Navigating the Complex World of Auto Insurance : A Vehicle Cost Analysis for Better Decision Making

TEAM ID :NM2023TMID10715

TEAM LEADER : RAJESH.R

TEAM MEMBER 1 : VENKATESHWARA MURTHY .D

TEAM MEMBER 2 : ABISHEK .K

TEAM MEMBER 3 : ESHWAR .N.R

PROJECT DOCUMENTATION

1) PROBLEM STATEMENT

The vehicle damage detection and The Estimated cost for the damaged vehicle parts is one of the most vital activities in the vehicle insurance.

Key Challenges:

- The estimation is carried out by manual process only ,so it takes more time for the customer .
- The estimation report is also not able to know that is correct or not .
- The damaged vehicle need to take to the Showroom for estimation .

Proposed Solution :

- Autoinsure is an online platform that helps users estimate the cost of vehicle insurance based on images of damaged parts.
- Using this platform, customers can upload images of the damaged parts of their vehicle and receive a report with an estimated insurance amount.
- The platform uses image analysis technology to assess the damage and generate the report, eliminating the need for a manual process.
- Users can quickly and easily estimate the cost of insurance or their damaged vehicles, saving time and hassle.
- By providing an accurate and convenient way to estimate insurance costs.

- AutoInsure goal is to make the process of obtaining vehicle insurance more efficient and user-friendly.

DESCRIPTION:

- The proposed system is the need of today ‘s vehicle insurance claim analysis.
- In our project we are using cloud technology for storing data and images of damaged parts of vehicle for Effective processing of Data.
- AutoInsure aims to make the process of obtaining vehicle insurance more efficient and user-friendly.
- This platform uses image analysis technology and eliminating the need for a manual process.
- Using this platform, customers can upload images of the damaged parts of their vehicle and receive an estimated insurance value.

2) REQUIREMENTS :

i) Functional Requirements :

a. User Registration :

User Registration is carried out using Forms ,generated in HTML pages .The Forms gets the input from the user and delivers it to the database .

b. User Confirmation :

The user provides the registered details to the login form to enter into the webpage to perform certain tasks.The provided details are validated from the database .

c. Image uploading and Image analysis :

After login ,user enters into the web page to estimate the insurance amount based upon the user uploaded image to the web page .The logined user can upload image to the website and Estimation report will be generated using Image Analysis .

d. Estimation of insurance amount :

The Estimation report will be displayed finally as the output from the webpage .

ii) Non-Functional Requirements :

a. Usability :

End User not having prior knowledge about technology can also use this product efficiently and easily.

b. Security :

As the system uses cloud application technology ,the user information are stored safely.Authentication is given to those users whose details are already exists in cloud database.

c. Reliability :

The database update are rolled back ,whenever the failure occurs the system is rolled back.

d. Performance :

User details are retrieved and stored faster through cloud application.

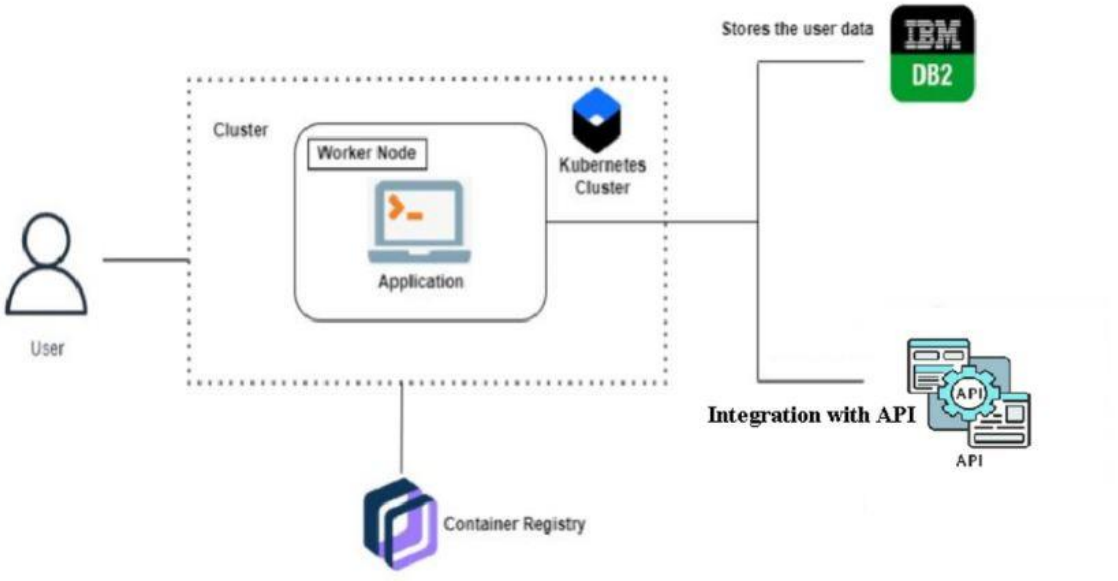
e. Availability :

User can login on the website at any time during the day.

f. Scalability :

The size of cloud can be increased or decreased based on number of users.

