PROJECT BASED EXPERIENTIAL LEARNING PROGRAM(NALAIYATHIRAN)

Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies

A PROJECT REPORT

Submitted by

LOGESH E (412919104006)
DEEPAK P (412919104002)
NAGARAJ V (412919104008)
KISHORE KUMAR B (412919104005)

TEAM ID : PNT2022TMID38414

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VI INSTITUTE OF TECHNOLOGY

Sirunkundram - 603108

NOVEMBER 2022

Contents

Project Report

INTRODUCTION

Project Overview:

Nowadays, a lot of money is being wasted in the car insurance businessdue to leakage claims. Claimsleakage Underwriting leakageis characterizedas the discrepancy between the actual paymentof 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 processingof claims.

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 sucha model is that it can be used by insurance companies for faster processing of claims if users can upload pics and the model canassess damage (be it dent scratch from and estimates the cost of damage.This model can also be used by lenders if they are underwritinga car loan, especially for a used car.

LITERATURE SURVEY

Existing problems:

Image analysis and pattern recognition are applied to automatically identifyand characterize automobile damage.

This approach requires3D computer aided design (CAD) modes of the considered vehicle to identifyhow it would look if it were undamaged.

Block chain, data analysis, machinelearning, AI for damage

CNN model is trained on Image Net dataset. After fine tuning the dataset,transfer learning with L2 regularization is Applied.

References:

LI Ying &Dorai Chitra, 2012

Srimal Jayewardene, 2013

M.Wassel,2019

Phyu Mar Kyu,KuntpongWoraratpanya, 2020

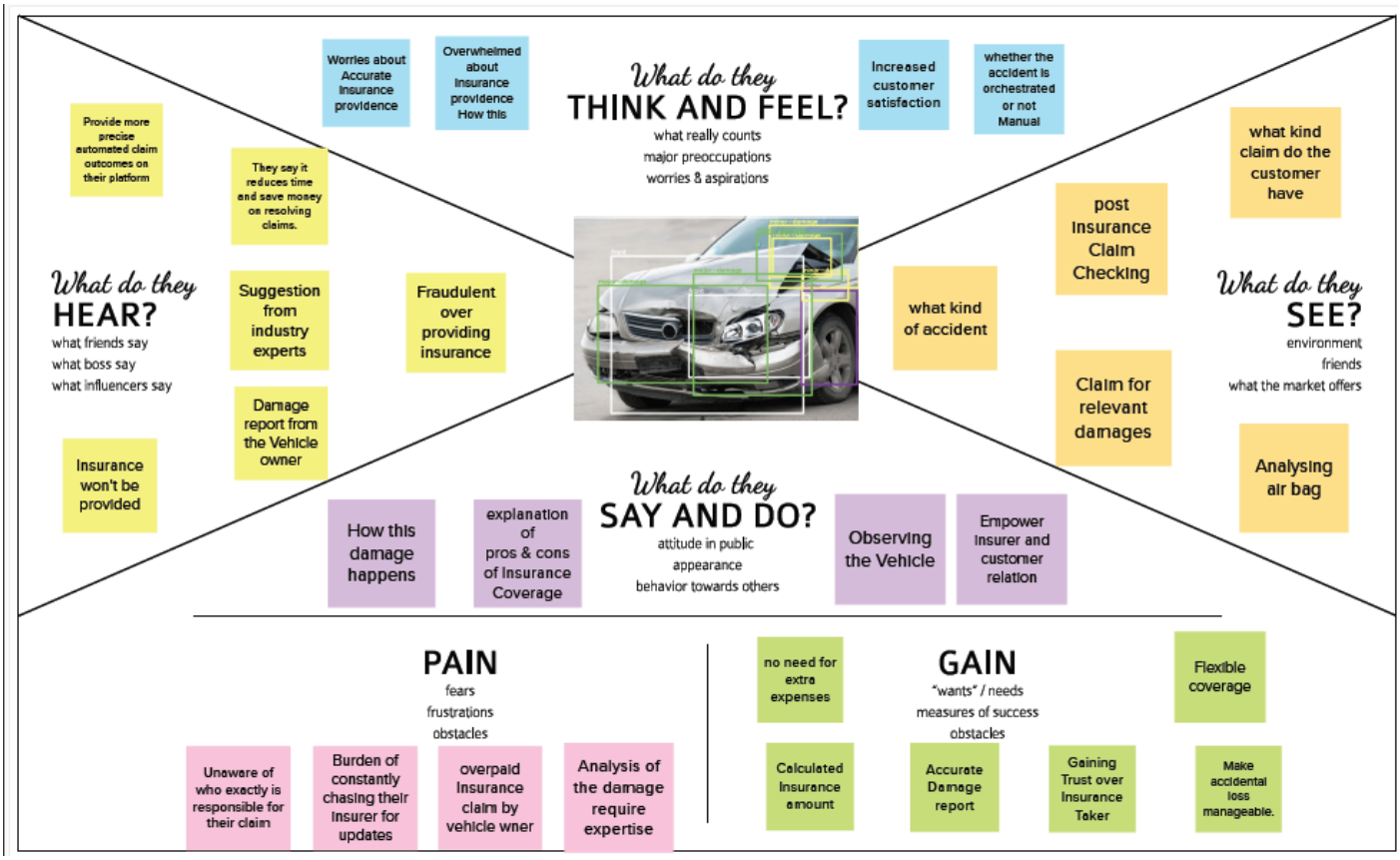
Problem StatementDefinition:

- Applying image analysis to auto insuranceTriage.
- Image based automatic vehicle damagedetection.

