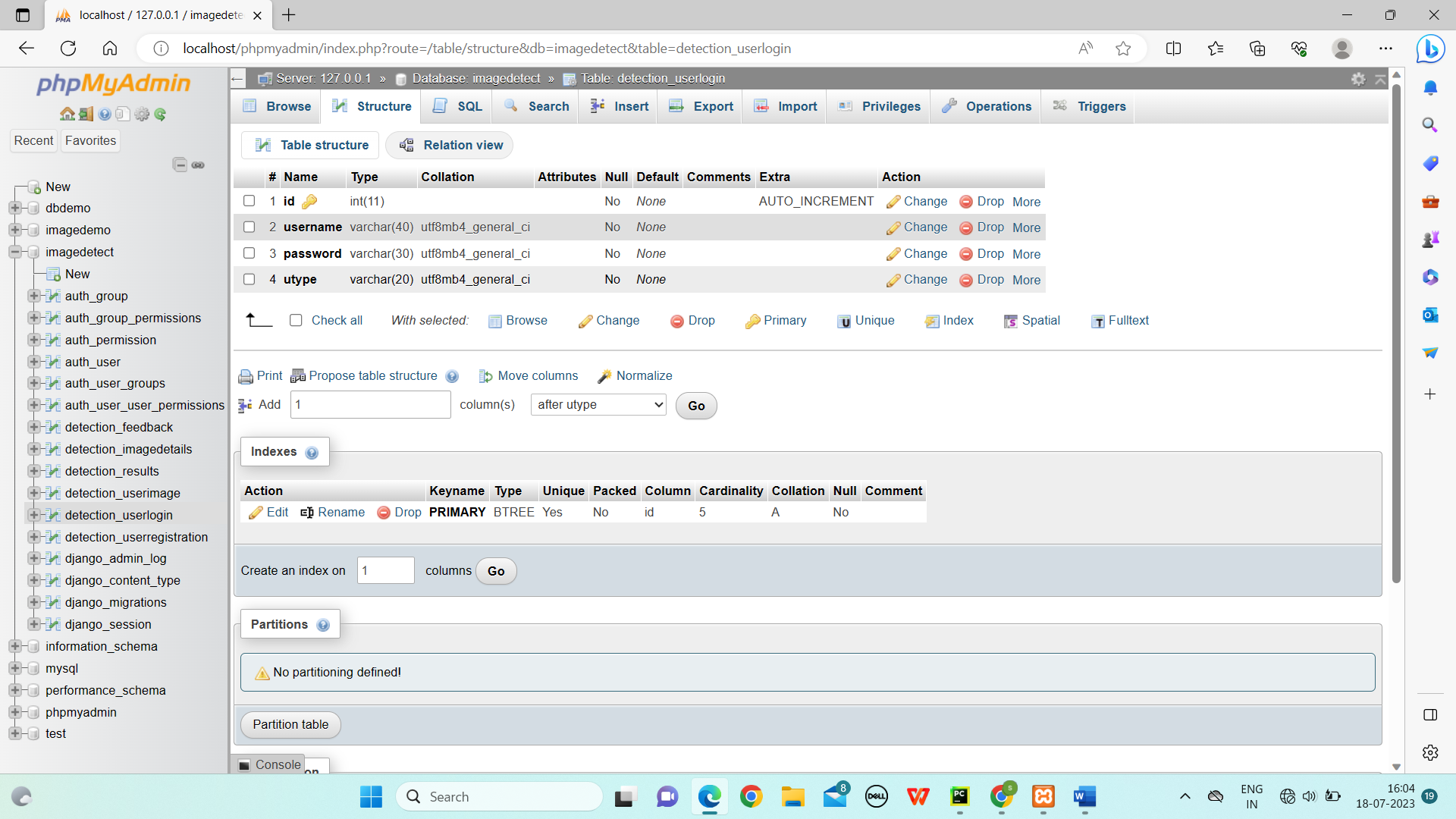
**Tables Used In Our Project:**

**1)Table structure for REGISTRATION table:**

**Primary Key: id**

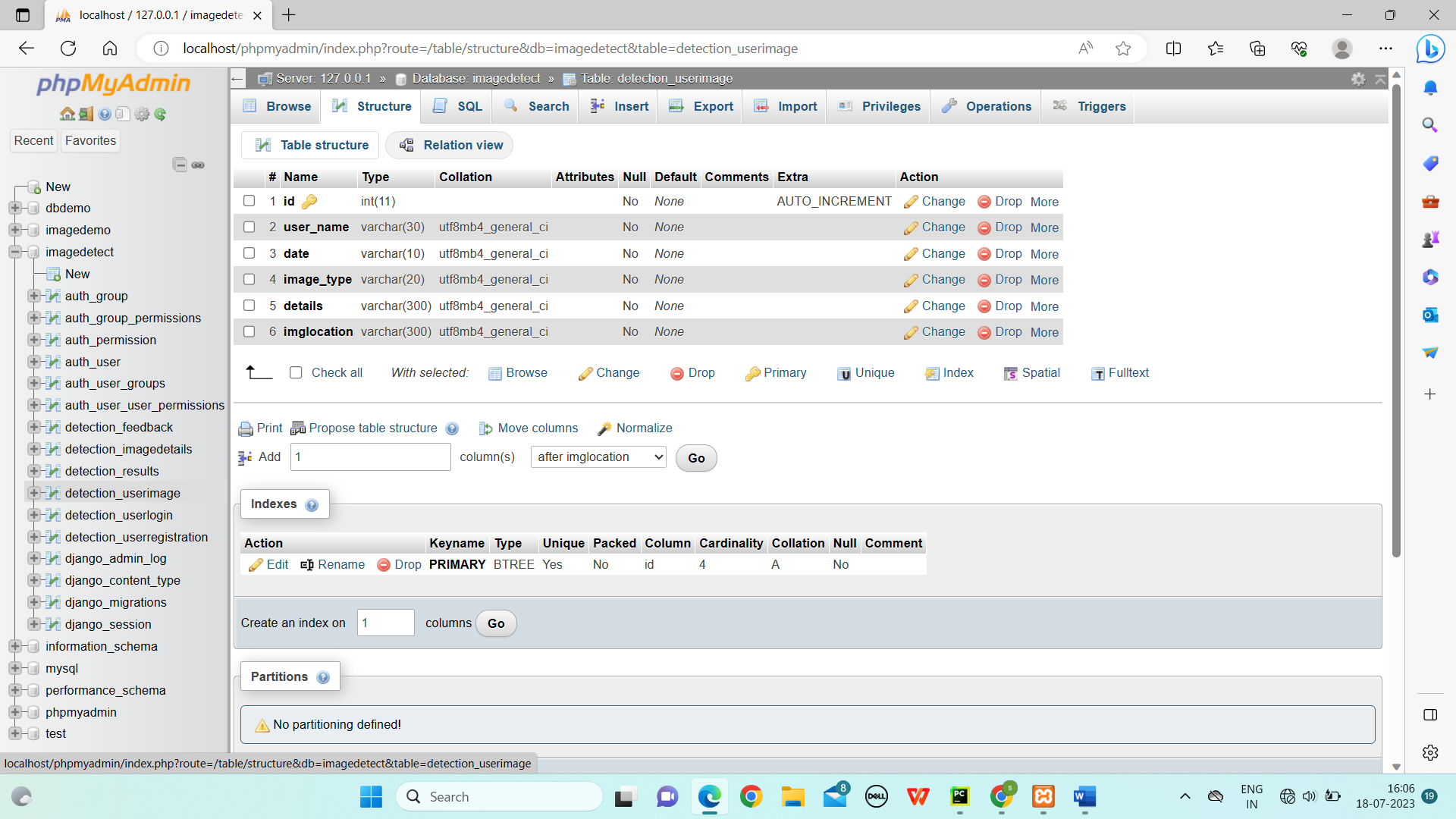


**2)Table structure for LOGIN table:**

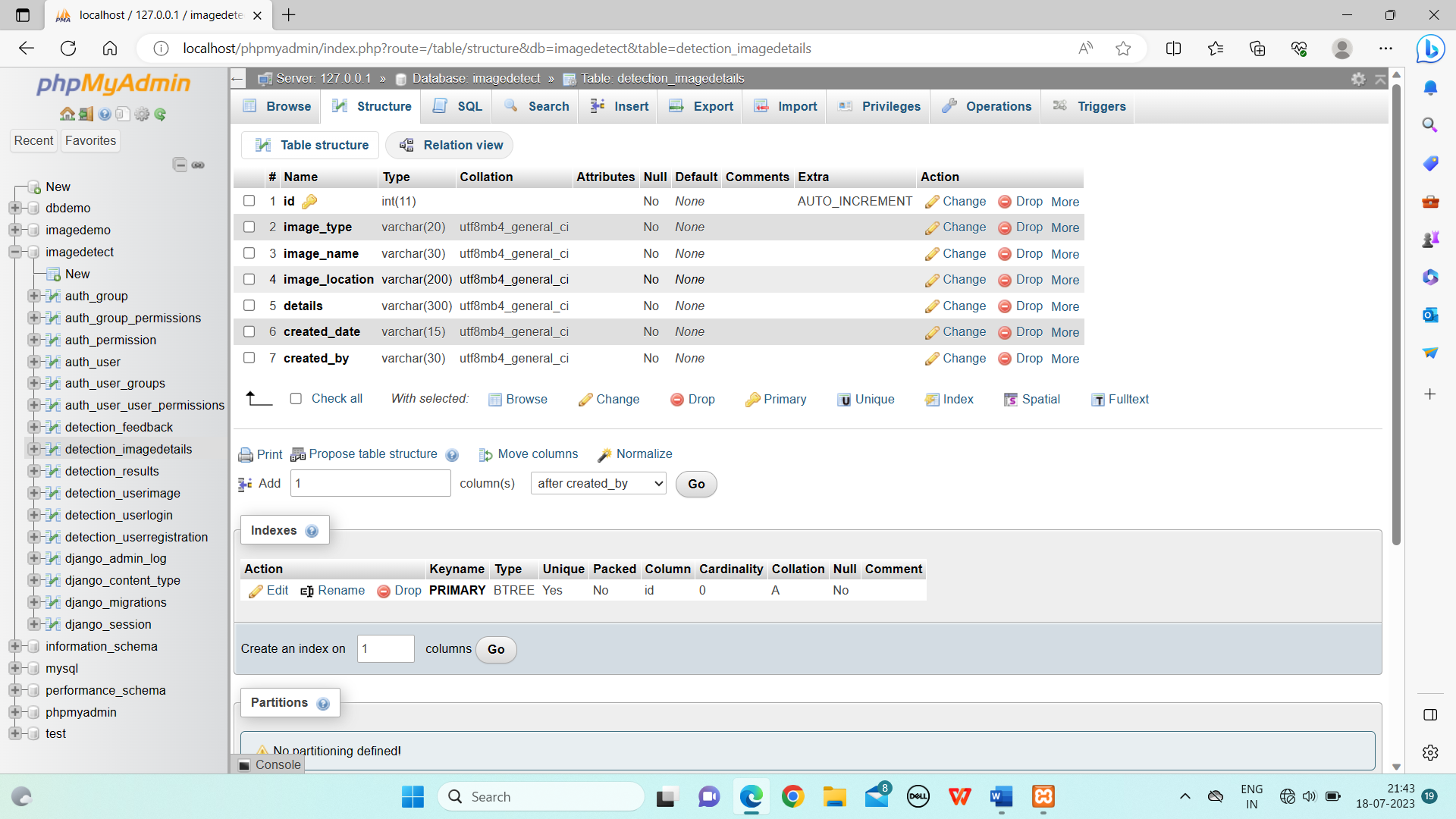