3) TECHNICAL ARCHITECTURE :



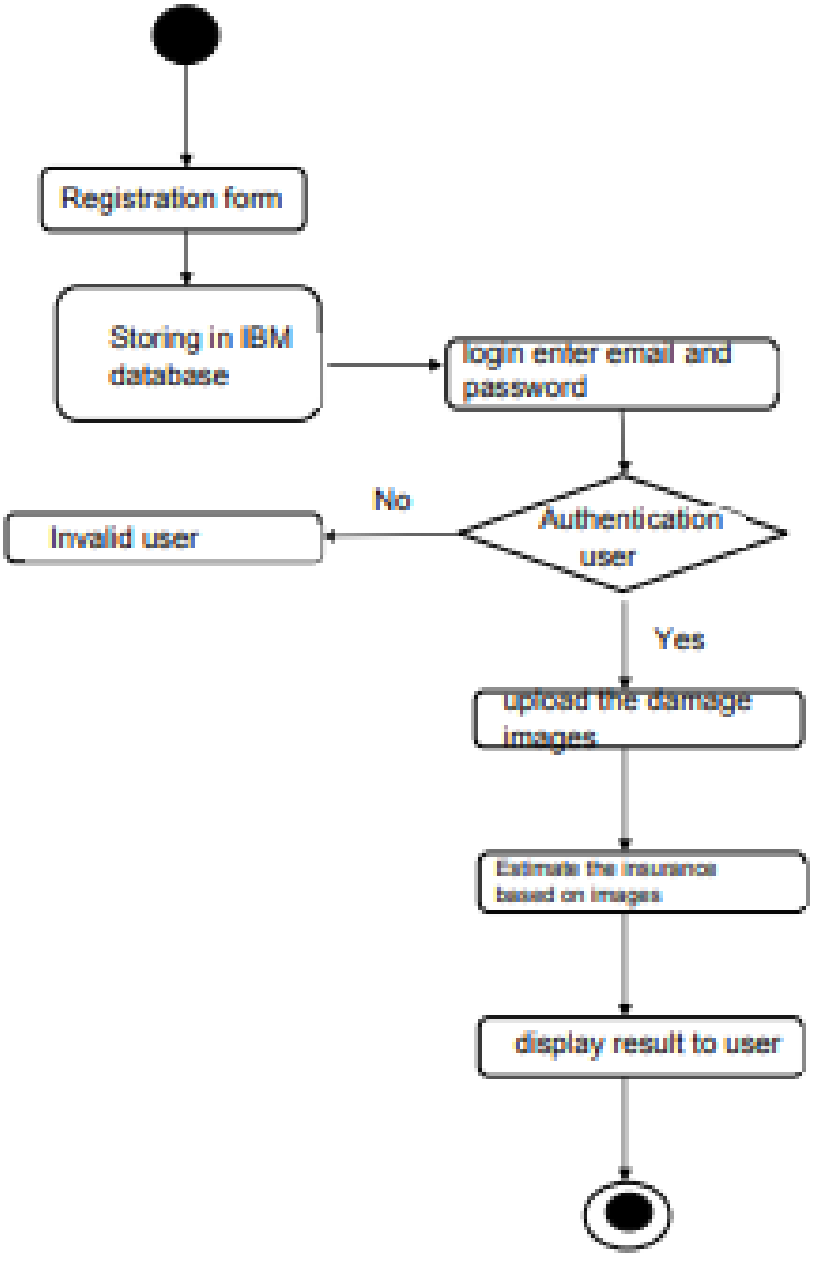
User Interface	Registration Form , Login Form	HTML, CSS/Boot Strap, JavaScript
Application Logic-1	User Registration And Login	Python
Application Logic-2	Image uploading and Image analysis.User can upload the damaged parts of the vehicle.	Python
Application Logic-3	Based on image analysis the cost of insurance is estimated.	Python

Database	Data Type, Configurations etc.	SQL
Cloud Database	Database Service on Cloud	IBM DB2
Cloud Object Storage	Object Storage	IBM Cloud object storage
Rapid API	From Rapid API, the damaged parts of a car are fetched and showcased in web browser by web browser library.	Vehicle Damage Assessment API etc.
Docker	Containerization way that allows them to run consistently across different environments.	Docker
Kubernetes	Deploy the application	Kubernetes

Application Characteristics :

