Project Report

1. INTRODUCTION

1.1 Project Overview :

Nowadays, a lot of money is being wasted in the car insurance business due to leakage claims. Claims leakage Underwriting leakage is characterized as the discrepancy between the actual payment of claims made and the sum that should have been paid if all of the industry's leading practices were applied. Visual examination and testing have been used to may these results. However, they impose delays in the processing of claims.

1.2 Purpose:

The aim of this project is to build a VGG16 model that can detect the area of damage on a car. The rationale for such a model is that it can be used by insurance companies for faster processing of claims if users can upload pics and the model can assess damage (be it dent, scratch form) and estimate the cost of damage. This model can also be used by lenders if they are underwriting a car loan, especially for a used car.

2. LITERATURE SURVEY

2.1 Existing problem:

Manual inspection needs to be done in case of an accident, which will consume a lot of time. Also, there is a chance of wastage of money due to insurance claims leakage, where there is a discrepancy between the actual payment of claims made and the sum that should have been paid if all of the industry's leading practices were applied. The customers may not be satisfied with the manual inspection and estimation given by an Insurance Inspector from Insurance Companies.

2.2 References:

1. **Automatic assessment of damage and repair costs in vehicles** (2018) by Vikas Taliwal, Siddhartha Dalal, Kaigang Li and Gaurav Sharma.
2. **Deep learning based car damage and detection of the automotive industry** (2022) by Kili Companies.
3. **Automated Car Damage Detection with AI for Remote Assessment and Improved Claims Processing** (2021) by Karamveer Verma.

4. **Automatic Car Damage Assessment System** (2021) by Wei Zhang and Zuan Cheng.

2.3 Problem Statement Definition:

Problem Statement.pdf

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

Empathy Map Canvas.pdf

3.2 Ideation & Brainstorming:

Brainstorm & Idea Prioritization.pdf

3.3 Proposed Solution:

Proposed Solution.pdf

3.4 Problem Solution fit:

Problem Solution Fit.pdf

4. REQUIREMENT ANALYSIS

Solution Requirements.pdf

5. PROJECT DESIGN

5.1 Data Flow Diagrams:

Data Flow Diagrams.pdf

5.2 Solution & Technical Architecture:

Solution Architecture.pdf
Technology Stack.pdf

5.3 User Stories:

User Stories.pdf

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation:

Milestone & Activity List.pdf

6.2 Sprint Delivery Schedule:

Sprint Delivery Plan.pdf

6.3 Reports from JIRA:

Reports from JIRA.pdf

7. CODING & SOLUTIONING

Python code for app.py:

```
from cloudant.client import Cloudant

client=Cloudant.iam('6f4f5183-072e-4bd0-b33f-8c0562a8e227-bluemix',
'IcNwIUOYQsMwH32_e2m3xg93E-Af0KVHeiAwVijUAWWC', connect=True)
my_database=client['my_database']

import numpy as np
import os
from flask import Flask, app,request,render_template,redirect,url_for,session
from tensorflow.keras import models
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.python.ops.gen_array_ops import concat
from tensorflow.keras.applications.inception_v3 import preprocess_input
import requests
import webbrowser

os.add_dll_directory
```

```
model1=load_model(r'D:\Usman\IBM Project\Model\body.h5')
model2=load_model(r'D:\Usman\IBM Project\Model\level.h5')

app=Flask(__name__)
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/index.html')
def home():
    return render_template("index.html")

@app.route('/register.html')
def register():
    return render_template("register.html")

@app.route('/afterreg',methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data={'_id':x[1],'name':x[0],'psw':x[2]}
    print(data)
    query={'_id':{'$eq':data['_id']}}
    docs=my_database.get_query_result(query)
    print(docs)
    print(len(docs.all()))
    if (len(docs.all())==0):
        url=my_database.create_document(data)
        return render_template("register.html",pred="Registration Successful, please login with  your details")
    else:
            return render_template("register.html",pred="You are already a member, please login using your registered details")

@app.route('/login.html')
```

```
def login():
    return render_template("login.html")

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    user=request.form['_id']
    passw=request.form['psw']
    print(user,passw)
    query={'_id':{'$eq':user}}
    docs=my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))

    if (len(docs.all())==0):
        return render_template("login.html",pred="The username or password is incorrect. Please login with correct details.")
    else:
        if((user==docs[0][0]['_id']and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            return render_template("login.html",pred="The username is not found or the details you've entered is incorrect.")

@app.route('/logout.html')
def logout():
    return render_template("logout.html")

@app.route('/prediction.html')
def prediction():
    return render_template("prediction.html")

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