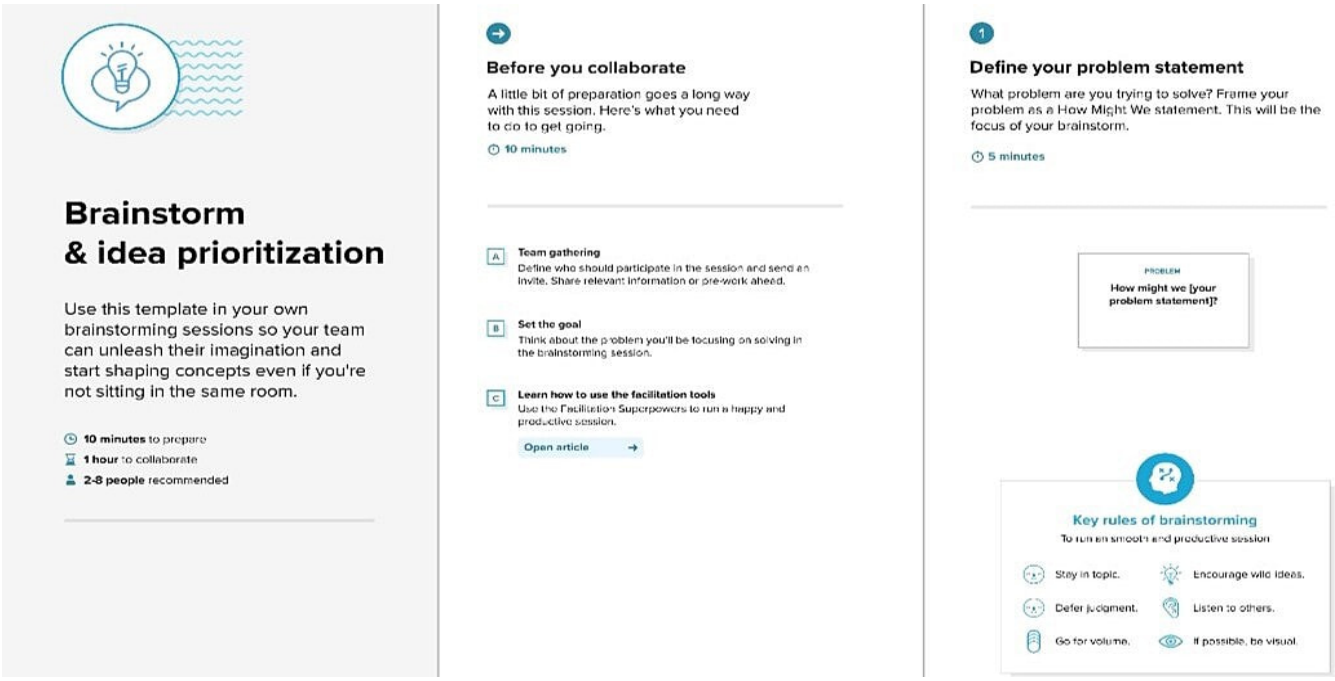
- A Secure AI- driven Architecture for Automated InsuranceSystems: Fraud Detectionand Risk Measurement.
- Car damage detectionand classification.

IDEATION & PROPOSED SOLUTION

Empathy Map Canvas:



Ideation & Brainstorming



2

Brainstorm

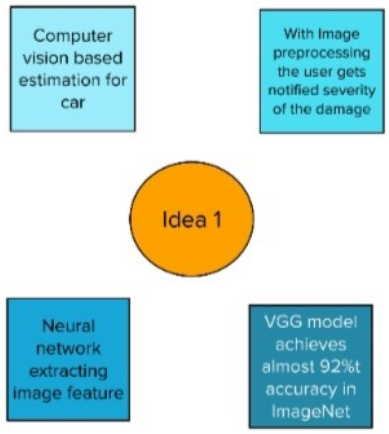
Write down any ideas that come to mind that address your problem statement.

10 minutes

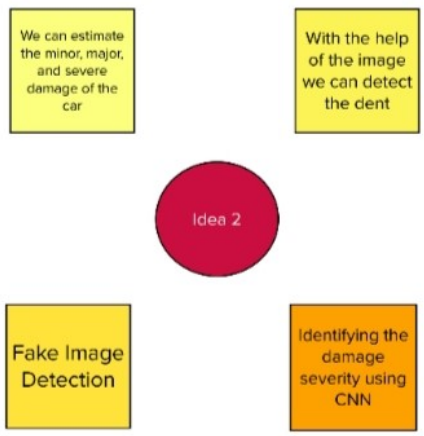
TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

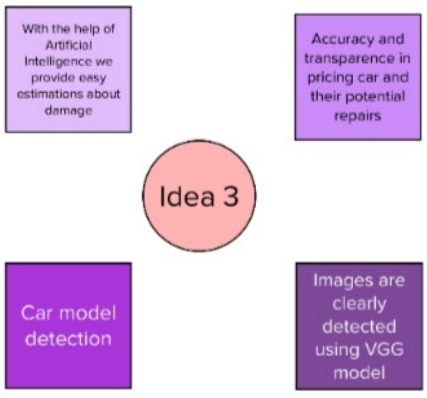
LOGESH E



DEEPAK P



NAGARAJ V



KISHORE KUMAR B

3

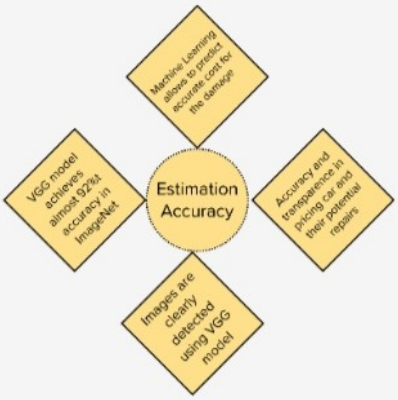
Group ideas

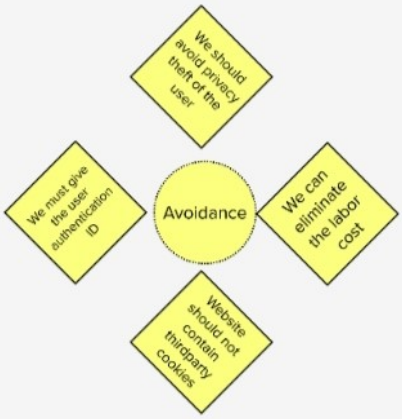
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.