Characteristics	Description	Technology
Open-Source Frameworks	Flask is a web framework, it's a Python module that lets you develop web applications easily.	Python
Security Implementations	As the system uses cloud application technology ,the user information are stored safely.	Cloud Application
Scalable Architecture	The size of cloud can be increased or decreased based on number of users.	Cloud Application
Availability	User can login on the website at any time during the day.	Python
Performance	User details are retrieved and stored faster through cloud application.	IBM DB2

Flowchart :


Registration form
Storing in IBM database
login enter email and password
Authentication user
No
Invalid user
Yes
upload the damage images
Estimate the insurance based on images
display result to user

- The new user need to register first to access the resources in the webpage .
- The details given by the user will be stored in the IBM D2 Database .
- The registered user can login now .
- The credentials given by the user are validated .
- If it is correct , it will redirect to the next page .
- Otherwise ,it shows the error message to the user .
- The validated user can now upload the damaged images into the webpage .
- The estimation of insurance will be analyzed by the Image Analaysis Process .
- The Estimation report will be generated and finally it will be displayed to the user .

MODULES :

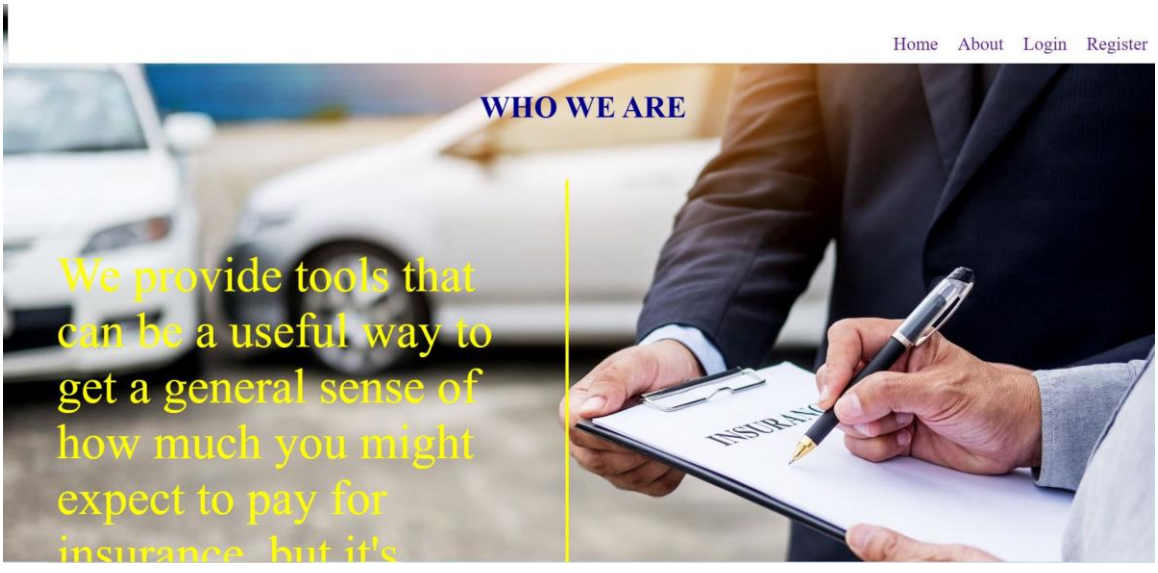
HOME PAGE

The Home page contains all the corresponding pages to the project ,

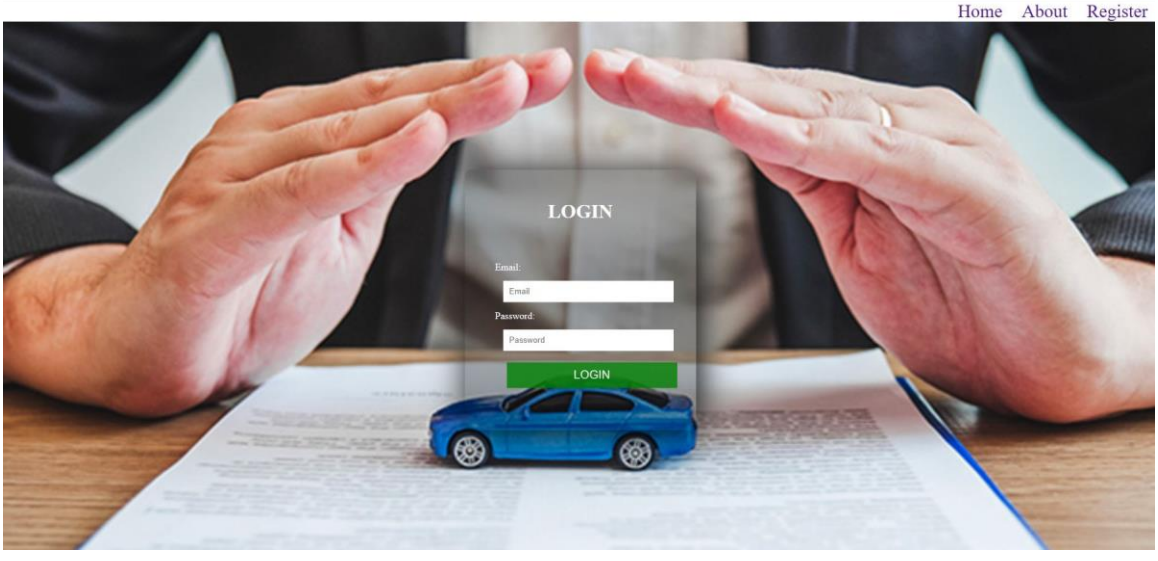
1) About
2) Login
3) Register

After login ,it will redirect to the Image Upload Page ,and it will lead to the Estimation Output Page .