****

**3)Table structure for User Image table:**

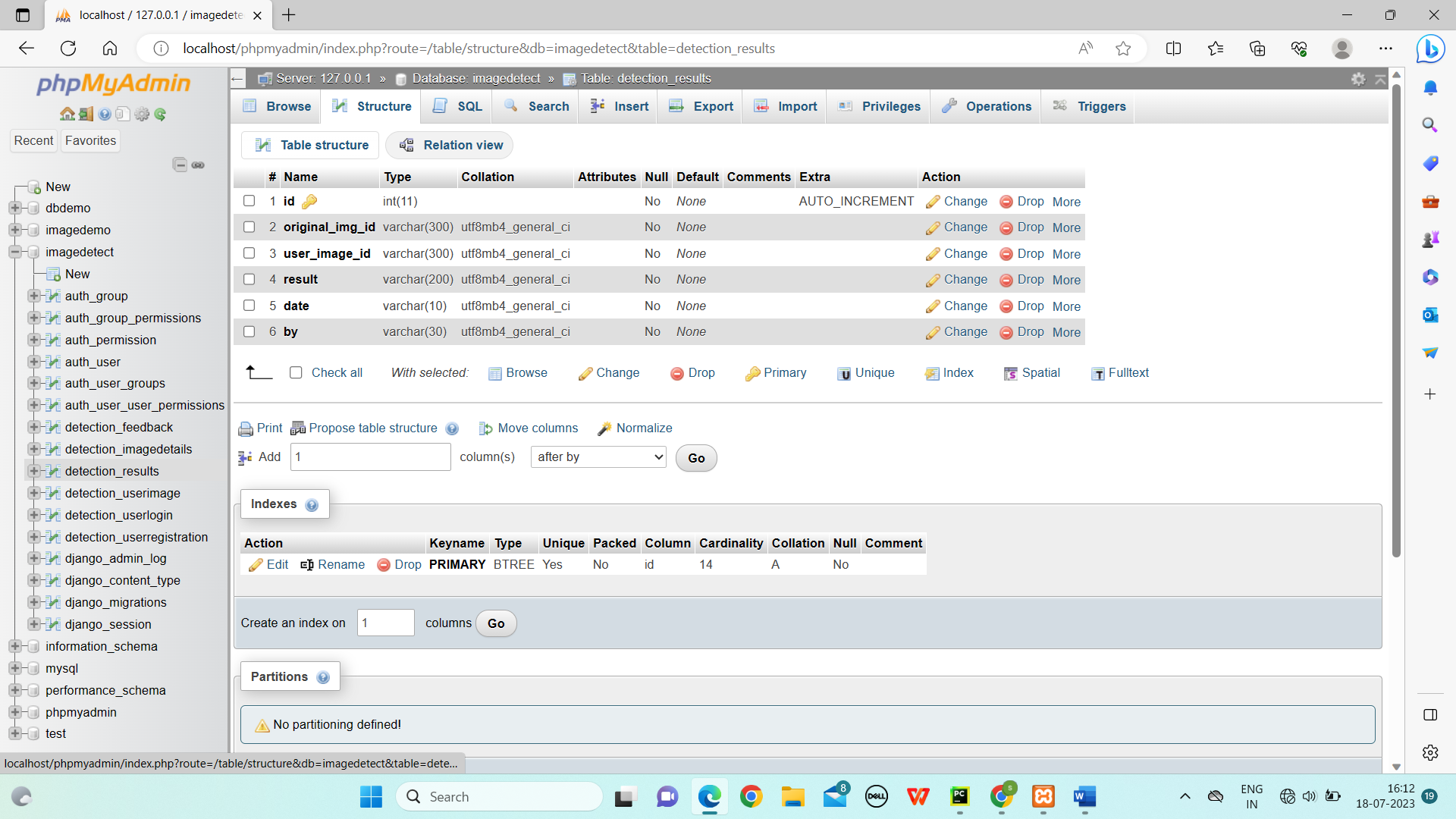
**Primary Key: id**

****

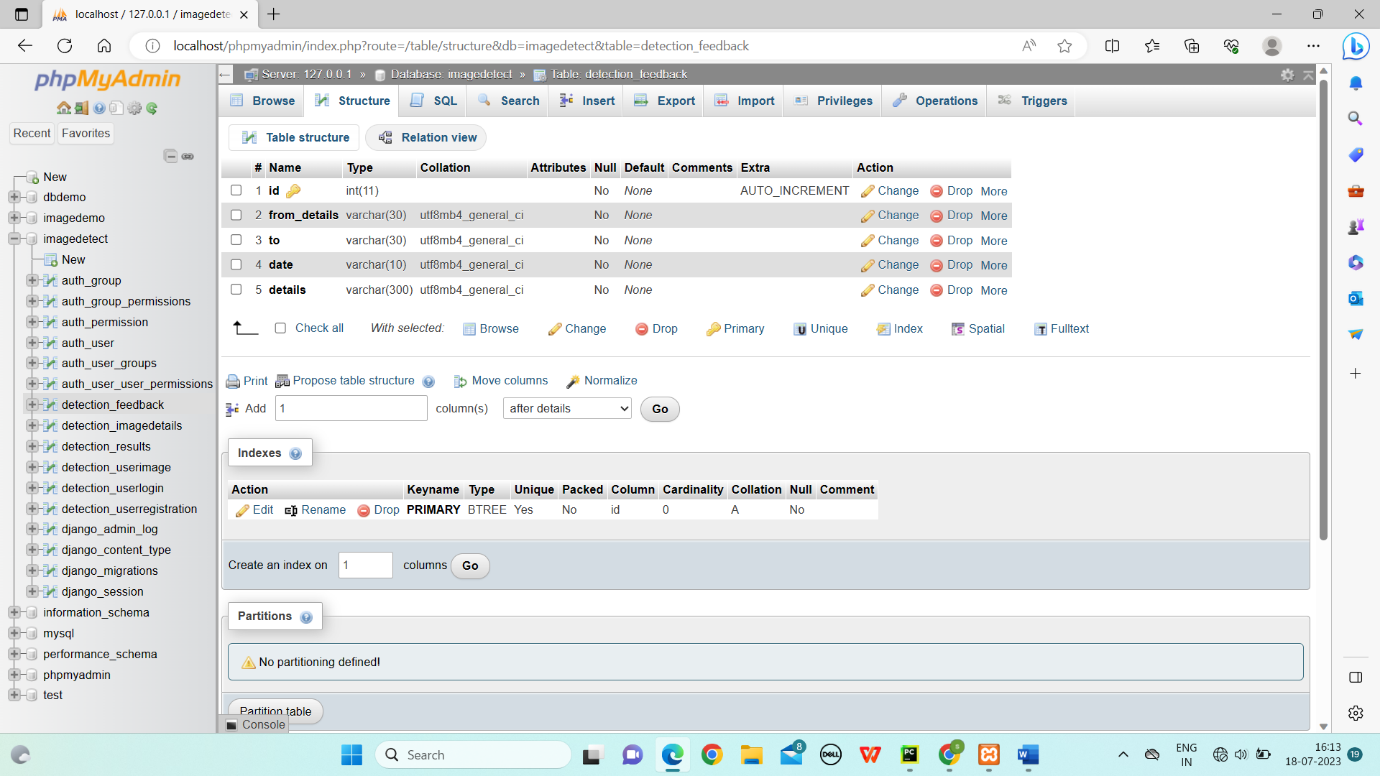
**4)Table structure for Original Image table:**