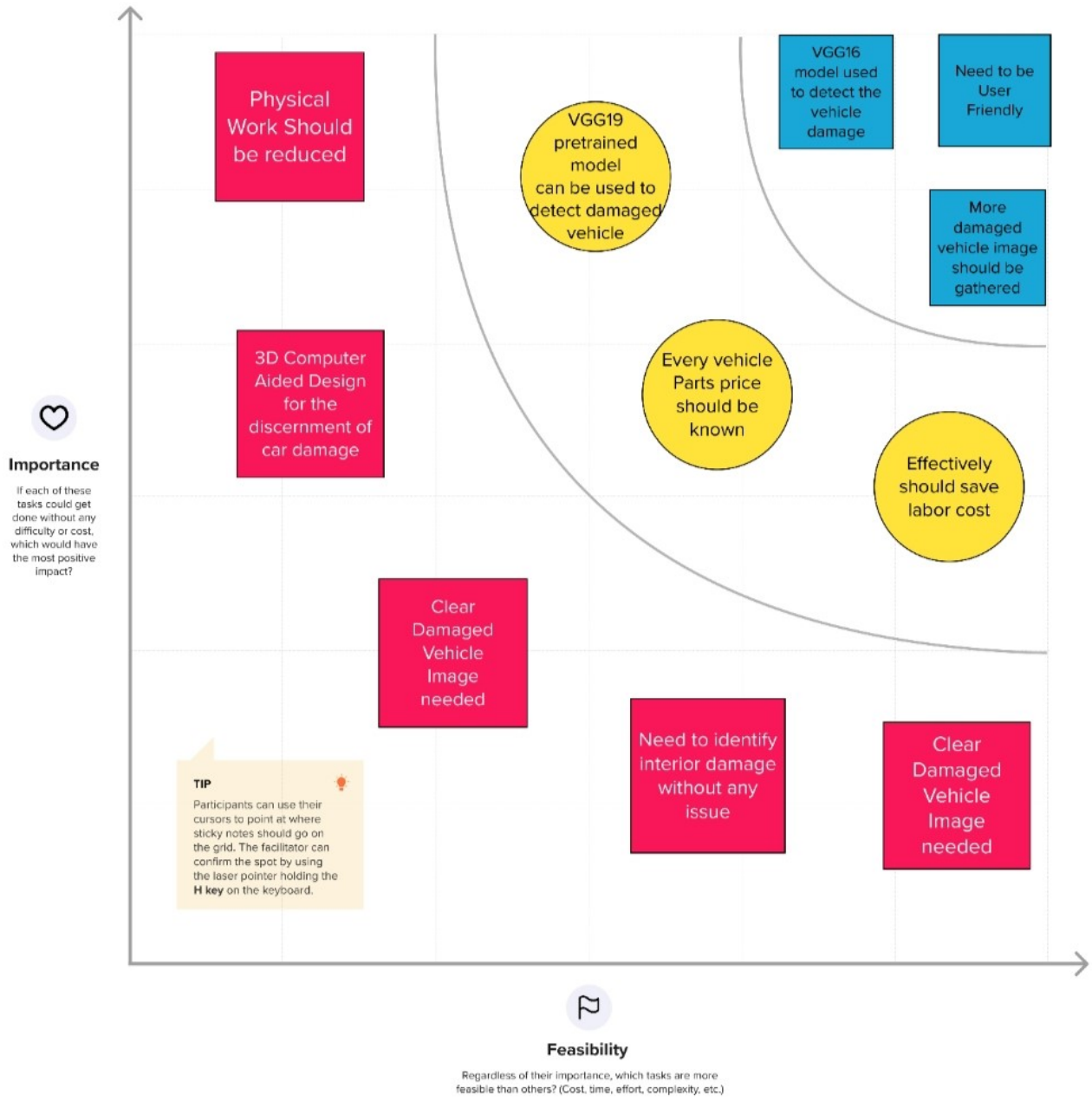


4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



Proposed Solution :

Novelty :

A collection of ML algorithms with an API that makes use of computer vision make up the Car Damage Recognition system. The algorithms, which are based on deep learning, automatically identify the body of a car and assess the severity of the damage. The analysis process can be accelerated by up to seconds using parallel machine learning and analytical pipelines.

1. Identify an automobile.
2. Choose the car's components.
3. Calculate the cost and preliminary damage to the car's components.

30 seconds for submitting a claim. Machine learning makes it possible to identify damaged auto parts, access damage, anticipate the type of repair that will be required, and calculate the potential cost of the repair.

Feasibility Of Ideas :

Companies can offer users an automatic examination of automotive damage thanks to a collection of tools and procedures called car damage assessment. It's crucial since it enables quick damage assessments and repair cost estimates without the need to wait for an inspector.

The installation of the required machine learning algorithms and the relevant training data have made vehicle damage detection possible. The following steps are necessary for each insurance claim to be processed:

1. Analyze the user-submitted image of the damaged car.
2. Examine a vehicle model.
3. Consider the angle at which the car is traveling.
4. Find faulty auto parts.
5. Evaluate the extent of component damage.
6. Produce a report.

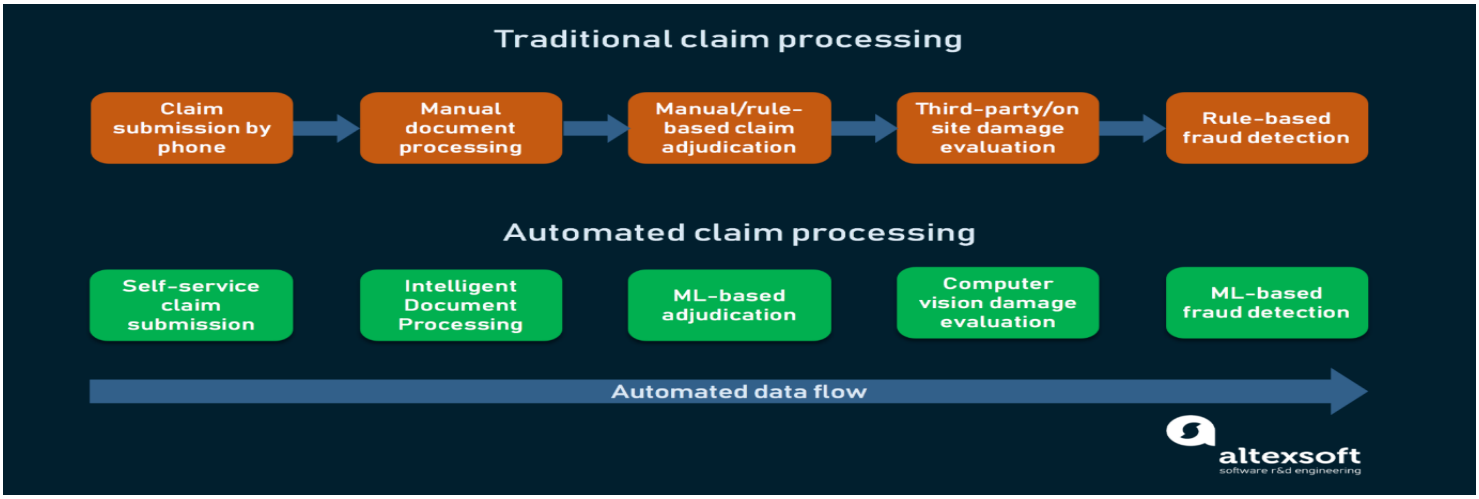
Framework for Car-Damage-Detection Algorithms:

The system for segmenting and detecting vehicle damage was developed in this paper using the Mask RCNN model. The graphic illustrates how an image of the damaged area of the car is chosenand gathered in accordance with the requirements, and the data annotated using theLabelMe annotation tool to create a dataset in the json format that is split into a training set and test.

Business Model :

The approach reduces the amount of time it takes to process data, protects against form fraud (by 80% or more), lowers the cost of hiring new employees, and occasionally speeds up image data analysis.

The application is utilized on-site and directs the user's actions to fulfill the photo requirements. Businesses that use Car Damage Recognition replace the time-consuming, human-operated claims processing and approval procedure with analytical technologies and machine learning algorithms.