```
if request.method=="POST":
    f=request.files['file']
    basepath=os.path.dirname("__file__")
    filepath=os.path.join(basepath,'uploads', f.filename)
    f.save(filepath)
    img=image.load_img(filepath,target_size=(256, 256))
    x=image.img_to_array(img)
    x=np.expand_dims(x,axis=0)
    img_data=preprocess_input(x)
    prediction1=np.argmax(model1.predict(img_data))
    prediction2=np.argmax(model2.predict(img_data))
    index1=['front','rear','side']
    index2=['minor','moderate','severe']
    result1=index1[prediction1]
    result2=index2[prediction2]
    print(result1)
    print(result2)
    if(result1=="front"and result2=="minor"):
        value="3000 - 5000 INR"
    elif(result1=="front"and result2=="moderate"):
        value="6000 - 8000 INR"
    elif(result1=="front"and result2=="severe"):
        value="9000 - 11000 INR"
    elif(result1=="rear"and result2=="minor"):
        value="4000 - 6000 INR"
    elif(result1=="rear"and result2=="moderate"):
        value="7000 - 9000 INR"
    elif(result1=="rear"and result2=="severe"):
        value="11000 - 13000 INR"
    elif(result1=="side"and result2=="minor"):
        value="6000 - 8000 INR"
    elif(result1=="side"and result2=="moderate"):
        value="9000 - 11000 INR"
    elif(result1=="side"and result2=="severe"):
```

```
            value="12000 - 15000 INR"
        else:
            value="16000 - 50000 INR"
        return render_template("result.html", prediction="The Estimated cost for the damage is: "+value)
url='http://127.0.0.1:8080'

if __name__=="__main__":
    webbrowser.open_new_tab(url)
    app.run(debug=False,port=8080)
```

Features:

7.1 Smart UI

The HTML pages are made in such a way that the user finds it easy to register, login, upload images and can see the results.

7.2 Secure

The user credentials are stored in IBM Cloudant and no one except the Insurance Company admin has access to.

7.3 Accurate Results

Our model is trained with around 1000 training images and tested with 200 test images which leads to an accuracy of 99.79%. Therefore, the results are accurate and reliable.

8. TESTING

8.1 Test Cases

Verify user is able to see login page
Verify if the user is able to login to the application or not?
Verify users are able to navigate to the register page?
Verify user is able to upload images
Verify results

8.2 User Acceptance Testing Report:

UAT Report.pdf

8.3 Test Cases Report:

Testcases Report.xlsx

9. RESULTS

9.1 Performance Metrics

Performance Testing - Artificial Intelligence.pdf

10. ADVANTAGES & DISADVANTAGES

10.1 Advantages:

- We can reduce the overall time for insurance claim procedure
- Accurate results
- Faster Prediction
- User Friendly Website

10.2 Disadvantages:

- Requires continuous support of Internet since this is a website
- Can be trained with more images to improve accuracy

11. CONCLUSION

The rationale for such a model is that it can be used by insurance companies for faster processing of claims if users can upload pics and the model can assess damage(be it dent or scratch) and estimate the cost of damage. This model can also be used by lenders if they are underwriting a car loan, especially for a used car.

12. FUTURE SCOPE

There is no need for any third person agent involvement in this system, hence his salary amount can be invested in the development of this system. More customers lead to more profit. Hence, this project will be a huge success if implemented by a reputed Insurance Company to compete with other Insurance Companies.

13. APPENDIX

Source Code: from cloudant.client import Cloudant