****

**5)Table structure for Result table:**

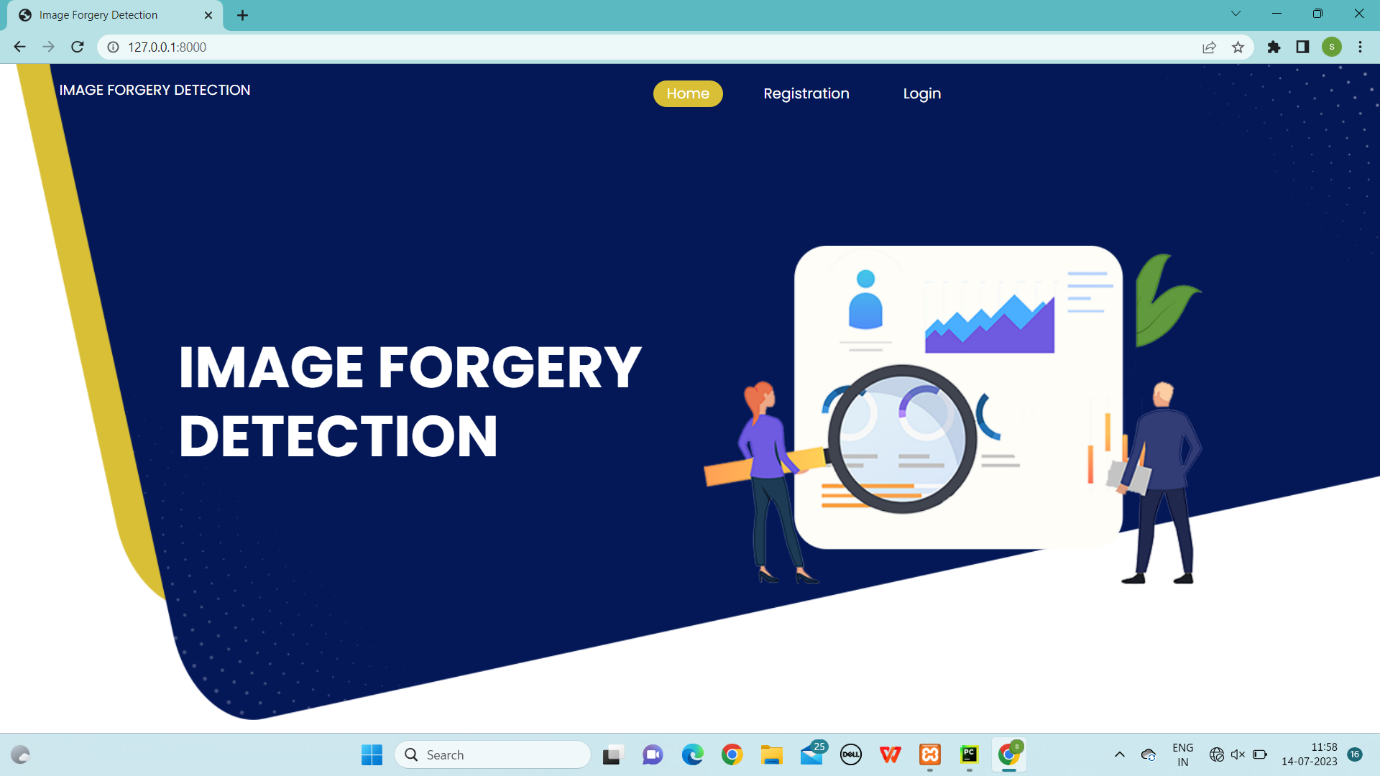
****

**5)Table structure for Feedback table:**

****

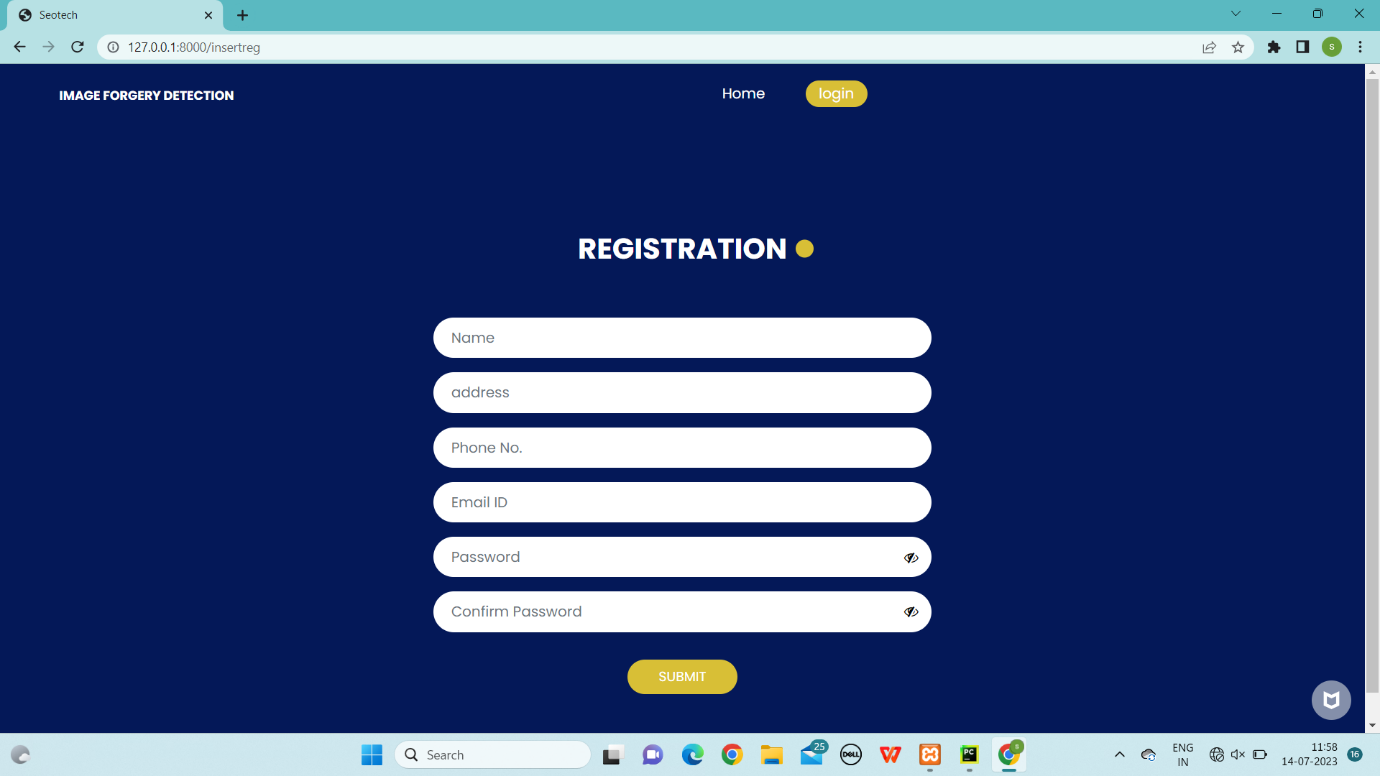
**7.SCREEN SHOTS**

**A VIEW OF HOME PAGE:**



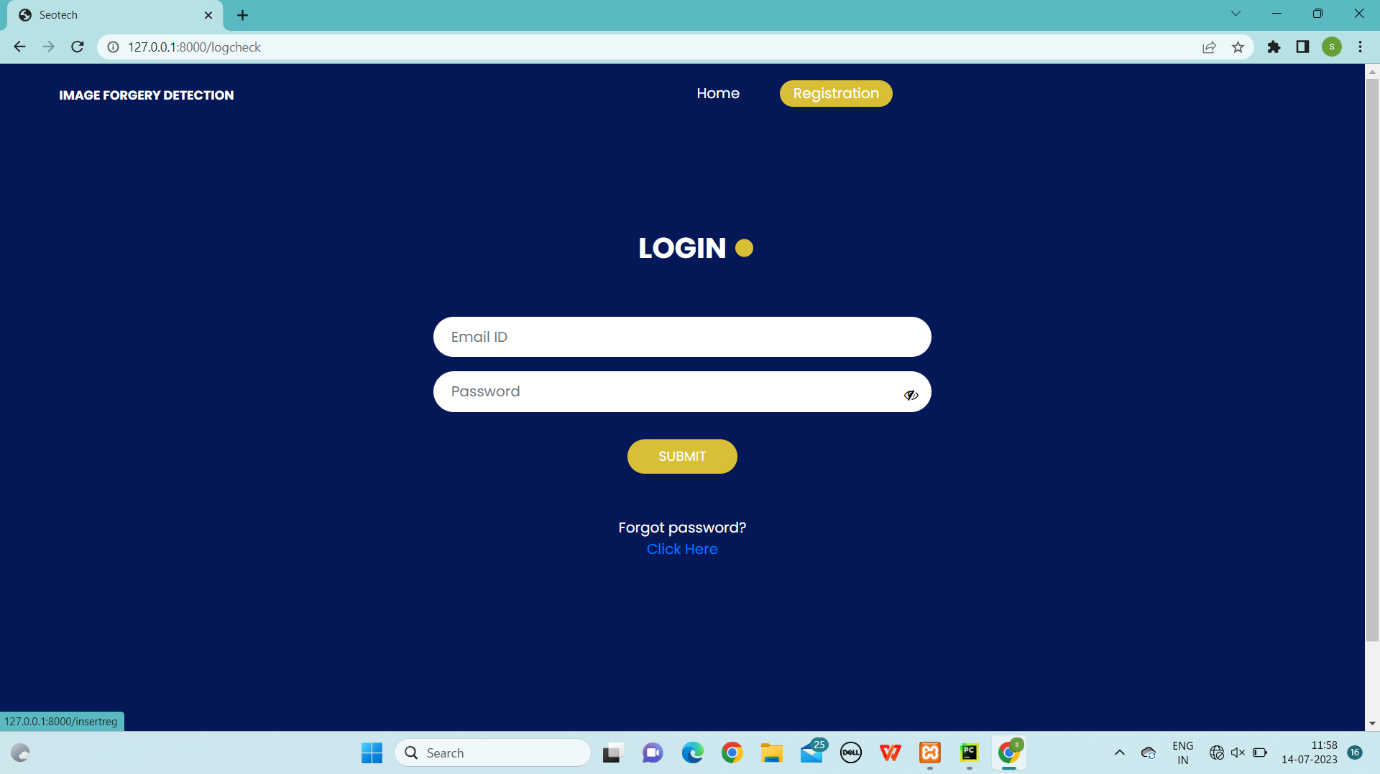
**Fig: 7.1**

**REGISTRATION PAGE:**

****

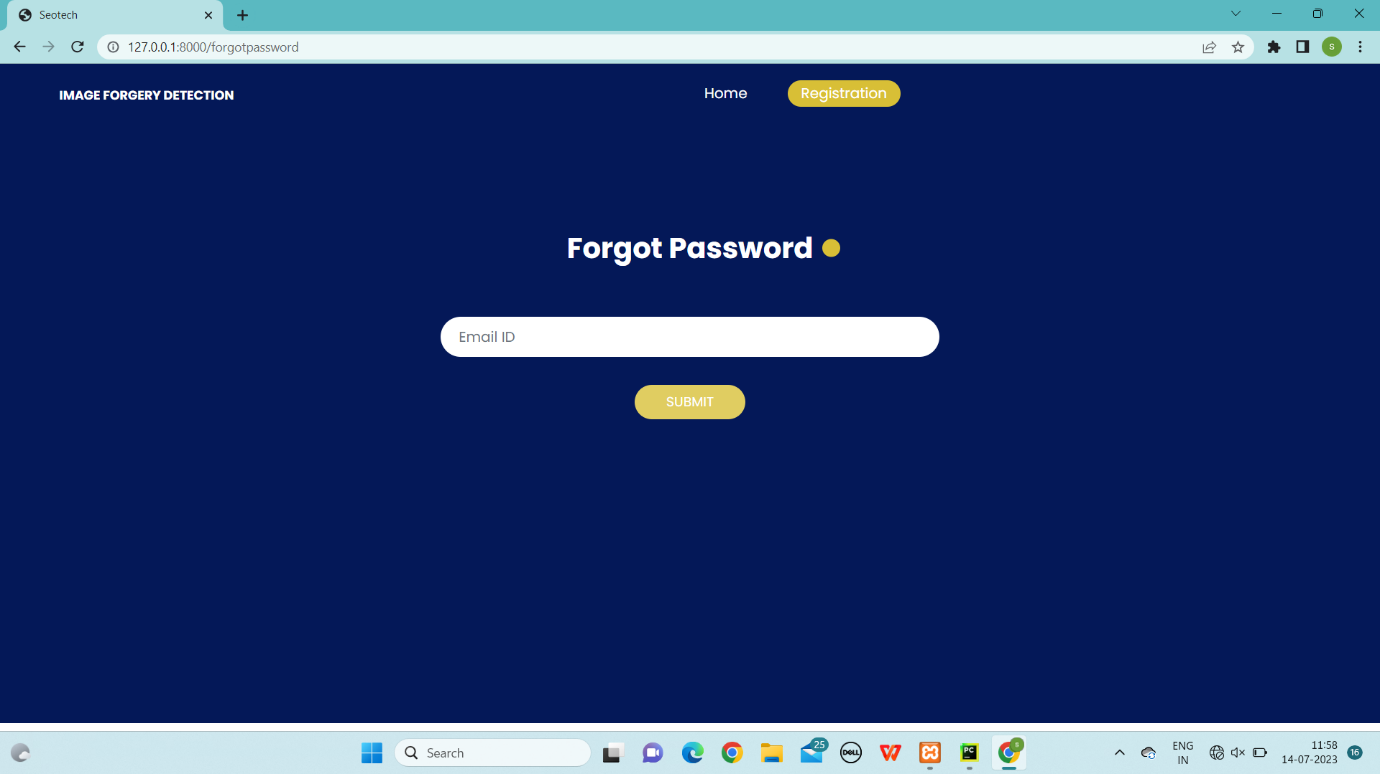
**Fig: 7.2**

**LOGIN PAGE:**

****

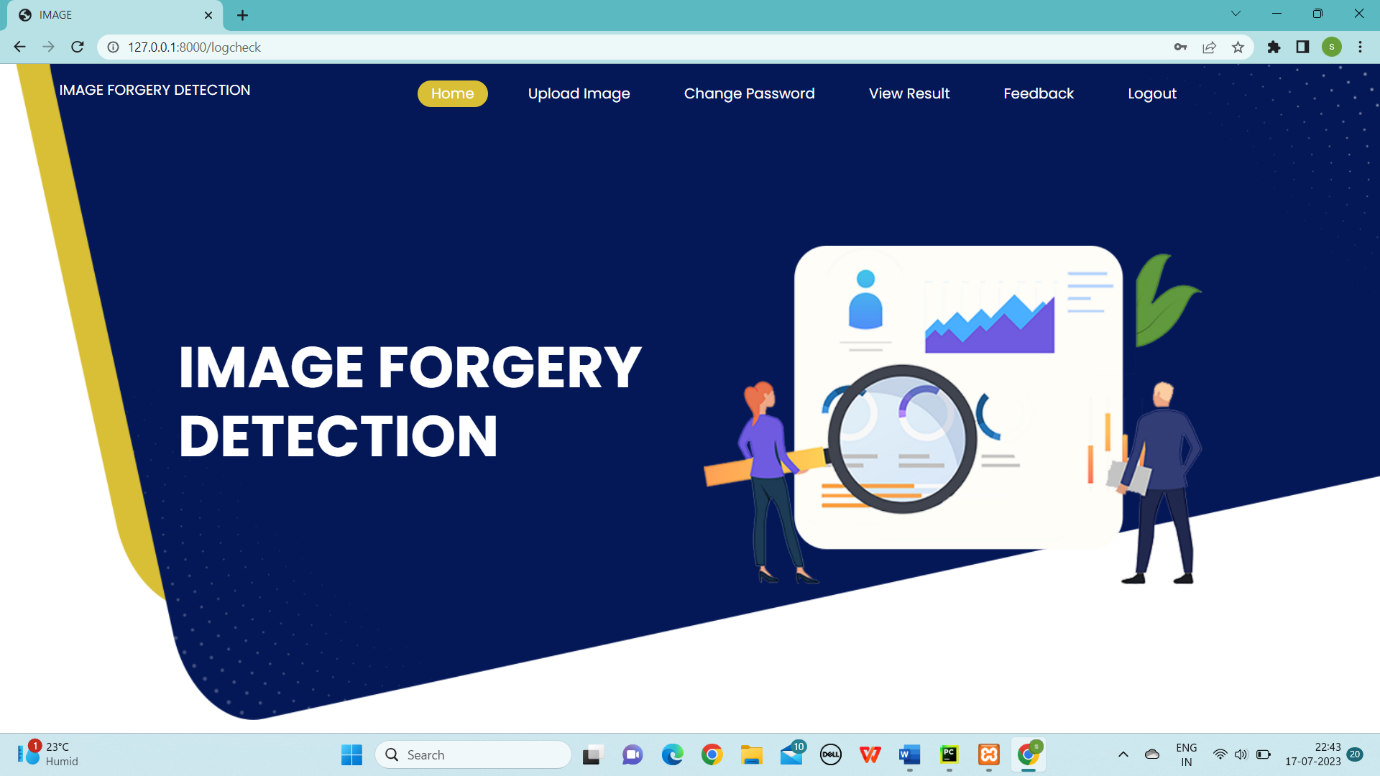
**Fig: 7.3**

**FORGOT PASSWORD:**

****

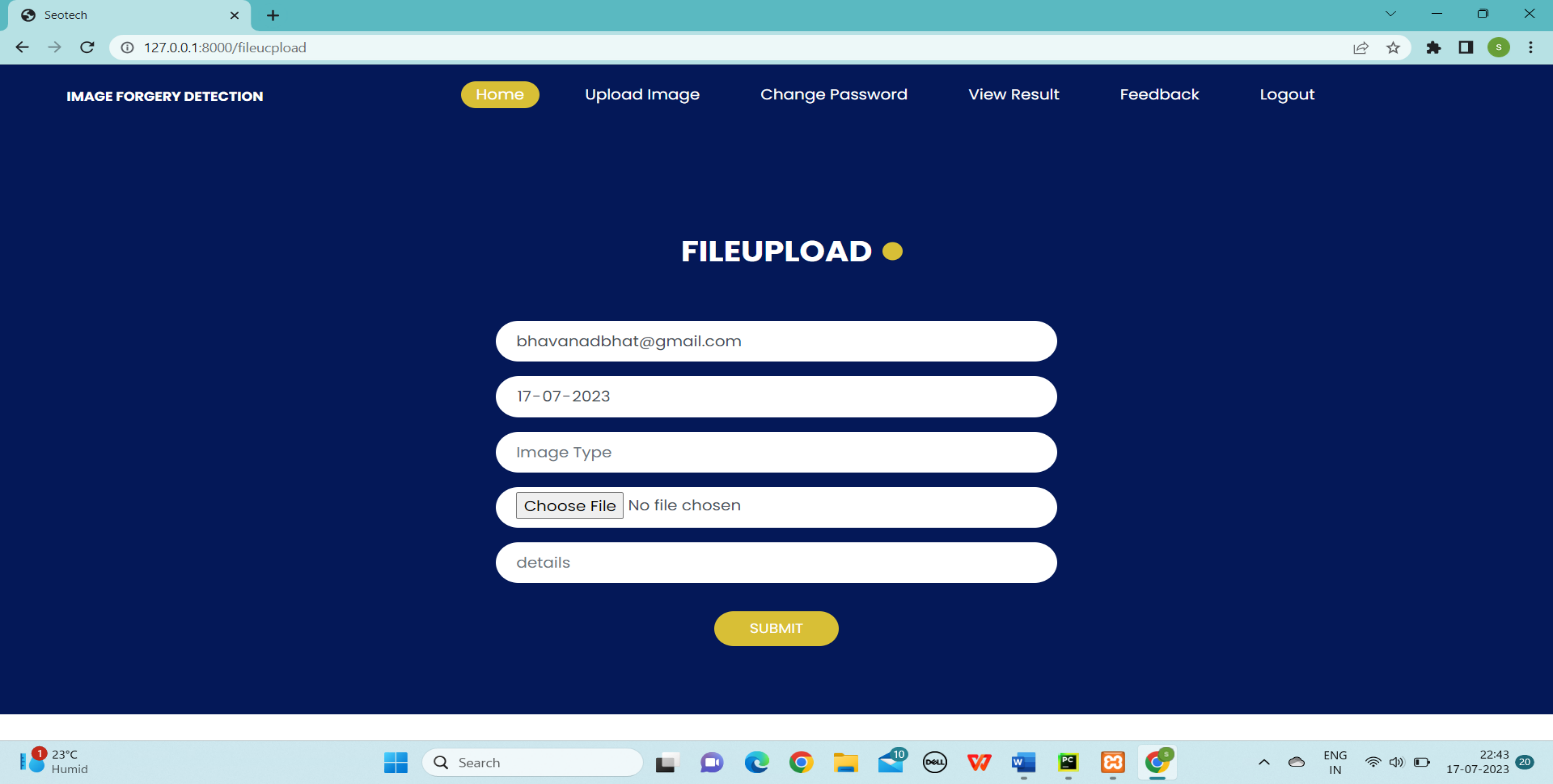
**Fig: 7.4**

**User Home:**

****

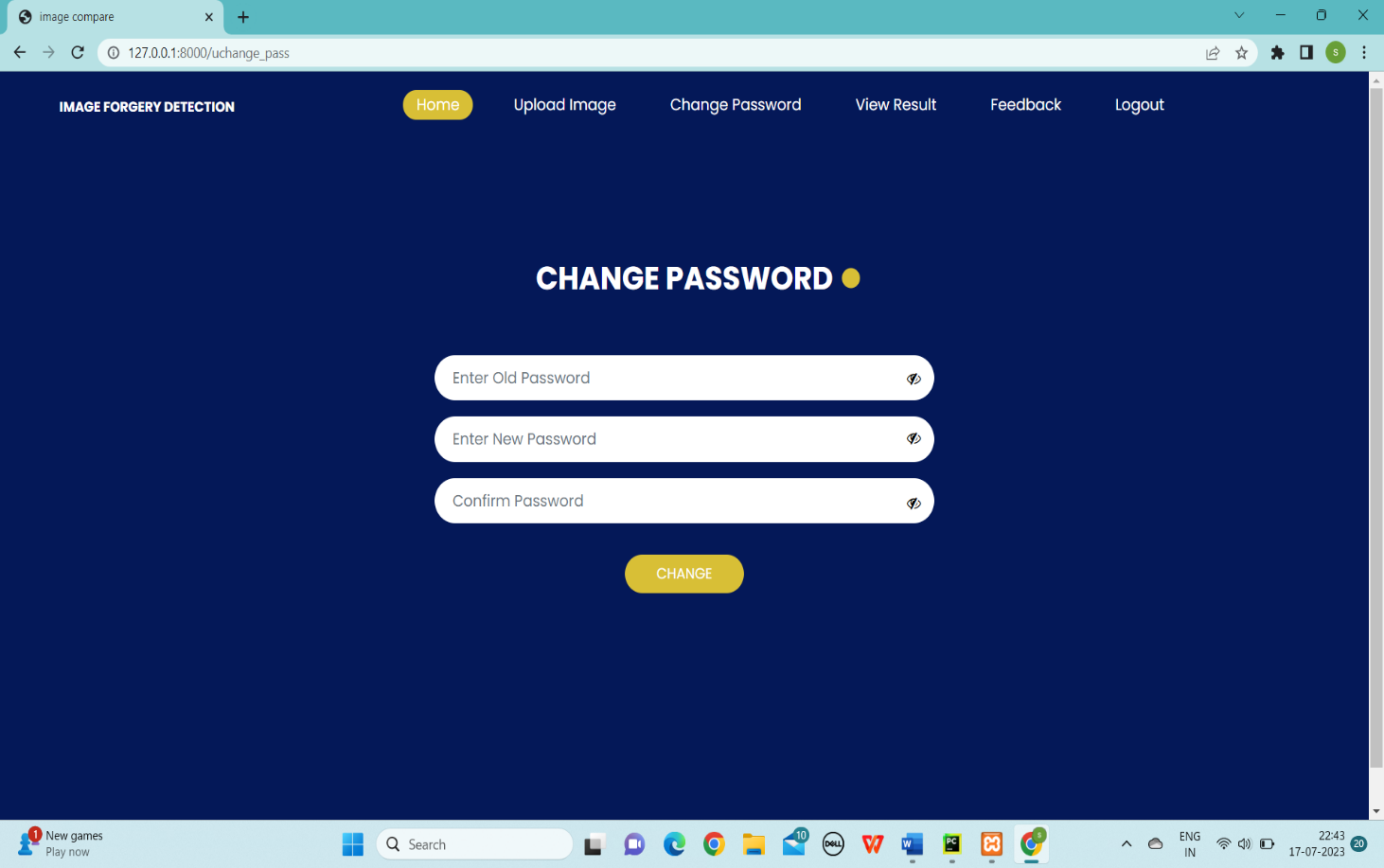
**Fig: 7.5**

**Upload User Image page:**

****

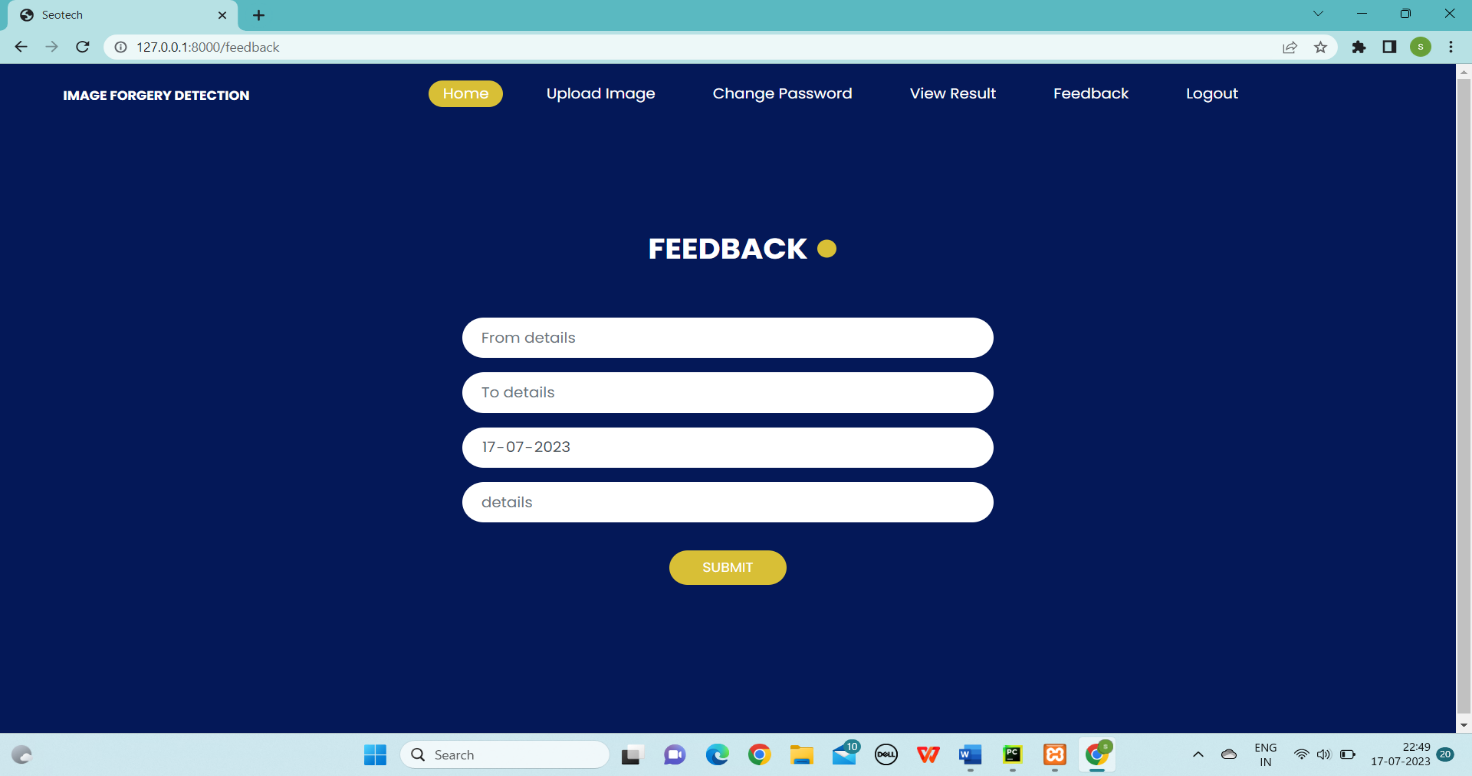
**Fig: 7.6**

**Change Password Page:**

****

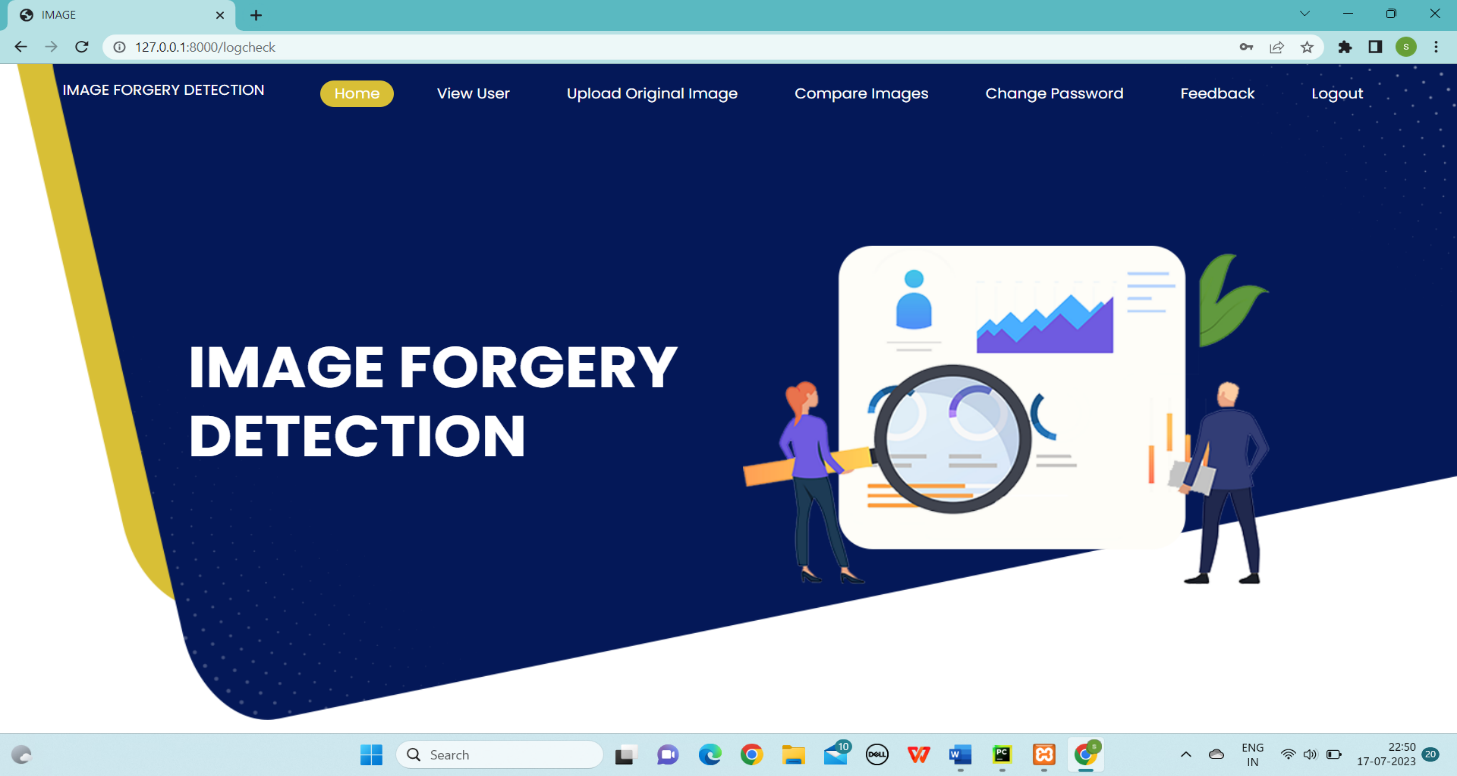
**Fig: 7.7**

**Feedback Page:**

****

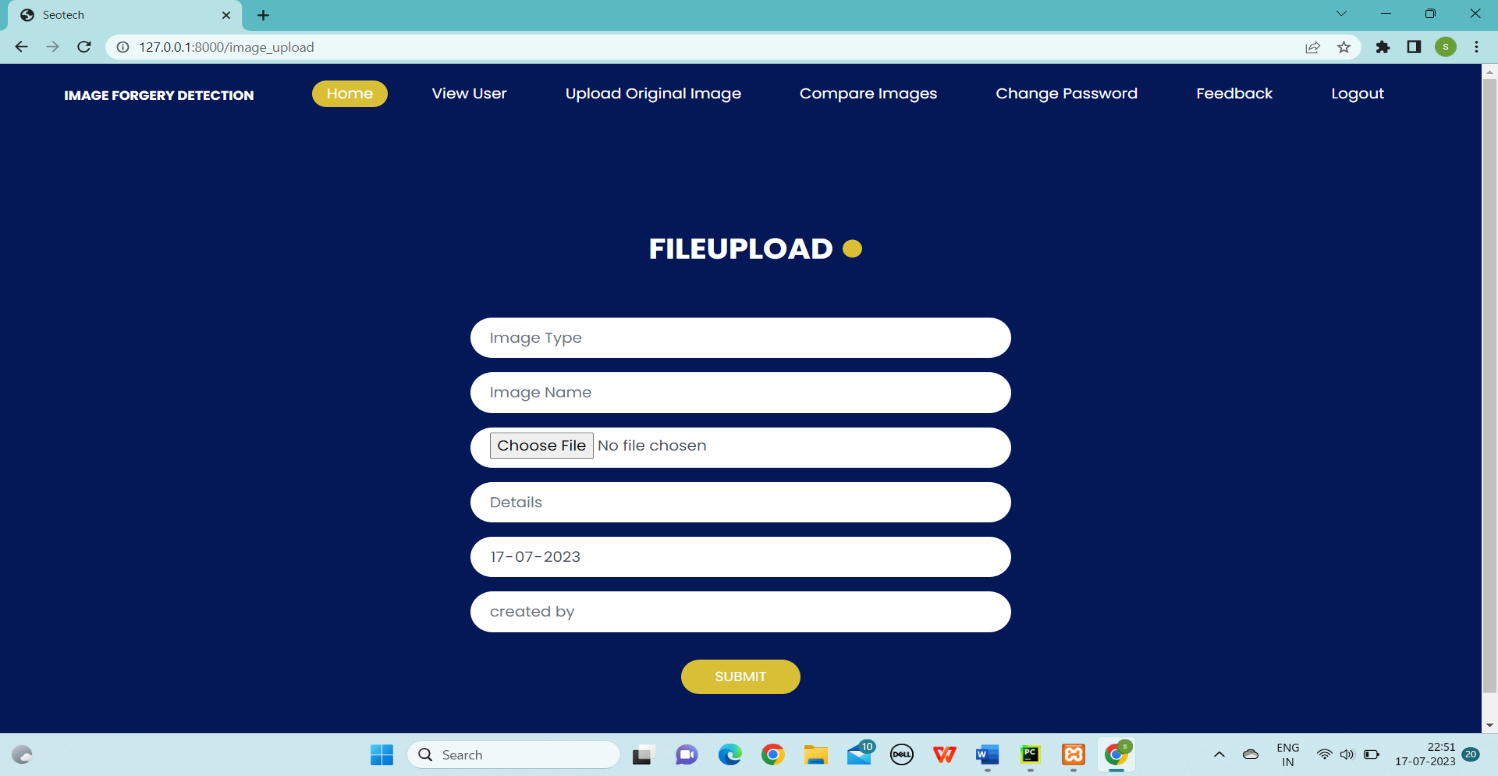
**Fig: 7.8**

**Admin Home:**

****

**Fig: 7.9**

**Original Image Upload Page:**

****

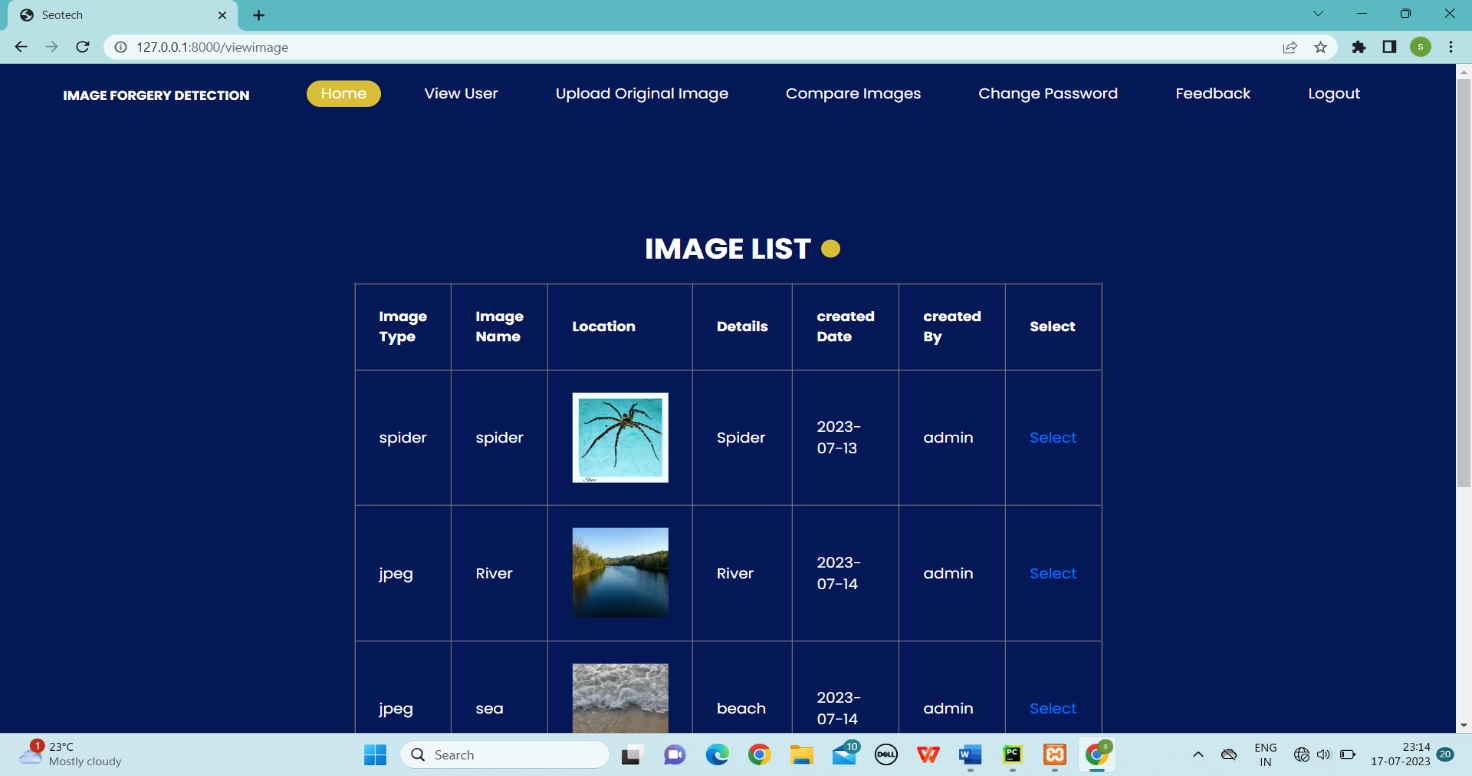
**Fig: 7.10**

**User List :**

****

**Fig: 7.11**

**Original Image list:**

****

**Fig: 7.12**

**7.SOURCE CODE**

**Views.py**

from django.shortcuts import render,redirect  
  
*# Create your views here.*from django.shortcuts import render  
import smtplib  
from detection.models import UserRegistration,UserLogin,UserImage,ImageDetails,Feedback,Results  
import os  
from django.core.files.storage import FileSystemStorage  
from image\_forgery.settings import BASE\_DIR  
import datetime  
from PIL import Image, ImageChops  
import cv2  
import numpy as np  
  
*# Create your views here.*def index(request):  
 return render(request,'index.html')  
  
def insertreg(request):  
 if request.method == "POST":  
 s1 = request.POST.get('t1', '')  
 s2 = request.POST.get('t2', '')  
 s3 = request.POST.get('t3', '')  
 s4 = request.POST.get('t4', '')  
 s5 = request.POST.get('t5', '')  
 s6 = request.POST.get('t6', '')  
 if s5==s6:  
 UserRegistration.objects.create(name=s1, address=s2, phone=s3, email=s4, password=s5)  
 UserLogin.objects.create(username=s4, password=s5, utype="user")  