ABOUT PAGE



LOGIN PAGE



REGISTER PAGE

AUTOINSURE

Home About Login

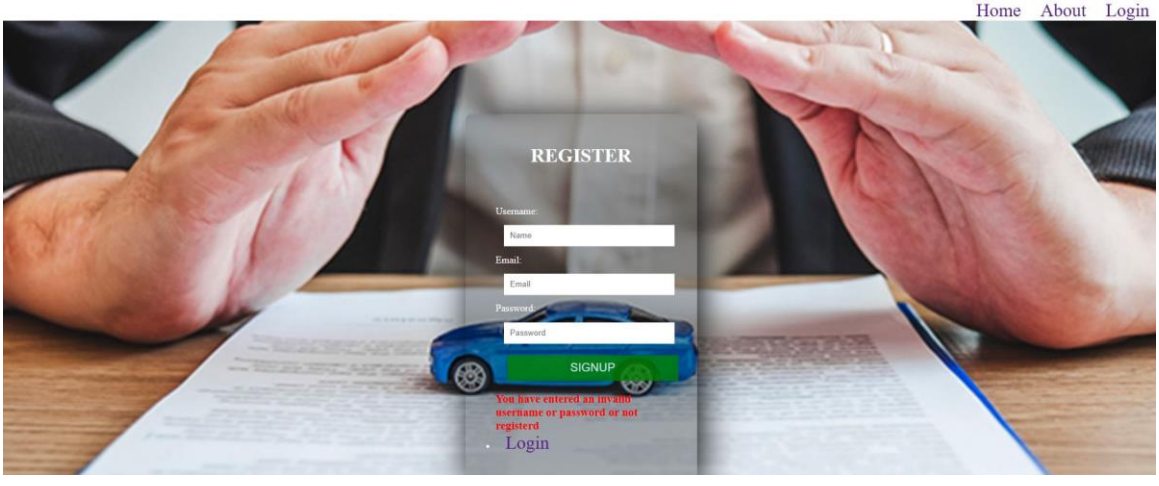
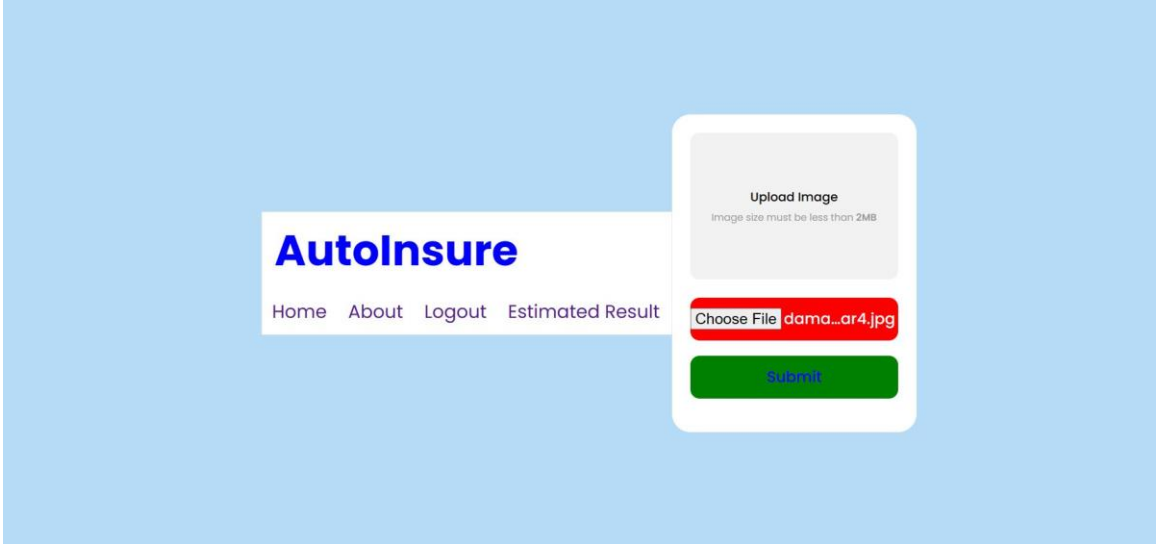


