# SYSTEM DESIGN

QUESTION PAPER

GENERATION

FACE DETECTION

FACE DETECT

**ARCHITECTURAL DIAGRAM**

LOGIN PAGE

USER

REGISTRATION

DATABA

SE

PAPER SUBMITTED

START TEST

# CONTEXT LEVEL DIAGRAM:

ADMIN

**Level 0**

Registration exam info

Exam info

ONLINE EXAMINATION

User

Admin

# DATAFLOW DIAGRAM

**LEVEL 1**

SELECT PREFFERED MODULE

DATABASE

LOGIN PAGE

START TEST

result

**LEVEL 2**

USER

DATABASE

ANSWER SCRIPT AND SCORE

REGISTRATION/ LOGIN

SELECT MODULE

START TEST

DETECT FACE

**YES**

FACE NOT

DETECTED?

GENERATE QUESTION PAPER

GENERATE ALERT MESSAGE

### MODULE DESCRIPTION

The module description of the system provide information about module that are provided in the system and its components which can be accessed in different manner . The “ONLINE APTITUDE EXAMINATION” system consists of two module :

* + - Admin
    - User

### ADMIN MODULE

In admin module the admin can login by providing the administration login details and can view the registered users , their score and the questions that has been asked by the users .the admin can clear the doubts of the users by answering their questions .

### USER MODULE

The user can register using the registration form and login to the system by thie username and password they are allowed to view their profile . They can take up the test and can view the test result immediately

## TEST CASES AND RESULTS

### TEST CASE 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project name :** Online Aptitude Test | | | | |
| **Test Case Id :** OAT 1 | | **Test Priority :** Medium | | |
| **Module name :** User login | | **Test Title :** Login Check | | |
| **Preconditions :** Provide valid User name and password | | **Test Description :** The user name and password retrieved from database | | |
| **Test Data** | **Expected Result** | **Actual Result** | **Status** | **Notes** |
| User name : pavi@20  Password: dharpavi | Login successful | Login successful, Move to next page | Pass |  |
| **Post conditions :** The user is validated with the data in the database | | | | |

**TEST CASE 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project name :** Online Aptitude Test | | | | |
| **Test Case Id :** OAT 2a | | **Test Priority :** Medium | | |
| **Module name :** Registration | | **Test Title :** Registration Check | | |
| **Preconditions :** Provide valid | | **Test Description :** Checks and validates the information provided by the user | | |
| **Test Data** | **Expected Result** | **Actual Result** | **Status** | **Notes** |
|  | Registration successful | Phone exceeds the mentioned number of digits | fail |  |
| **Post conditions :** The details are not stored in the database | | | | |

### TEST CASE 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project name :** Online Aptitude Test | | | | |
| **Test Case Id :** OAT 2b | | **Test Priority :** Medium | | |
| **Module name :** Registration | | **Test Title :** Registration Check | | |
| **Preconditions :** | | **Test Description :** Checks and validates the information provided by the user | | |
| **Test Data** | **Expected Result** | **Actual Result** | **Status** | **Notes** |
|  | Registration successful | Registration successful | pass |  |
| **Post conditions :**The registered user details are stored in the database. | | | | |

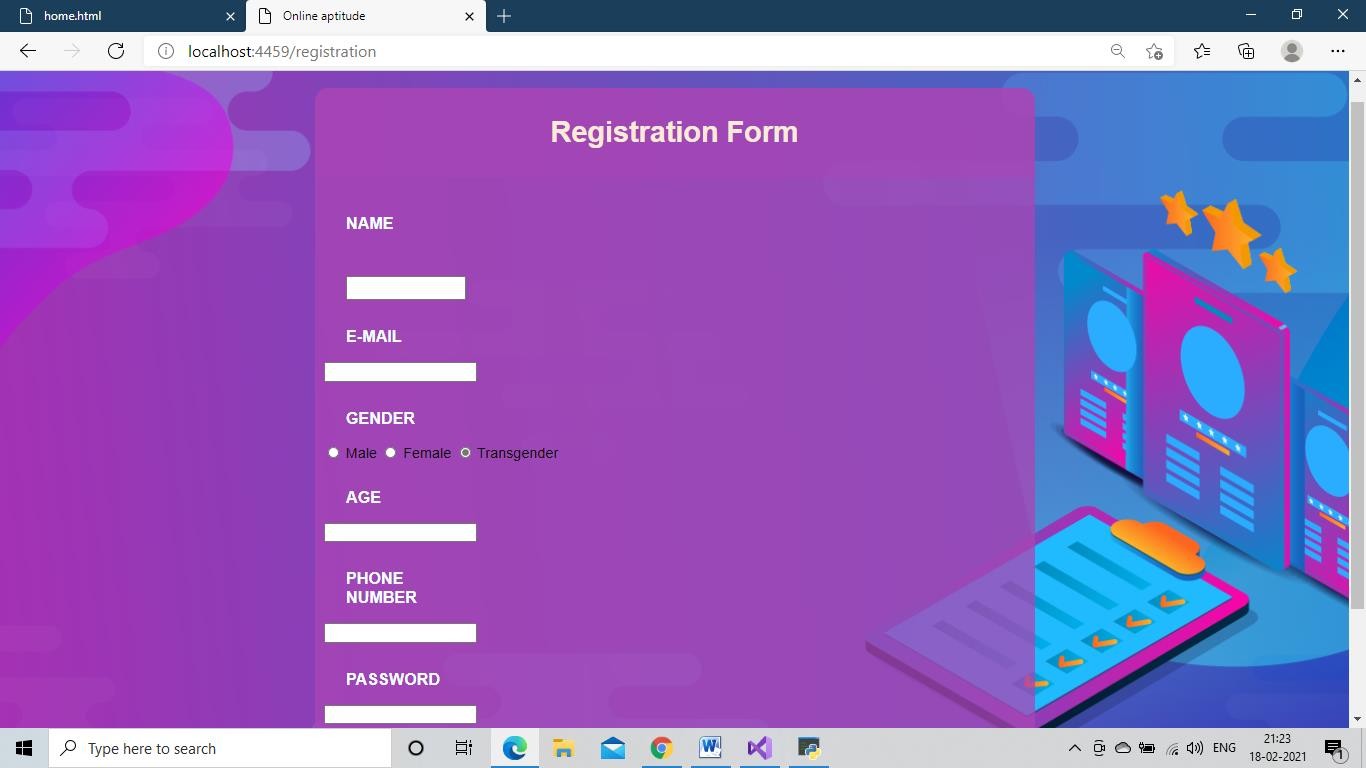
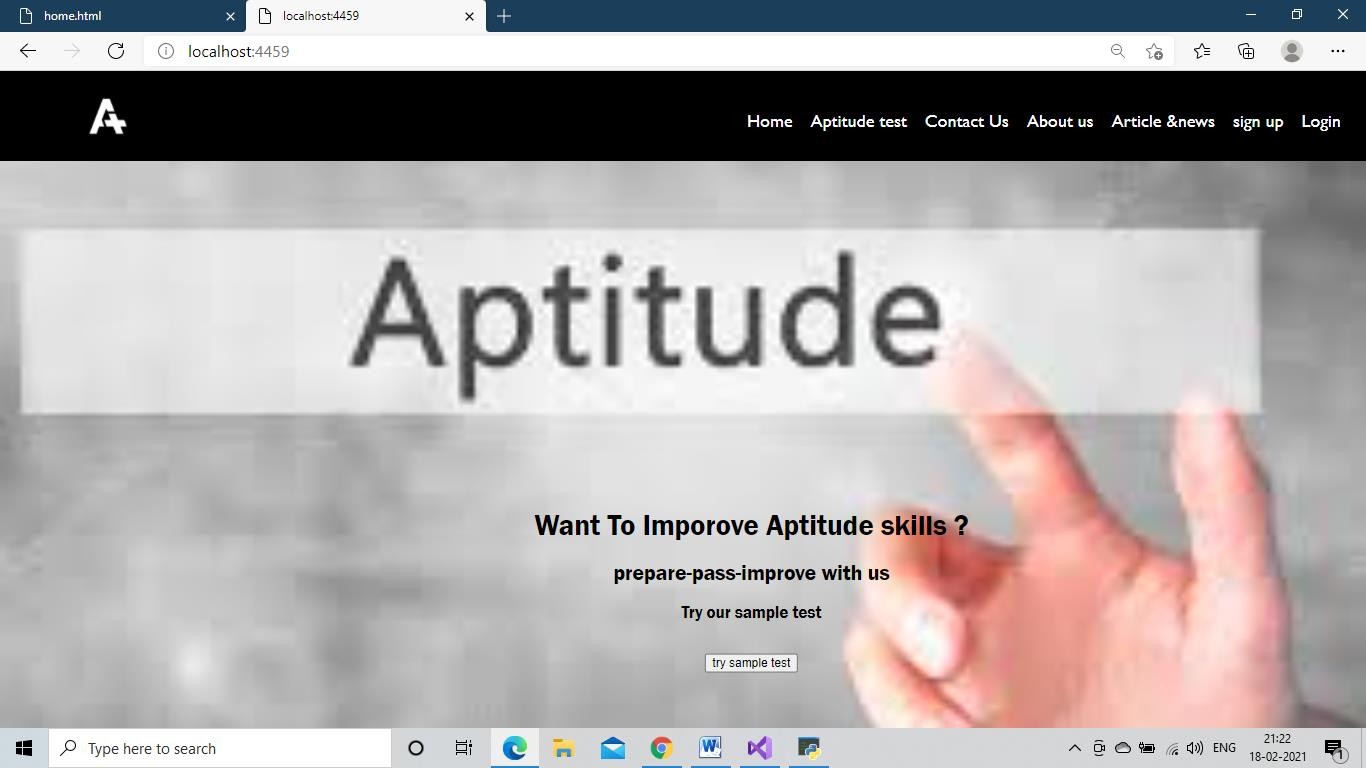
**TEST CASE 4**

