

User Authentication

Mohammed Fahad C

Authentication is important because it enables organizations to keep their networks secure by permitting only authenticated users (or processes) to access its protected resources, which may include computer systems, networks, databases, websites and other network-based applications or services.

It is a simple application that describes how authentication works.

Stacks

- Html
- Bootstrap
- Php
- MySql

Files

- login.html
- register.html
- login.php
- register.php
- logout.php
- home.php

register.html

This file contains a CDN of Bootstrap for connecting to Bootstrap and a form for registration, it includes email, phone, and password. When click the submit button it will call register.php file.

register.php

This PHP file first read all the user inputs.

```
$name=$_POST["name"];  
$email=$_POST['email'];  
$password=$_POST['password'];
```

Then it will hash the password using password_hash() function

```
$passwordHash=password_hash($password,PASSWORD_DEFAULT);
```

Next step is to connect with database

```
$conn=new mysqli('localhost','root','','auth_form');
```

the root is the user name and not set any password, so just given that as an empty string and auth_form is the name of the database.

The next step is to check whether the connection established or not if any error occurs show appropriate messages to the user.

If the connection established successfully then first check whether the email has given already exists or not

```
$sql=" select * from users where email='$email'";
```

if exist show error message otherwise store all the data to the database and navigate to login.html

```
header("Location:http://localhost/auth_form/login.html");
```

login.html

This file contains a CDN of Bootstrap for connecting to Bootstrap and a form for registration, it includes email and password. When click the submit button it will call login.php file.

login.php

This PHP file first read all the user inputs. Then it will hash the password using password_hash() function

Next step is to connect with database

The next step is to check whether the connection established or not if any error occurs show appropriate messages to the user.

If the connection established successfully then first check whether the email has given already exists or not

```
$sql=" select * from users where email='$email' ";
```

if not exist show message *"User not exist"*.

If exist, verify the password given with the actual password using password_verify() function

```
$verify=password_verify($password,$pswd);
```

If password matches it will create a session using the name of the user and navigate to home.php

```
$_SESSION['name']=$user_name;
```

```
header("Location:http://localhost/auth_form/home.php");
```

if not matches the passwords show message "*Incorrect password*"

home.php

When loading this file it will check for the session, if the session does not exist it will redirect to login.html

```
if($_SESSION['name']==null){
```

```
header("Location:http://localhost/auth_form/login.html");
```

```
}
```

If the session exists show a welcome message with the name of the user and a logout button.

```
<a href="logout.php">Logout</a>
```

```
<h1>Welcome <?php echo $_SESSION['name']; ?> </h1>
```

When clicking the logout button it will navigate to logout.php.

logout.php

It will destroy the session and redirect to the login page.

```
session_destroy();
```

```
header('location:login.html');
```

Documentation

Login/Signup

what are you?

Developer

Stacks:

- ◆ HTML
- ◆ PHP
- ◆ CSS
- ◆ JavaScript
- ◆ Bootstrap

Resources:

GitHub: <https://github.com/pranchi3578/kingslab-registration-login-php>

Bootstrap-Setup: <https://getbootstrap.com/docs/3.3/getting-started/>

Ref: <https://www.youtube.com/watch?v=5GcQtLDGXy8>

Files:

Html:

home.html
login.html
welcome.html

Php:

home.php
Login.php
Session.php
Logout.php

Css:

home.css

home.html

This html page addresses user Signup. Bootstrap is integrated to the file using CDN. Reference to this is provided in the Resources above.

For Sign-Up purpose username, phone number (unique), password are collected from the user side(As of now no constraints are set on any of these fields at the user side).