IMAGE UPLOAD PAGE



ESTIMATION OUTPUT PAGE

PROGRAM :

app.py ● DigiCertGlobalRootCA.crt

app.py > img

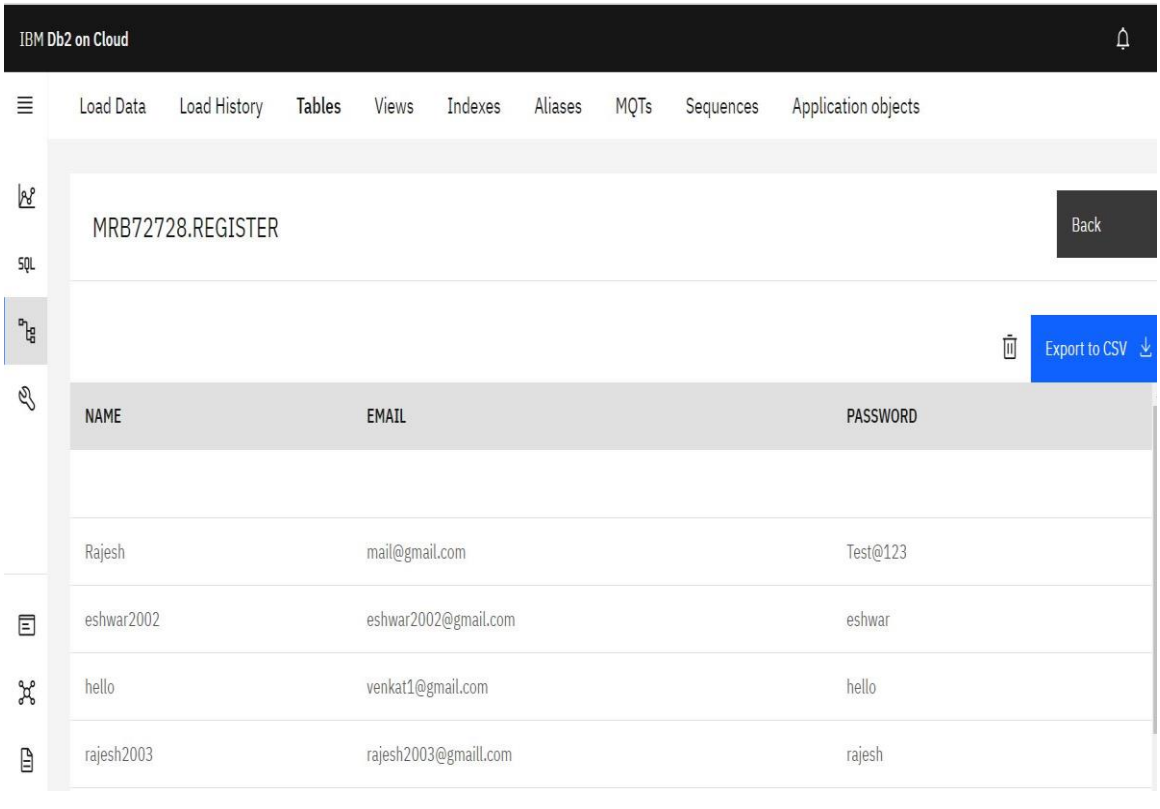
```python
from flask import Flask, render_template,request,session
import ibm_boto3
from ibm_botocore.client import Config, ClientError
import requests
import ibm_db
import re
import json
import webbrowser
import os.path
app = Flask(__name__)
slight =["slight_scratch","slight_deformation","car_light_crack","side_mirror_scratch","side_mirror_crack","side_mirror_drop_off"]
moderate =["fender/headlight_damage","fender/bumper_damage","car_light_severe_crack","car_light_damage","windshield_damage"]
severe=["severe_scratch","medium_deformation","severe_deformation","crack_and_hole"]
app.secret_key = 'a'
con = ibm_db.connect("DATABASE=bludb;HOSTNAME=98538591-7217-4024-b027-8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=3087
print("connection successful")
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/about')
def home1():
    return render_template('about.html')
@app.route('/login')
def login1():
    return render_template('login.html')
@app.route('/register')
def register1():
    return render_template('register.html')
@app.route('/image')
```