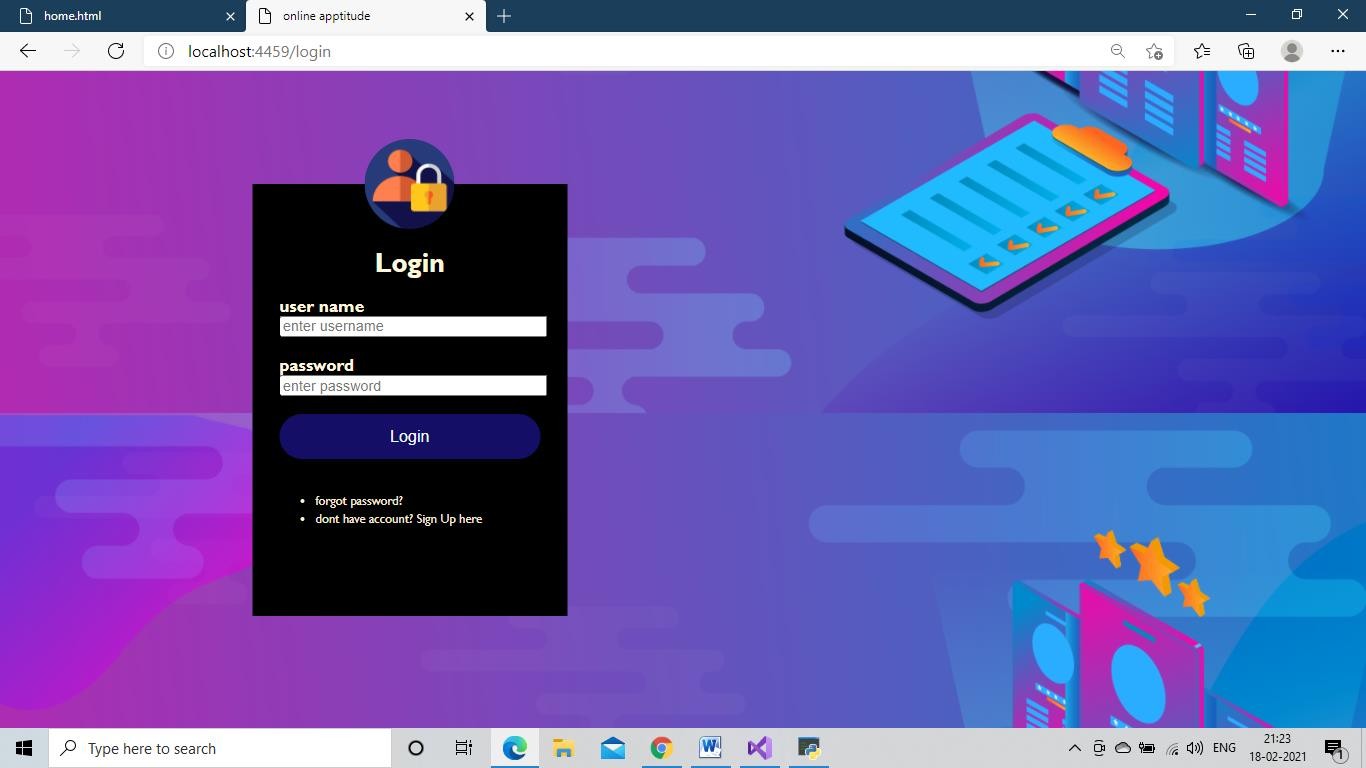
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project name :** Online Aptitude Test | | | | |
| **Test Case Id :** OAT 3 | | **Test Priority :** Medium | | |
| **Module name :** Exam | | **Test Title :** Timer check | | |
| **Preconditions :**complete the exam within the stipulated time. | | **Test Description :** stops the exams when the timer goes off | | |
| **Test Data** | **Expected Result** | **Actual Result** | **Status** | **Notes** |
| -00.00 | time up | time up | pass |  |
| **Post conditions :**The result of the exam will be stored in the database. | | | | |