On submission the details entered in the form are stored in MySql Database by \$POST method. Refer “Database” for more understanding.

home.php

Content passed from home.html by \$POST method is processed here:

```
$phone=$_POST['phone'];  
  
$password=md5($_POST['password']);  
  
$username=$_POST['username'];
```

Connection to MySql Database is set up here:

```
$conn=new mysqli('localhost','root','','sample');
```

If connection is success \$conn value will become true, and data is written into the database using the code snippet

```
$stmt=$conn->prepare("insert into registration(username,phone,password)values(?,?,?)");  
  
$stmt->bind_param("sss",$username,$phone,$password);  
  
$stmt->execute();  
  
echo "registration success..";
```

Now close the streams using obj-->close();

login.html

This is a cone of the SignUp page. Phone number and Password is collected from the user side. Phone number, rather than username because it is set as unique in the DataBase.

login.php

Phone number and password collected from user side is processed here.

The row which happens to meet the conditions specified in the query is returned. If the number of rows returned is exactly one then the user is a valid user. This is depicted by the code below:

```
$sql="SELECT * FROM registration WHERE phone='".  
$phone.'"AND password='". $password.'"';
```

```
$result = mysqli_query($conn, $sql);  
if(($result->num_rows)==1){  
$_SESSION['phone']=$phone;  
header('location:welcome.html');  
}
```

If the user is successfully logged in, we create a session variable and initialize it with a value fetched from the DB or a value generated using the same. This is later saved in cookies. use-case: restricting authorized access to certain pages.

Here since “phone “ is the unique variable, we store it in a session variable ‘phone’

```
$_SESSION['phone']=$phone;
```

Session.php

```
session_start();  
echo "the logged in user's phone no (unique token) " .  
$_SESSION['phone'];
```

This file simply echos or displays a text with the Session variable,
This implies that the session variable is accessible from another php page.

Welcome.html

If the user is successfully logged in, he/she is directed to this html page. The page gives the user two options. Either to view the session value, or to logout.

If the user chooses to logout , logout.php is executed

Logout.php

Here the session is unset or destroyed, and the user is redirected to login.html .

PHP MySQL Authentication System

Sabin Saleem



User authentication is very common in modern web application. It is a security mechanism that is used to restrict unauthorized access.

In this Project we'll create a simple registration and login system using the PHP and MySQL. This Documentation is comprised of two parts: in the first part we'll create a user registration form, and in the second part we'll create a login form, as well as a welcome page and a logout script.

• Building the Registration System

In this part we'll build a registration system that allows users to create a new account by filling out a web form. But, first we need to create a table that will hold all the user data.

✓ Step 1: Creating the Database Table

```
CREATE TABLE users ( id INT NOT NULL PRIMARY KEY  
AUTO_INCREMENT, username VARCHAR(50) NOT NULL UNIQUE,  
password VARCHAR(255) NOT NULL
```

Create the Database Table by the above query or you can directly make it by using the admin interface of mysql database.

✓ Step 2: Creating the Config File

After creating the table, we need create a PHP script in order to connect to the MySQL database server. The code needed to connect the Project to the Database is shown below.

```
$mysqli = new mysqli("localhost","root","","userlogin");  
  
// Check connection  
if ($mysqli->connect_errno) {  
    echo "Failed to connect to MySQL: " . $mysqli->connect_error;  
    exit();  
}  
  
if(!mysqli_select_db($mysqli,'userlogin'))  
{  
    echo 'Not Selected the Database';  
}
```