 return render(request, 'registration.html', context={'msg': 'registration successfull'})  
 return render(request, 'registration.html')  
  
def logcheck(request):  
 if request.method == "POST":  
 uname = request.POST.get('t1', '')  
 request.session['username'] = uname  
 pwd = request.POST.get('t2', '')  
 checklogin = UserLogin.objects.filter(username=uname).values()  
 for a in checklogin:  
 utype = a['utype']  
 upass = a['password']  
 if (upass == pwd):  
 if (utype == "admin"):  
 return render(request, 'admin\_home.html')  
 if (utype == "user"):  
 return render(request, 'user\_home.html')  
 else:  
 return render(request, 'login.html', context={'msg': 'check username or password'})  
 return render(request, "login.html", context={'msg': 'check username or password'})  
 return render(request, "login.html", context={'msg': 'check username or password'})  
 return render(request, "login.html")  
  
def showuser(request):  
 userdict = UserRegistration.objects.all()  
 return render(request, 'viewregistration.html', {'userdict': userdict})  
  
def deluser(request,pk):  
 rid = UserRegistration.objects.get(id=pk)  
 rid.delete()  
 userdict = UserRegistration.objects.all()  
 return render(request, 'viewregistration.html', {'userdict': userdict})  
  
def userhome(request):  
 return render(request,'user\_home.html')  
  
def adminhome(request):  
 return render(request,'admin\_home.html')  
  
def image\_upload(request):  
 if request.method=="POST" and request.FILES['myfile']:  
 imgtype = request.POST.get('t1')  
 imgname=request.POST.get('t2')  
 myfile=request.FILES['myfile']  
 details=request.POST.get('t3')  
 date=request.POST.get('t4')  
 by = request.POST.get('t5')  
  
 fs = FileSystemStorage()  
 filename = fs.save(myfile.name, myfile)  
 uploaded\_file\_url = fs.url(filename)  
 pat = os.path.join(BASE\_DIR, '/media/' + filename)  
 ImageDetails.objects.create(image\_type=imgtype,image\_name=imgname,image\_location=myfile,details=details,created\_date=date,created\_by=by)  
 return render(request,"imageupload.html")  
  
def fileupload(request):  
 username = request.session['username']  