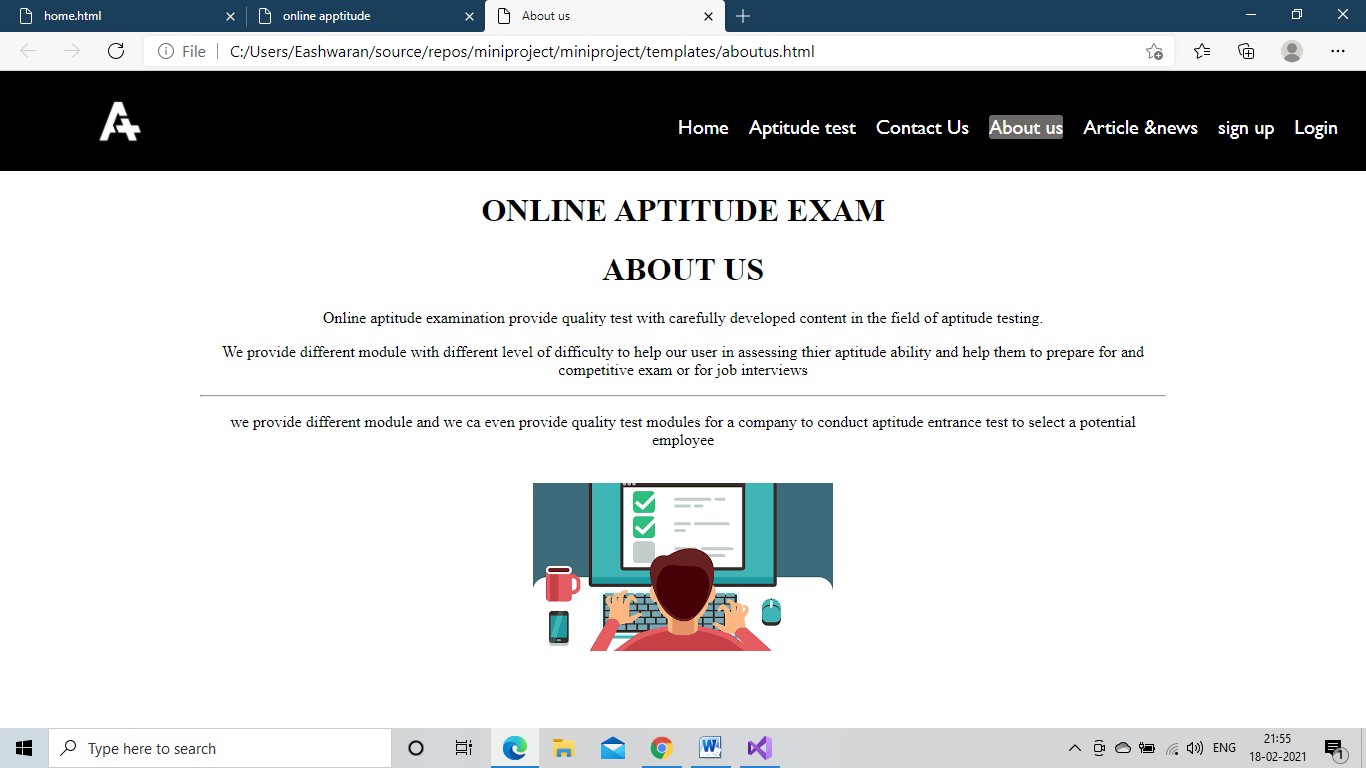
**TEST CASE 5**

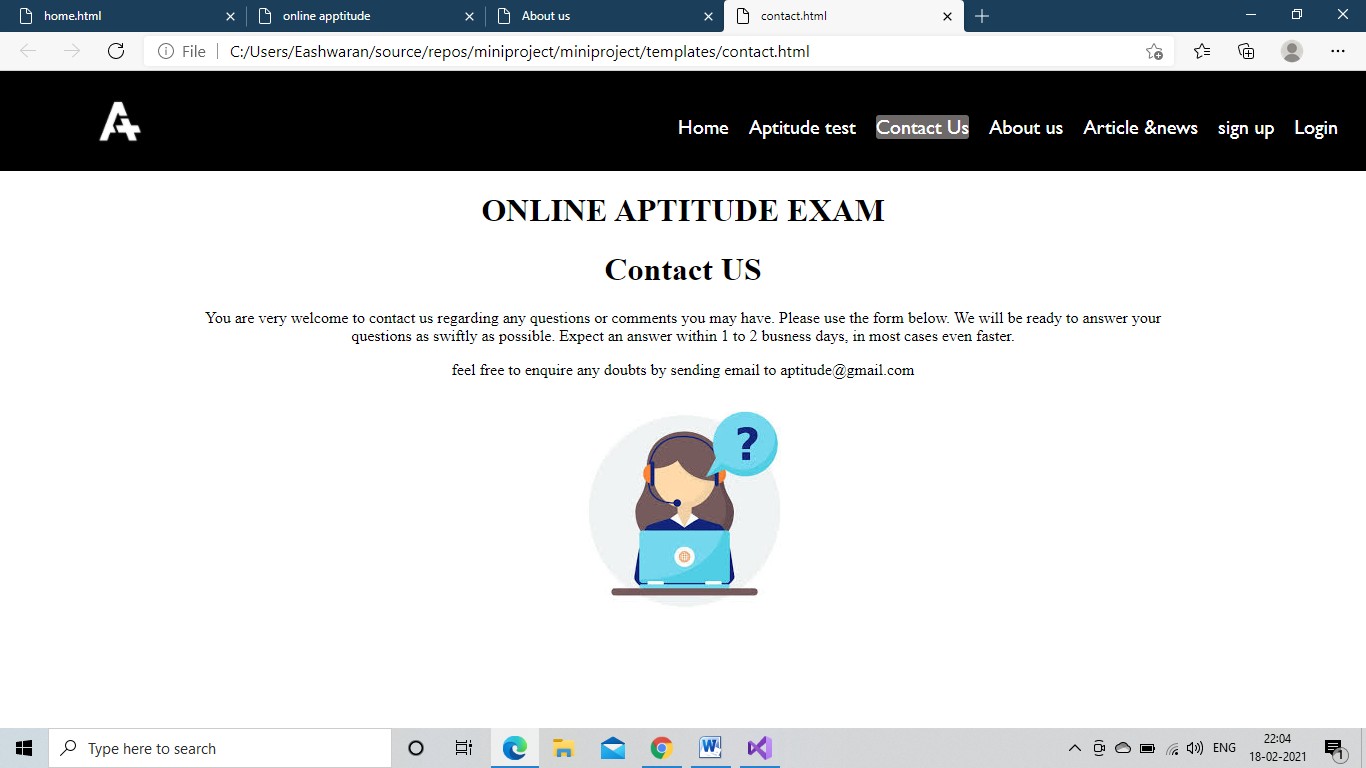
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project name :** Online Aptitude Test | | | | |
| **Test Case Id :** OAT 4 | | **Test Priority :** Medium | | |
| **Module name :** Exam | | **Test Title :** face detection | | |
| **Preconditions :**Webcam should be enabled | | **Test Description :** starts the exam when face is detected | | |
| **Test Data** | **Expected Result** | **Actual Result** | **Status** | **Notes** |
| Face detected | Exam starts | Exam starts | pass |  |
| **Post conditions :**The result of the exam will be stored in the database. | | | | |