✓ Step 3: Creating the Registration Form

Let's create another PHP file "register.php" and put the following code in it. This example code will create a web form that allows user to register themselves.

```
$name = $_POST['user'];
$pass = $_POST['password'];
$pass = md5($pass);

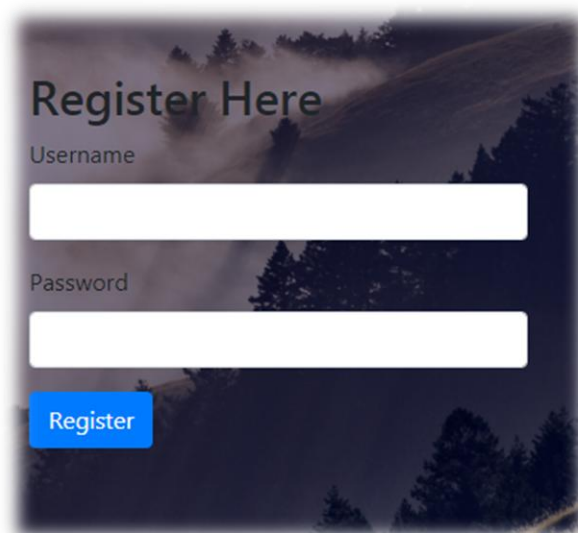
$s = "select * from usertable where name = '$name'";

$result = mysqli_query($mysqli, $s);

$num = mysqli_num_rows($result);

if($num == 1){
    echo "User already exist";
}else{
    $reg= "insert into usertable(name , password) values ('$name' , '$pass')";
    mysqli_query($mysqli, $reg);
    echo" Registration Succesfull";
}
```

We used md5 Encryption for storing password in the form of hash value in the database. Username and password is saved in the database while a new user creates his/her account and also we make sure that the username is not already exists in the database also.



The image shows a web form titled "Register Here" set against a dark, atmospheric background of a forest at night. The form consists of two white input fields. The first field is labeled "Username" and the second is labeled "Password". Below these fields is a blue button with the word "Register" in white text.

• Building the Login System

In this part we'll create a login form where user can enter their username and password. When user submit the form these inputs will be verified against the credentials stored in the database, if the username and password match, the user is authorized and granted access to the site, otherwise the login attempt will be rejected.

✓ Step 1: Creating the Login Form

```
$name = $_POST['user'];
$pass = $_POST['password'];
$pass = md5($pass);

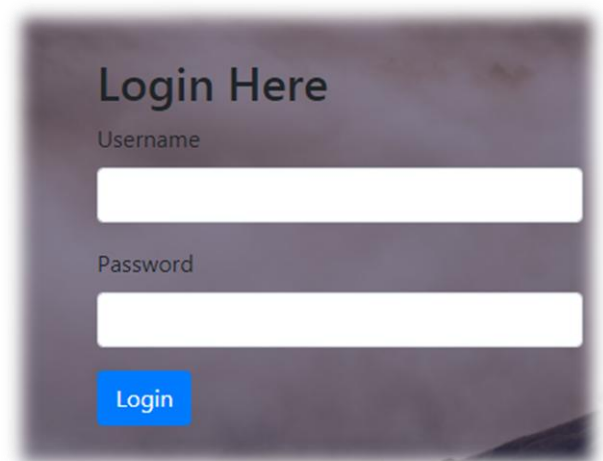
$s = "select * from usertable where name = '$name' && password = '$pass'";

$result = mysqli_query($mysqli, $s);

$num = mysqli_num_rows($result);

if($num === 1){
    header('location:home.php');
}else{
    header('location:login.php');
}
```

As you can see the above code if the user entered input matches the user and password stored in the database then the user will be redirected to the home.php page which is the homepage for the user. Otherwise the user will remain in the login page due to an invalid input.

A login form titled "Login Here" is shown. It has two input fields: "Username" and "Password". Below the "Password" field is a blue button labeled "Login". The form is set against a dark, textured background.

✓ Step 2: Creating the Logout Script

Now, let's create a "logout.php" file. When the user clicks on the log out or sign out link, the script inside this file destroys the session and redirect the user back to the login page.

```
logout.php
1  <?php
2
3  session_start();
4  session_destroy();
5
6  header('location:login.php');
7
8  >
```

When the user logouts the session that was created will be destroyed and redirects to the login.php page as you see it from the above code.