 if request.method=="POST" and request.FILES['myfile']:  
 name = request.POST.get('t1')  
 myfile=request.FILES['myfile']  
 date=request.POST.get('t2')  
 details=request.POST.get('t4')  
 type=request.POST.get('t3')  
  
 fs = FileSystemStorage()  
 filename = fs.save(myfile.name, myfile)  
 uploaded\_file\_url = fs.url(filename)  
 pat = os.path.join(BASE\_DIR, '/media/' + filename)  
 UserImage.objects.create(user\_name=name,image\_type=type,date=date,details=details,imglocation=myfile)  
 return render(request,"fileupload.html",{'username':username})  
  
def showimage(request):  
 userdict = UserImage.objects.all()  
 return render(request, 'user\_img\_list.html', {'userdict': userdict})  
  
def viewimage(request):  
 userdict = ImageDetails.objects.all()  
 return render(request, 'original\_img\_list.html', {'userdict': userdict})  
  
  
def selectimage(request,pk):  
 oid=ImageDetails.objects.filter(id=pk).all()  
 userdict = UserImage.objects.all()  
 return render(request, 'user\_img\_list.html', {'userdict': userdict,'oid':oid})  
  
def showimg(request,id1,id2):  
 request.session['oimage'] = id1  
 request.session['uimage'] = id2  
 context = {}  
 context['id1'] = id1  
 context['id2'] = id2  
 return render(request, 'showimage.html', context=context)  
  
def compare(request):  
 oimg = request.session['oimage']  
 uimg = request.session['uimage']  
  
 image1 = Image.open('media/'+oimg)  
 image2 = Image.open('media/'+uimg)  
 percentage\_diff=""  
 try:  
 diff = ImageChops.difference(image1, image2)  
  
 pixels\_diff = sum(diff.getdata()) / 255 *# Count the differing pixels* total\_pixels = image1.size[0] \* image1.size[1] *# Calculate total number of pixels* percentage\_diff = (pixels\_diff / total\_pixels) \* 100  
 except:  
  
 percentage\_diff="both images are diffrent"  
  
 *#Results.objects.create(original\_img\_id=oimg,user\_image\_id=uimg,result=percentage\_diff,date='1',by='admin')* return render(request,"result.html",{'oimg':oimg,'uimg':uimg,'percentage\_diff':percentage\_diff})  
  
  
def imagecompare\_diff(request):  
 oimg = str(request.session['oimage'])  
 uimg = str(request.session['uimage'])  