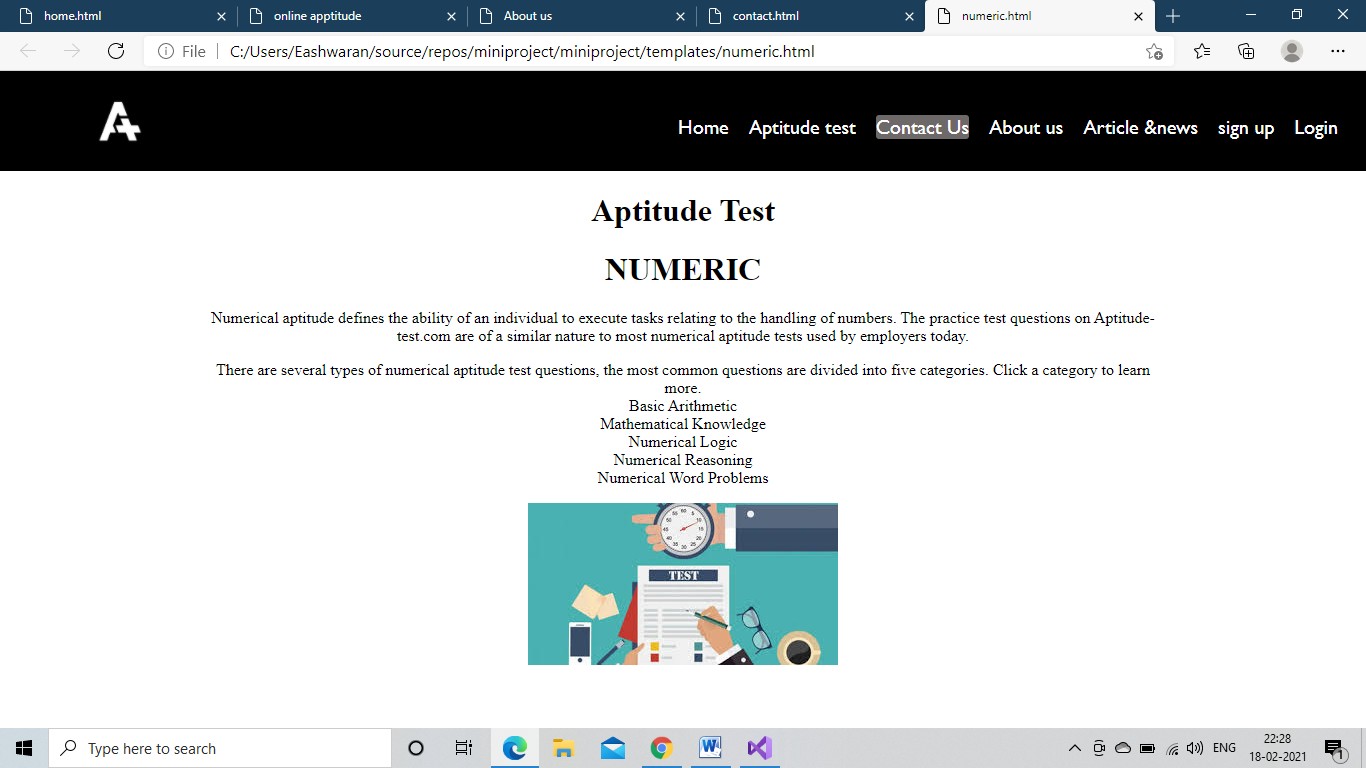
### Appendix 1 Superfluous Screen Shots

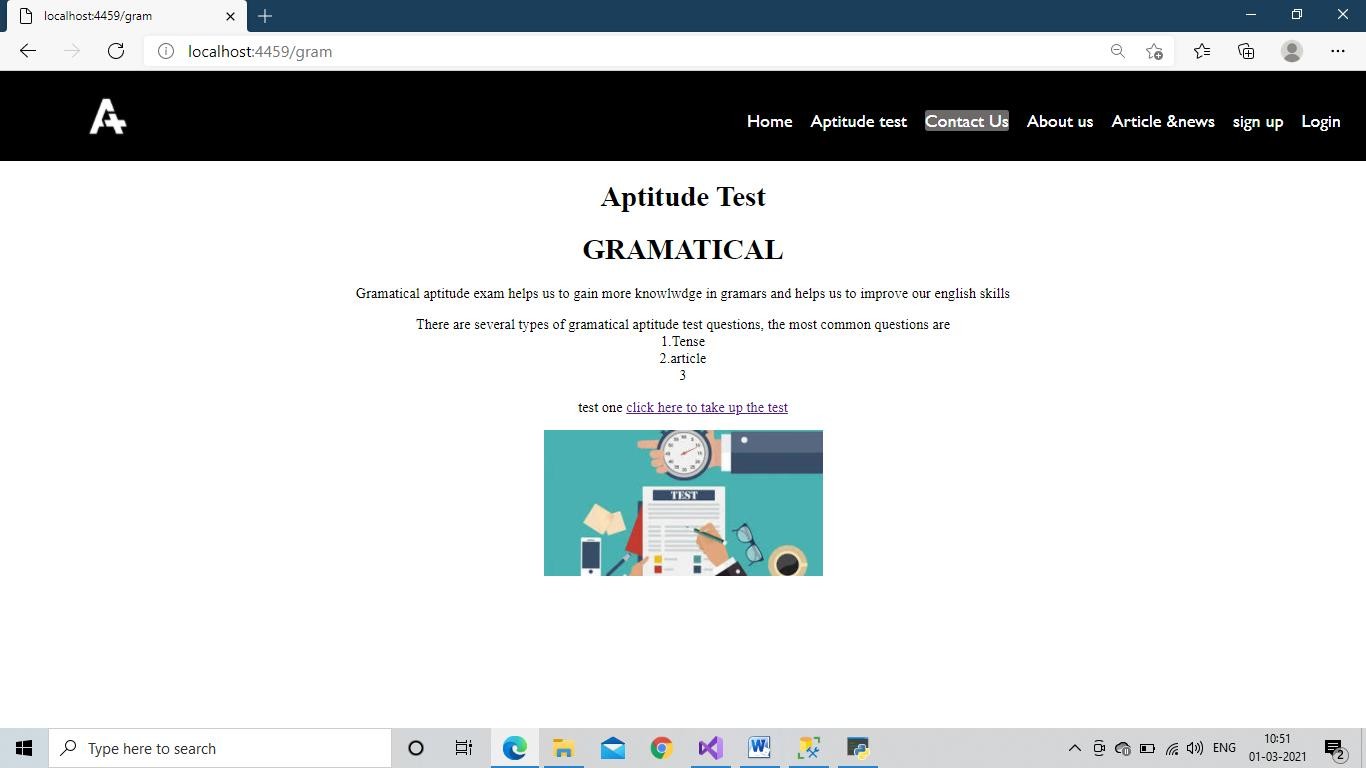


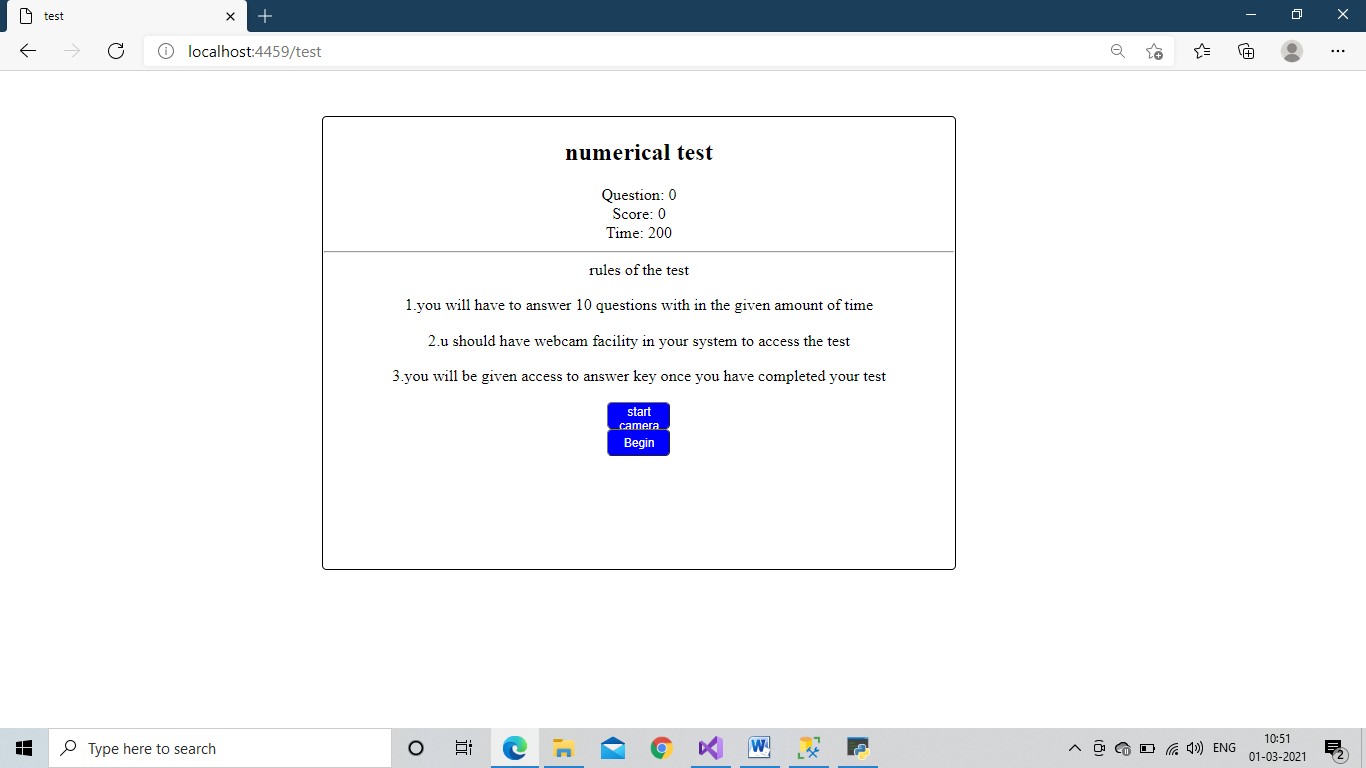


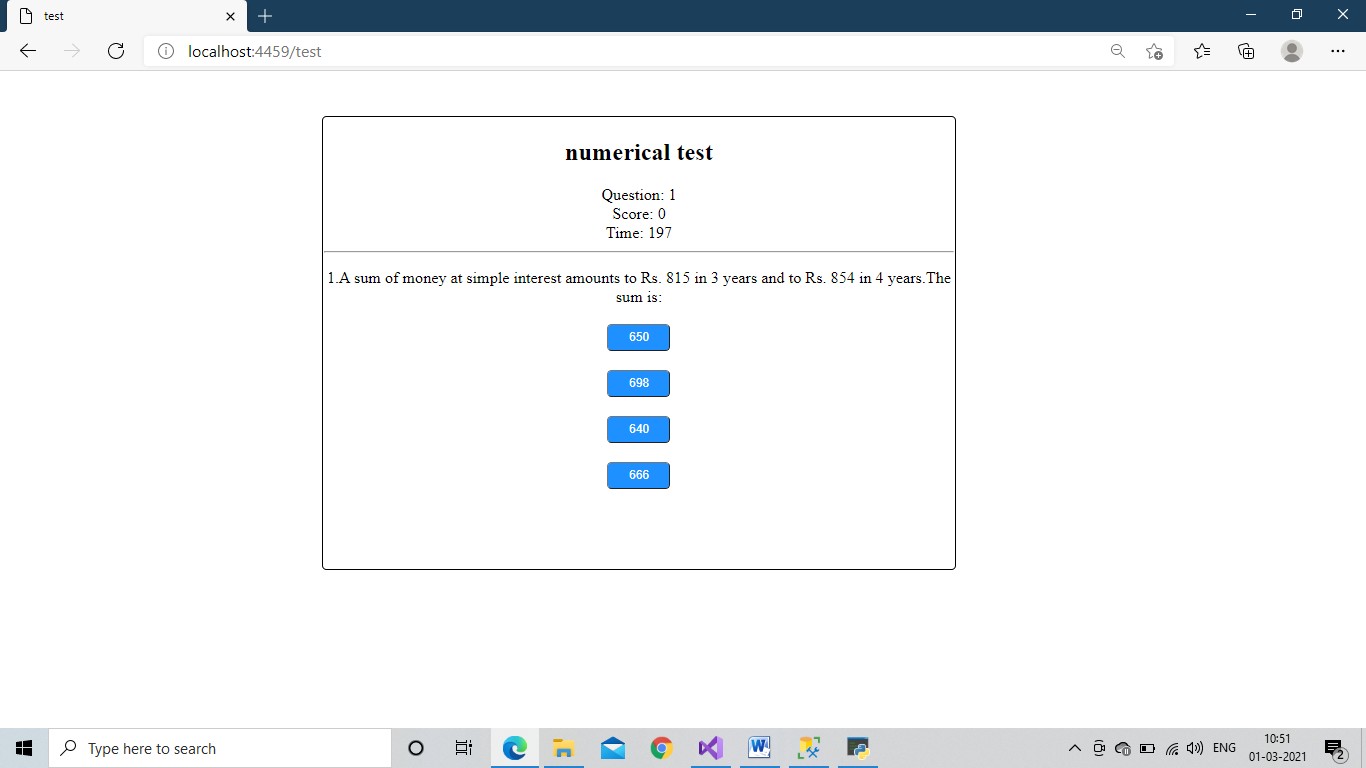


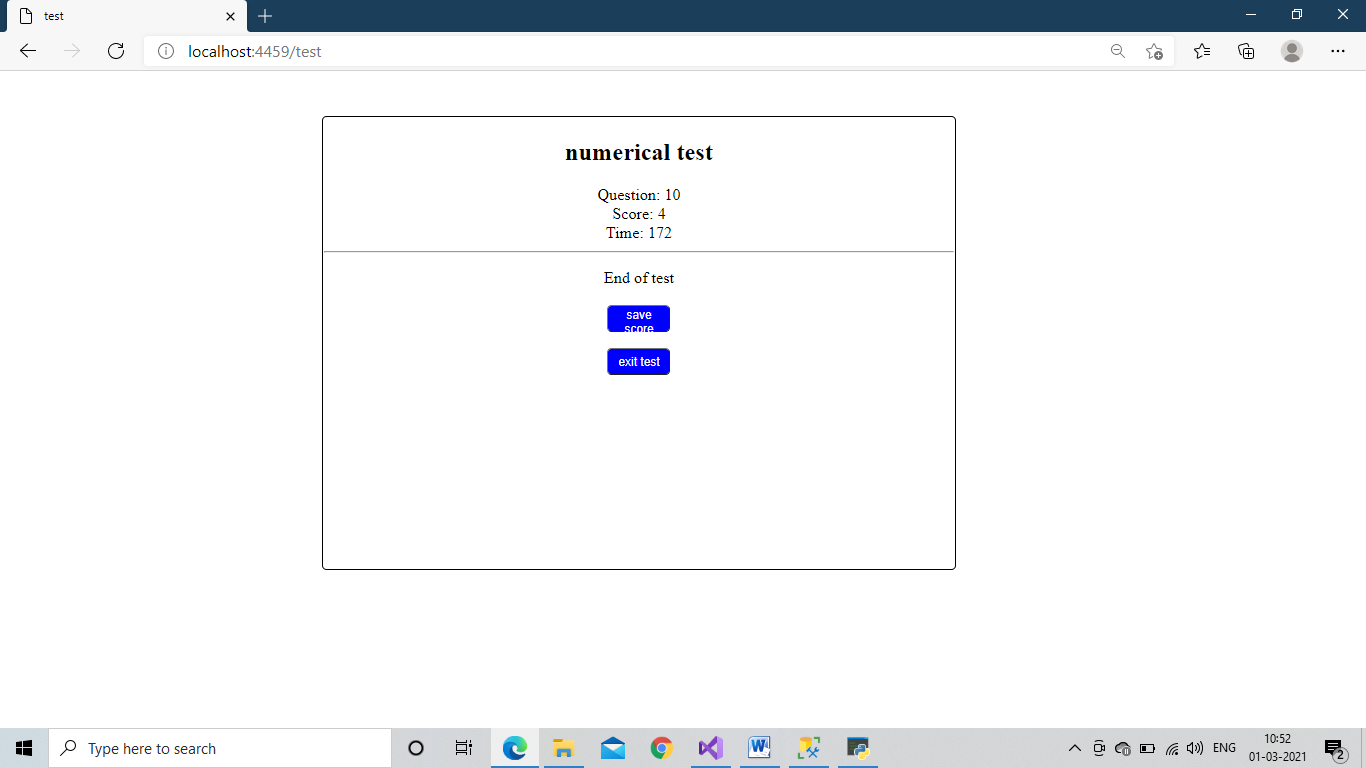


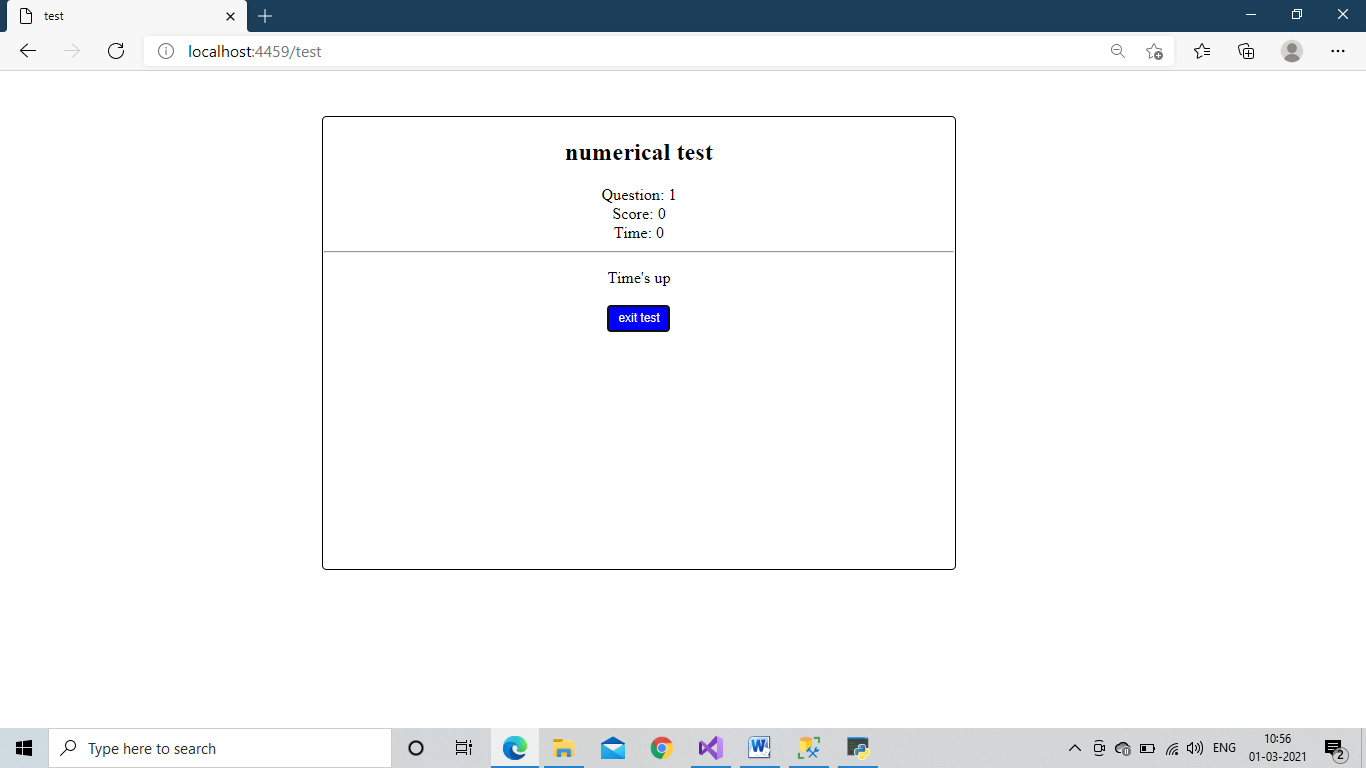


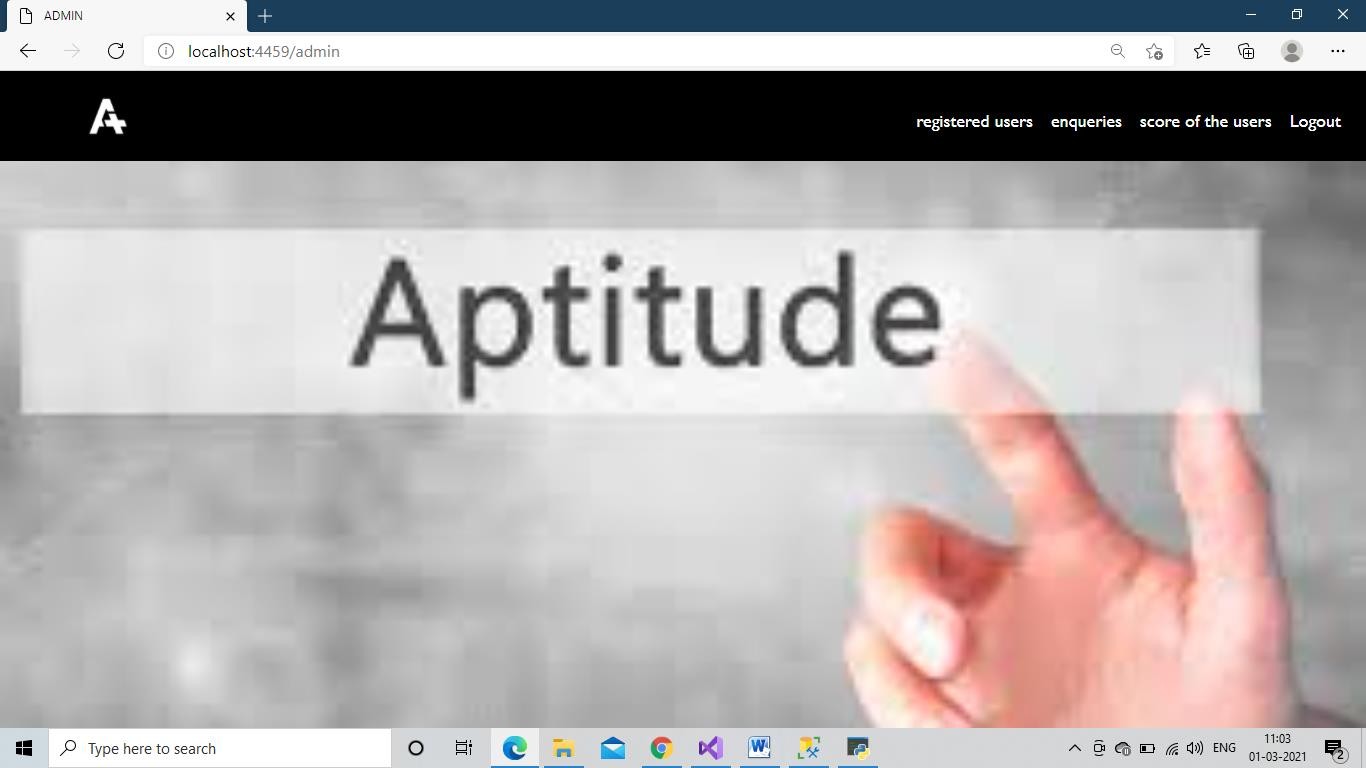












Appendix 2 Code

Python:

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

import pypyodbc

import cv2 import numpy import os import sys

import random, copy

app = Flask( name ) app.secret\_key='something'

@app.route('/') def index():

return render\_template('home.html' ) @app.route('/login', methods=['POST','GET']) def login():

msg = ''

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

username = request.form['username'] password = request.form['password'] if (username == admin ):

msg='welcome admin'

return flask.redirect('home.html',value=msg) connection = pypyodbc.connect('Driver={SQL

Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi') cursor=connection.cursor()

cursor.execute('SELECT \* FROM login WHERE username = %s AND password = %s'% (username,password))

account = cursor.fetchone() if account:

session['loggedin'] = True session['id'] = account['id']

session['username'] = account['username'] msg = 'Logged in successfully !'

data= '%s' % username