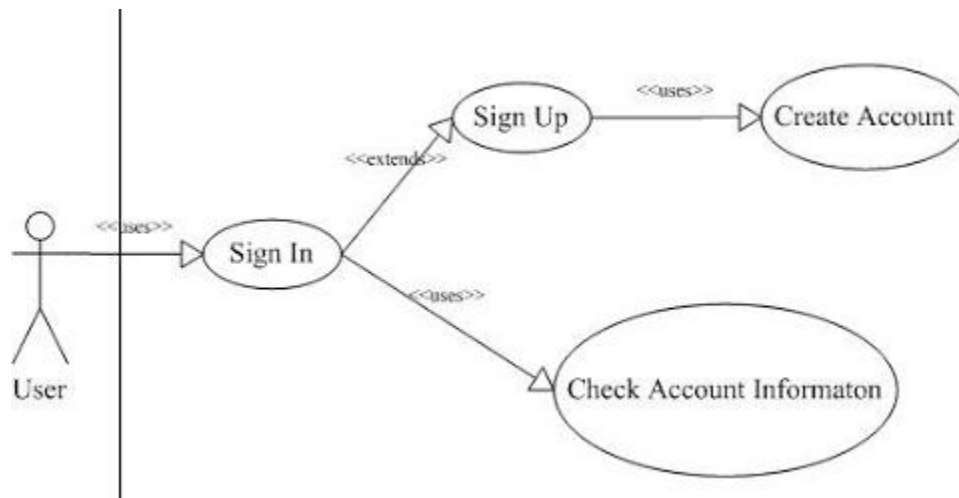
[LOGOUT](#)

Welcome

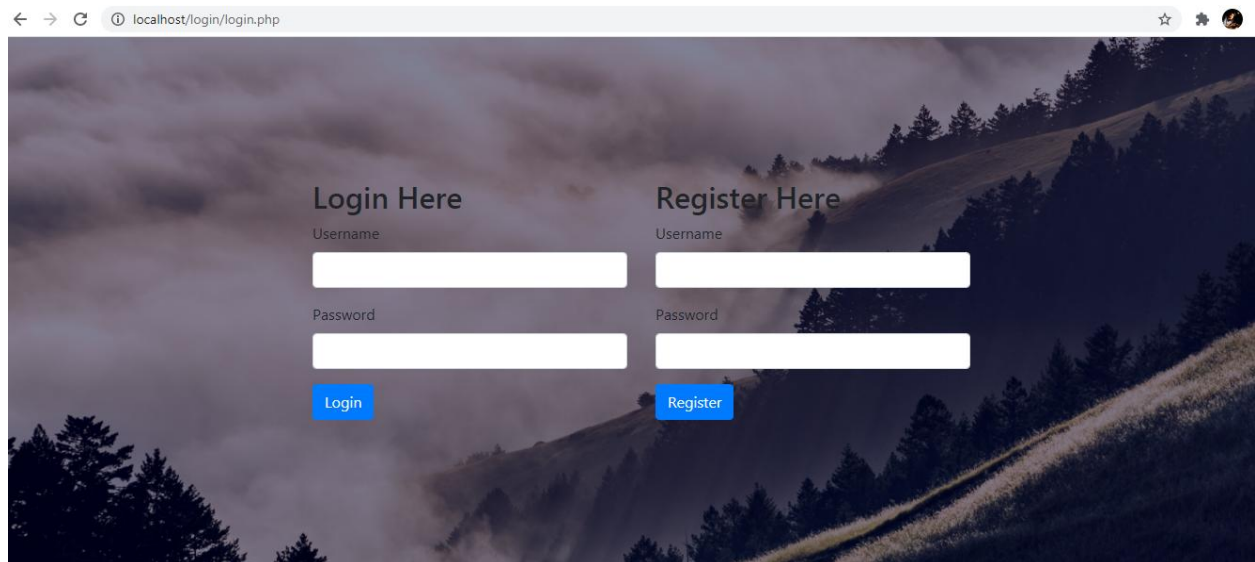
When a user login successfully they will directed to a page like this as shown above with a logout link at the top of the browser to end the session.

```
home.php
1  <?php
2  session_start();
3  >
4  <!DOCTYPE html>
5  <html lang="en">
6  <head>
7      <meta charset="UTF-8">
8      <meta name="viewport" content="width=device-width, initial-scale=1.0">
9      <title>Homepage</title>
10 </head>
11 <body>
12     <a href="logout.php"> LOGOUT </a>
13 <h1>Welcome </h1>
14 </body>
15 </html>
```