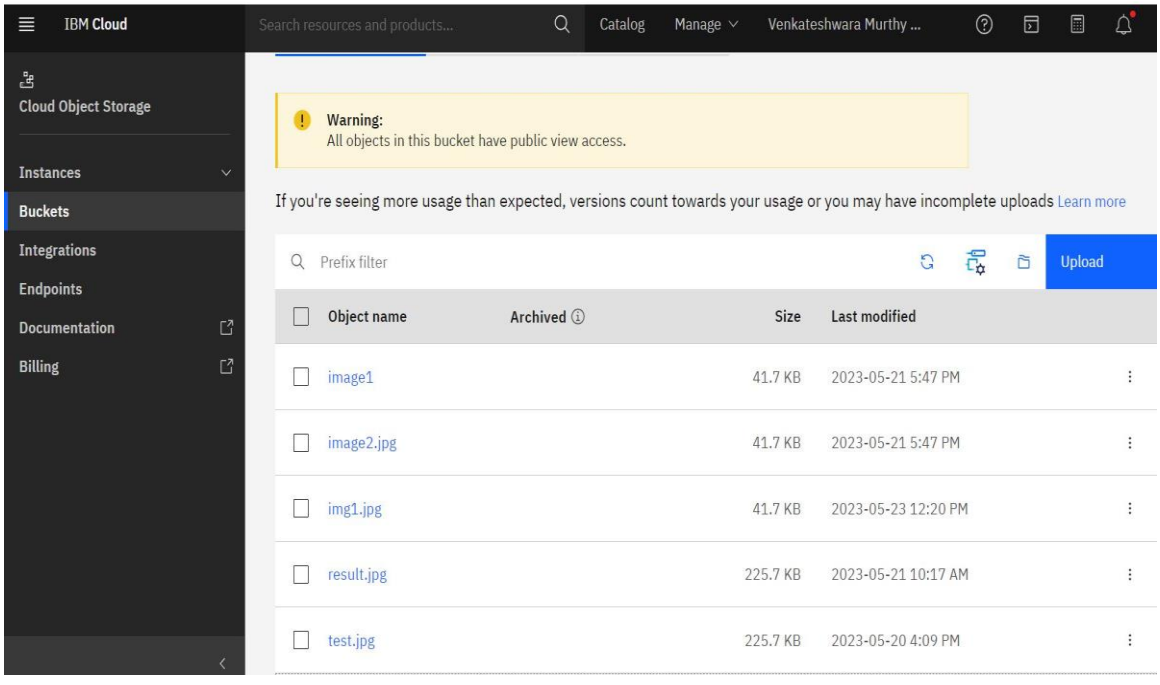
app.py > register

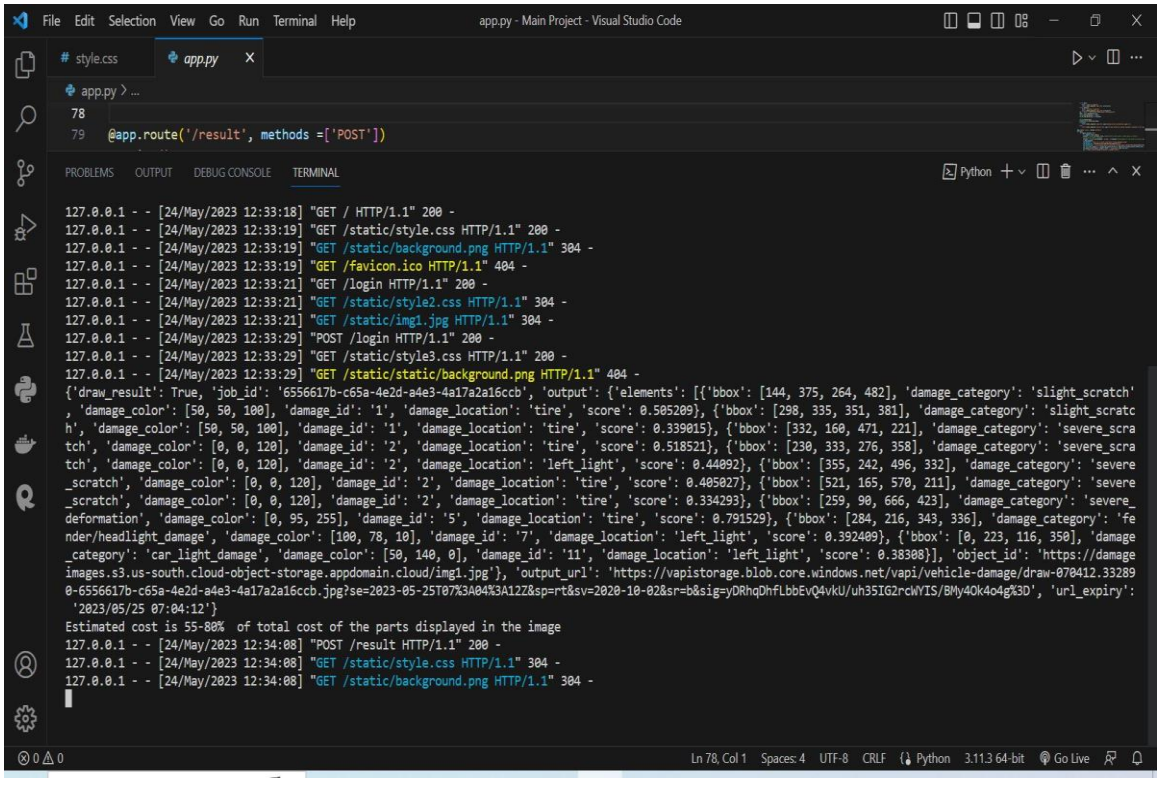
```python
def img1():
    return render_template('image.html')
@app.route('/result')
def result1():
   return render_template('result.html')


@app.route('/register',methods=['POST'])
def register():
    Username=request.form['Username']
    Email=request.form['Email']
    Password=request.form['Password']
    insert_sql = "INSERT INTO REGISTER VALUES (?,?,?)"
    prepSql=ibm_db.prepare(con,insert_sql)
    ibm_db.bind_param(prepSql, 1, Username)
    ibm_db.bind_param(prepSql, 2, Email)
    ibm_db.bind_param(prepSql, 3, Password)
    result=ibm_db.execute(prepSql)
    return render_template('register.html',msg='Thank you for registering click above to login ')
@app.route('/login',methods=['POST'])
def login():
    Email=request.form['Email']
    Password=request.form['Password']
    sql = "SELECT * FROM REGISTER WHERE Email=? AND Password=?"
    smtp = ibm_db.prepare(con,sql)
    ibm_db.bind_param(smtp, 1, Email)
    ibm_db.bind_param(smtp, 2, Password)

    ibm_db.execute(smtp)
    account=ibm_db.fetch_assoc(smtp)
```

```python
    account=ibm_db.fetch_assoc(smtp)
    if account:
        return render_template('image.html',msg='Welcome you are successfully logged in')
    else:
        return render_template('register.html',msg='You have entered an invalid username or password or not registerd')

@app.route('/result', methods =['POST'])
def img():
     if request.method=='POST':
         f = request.files['images']
         basepath = os.path.dirname(__file__) #getting the current path i.e where app.py is present
         #print("current path", basepath)
         filepath = os.path.join(basepath, 'uploads', f.filename) #from anywhere in the system we can give image but we want that image la
         f.save(filepath)
         COS_ENDPOINT = "https://s3.us-south.cloud-object-storage.appdomain.cloud"
         COS_API_KEY_ID = "1PI8G9m1woxT5UIv5dH68iMGunKdutxWKHSv4OIfxc11"
         COS_INSTANCE_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/07b15bf7929c4d94bae59d59774f0d76:a8fda193-7429-4a80-874f-