return falsk.redirect('home.html', msg = msg ,value=data)

else:

msg = 'incorrect username or password'

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

@app.route('/logout') def logout():

session.pop('loggedin', None) session.pop('id', None) session.pop('username', None) return redirect(url\_for('login'))

@app.route('/registration', methods =['GET'])

def register():

return render\_template('registration.html') @app.route('/a', methods =['POST'])

def a():

name,email,gender,age,phone = flask.request.form['name'], flask.request.form['email'], flask.request.form['gender'], flask.request.form['age'],flask.request.form['phone']

flask.session['completed'] = True

connection = pypyodbc.connect('Driver={SQL Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi')

cursor=connection.cursor()

SQLCommand=("INSERT INTO register(name,email,gender,age,phone) VALUES (?,?,?,?,?)")

values=[name,email,gender,age,phone] cursor.execute(SQLCommand,values) connection.commit() connection.close()

return flask.redirect("/login") original\_questions = {

#Format is 'question':[options]

'Taj Mahal':['Agra','New Delhi','Mumbai','Chennai'],

'Great Wall of China':['China','Beijing','Shanghai','Tianjin'], 'Petra':['Ma\'an Governorate','Amman','Zarqa','Jerash'],

'Machu Picchu':['Cuzco Region','Lima','Piura','Tacna'],