 print(oimg)  
 print(uimg)  
  
 image1 = Image.open('media/' + oimg)  
 image2 = Image.open('media/' + uimg)  
 x = np.array(image1.histogram())  
 y = np.array(image2.histogram())  
  
 *# image1 = Image.open('D:/project\_final/image\_forgery/media/one.jpg')  
 #  
 # image2 = Image.open('D:/project\_final/image\_forgery/media/two.jpg')  
  
 #original = cv2.imread(image1)  
 #duplicate = cv2.imread(image2)* original = cv2.imread('media/' + oimg)  
 duplicate = cv2.imread('media/' + uimg)  
  
 if (original.shape == duplicate.shape):  
 print("the images have same size and channels")  
 e1 = "images are very similar"  
 difference=cv2.subtract(original, duplicate)  
 b, g, r=cv2.split(difference)  
  
 if (cv2.countNonZero(b) == 0 and cv2.countNonZero(g) == 0 and cv2.countNonZero(r) == 0):  
 print("The images are completely equal");  
 error = np.sqrt(((x - y) \*\* 2).mean())  
 error = str(error)[:2]  
 actual\_error = float(100) - float(error)  
 e1="Images are completely equal. "+str(actual\_error)+"% matched"  
 else:  
 print("the images are not equal");  
 error = np.sqrt(((x - y) \*\* 2).mean())  
 error = str(error)[:2]  
 actual\_error = float(100) - float(error)  
 e1="Images are not equal."+str(actual\_error)+ "% not matched"  
  
 udata = UserImage.objects.filter(imglocation=uimg).values()  
 for i in udata:  
 user= i['user\_name']  
 now = datetime.datetime.now()  
 idate = now.strftime("%d-%m-%Y")  
 itime = now.strftime("%X")  
  
 Results.objects.create(original\_img\_id=oimg,user\_image\_id=uimg,result=e1,date=idate,by=user)  
 return render(request,"ans.html",{'percentage\_diff':e1})  
  
def ans(request):  
 return render(request,'ans.html')  
  
def delshowimg(request,pk):  
 rid = UserImage.objects.get(id=pk)  
 rid.delete()  
 userdict = UserImage.objects.all()  
 return render(request, 'user\_img\_list.html', {'userdict': userdict})  
  
  
  
def delviewimg(request,pk):  
 rid = ImageDetails.objects.get(id=pk)  
 rid.delete()  
 userdict = ImageDetails.objects.all()  
 return render(request, 'original\_img\_list.html', {'userdict': userdict})  
  