Scalability :

enables the scaling of the claim settlement process utilizing an automated framework based on cutting-edge methods and algorithms. The ability to rapidly and affordably fix the faults the system detects benefits insurance companies, car rental businesses, and auto repair shops. We used a number of strategies that had significantly better results than the traditional ones to increase accuracy and speed up the training process. It is crucial to identify the ideal learning rate area because it has a significant impact on the network's efficiency and speed. By increasing the learning rate until the loss stops dropping, as described by Leslie N. Smith's method, a good learning rate bound can be determined. Then, by estimating "acceptable boundaries," we select an ideal learning rate.

PROBLEM STATEMENT

Why do we need an Intelligent vehicle damage cost assessment system?

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.

OUR PLAN:

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 from scratch from 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.

List of problem statements:

❖ A car insurance settlement claim is a process that requires near-perfect accuracy in order to avoid deceiving the customer. If such models are to be trained on the huge data sets required to achieve such accuracy, it is difficult and time-consuming to obtain such sets. In addition, these large datasets also require substantial amounts of storage space and processing resources.

❖ Furthermore, the training and evaluation phases of such systems usually take a long time to complete, which places significant restrictions on the scalability of the system.

❖ The field of Computer Vision is still in its inchoate state and is not mature enough to deal with modular phone camera quality images. Angle, lighting, and resolution are factors that can easily cause major disruptions in image classification

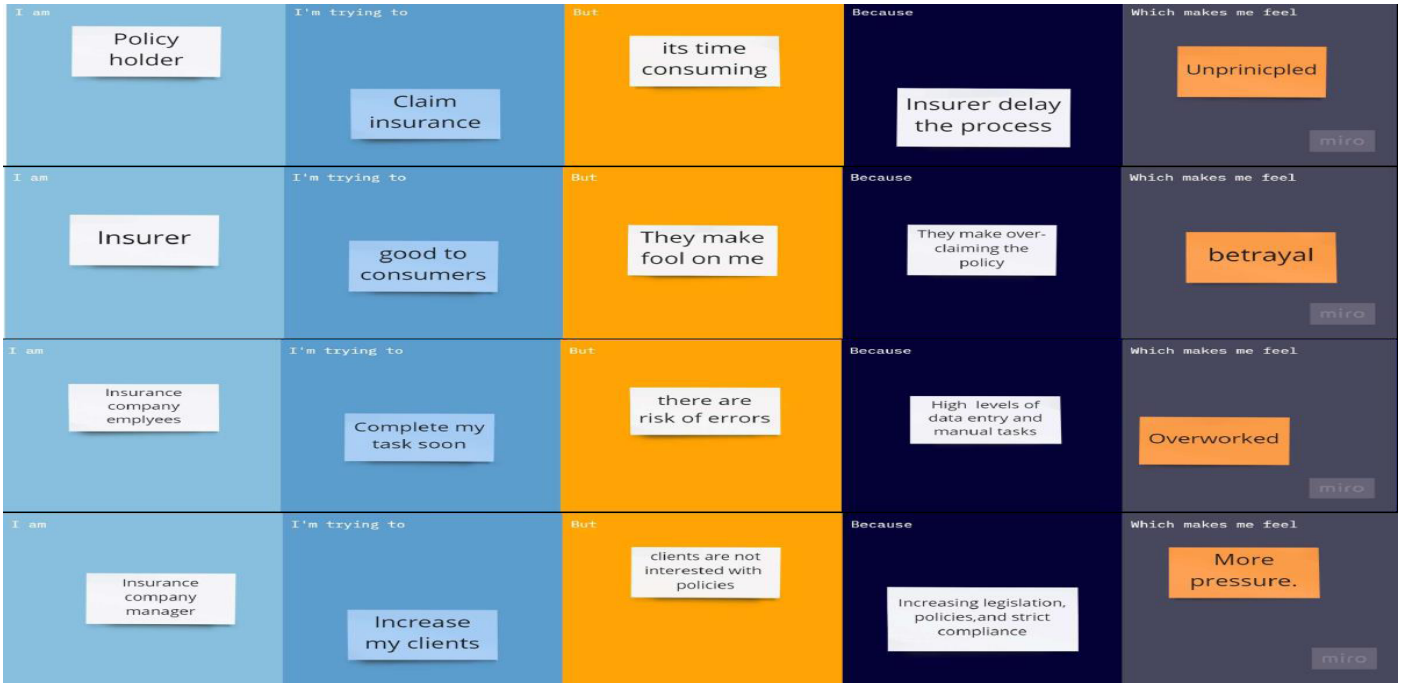
❖ Car insurance settlement claims require near-perfect accuracy to avoid deceiving the customer in the process. Such models have to be trained on huge data sets that are very difficult to obtain.

❖ Running such large datasets to ensure maximum accuracy imposes hardware limitations. Storing, training, and delivering such large datasets via the cloud requires expensive architectures.