'Egypt Pyramids':['Giza','Suez','Luxor','Tanta'],

'Colosseum':['Rome','Milan','Bari','Bologna'],

'Christ the Redeemer':['Rio de Janeiro','Natal','Olinda','Betim']

}

questions = copy.deepcopy(original\_questions)

def shuffle(q):

"""

This function is for shuffling the dictionary elements.

"""

selected\_keys = [] i = 0

while i < len(q):

current\_selection = random.choice(q.keys()) if current\_selection not in selected\_keys: selected\_keys.append(current\_selection)

i = i+1

return selected\_keys

@app.route('/qs', methods=['POST', 'GET']) def quiz():

return render\_template('main.html', o = questions) @app.route('/quiz', methods=['POST'])

def quiz\_answers():

correct = 0

for i in questions.keys():

answered = request.form[i]

if original\_questions[i][0] == answered:

correct = correct+1

return '<h1>Correct Answers: <u>'+str(correct)+'</u></h1>' @app.route('/aboutus', methods =['GET'])

def aboutus():

return render\_template('aboutus.html') @app.route('/contact', methods =['GET']) def contact():

return render\_template('contact.html') @app.route('/b', methods =['POST']) def question():

username, password = flask.request.form['question'], flask.request.form['email']

flask.session['completed'] = True

conn = pypyodbc.connect('Driver={SQL Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi')

cursor=conn.cursor()