UML Diagram of the Task Performed



Website Outlook



SIGN BOARD TRANSLITERATION

OVERVIEW

The goal of this project is to develop an interface that transliterates the text written on a signboard from one script to another as desired by the user. An average traveller, on a business or pleasure trip, often gets confused by the various signboards written in an unfamiliar language in a new region. The scenario is so common that people have started considering Hindi in day-to-day language usage in the Indian metropolis. So, the signboards may allow the text to be written in 1-2 languages only, in which the Hindi language is a must. Our work intends to build up an application that can recognize the Hindi content present on signboard pictures captured using any device, transliterate the content from Hindi to English, and display the transliterated English text back onto the screen of the phone. We will design a system that works for Hindi text names such as road names, city names, organization names shop names, etc. In the future, more languages can be supported. Experiments have been conducted on various signboard pictures and the outcomes demonstrate the viability of the proposed approach.

In this work, we concentrate on recognizing text on signs. The signboard text transliterate application uses a smartphone to capture the signboard image containing signs, recognizes it, and transliterate it into user specified language. While developing the project, we have decided to use an agile software development methodology. For training such an engine we need parallel data between the two languages. More specifically, for training a transliteration engine for Hindi-to-English we need training data containing multiple English-Hindi word pairs (for example, Mumbai- मुंबई, Chennai- चेन्नई and so on). Such parallel data is not easily available and we will have to collect such data as a part of the project. Because of the limited time and available resources, datasets of all Vernacular Languages are not gatherable. Acknowledging Hindi as a day-to-day language in the Indian metropolis, we are for the time being considering only Hindi-to-English transliteration. Detection and recognition of content from common scenes are requirements for a signboard text transliteration. The main challenge lies in the assortment of text: it can vary in text style, size, font, and orientation. There may also blur in the text and can be blocked by other objects in the signboard. As we know signs exist in three-dimensional space, content on signs can be miss-shaped by inclination, tilt, and shape of objects on which they are found.

Numerous OCR frameworks function very well on high-quality images; however, they perform poorly for signboard images because of the low-quality nature of them. The proposed approach uses a Yolov3 which is the latest variant of a popular object detection algorithm YOLO (You only look once) real-time object detection system. Yolov3 is used on darknet framework to detect the text from the acquired image and it gives a bounding box around the detected objects. Recognition is done using Convolutional Recurrent Neural Network (CRNN), a combination of CNN, RNN. This reads an example image and recognizes

its text content. And finally, transliteration is done. Transliteration module provides a way to transliterate from Hindi to the English Language. We have successfully applied the proposed approach to a Hindi-English sign text translation framework, which can recognize Hindi signs caught from a camera, and translate the recognized text into English. To our knowledge, Hindi to English signboard translator has not been investigated previously.

We have used Anvil as front-end. The detected text with the bounding box and the transliterated output also will be shown.

Sample codes

- We use yolo v3 framework for the detection of the text from an image. Below given image is the code used for converting annotations:

```
import os
import math
import fnmatch
from PIL import Image
new_list = []
for root, dirs, files in os.walk("obj"):
    for fil in files:
        f=open(os.path.join(root, fil), 'r')
        i=0
        l=Image.open("obj1/"+fil.replace(".txt", ".jpg"))
        while True:
            i=i+1
            new_list=""
            line=f.readline()
            if not line:
                break
            li=list(map(str,line.split()))
            x1,x2,x3,x4,y1,y2,y3,y4 =(float(li[0]),float(li[1]),float(li[2]),float(li[3]),
            float(li[4]),float(li[5]),float(li[6]),float(li[7]))
            s=li[8]
            print(s)
            coords=(x1,y1,x3,y3)
            cropped_image = l.crop(coords)
            try:
                cropped_image.save("obj_new1/"+fil.replace(".txt", "("+str(s)+").jpg"))
            except:
                print("error occured")
```