  
  
  
def change\_pass(request):  
 username=request.session['username']  
 if request.method=="POST":  
 current\_pass=request.POST.get("t1")  
 new\_pass=request.POST.get("t2")  
 cofirm\_pass=request.POST.get("t3")  
 udata=UserLogin.objects.filter(username=username).filter(password=current\_pass).count()  
 if udata>=1:  
 if new\_pass==cofirm\_pass:  
 UserLogin.objects.filter(username=username).update(password=new\_pass)  
 return render(request,'admin\_home.html')  
 else:  
 return render(request,'change\_password.html',context={'msg': 'check your old password'})  
 else:  
 return render(request, 'change\_password.html',context={'msg': 'check your old password'})  
 return render(request, 'change\_password.html',{'username':username})  
  
def uchange\_pass(request):  
 username=request.session['username']  
 if request.method=="POST":  
 current\_pass=request.POST.get("t1")  
 new\_pass=request.POST.get("t2")  
 cofirm\_pass=request.POST.get("t3")  
 udata=UserLogin.objects.filter(username=username).filter(password=current\_pass).count()  
 if udata>=1:  
 if new\_pass==cofirm\_pass:  
 UserLogin.objects.filter(username=username).update(password=new\_pass)  
 return render(request,'user\_home.html')  
 else:  
 return render(request,'change\_user\_pwd.html',context={'msg': 'check your old password'})  
 else:  
 return render(request, 'change\_user\_pwd.html',context={'msg': 'check your old password'})  
 return render(request, 'change\_user\_pwd.html',{'username':username})  
  
def feedback(request):  
 if request.method == "POST":  
 s1 = request.POST.get('t1', '')  
 s2 = request.POST.get('t2', '')  
 s3 = request.POST.get('t3', '')  
 s4 = request.POST.get('t4', '')  
 s4 = request.POST.get('t4', '')  
 Feedback.objects.create(from\_details=s1, to=s2, date=s3, details=s4)  
 return render(request, 'feedback.html')  
 return render(request, 'feedback.html')  
  
def showfeedback(request):  
 userdict = Feedback.objects.all()  
 return render(request, 'feedback\_list.html', {'userdict': userdict})  
  
  
def forgotpassword(request):  
 if request.method=="POST":  
 uname = request.POST.get('t1', '')  
 user = UserLogin.objects.filter(username=uname).count()  
 if user >= 1:  
 userlog = UserLogin.objects.filter(username=uname).values()  
 for u in userlog:  
 upass= u['password']  
 content = upass  
 mail = smtplib.SMTP('smtp.gmail.com', 587)  
 mail.ehlo()  
 mail.starttls()  
 mail.login('shwethasirsi16@gmail.com', 'ezwbwyguthzsklog')  
 mail.sendmail('shwethasirsi16@gmail.com', uname , content)  
 mail.close()  
 return render(request,'login.html', {'msg': 'Your password has been sent to your E-mail'})  
 else:  
 return render(request,'forgotpassword.html', {'msg': 'Enter a valid username'})  
 return render(request,'forgotpassword.html')  
  
def viewresult(request):  
 username = request.session['username']  
 userdict = Results.objects.filter(by=username).values()  
 return render(request, 'viewresult.html', {'userdict':userdict})

**models.py**

from django.db import models  
  
*# Create your models here.*class UserRegistration(models.Model):  
 name = models.CharField(max\_length=40)  
 address = models.CharField(max\_length=100)  
 phone = models.CharField(max\_length=13)  
 email = models.CharField(max\_length=40)  
 password = models.CharField(max\_length=30)  
  
class UserLogin(models. Model):  
 username = models.CharField(max\_length=40)  
 password = models.CharField(max\_length=30)  
 utype = models.CharField(max\_length=20)  
  
class ImageDetails(models.Model):  
 image\_type = models.CharField(max\_length=20)  
 image\_name = models.CharField(max\_length=30)  
 image\_location = models.CharField(max\_length=200)  
 details = models.CharField(max\_length=300)  
 created\_date = models.CharField(max\_length=15)  
 created\_by = models.CharField(max\_length=30)  
  
class UserImage(models.Model):  
 user\_name = models.CharField(max\_length=30)  
 date = models.CharField(max\_length=10)  
 image\_type = models.CharField(max\_length=20)  
 details = models.CharField(max\_length=300)  
 imglocation = models.CharField(max\_length=300)  
  
  
class Results(models.Model):  
 original\_img\_id = models.CharField(max\_length=300)  
 user\_image\_id = models.CharField(max\_length=300)  
 result = models.CharField(max\_length=200)  
 date = models.CharField(max\_length=10)  
 by = models.CharField(max\_length=30)  
  
class Feedback(models.Model):  
 from\_details = models.CharField(max\_length=30)  
 to = models.CharField(max\_length=30)  
 date = models.CharField(max\_length=10)  
 details = models.CharField(max\_length=300)

**urls.py**

*"""Image\_forgery\_detection URL Configuration  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/2.2/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""*from django.contrib import admin  
from django.urls import path  
from detection import views  
from django.conf.urls import url  
from django.conf import settings  
from django.conf.urls.static import static  
  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 url('^$', views.index, name="index"),  
 url('^insertreg', views.insertreg, name="insertreg"),  
 url('^logcheck', views.logcheck, name="logcheck"),  
 url('^showuser', views.showuser, name="showuser"),  
 url('^fileucpload', views.fileupload, name="fileupload"),  
 url('^deluser(?P<pk>\d+)/$', views.deluser, name="deluser"),  
 url('^image\_upload', views.image\_upload, name="image\_upload"),  
 url('^showimage', views.showimage, name="showimage"),  
 url('^viewimage', views.viewimage, name="viewimage"),  
 url('^change\_pass', views.change\_pass, name="change\_pass"),  
 url('^uchange\_pass', views.uchange\_pass, name="uchange\_pass"),  
 url('^userhome', views.userhome, name="userhome"),  
 url('^adminhome', views.adminhome, name="adminhome"),  
 url('^feedback', views.feedback, name="feedback"),  
 url('^showfeedback', views.showfeedback, name="showfeedback"),  
 url('^delshowimg(?P<pk>\d+)/$', views.delshowimg, name="delshowimg"),  
 url('^delviewimg(?P<pk>\d+)/$', views.delviewimg, name="delviewimg"),  
 *# url('^compare(P<oimage>\w+)/(P<uimage>\w+)/$', views.compare, name="compare"),* path('showimg/<str:id1>/<str:id2>/', views.showimg, name = 'showimg'),  
 url('^compare', views.compare, name="compare"),  
 *#url('^comparediff', views.comparediff, name="comparediff"),  
 # url('^showimg/<str:param1>/<str:param2>$', views.showimg,name='showimg'),* url('^selectimage(?P<pk>\d+)/$', views.selectimage, name="selectimage"),  
 url('^forgotpassword', views.forgotpassword, name="forgotpassword"),  
 path('imagecompare\_diff',views.imagecompare\_diff,name='imagecompare\_diff'),  
 path('ans',views.ans,name='ans'),  
 url('^viewresult', views.viewresult, name="viewresult")  
]  
  
  
if settings.DEBUG:  
 urlpatterns += static(settings.MEDIA\_URL,document\_root=settings.MEDIA\_ROOT)

**8.System Testing & Resulting**

**Introduction:**

Testing is a process of executing a program with the indent of finding an error. Testing is a crucial element of software quality assurance and presents ultimate review of specification, design and coding. System Testing is an important phase. Testing represents an interesting anomaly for the software. Thus a series of testing are performed for the proposed system before the system is ready for user acceptance testing. The code is tested at various levels in software testing. Unit, system and user acceptance testings are often performed.

**Testing Objectives**

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a probability of finding an as yet undiscovered error.
* A successful test is one that uncovers an undiscovered error.

**Testing Principles**

* All tests should be traceable to end user requirements.
* Tests should be planned long before testing begins.
* Testing should begin on a small scale and progress towards testing in large.
* Exhaustive testing is not possible.
* To be most effective testing should be conducted by a independent third party.\*

The primary objective for test case design is to derive a set of tests that has the highest livelihood for uncovering defects in software. To accomplish this objective two different categories of test case design techniques are used. They are:

* White box testing.
* Black box testing.

# White-Box Testing and Black-Box Testing:

**White Box Testing:** White box testing focus on the program control structure. Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.

**Black Box Testing:** Black box testing is designed to validate functional requirements without regard to the internal workings of a program. Black box testing mainly focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides through test coverage. Incorrect and missing functions, interface errors, errors in data structures, error in functional logic are the errors falling in this category.

# Testing strategies

A strategy for software testing must accommodate low-level tests that are necessary to verify that all small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.There are two general strategies for testing software. They are as follows:

**Code Testing:** This examines the logic of the program. To follow this test, cases are developed such that every path of program is tested.

**Specification Testing:** Specification Testing examines the specification, starting what the program should do and how it should perform under various conditions. Then test cases are developed for each condition and combinations of conditions and to be submitted for processing.

**Levels of Testing**

The stages of Testing Process are:

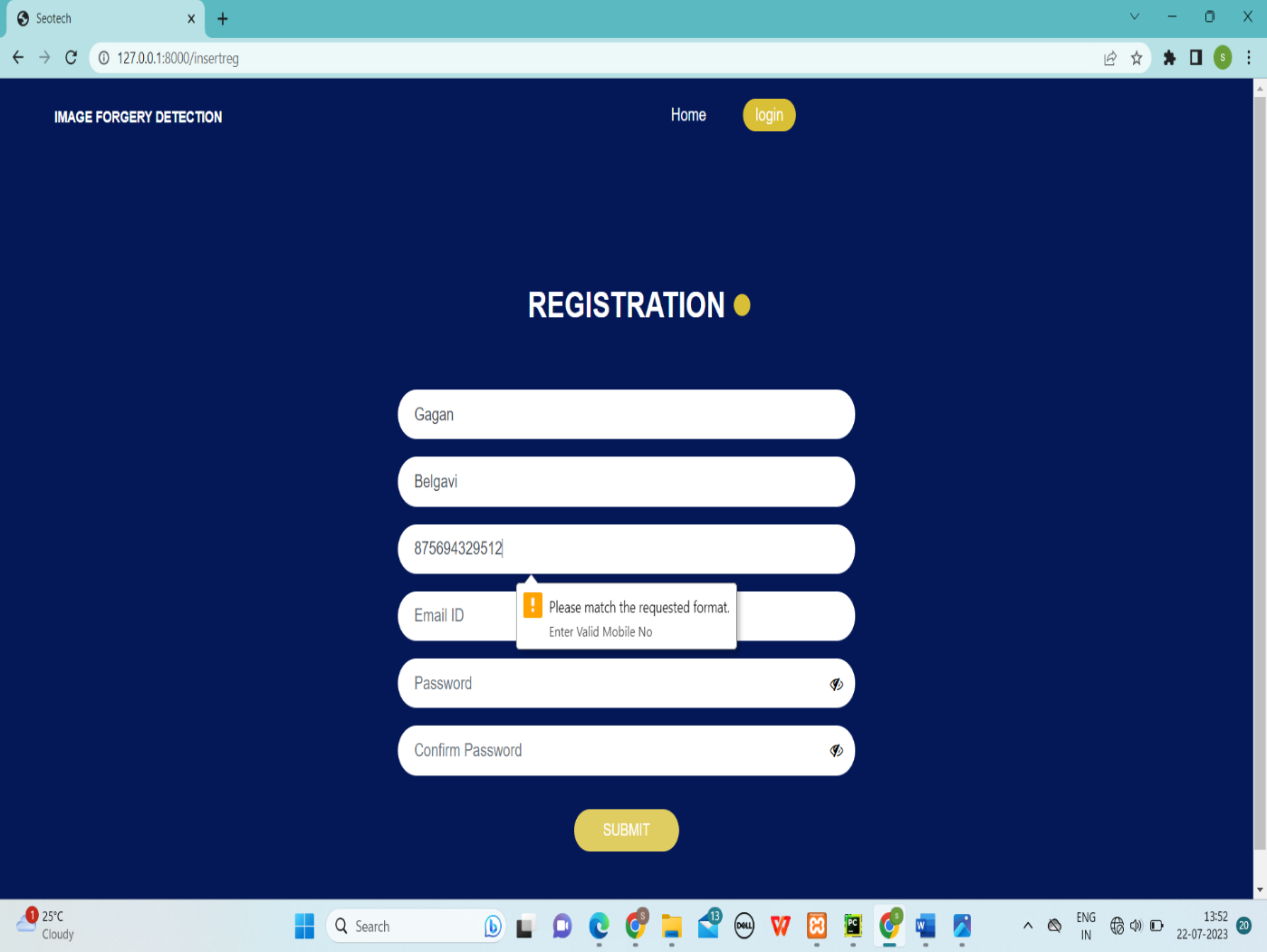
**Unit Testing:** Individual components are tested to ensure that they operate correctly. Each component tested independently without other system components. Ex. Check for Username and Password with the table, after the next module is loaded.