SQLCommand=("INSERT INTO contact(question,email) VALUES (?,?)") values=[username,password]

cursor.execute(SQLCommand,values) conn.commit()

conn.close()

return flask.redirect('/contact') @app.route('/aan', methods =['GET']) def aan():

return render\_template('aan.html') @app.route('/verbal',methods=['POST' , 'GET']) def verbal():

return render\_template('verbal.html') @app.route('/nv',methods=['POST' , 'GET']) def nonverbal():

return render\_template('nonverbal.html') @app.route('/rea',methods=['POST' , 'GET']) def reason():

return render\_template('reasoning.html') @app.route('/gram',methods=['POST' , 'GET']) def gramatical():

return render\_template('gramatical.html') @app.route('/mix',methods=['POST' , 'GET']) def mixed():

return render\_template('mixed.html') @app.route('/numeric',methods=['POST' , 'GET']) def numeric():

return render\_template('numeric.html') @app.route('/test',methods=['POST' , 'GET'])

def test():

return render\_template('test.html')

@app.route('/rule', methods=['POST', 'GET']) def rule():

return render\_template('rule.html') @app.route('/c', methods=['POST' , 'GET']) def sco\_data():

score = flask.request.form['score001'] email = 'pavies212000@gmail.com' module = 'numeric'

qap='test one' flask.session['completed'] = True

conn = pypyodbc.connect('Driver={SQL Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi')

cursor=conn.cursor()

SQLCommand=("INSERT INTO contact(email,qap,module,score) VALUES (?,?,?,?)")

values=[email,qap,module,score] cursor.execute(SQLCommand,values) conn.commit()

conn.close()

return flask.redirect('/contact')

@app.route('/face', methods = ['GET','POST']) def sleep():

eye\_cascPath = cv2.data.haarcascades + 'haarcascade\_eye.xml' #eye detect model

face\_cascPath = cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml' #face detect model

faceCascade = cv2.CascadeClassifier(face\_cascPath) eyeCascade = cv2.CascadeClassifier(eye\_cascPath)

cap = cv2.VideoCapture(0) while 1:

ret, img = cap.read() if ret:

frame = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) # Detect faces in the image

faces = faceCascade.detectMultiScale( frame,

scaleFactor=1.1, minNeighbors=5, minSize=(30, 30),

# flags = cv2.CV\_HAAR\_SCALE\_IMAGE

)

# print("Found {0} faces!".format(len(faces))) if len(faces) > 0:

# Draw a rectangle around the faces for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

frame\_tmp = img[faces[0][1]:faces[0][1] + faces[0][3], faces[0][0]:faces[0][0] + faces[0][2]:1, :]

frame = frame[faces[0][1]:faces[0][1] + faces[0][3],

faces[0][0]:faces[0][0] + faces[0][2]:1]

eyes = eyeCascade.detectMultiScale( frame,

scaleFactor=1.1, minNeighbors=5, minSize=(30, 30),

# flags = cv2.CV\_HAAR\_SCALE\_IMAGE

)

frame\_tmp = cv2.resize(frame\_tmp, (400, 400), interpolation=cv2.INTER\_LINEAR)

cv2.imshow('Face Recognition', frame\_tmp) waitkey = cv2.waitKey(1)

if waitkey == ord('q') or waitkey == ord('Q'): cv2.destroyAllWindows()

break

conn = pypyodbc.connect('Driver={SQL Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi')

cursor=conn.cursor()

cursor.execute("SELECT \* FROM register") s = "<table style='border:1px solid red'>"

for row in cursor: s = s + "<tr>"

for x in row:

s = s + "<td>" + str(x) + "</td>" s = s + "</tr>"

connection.close() @app.route('/reg',methods=['POST' , 'GET']) def ruser():

return render\_template('users.html')

conn = pypyodbc.connect('Driver={SQL Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi')

cursor=conn.cursor() cursor.execute("SELECT \* FROM register") p = "<table style='border:1px solid red'>" for row in cursor:

p = p + "<tr>" for x in row:

p = p + "<td>" + str(x) + "</td>" p = p + "</tr>"

@app.route('sc',methods=['POST' , 'GET']) def ascore():

return render\_template('adscore.html') @app.route("/admin" , methods=['POST' , 'GET'])

def admin():

return render\_template('admin.html' ) conn = pypyodbc.connect('Driver={SQL

Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi') cursor=conn.cursor()

cursor.execute("SELECT \* FROM register") s = "<table style='border:1px solid red'>"

for row in cursor: s = s + "<tr>"

for x in row:

s = s + "<td>" + str(x) + "</td>" s = s + "</tr>"

conn.close() @app.route('/reg',methods=['POST' , 'GET']) def ruser():

return "<html><body>"+s+ "</body></html>"

conn = pypyodbc.connect('Driver={SQL Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi')

cursor=conn.cursor()

cursor.execute("SELECT \* FROM score JOIN register ON score.email=register.email")

p = "<table style='border:1px solid red'>" for row in cursor:

p = p + "<tr>" for x in row:

p = p + "<td>" + str(x) + "</td>" p = p + "</tr>"

conn.close() @app.route('/sc',methods=['POST' , 'GET']) def ascore():

return "<html><body align:center>"+p+ "</body></html>" @app.route("/admin" , methods=['POST' , 'GET'])

def admin():

return render\_template('admin.html' ) conn = pypyodbc.connect('Driver={SQL

Server};Server=.\\sqlex;Database=aptitudemini;uid=minipro;pwd=dharpavi') cursor=conn.cursor()

cursor.execute("SELECT \* FROM contact") s = "<table style='border:1px solid red'>" for row in cursor:

a = a + "<tr>" for x in row:

a = a + "<td>" + str(x) + "</td>" a = a + "</tr>"

conn.close() @app.route('/reg',methods=['POST' , 'GET']) def ruser():

return "<html><body>"+a+ "</body></html>"

if name == " main ": app.run('localhost', 4459) home.html:

<!DOCTYPE html>

<html lang="en" xmlns=["h](http://www.w3.org/1999/xhtml)t[tp://www.w3.org/1999/xhtml"](http://www.w3.org/1999/xhtml)>

<head>

<meta charset="utf-8" />

<title></title>

<link rel="stylesheet" type="text/css" href="**{{** url\_for('static', filename='home.css')**}}**" />

</head>

<body>

<div class="wrapper">

<nav class="navbar">

<img class="logo" src="../static/images/images.png" />

<ul>

<p>**{{**session.username**}}**</p>

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

<li>

<a class="active" href="#">Aptitude test</a>

<div class="submenu">

<ul>

<li><a href="/numeric">numeric</a></li>

<li><a href="/verbal">verbal</a></li>

<li><a href="/gram">gramatical</a></li>

<li><a href="/rea">reasoning</a></li>

<li><a href="/mix">mixed</a></li>

<li><a href="/nv">non-verbal</a></li>

</ul>

</div>

</li>

<li><a class="active" href="/contact">Contact Us</a></li>

<li><a class="active" href="/aboutus">About us</a></li>

<li><a class="active" href="/aan">Article &news</a></li>

<li><a class="active" href="/registration">sign up</a></li>

<li><a class="active" href="/login">Login</a></li>

</ul>

</nav>

<div class="center">

<p>**{{** value **}}**</p>

<h1>Want To Imporove Aptitude skills ?</h1>

<h2>prepare-pass-improve with us</h2>

<form action='/rule' method='POST'>

<input type="submit" value="try sample test" />

</form>

</div>

</div>

</body>

</html> Login.html:

<!DOCTYPE html>

<html lang="en" xmlns=["h](http://www.w3.org/1999/xhtml)t[tp://www.w3.org/1999/xhtml"](http://www.w3.org/1999/xhtml)>

<head>

<meta charset="utf-8" />

<title>online apptitude</title>

<link rel="stylesheet" type="text/css" href="**{{** url\_for('static', filename='login.css')**}}**" />

</head>

<body>

<div class="loginbox">

<img src="../static/images/logo1.jpg" class="avatar" />

<h1> Login</h1>

<form action="**{{** url\_for('login') **}}**" method="post">

<div class="msg">**{{** msg **}}**</div>

<p>user name</p>

<input type="text" name="username" placeholder="enter username" required

/>

<p>password</p>

<input type="password" name="password" placeholder="enter password"

required />

<input type="submit" class="buttons" value="Login" />

<nav>

<ul>

<li><a class="bottom" href="/registration"> dont have account? Sign Up here </a></li>

</ul>

</nav>

</form>

</div>

</body>

</html> Test.html:

<!DOCTYPE html>

<html lang="en" xmlns=["h](http://www.w3.org/1999/xhtml)t[tp://www.w3.org/1999/xhtml"](http://www.w3.org/1999/xhtml)>

<head>

<meta charset="utf-8" />

<title>test</title>

</head>

<body>

<style>

body {

background-image: url(images/app.png);

}

#frame001 {

border-radius: 5px; background-color: white; border: 2px solid black; height: 500px;

width: 700px; margin-left:350px; margin-top:50px; text-align:center;

}

.buttons001 {

background-color: dodgerblue; color: white;

width: 70px; height: 30px; border-radius: 5px;

}

.buttons002 {

background-color: blue; color: white;

width: 70px; height: 30px; border-radius: 5px;

}

#font001 {

text-align: center; font-size: large; color: black;

}

</style>

<div id="frame001">

<div id="font001">

<text id="001">{{ value }}</text>

<h2>numerical test</h2><text ></text><text>Question: <text id="number001">0</text></text><br />

<form>

<text>Score: <text id="score001">0</text></text><br />

</form>

<text>Time: <text id="time001">200</text></text><br />

<hr />

<div id="disappear001">

<span>rules of the test</span>

<p>1.you will have to answer 10 questions with in the given amount of time</p>

<p>2.u should have webcam facility in your system to access the test</p>

<p>3.you will be given access to answer key once you have completed your test </p>

<form action='/face' method="POST">

<button class="buttons002">start camera</button>

</form>

<button class="buttons002" onclick="begin001()">Begin</button>

</div>

<p id="message001"></p>

<p id="message002"></p>

<p id="message003"></p>

<p id="message004"></p>

</div>

</div>

<script>

var question001 = ["1.A sum of money at simple interest amounts to Rs. 815 in 3 years and to Rs. 854 in 4 years.The sum is:", "A sum fetched a total simple interest of Rs. 4016.25 at the rate of 9 p.c.p.a. in 5 years. What is the sum?", "How much time will it take for an amount of Rs. 450 to yield Rs. 81 as interest at 4.5% per annum of simple interest?",

"Reena took a loan of Rs. 1200 with simple interest for as many years as the rate of interest. If she paid Rs. 432 as interest at the end of the loan period, what was the rate of interest?", "A man standing at a point P is watching the top of a tower, which makes an angle of elevation of 30° with the man's eye. The man walks some distance towards the tower to watch its top and the angle of the elevation becomes 60°. What is the distance between the base of the tower and the point P?", "An observer 1.6 m tall is 203 away from a tower. The angle of elevation from his eye to the top of the tower is 30°. The height of the tower is:", "From a point P on a level ground, the angle of elevation of the top tower is 30°. If the tower is 100 m high, the distance of point P from the foot of the tower is:",

"The cost price of 20 articles is the same as the selling price of x articles. If the profit is 25%, then the value of x is:", "If selling price is doubled, the profit triples.

Find the profit percent.","A vendor bought toffees at 6 for a rupee. How many for a rupee must he sell to gain 20%?"

];

var options001 = ["<button class=buttons001 onclick=q1i()>650</button><br

/><br /><button class=buttons001 onclick=q1c()>698</button><br/><br/><button class=buttons001 onclick=q1i()>640</button><br /><br /><button class=buttons001 onclick=q1i()>666</button>"];

var options002 = ["<button class=buttons001 onclick=q1c()>8925</button><br

/><br /><button class=buttons001 onclick=q1i()>4462.50</button><br/><br/><button class=buttons001 onclick=q1i()>8032.50</button><br /><br /><button class=buttons001 onclick=q1i()>8900</button>"];

var options003 = ["<button class=buttons001 onclick=q1i()>4.1</button><br

/><br /><button class=buttons001 onclick=q1i()>4.5</button><br/><br/><button class=buttons001 onclick=q1i()>4.8</button><br /><br /><button class=buttons001 onclick=q1c()>4</button>"];

var options004 = ["<button class=buttons001 onclick=q1i()>5</button><br

/><br /><button class=buttons001 onclick=q1c()>6</button><br/><br/><button class=buttons001 onclick=q1i()>18</button><br /><br /><button class=buttons001 onclick=q1i()>13</button>"];

var options005 = ["<button class=buttons001 onclick=q1i()>4 units</button><br

/><br /><button class=buttons001 onclick=q1i()>8units</button><br/><br/><button class=buttons001 onclick=q1c()>Data inadequate</button><br /><br /><button class=buttons001 onclick=q1i()>12units</button>"];

var options006 = ["<button class=buttons001 onclick=q1i()>23.2</button><br

/><br /><button class=buttons001 onclick=q1c()>21.6</button><br/><br/><button class=buttons001 onclick=q1i()>24.72</button><br /><br /><button class=buttons001 onclick=q1i()>20.5</button>"];

var options007 = ["<button class=buttons001 onclick=q1c()>173</button><br

/><br /><button class=buttons001 onclick=q1i()>200</button><br/><br/><button class=buttons001 onclick=q1i()>156</button><br /><br /><button class=buttons001 onclick=q1i()>210</button>"];

var options008 = ["<button class=buttons001 onclick=q1c()>16</button><br

/><br /><button class=buttons001 onclick=q1i()>15</button><br/><br/><button class=buttons001 onclick=q1i()>18</button><br /><br /><button class=buttons001 onclick=q8i()>25</button>"];

var options009 = ["<button class=buttons001 onclick=q1c()>100</button><br

/><br /><button class=buttons001 onclick=q1i()>66</button><br/><br/><button class=buttons001 onclick=q1i()>105</button><br /><br /><button class=buttons001 onclick=q9i()>120</button>"];

var options010 = ["<button class=buttons001 onclick=q1i()>6</button><br

/><br /><button class=buttons001 onclick=q1i()>3</button><br/><br/><button class=buttons001 onclick=q1i()>4</button><br /><br /><button class=buttons001 onclick=q1c()>5</button>"];

var a = 0; a++;

var b = 0;

b++;

function begin001() { c = 200;

disappear001.innerHTML = ""; message001.innerHTML = question001[0]; message002.innerHTML = options001; number001.innerHTML = a++;

}

function q1c() {

score001.innerHTML = b++;

message004.innerHTML = "<button class=buttons002 onclick=next001()>Next</button>";

}

function q1i() {

message003.innerHTML = "Incorrect";

message002.innerHTML = ""; message004.innerHTML = "<button class=buttons002

onclick=next001()>Next</button>";

}

function next001() { if (a == "2") {

message001.innerHTML = question001[1]; message002.innerHTML = options002; message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

}

else if (a == "3") {

message001.innerHTML = question001[2]; message002.innerHTML = options003; message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

}

else if (a == "4") {

message001.innerHTML = question001[3]; message002.innerHTML = options004; message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

}

else if (a == "5") {

message001.innerHTML = question001[4]; message002.innerHTML = options005; message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

}

else if (a == "6") {

message001.innerHTML = question001[5]; message002.innerHTML = options006; message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

}

else if (a == "7") {

message001.innerHTML = question001[6]; message002.innerHTML = options007; message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

}

else if (a == "8") {

message001.innerHTML = question001[7]; message002.innerHTML = options008;

message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

}

else if (a == "9") {

message001.innerHTML = question001[8]; message002.innerHTML = options009; message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

}

else if (a == "10") {

message001.innerHTML = question001[9]; message002.innerHTML = options010; message003.innerHTML = ""; number001.innerHTML = a++; message004.innerHTML = "";

} else {

window.clearInterval(update); c = "-";

message001.innerHTML = "End of test"; message002.innerHTML = "";

message003.innerHTML = "<button class=buttons002 onclick='/c'>save score</button>";

message004.innerHTML = "<button class=buttons002 onclick='/numeric'>exit test</button>";

}

}

function timer001() { c = c - 1;

if (c < 200) { time001.innerHTML = c;

}

if (c < 1) { window.clearInterval(update);

message001.innerHTML = "Time's up"; message002.innerHTML = ""; message003.innerHTML = "";

message004.innerHTML = "<button class=buttons002 onclick='/numeric'>exit test</button>";

}

}

update = setInterval("timer001()", 1000);

</script>

</body>

</html>