At first starting of the project itself we downloaded pre-trained yolo weights. Below is the code:

```
# get yolov3 pretrained coco dataset weights
```

```
!wget https://pjreddie.com/media/files/yolov3.weights
```

```
# define helper functions
```

```
def imShow(path):  
    import cv2  
    import matplotlib.pyplot as plt  
    % matplotlib inline  
    image = cv2.imread(path)  
    height, width = image.shape[:2]  
    resized_image = cv2.resize(image,(3* width, 3* height), interpolation = cv2.INTER_CUBIC)  
    fig = plt.gcf()  
    fig.set_size_inches(18, 10)  
    plt.axis("off")  
    plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))  
    plt.show()  
  
# use this to upload files  
def upload():  
    from google.colab import files  
    uploaded = files.upload()  
    for name, data in uploaded.items():  
        with open(name, 'wb') as f:  
            f.write(data)  
        print ('saved file', name)  
  
# use this to download a file  
def download(path):  
    from google.colab import files  
    files.download(path)
```


- For Recognition we use Keras framework with CRNN with CTC decoder model. Below is the recognized text from image example:

The demo reads an example image and recognizes its text content.



Expected output:

```

Loading model...
Model loaded from ../model/crnn_demo/model.t7
Recognized text: available (raw: a-----v--a-i-l-a-bb-l-e---)

```



```

Recognized text: shakeshack (raw: ss-h-a--k-e-ssh--aa-c--k--)

```

- For transliteration we used Indic-trans framework. Code is shown below.

```

[ ] import anvil.server

    anvil.server.connect("2RJPITLOZUHVQH35JI3TK56F-GPYSLJPZ66SWEPHY")

[ ] import anvil.media
    @anvil.server.callable
    def classify_image(file):
        with anvil.media.TempFile(file) as filename:
            a=outout(filename)
            h=anvil.media.from_file("predictions.jpg", ["jpeg"])
            !rm predictions.jpg
            return(a,h)

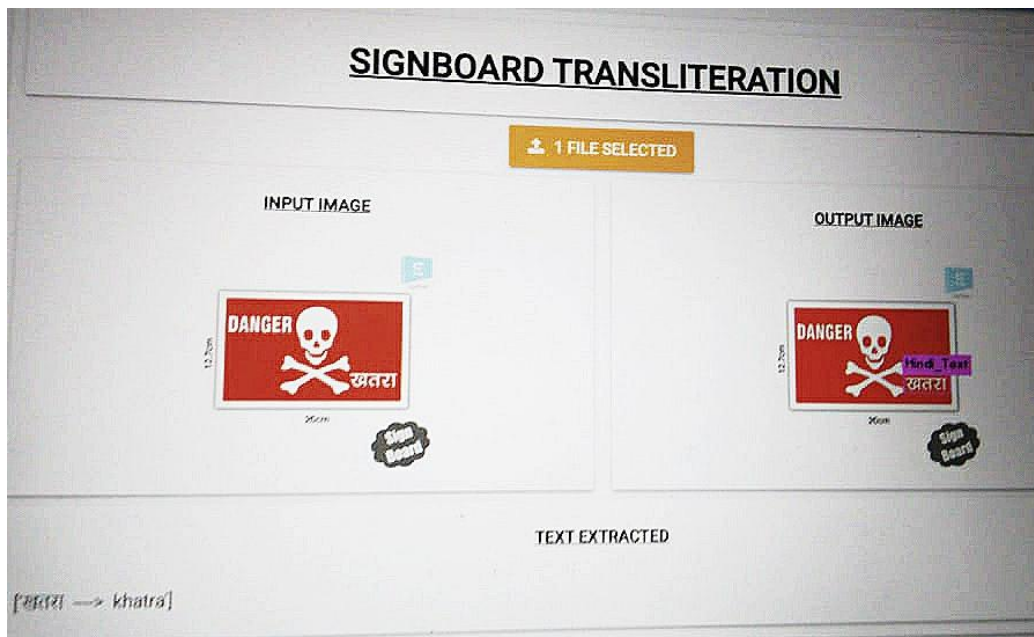
[ ] """b=outout("/content/dan.jpg")"""
    ↗ 'b=outout("/content/dan.jpg")'

[ ] """print(b)"""
    ↗ 'print(b)'