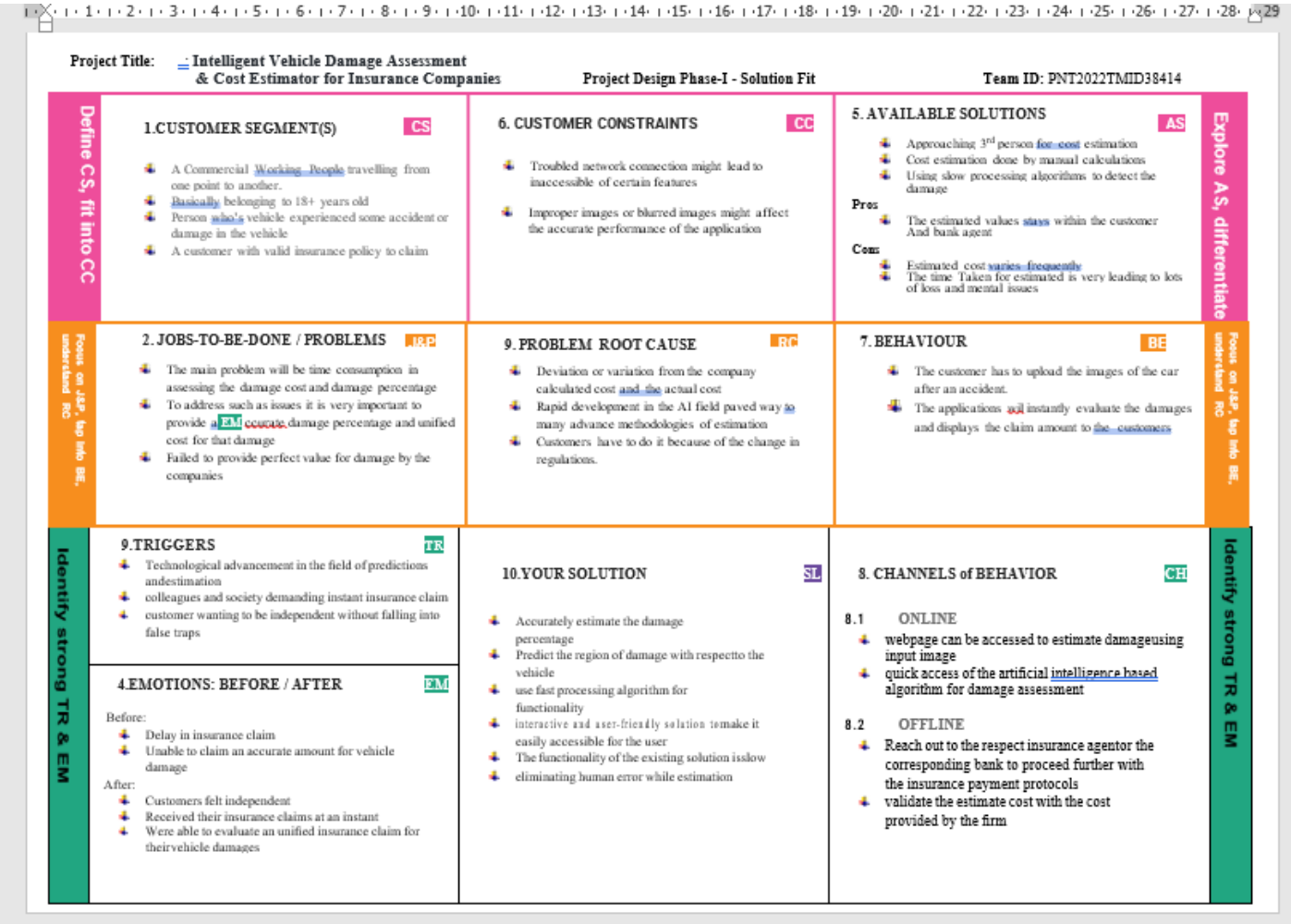
❖ The task of manually approving or disputing a claim falls on staff who must be both well-trained and well-equipped to deal with a variety of situations, both expected and unexpected.

❖ Manual approval processes are often time-consuming and require a significant amount of staff to be trained to handle a variety of claims.

PROBLEM STATEMENTS:



PROBLEM SOLUTION FIT:



Project Title: Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

Project Design Phase-I - Solution Fit

Team ID: PNT2022TMID38414

Define CS, fit into CC

1.CUSTOMER SEGMENT(S) CS

- A Commercial Working People travelling from one point to another.
- Basically belonging to 18+ years old
- Person who's vehicle experienced some accident or damage in the vehicle
- A customer with valid insurance policy to claim

6. CUSTOMER CONSTRAINTS CC

- Troubled network connection might lead to inaccessible of certain features
- Improper images or blurred images might affect the accurate performance of the application

5. AVAILABLE SOLUTIONS AS

- Approaching 3rd person for cost estimation
- Cost estimation done by manual calculations
- Using slow processing algorithms to detect the damage

Pros:

- The estimated values stays within the customer And bank agent

Cons:

- Estimated cost varies frequently
- The time Taken for estimated is very leading to loss of loss and mental issues

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

- The main problem will be time consumption in assessing the damage cost and damage percentage
- To address such as issues it is very important to provide a EM ccurate damage percentage and unified cost for that damage
- Failed to provide perfect value for damage by the companies

9. PROBLEM ROOT CAUSE RC

- Deviation or variation from the company calculated cost and the actual cost
- Rapid development in the AI field paved way to many advance methodologies of estimation
- Customers have to do it because of the change in regulations.

7. BEHAVIOUR BE

- The customer has to upload the images of the car after an accident.
- The applications will instantly evaluate the damages and displays the claim amount to the customers

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

9.TRIGGERS TR

- Technological advancement in the field of predictions andestimation
- colleagues and society demanding instant insurance claim
- customer wanting to be independent without falling into false traps

4.EMOTIONS: BEFORE / AFTER EM

Before:

- Delay in insurance claim
- Unable to claim an accurate amount for vehicle damage

After:

- Customers felt independent
- Received their insurance claims at an instant
- Were able to evaluate an unified insurance claim for theirvehicle damages

10.YOUR SOLUTION SL

- Accurately estimate the damage percentage
- Predict the region of damage with respectto the vehicle
- use fast processing algorithm for functionality
- interactive and user-friendly solution tomake it easily accessible for the user
- The functionality of the existing solution isslow
- eliminating human error while estimation

8. CHANNELS of BEHAVIOR CH

8.1 ONLINE

- webpage can be accessed to estimate damageusing input image
- quick access of the artificial intelligence based algorithm for damage assessment

8.2 OFFLINE

- Reach out to the respect insurance agentor the corresponding bank to proceed further with the insurance payment protocols
- validate the estimate cost with the cost provided by the firm

Identify strong TR & EM

Prerequisites

Software Required:

1. Anaconda 3 - JupyterNoteBook
2. Cloudant DB

Deployment Phases:

1. Front End :

a. HTML

b. CSS

2. BackEnd :

a. Python code

b. Flask

c. Cloudant DB

Python Packages :

1. Numpy
2. Pandas
3. Tensorflow 2.3.2
4. Keras 2.3.1
5. Scikit learn
6. Fask

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story/ Sub-Task)
FR-1	User registration	Download the app Registration through Gmail Create an account Follow the instructions
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Interface	Good Interface to userto operate
FR-4	Accessing datasets	Details about user Details aboutvehicle Details about insurance companies
FR-5	Mobile application	AI and camerasensor in the field can be access bymobile application.