```
client=Cloudant.iam('6f4f5183-072e-4bd0-b33f-8c0562a8e227-bluemix',
'IcNwIUOYQsMwH32_e2m3xg93E-Af0KVHeiAwVijUAWWC', connect=True)
my_database=client['my_database']

import numpy as np
import os
from flask import Flask, app,request,render_template,redirect,url_for,session
from tensorflow.keras import models
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.python.ops.gen_array_ops import concat
from tensorflow.keras.applications.inception_v3 import preprocess_input
import requests
import webbrowser

os.add_dll_directory

model1=load_model(r'D:\Usman\IBM Project\Model\body.h5')
model2=load_model(r'D:\Usman\IBM Project\Model\level.h5')

app=Flask(__name__)
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/index.html')
def home():
    return render_template("index.html")

@app.route('/register.html')
def register():
```

```
    return render_template("register.html")

@app.route('/afterreg',methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data={'_id':x[1],'name':x[0],'psw':x[2]}
    print(data)
    query={'_id':{'$eq':data['_id']}}
    docs=my_database.get_query_result(query)
    print(docs)
    print(len(docs.all()))
    if (len(docs.all())==0):
        url=my_database.create_document(data)
        return render_template("register.html",pred="Registration Successful, please login with  your details")
    else:
            return render_template("register.html",pred="You are already a member, please login using your registered details")

@app.route('/login.html')
def login():
    return render_template("login.html")

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    user=request.form['_id']
    passw=request.form['psw']
    print(user,passw)
    query={'_id':{'$eq':user}}
    docs=my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))
```

```
    if (len(docs.all())==0):
        return render_template("login.html",pred="The username or password is incorrect. Please login with correct details.")
    else:
        if((user==docs[0][0]['_id']and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            return render_template("login.html",pred="The username is not found or the details you've entered is incorrect.")

@app.route('/logout.html')
def logout():
    return render_template("logout.html")

@app.route('/prediction.html')
def prediction():
    return render_template("prediction.html")

@app.route('/result',methods=["GET","POST"])
def result():
    if request.method=="POST":
        f=request.files['file']
        basepath=os.path.dirname("__file__")
        filepath=os.path.join(basepath,'uploads', f.filename)
        f.save(filepath)
        img=image.load_img(filepath,target_size=(256, 256))
        x=image.img_to_array(img)
        x=np.expand_dims(x,axis=0)
        img_data=preprocess_input(x)
        prediction1=np.argmax(model1.predict(img_data))
        prediction2=np.argmax(model2.predict(img_data))
        index1=['front','rear','side']
        index2=['minor','moderate','severe']
        result1=index1[prediction1]
```

```
        result2=index2[prediction2]
        print(result1)
        print(result2)
        if(result1=="front"and result2=="minor"):
            value="3000 - 5000 INR"
        elif(result1=="front"and result2=="moderate"):
            value="6000 - 8000 INR"
        elif(result1=="front"and result2=="severe"):
            value="9000 - 11000 INR"
        elif(result1=="rear"and result2=="minor"):
            value="4000 - 6000 INR"
        elif(result1=="rear"and result2=="moderate"):
            value="7000 - 9000 INR"
        elif(result1=="rear"and result2=="severe"):
            value="11000 - 13000 INR"
        elif(result1=="side"and result2=="minor"):
            value="6000 - 8000 INR"
        elif(result1=="side"and result2=="moderate"):
            value="9000 - 11000 INR"
        elif(result1=="side"and result2=="severe"):
            value="12000 - 15000 INR"
        else:
            value="16000 - 50000 INR"
        return render_template("result.html", prediction="The Estimated cost for the damage is: "+value)
url='http://127.0.0.1:8080'

if __name__=="__main__":
    webbrowser.open_new_tab(url)
    app.run(debug=False,port=8080)
```

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-23701-1659895004

Project Demo Link: https://drive.google.com/file/d/1IsVfhX3f7yEV6J6VJe1e-iEM2Y6rL4K_/view?usp=sharing