```

The example of output is given below:

AS you can see the bounding box is shown in the output image as Hindi text(right image) and the transliterated word is given below as Text Extracted.



Recommended System Requirements

- Windows 7 or Higher
- Intel Pentium 4 or higher processor
- RAM 2GB(Minimum)
- Storage space of 2GB(minimum)
- Graphics 1GB(minimum)
- Python IDE - A python script is used to convert the annotation.
- Stable Internet Connection
- Google account for using Colab

How To Use:

The pre-trained weight are given into the code.

1. Open the .ipynb notebook in the collabatory
2. The input image can be uploaded into the specified folder in Google drive.
3. Now selection the run tab from the top view pane and click run all.
4. When asked to authorise your Google account, copy and paste the generated code.
5. Now after completing executing all cells, there will be a link given as inside brackets of anvil.connect.server
6. Copy and paste the link in anvil site
7. Now the result will be shown

Further information about the tasks can be found at:

<https://ai4bharat.org/>

<https://pjreddie.com/darknet/yolo/>

<https://github.com/theAlGuysCode>

Color Detection Using OpenCV Python Project

OpenCV is a Computer Vision library. It is a collection of C functions with a few C++ classes that implement popular Image Processing and Computer Vision algorithms. Computer vision is the science that means to give a comparative, if not better, capacity to a machine or PC. Computer vision is worried about the programmed extraction, investigation and comprehension of valuable data from a single picture or a grouping of pictures. Some of the basic image processing capabilities include filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms and many more. Color detection using OpenCV has many advantages like, it allows the detection of a specific color in a livestream video content. In this OpenCV color detection system there are four major modules, activated webcam, scan object, match frame parts and system results. Users can open webcam by clicking the webcam button. Then the algorithm analysis the pattern of the framed part of webcam. Pattern is matched with defined color pattern by RGB color model. If the pattern matched with the potential pattern of RGB color model then the system results with the correct output.

❖ **Modules:**

The system comprises of 4 major modules as follows:

- **Activate Webcam:**
 - User opened the webcam by clicking button on screen
- **Scan object part within the camera frame:**
 - Algorithm analysis the pattern of framed part.
- **Matching framed part:**
 - Pattern matched with defined colour pattern by rgb colour model
- **System result:**
 - If pattern matched with potential pattern of rgb colour model then system output the correct result

Project Lifecycle:

Description

The waterfall Model is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach that was used for software development.

❖ **Hardware Requirement:**

- i3 Processor Based Computer or higher
- Memory: 1 GB RAM
- Hard Drive: 50 GB
- Monitor
- Internet Connection

❖ **Software Requirement:**

- Windows 7 or higher
- Python
- Django
- MySQL database

Documentation

Todo List application

Languages used:

HTML

CSS

JavaScript

Resources:

Bootstrap 4

Github

visual studio code

Objective

this is very simple web application designed with a very minimal requirements and functionalities, this contains a single html page which shows an input field to edit and an add button to add the item entered in input field, once we entered, the new item will be appended below

task3.html

This is the only file used for this project, but this file is further divided into parts to enhance this project with different functionalities and to bring the different languages used within this file