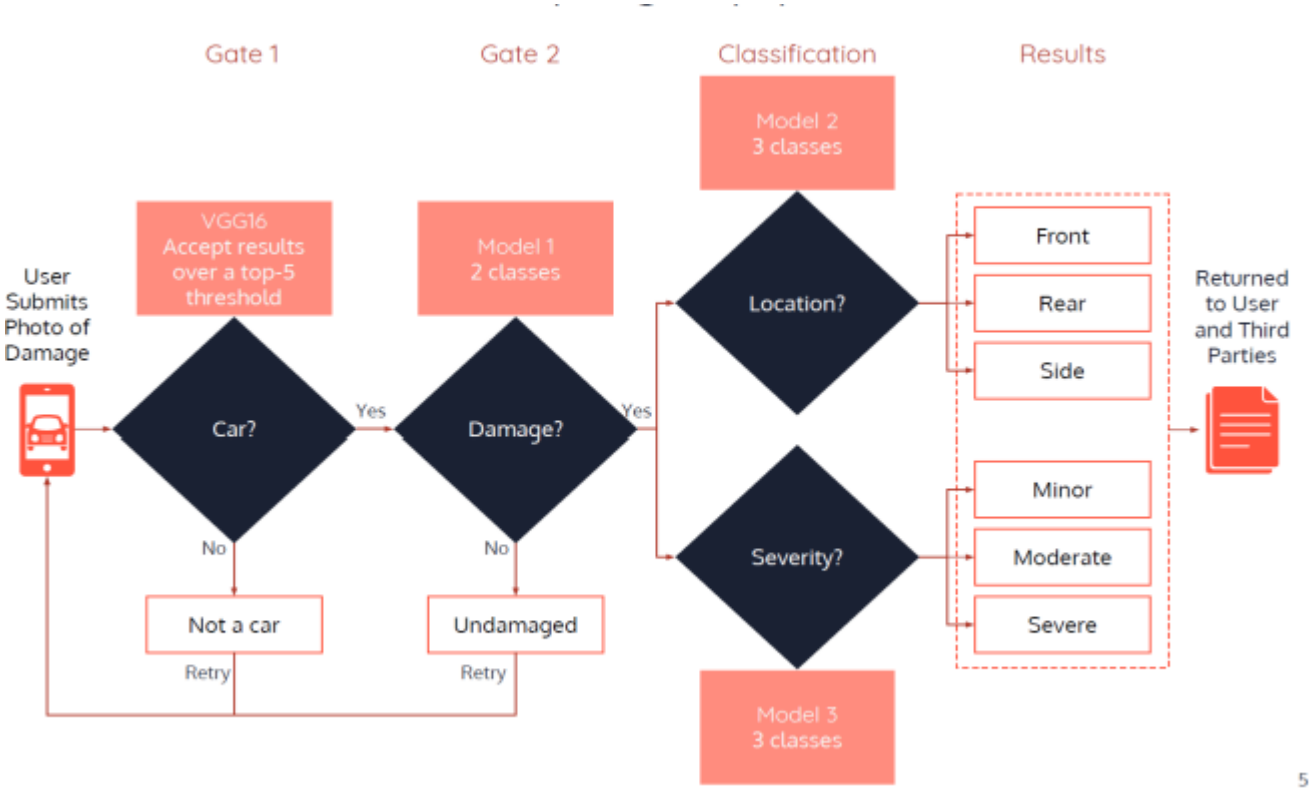
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	1. The smart claiming system for vehicle damageinsurance in bank companies
NFR-2	**Security**	2. We have designed this project to user easyto claim the insurance .
NFR-3	**Reliability**	3. This project will help the userto claim the insurancecost based on vehicle damage .It gives the exact valueto user.This helpsuser to get correct cost without any failure.

NFR-4	**Performance**	4. AI devices and sensorsare used to indicate the userto estimated the cost of the vehicle.AI camera to scan the damaged vehicle and gives exact cost insurance to user.
NFR-5	**Availability**	5. This application is designed for all devices and also Available in apk.
NFR-6	**Scalability**	6. This project is more scalability in our present andfuture uses to estimate the cost exactly to user.

Data Flow Diagrams :

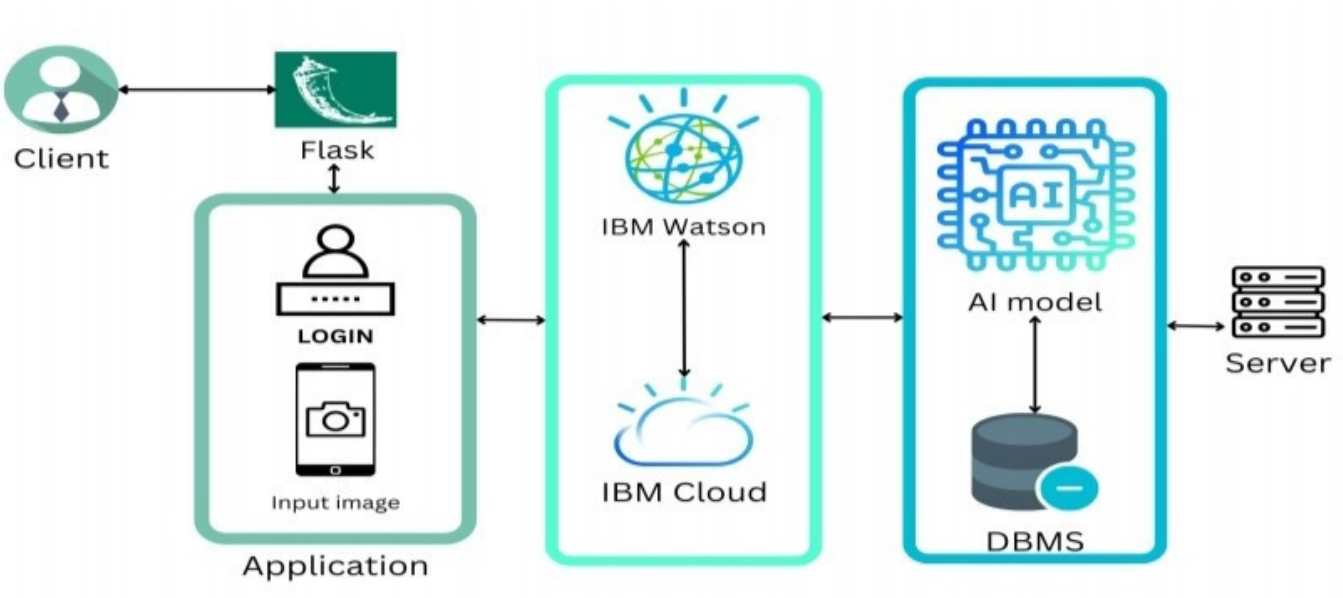


- The Project is Based AI Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies.
- It is application is use for claim insurance for damaged vehicle to pay a correct amount.
- We have best customer support the user.
- Application is user-friendly interface to all users.
- It give exact estimated value for the damaged vehicle.
- This model can also be used by lenders if they are underwriting a car loan, especially for a used car.