**Integration Testing:** Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. This testing is done using the bottom-up approach to integrate the software components of the software system in to functioning whole.

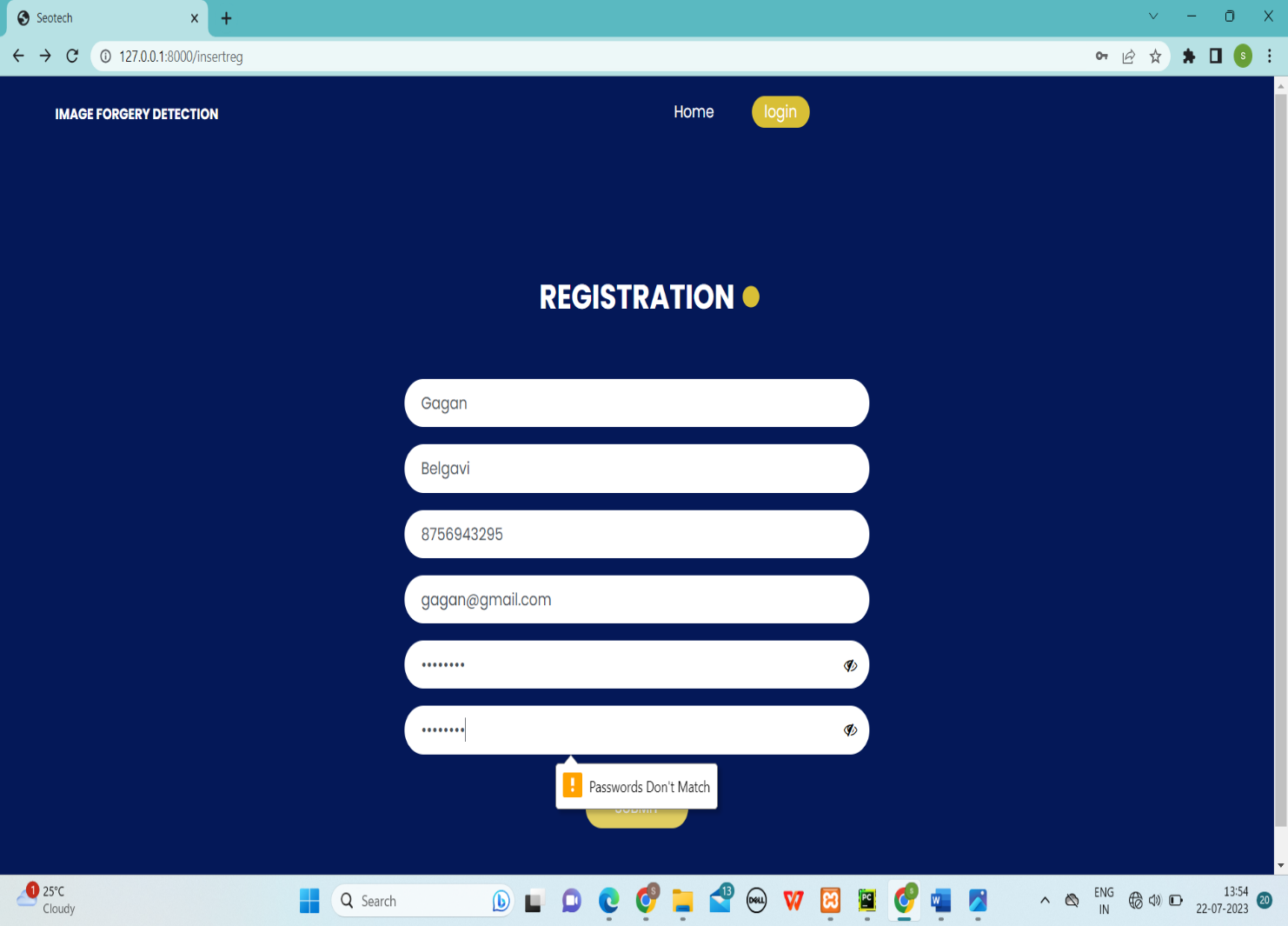
**System Testing:** System testing is actually a series of different tests whose primary purpose is fully to exercise the computer-based system. The system tests that where applied are recovery testing and performance testing. Finally a review or audit is conducted which is a final evaluation that occurs only after operating the system long enough for user to have gained a familiarity with it. System testing was done by the inspection team to verify that all the functionality identified is the software requirement specification has been implemented. Defects that crept in the system has been found defect free and is working well. System testing is concerned with interfaces, design logic, control flow recovery, procedures throughput, capacity and timing characteristics of the entire system. For blank field, alphabets, number and special character validation.

**Acceptance Testing:** User acceptance of the system is the key factor for the success of any system. This is done by user. The system is given to the user and they test it with live data. Acceptance testing involves the planning and execution of functional test. Performance tests, stress tests in order to demonstrate that the implemented system satisfies its requirements. Two sets of acceptance test can be run, those developed by the customer. The system has been tested for its performance at unit level by the individuals through performance testing that is designed to test the run time performance of the software. The performance of the fully integrated system is tested and was found good.

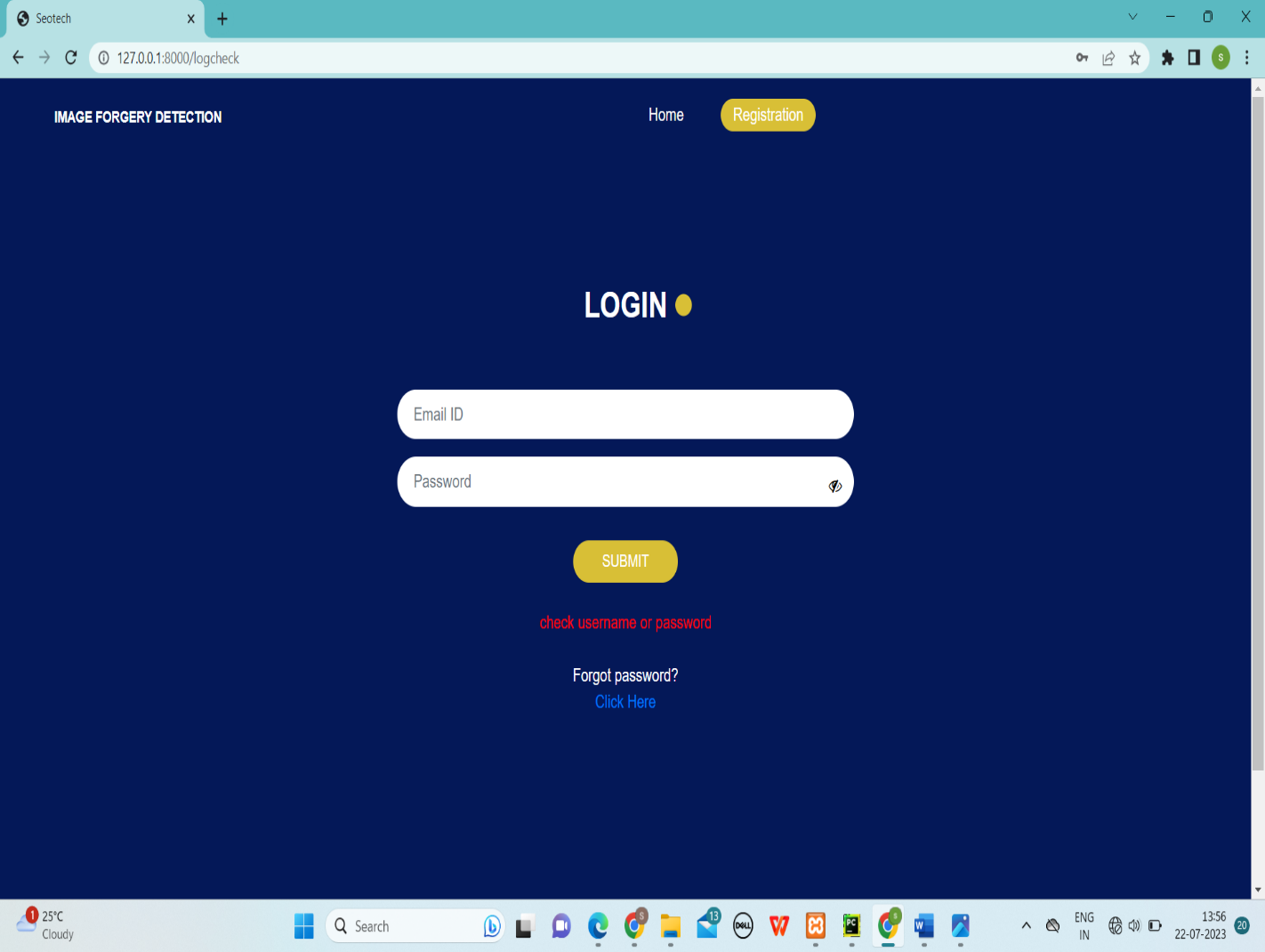
**Validation**

****

**Fig: 8.1 Mobile number validation**

****

**Fig: 8.2 Password validation**

****

**Fig: 8.3 Login page validation**

****

**Fig: 8.4 Change password Validation**

**9.CONCLUSION**

Website is said to have attained its objective only when it need all requirements of the user, further the user himself is the person to judge the success of the system. Every attempt has been made to ensure that the system is fully functional and works effectively and efficiently. The system has been tested with simple data to cover all possible options and checked for all outputs. Since the system is flexible and modular, further modification of this packge can be easily incorporated.

**Importance of the system:**

* Less manual work.
* Increased efficiency.
* Decreases the rate of errors.
* It reduces the time consumption.
* Quick (instant) result.

**Future Enhancement:**

* Mobile Applications can be developed.
* Improved comparision using AI tools.
* Comparion can also be done on black and white pictures.
* Link to facebook, twitter etc.
* Credit debit and net banking payment can be done.

**10.BIBLIOGRAPHY**

* **Books :**
  + “Python Carsh Course” By Eric Matthes(3rd Edition) 2023
  + Ian Summerville “Software Engineering” pearson Education Ltd 10th edition 2021.
* Django Essentials, Samuel Dauzon
* Elias M. Awrad System Analysis and Design, Golgotia 2010.
* Django Design Patterns and Best Practices, Arun Ravindran
* Practical Programming using python 2016.

**Website links:**

1. http://in.youtube.com/
2. <https://www.wikepedia.com>
3. <http://www.stackoverflow.com>
4. [www.tutorialspoint.com](http://www.tutorialspoint.com)
5. <https://learn.coderslang.com/0022-how-to-compare-strings-in-javascript>
6. [www.w3schools.com/](http://www.w3schools.com/)
7. <https://www.instagram.com/reel/CgjesAeAqla/?igshid=YmMyMTA2M2Y>
8. <https://www.w3schools.com/django/django_getstarted.php>
9. <https://www.w3schools.com/django/django_create_app.php>
10. <https://www.tutorialspoint.com/javascript/index.htm>
11. <https://www.udemy.com/share/103J8C3@1LUuWg14Wvx3ayaaLHDTAhmEGMWs4szBaoiU3th5PVK_GfAdalV1pXnULcU_bNP2jg==/>
12. <https://www.javatpoint.com/xampp>
13. <https://www.javatpoint.com/mysql-tutorial>