<html>

this represents the root of an HTML document and every code of this project is written within this tags

<link>

this tag defines the relationship between the current document and an external resource. here it is used to link to the external css open resource document: **Bootstrap**

<link

```
rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-9alt2nRpC12Uk9gS9baDI411NQApFmC26EwAOH8WgZl5MYxHfFc+NcPb1dKGj7Sk"
crossorigin="anonymous"
/>
```

<style>

used to define style information (CSS) for a document. Inside the <style> element you specify how HTML elements should render in a browser. here we can customize the style of various html elements and classes, id's etc (if they are defined in html tag attributes)

```
.itemlistdata {
margin-bottom: 0px;
padding-top: 5px;
margin-right: auto;
width: 100%;
}
```

<body>

The <body> tag defines the document's body. The <body> element contains all the contents of an HTML document, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

<div>

The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript. The <div> tag is easily styled by using the class or id attribute.

Attributes

HTML attributes are special words used inside the opening tag to control the element's behaviour. HTML attributes are a modifier of an HTML element type. some of the attributes we used in this project are:

Attribute Description

id	Specifies a unique id for an element
class	Specifies class for an element, which can be used for styling
type	specifies the type of input elements,
onClick()	onclick attribute is part of the Event Attributes, it fires on a mouse click

ANGULAR ROUTING

AngularJS supports SPA(Single page application) using routing module `ngRoute`. This routing module acts based on the url. When a user requests a specific url, the routing engine captures that url and renders the view based on the defined routing rules.

We will be building an application, which will display a login page when a user requests for base url – “<http://localhost/>.” Once the user logs in successfully, we will redirect it to student page <http://localhost/student/{username}> where username would be logged in user's name.

In our example, we will have one layout page - [index.html](#), and two HTML templates - [login.html](#) and [student.html](#).

1. [Index.html](#) - layout view
2. [login.html](#) - template
3. [student.html](#) – template

1. The first step is to include [angular.js](#), [angular-route.js](#), and [bootstrap.css](#) in the [index.html](#). The [angular-route.js](#) includes necessary functions for routing.
2. Apply [ng-app](#) directive.
3. Apply [ng-view](#) directive to `<div>` or other elements where you want to inject another child view. AngularJS routing module uses [ng-view](#) directive to inject another child view where it is defined. Therefore, Angular will inject [login.html](#) or [student.html](#) inside this div element.
4. Now, create an application module and specify '[ngRoute](#)' as a dependency module.
5. Now, we need to configure the routing rules that need to compile before any other module of an application. So, use [config\(\)](#) method to configure the routing rules using [\\$routeProvider](#) object.
6. Use [\\$routeProvider.when\(path, route\)](#) method to configure routing rules, where the first parameter is request URL and the second parameter is an object which contains controller, template, or other properties. In the above example, we specified that if user request for "/" URL, meaning the base url then inject

`login.html` and `loginController`. In the same way, if a user requests for `"/student/:username"` url then inject `student.html` and `studentController`. The `:username` would be url parameter.

7. Use `otherwise()` method to redirect to base url if user request for the URL other than configured rules.
8. Now, define `loginController` which attaches `authenticate()` function to the `$scope`. The `authenticate()` method redirects to `"/student/username/"` using `$location` service.
9. Define `studentController` which attaches `username` property to `$scope`, to display user name in the view. Notice that `$routeParams` is used to get the value of url parameter supplied from login view.

Create `login.html` which contains username and password input box with validation. Here we using `bootstrap.css`.

Notice that `login.html` and `student.html` starts from `<form>` tag, because they are going to be injected in to layout page - `index.html`. The layout page already contains head and body tag.

WORKING:

When you run the application, it will display a login page.

Once you enter username and password, it will display student page with supplied username.

To redirect to `'/student'`, the url would be `"http://localhost/#/student"`

After click logout it will redirect to login page. And continues process.