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
Customer Details	Login	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
Customer Uses	Dashboard	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
Customer options	Details about banks	USN-4	As a user, I can register for the application through Gmail	I can register & access the Detaile through gmail	Medium	Sprint-2
Customer Must do	Camera scanner	USN-5	As a user, I can log into the application by entering email & password	I can scan the entire vehicle In camera	High	Sprint-1
Customer Value	Details about cost based on damage	USN-6	It gives the estimation cost based on the damage.	I can get the estimated cost price.	Medium	Sprint-2
Customer Care Executive	Good customer support	USN-7	We have good customer support to the user to apply the insurance	I can get good customer support.	Low	Sprint-2
Administrator	To finish the Customer Work	USN-8	We will finish the customer needs in good manner without any failure.	I can get good customer support.	High	Sprint-1

Technical Architecture:



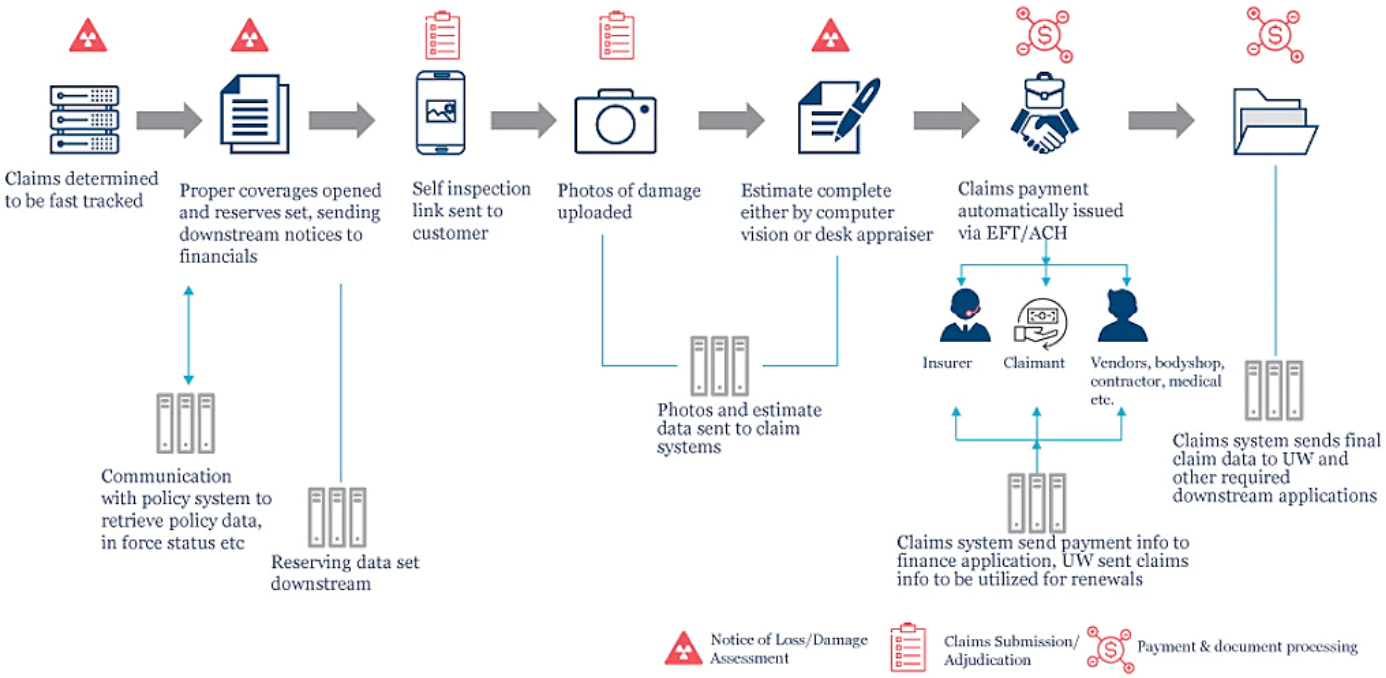


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user interacts with the web UI application	HTML,CSS, Python
2.	Application Logic-1	Getting userinput image	Python
3.	Application Logic-2	Getting modeloutput for damageprediction	IBM Watson,Python
4.	Application Logic-3	Getting modeloutput for cost estimation	IBM Watson,Python
5.	Database	Data Type – Images and user inputsdetails arestored	MySQL, Js, IBM DB2
6.	Cloud Database	DatabaseService on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	Receiveduser details and received userinputimages of the vehicleis stored in cloud	IBM Block Storage, IBM cloud
8.	MachineLearning Model	Purpose of the AI Model is for estimating the cost of the damaged vehicle.	ObjectRecognition Model, and CNN based model for damage estimation
9.	Infrastructure (Server / Cloud)	On cloud serverwe will be deploying the AI Modelusing flask in the webpage	PythonFlask

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Open-source frameworks usedis IBM Watson	Technology of OpenSource framework-IBM Watson
2.	Security Implementations	IBM Cloud	Certified Watson assistant for Encrypted file systems, Encrypted storagesystems, Key management systems.
3.	Scalable Architecture	Web server - static and dynamic website contentpresent in the website willbe update baseduponuser demandsand suggestion Application server - updation of the basic functionality of the website and integration of newlogicwithin the websitecan be done Database server- based uponthe varying inputsgiven by the user the database will be modified constantly	IBM Watson Assistant, Python, MySQL
4.	Availability	The AI model is made available instantly to user at any point of time	IBM Watson Cloudassistance
5.	Performance	IBM Watson –automate processes, The deep learning modelis trained usingIBM Watson studio for better performance and quick accessibility .	IBM Watson Assistant

Product Backlog, Sprint Schedule& Estimation (4 Marks)

Sprint	Milestone	User Story Number	Description	Duration	Priority	Team Members
Sprint 1	Project Objectives	USN-1	Project Objectives are what you plan tom achieve.	1Week	Low	Logesh E
Sprint 1	Data Collection	USN-2	It is the process of gathering and measuring variables in an establish system which than enables one to answer relevant questions and evaluate the outcomes.	2 weeks	Medium	Logesh E

Sprint 1	Image Preprocessing	USN-3	It is a system to perform some operations on an image, in order to get an enhanced image to tries	1 Week	High	Logesh E Deepak P
Sprint 2	Model Building	USN-4	Is the process of developing a probabilistic model that best describes the relation between the depended and independent variables.	2 Week	High	Logesh E Deepak P
Sprint 2	Import & Load the Model	USN-5	With both the training data defined and model defined, its time to configure the learning process.	1 Week	Low	Logesh E Nagaraj V
Sprint 2	Train & Test the Model	USN-6	As a user, let us train our model with image dataset.		Low	Kishore Kumar B
Sprint 2	Save the Model	USN-7	As a user, the model is saved and integrated with an android application or web application in order to predict something.		Low	Logesh E Nagaraj V
Sprint 3	Cloudant Databasse	USN-8	Higher levels of compliance,security and administrator are made possible by IBM Cloud's solutions,which also featuretried-and-true architecture patterns and procedures for quick delivery of mission-critical workloads.	1 Week	Medium	Logesh E Deepak P
Sprint 3	Application Building	USN-9	The process of writing a computer programme is called application.Create our flask application in this phase,which will have an interface and operate in our local browser.	2 Week	High	Logesh E Deepak P