         cos =ibm_boto3.client("s3", ibm_api_key_id=COS_API_KEY_ID, ibm_service_instance_id=COS_INSTANCE_CRN, config= Config(signature_ver
         cos.upload_file(Filename= filepath, Bucket = 'damageimages',Key='img1.jpg')
         url = "https://vehicle-damage-assessment.p.rapidapi.com/run"
         payload = {"draw_result": True, "remove_background": True,"image": "https://damageimages.s3.us-south.cloud-object-storage.appdoma
         headers = {"content-type": "application/json", "X-RapidAPI-Key": "aace9c9262mshe0df8296e0e571dp12b23cjsn5f6e5e4b1c09", "X-RapidAP
         response = requests.request("POST",url, json=payload, headers=headers)
         output=response.json()
         print(output)
         webbrowser.open(url)
         cos.upload_file(Filename ="uploads/car.jpg", Bucket='damageimages', Key='img1.jpg')
         a=0
         b=0
         c=0
```

```
        l=[]
        for i in range(0,len(output['output']["elements"])):
              d = output['output']["elements"][i]["damage_category"]
              l.append(d)
        for i in range(0,len(l)):
            for j in range(0,len(slight)):
                if l[i]==slight[j]:
                    a = a+1
        for i in range(0,len(l)):
            for j in range(0,len(moderate)):
                if l[i]==moderate[j]:
                    b = b+1
        for i in range(0, len(l)):
            for j in range(0,len(severe)):
                if l[i]==severe[j]:
                    c = c+1
        percentage = (a*30 + b*50 +c*80)/(a+b+c)
        damage_parts =set()
        for i in range(0,len(output['output']["elements"])):
              d = output['output']["elements"][i]["damage_location"]
              damage_parts.add(d)
        damage_parts=list(damage_parts)
        if percentage<30:
            result = "Estimated cost is"+ " "+ "20" +"-"+ "30" + "% "+ " " + "of total cost of the parts displayed in the image"
        elif percentage<50:
            result = "Estimated cost is"+ " "+ "30" +"-"+ "50" + "% "+ " " + "of total cost of the parts displayed in the image"
        else:
            result = "Estimated cost is"+ " "+ "55" +"-"+ "80" + "% "+ " " + "of total cost of the parts displayed in the image"
        print(result)
```




IBM Cloud
Search resources and products...
Catalog
Manage
Venkateshwara Murthy ...
Cloud Object Storage
Instances
Buckets
Integrations
Endpoints
Documentation
Billing
Warning:
All objects in this bucket have public view access.
If you're seeing more usage than expected, versions count towards your usage or you may have incomplete uploads Learn more
Prefix filter
Upload
Object name
Archived
Size
Last modified
image1 41.7 KB 2023-05-21 5:47 PM
image2.jpg 41.7 KB 2023-05-21 5:47 PM
img1.jpg 41.7 KB 2023-05-23 12:20 PM
result.jpg 225.7 KB 2023-05-21 10:17 AM
test.jpg 225.7 KB 2023-05-20 4:09 PM


File Edit Selection View Go Run Terminal Help
app.py - Main Project - Visual Studio Code
style.css
app.py
app.py > ...
78
79 @app.route('/result', methods =['POST'])
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
127.0.0.1 - - [24/May/2023 12:33:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/May/2023 12:33:19] "GET /static/style.css HTTP/1.1" 200 -
127.0.0.1 - - [24/May/2023 12:33:19] "GET /static/background.png HTTP/1.1" 304 -
127.0.0.1 - - [24/May/2023 12:33:19] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [24/May/2023 12:33:21] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [24/May/2023 12:33:21] "GET /static/style2.css HTTP/1.1" 304 -
127.0.0.1 - - [24/May/2023 12:33:21] "GET /static/img1.jpg HTTP/1.1" 304 -
127.0.0.1 - - [24/May/2023 12:33:29] "POST /login HTTP/1.1" 200 -
127.0.0.1 - - [24/May/2023 12:33:29] "GET /static/style3.css HTTP/1.1" 200 -
127.0.0.1 - - [24/May/2023 12:33:29] "GET /static/static/background.png HTTP/1.1" 404 -
{'draw_result': True, 'job_id': '6556617b-c65a-4e2d-a4e3-4a17a2a16ccb', 'output': {'elements': [{'bbox': [144, 375, 264, 482], 'damage_category': 'slight_scratch', 'damage_color': [50, 50, 100], 'damage_id': '1', 'damage_location': 'tire', 'score': 0.505209}, {'bbox': [298, 335, 351, 381], 'damage_category': 'slight_scratch', 'damage_color': [50, 50, 100], 'damage_id': '1', 'damage_location': 'tire', 'score': 0.339015}, {'bbox': [332, 160, 471, 221], 'damage_category': 'severe_scratch', 'damage_color': [0, 0, 120], 'damage_id': '2', 'damage_location': 'tire', 'score': 0.518521}, {'bbox': [230, 333, 276, 358], 'damage_category': 'severe_scratch', 'damage_color': [0, 0, 120], 'damage_id': '2', 'damage_location': 'left_light', 'score': 0.44092}, {'bbox': [355, 242, 496, 332], 'damage_category': 'severe_scratch', 'damage_color': [0, 0, 120], 'damage_id': '2', 'damage_location': 'tire', 'score': 0.405027}, {'bbox': [521, 165, 570, 211], 'damage_category': 'severe_scratch', 'damage_color': [0, 0, 120], 'damage_id': '2', 'damage_location': 'tire', 'score': 0.334293}, {'bbox': [259, 90, 666, 423], 'damage_category': 'severe_deformation', 'damage_color': [0, 95, 255], 'damage_id': '5', 'damage_location': 'tire', 'score': 0.791529}, {'bbox': [284, 216, 343, 336], 'damage_category': 'fender/headlight_damage', 'damage_color': [100, 78, 10], 'damage_id': '7', 'damage_location': 'left_light', 'score': 0.392409}, {'bbox': [0, 223, 116, 350], 'damage_category': 'car_light_damage', 'damage_color': [50, 140, 0], 'damage_id': '11', 'damage_location': 'left_light', 'score': 0.38308}], 'object_id': 'https://damage-images.s3.us-south.cloud-object-storage.appdomain.cloud/img1.jpg'}, 'output_url': 'https://vapistorage.blob.core.windows.net/vapi/vehicle-damage/draw-070412.332890-6556617b-c65a-4e2d-a4e3-4a17a2a16ccb.jpg?se=2023-05-25T07%3A04%3A12Z&sp=rt&sv=2020-10-02&sr=b&sig=yDRhqDhfLbbEvQ4vkU/uh35IG2rcWYIS/BMy4Ok4o4g%3D', 'url_expiry': '2023/05/25 07:04:12'}
Estimated cost is 55-80% of total cost of the parts displayed in the image
127.0.0.1 - - [24/May/2023 12:34:08] "POST /result HTTP/1.1" 200 -
127.0.0.1 - - [24/May/2023 12:34:08] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [24/May/2023 12:34:08] "GET /static/background.png HTTP/1.1" 304 -
Ln 78, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live