Sprint 4	Train The Model on IBM	USN-10	A Deep learning network architecture that doesn't require human feature extraction because it learns straight from the data.	1 Week	Medium	Logesh E Kishore Kumar B
Sprint 4	Cloud Deployment	USN-11	As a user I an access the web application and make the use of the product from anywhere.	1 Week	High	Logesh E Deepak P

Project Tracker, Velocity& Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint StartDate	Sprint End Date (Planned)	Story Points Completed (as on Planned EndDate)	Sprint ReleaseDate (Actual)
Sprint-1	20	6 Days	24 Oct2022	29 Oct2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct2022	05 Nov2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov2022	12 Nov2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov2022	19 Nov2022	20	19 Nov 2022

Velocity:

Imagine we have a 10-day sprintduration. The velocityof the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iterationunit (story pointsper day)

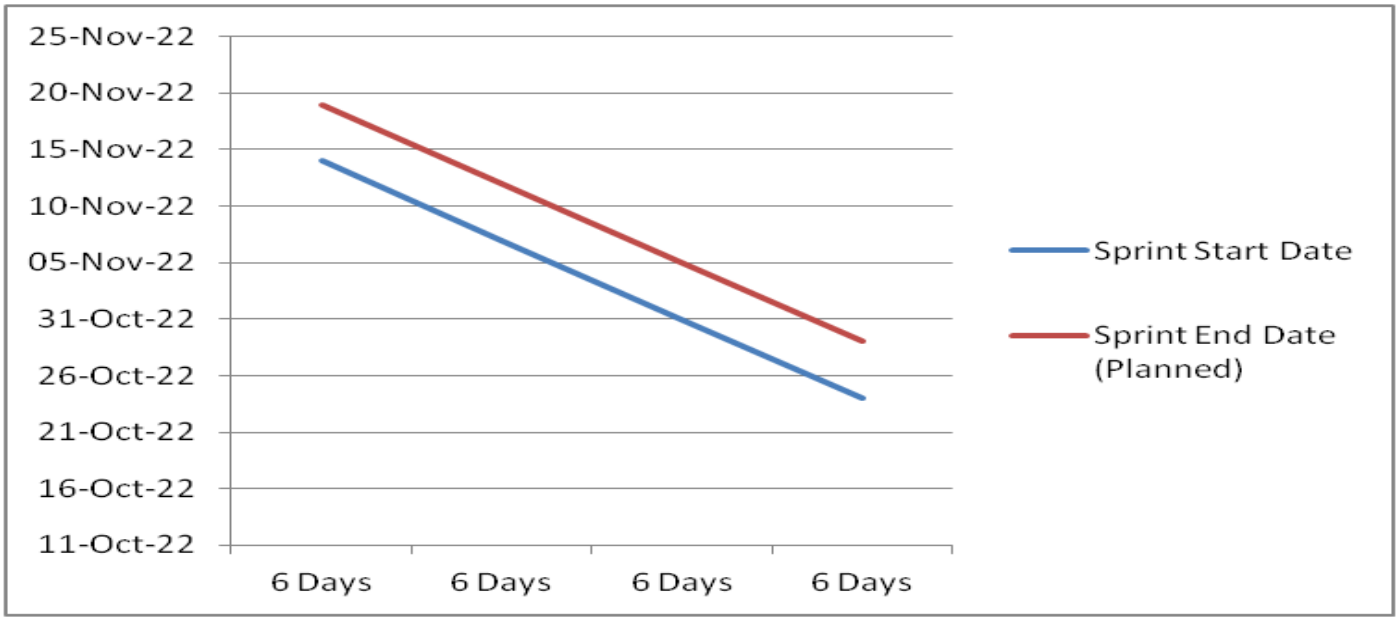
AV = sprint duration / velocity

= 20 / 6

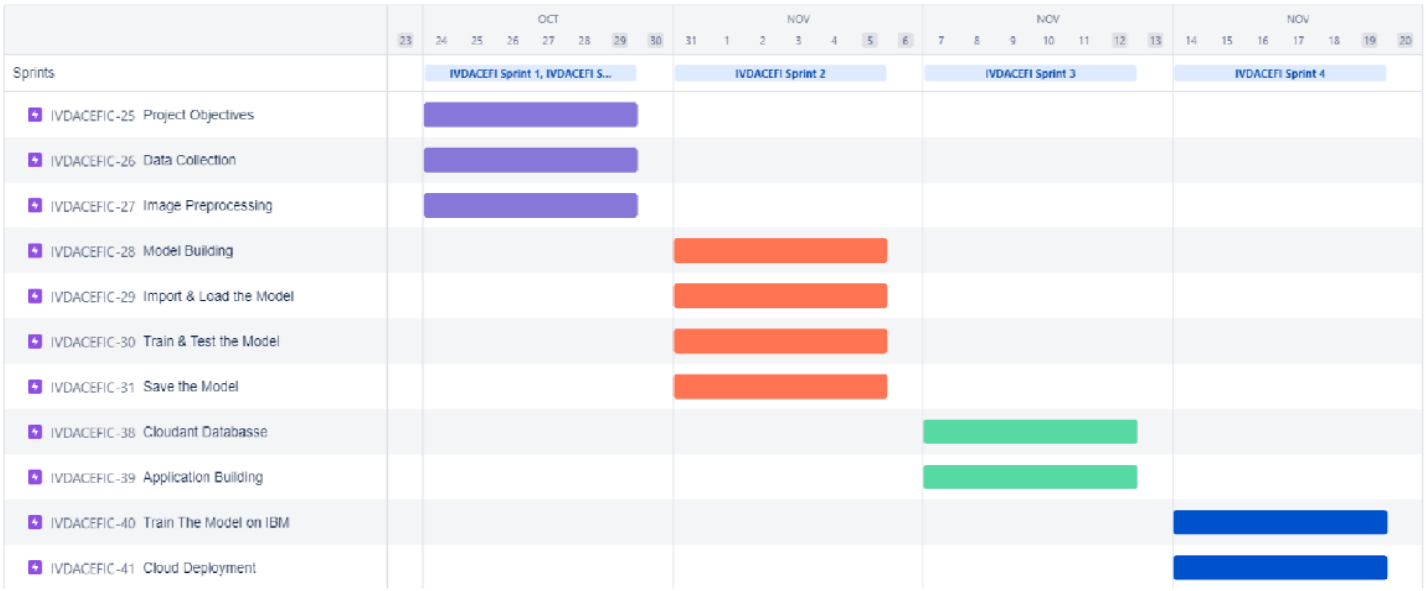
= **3.33**

Burndown Chart:

A burn-down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn-down chartscan be applied to any project containing measurable progress over time.



JIRA STORY POINTS :

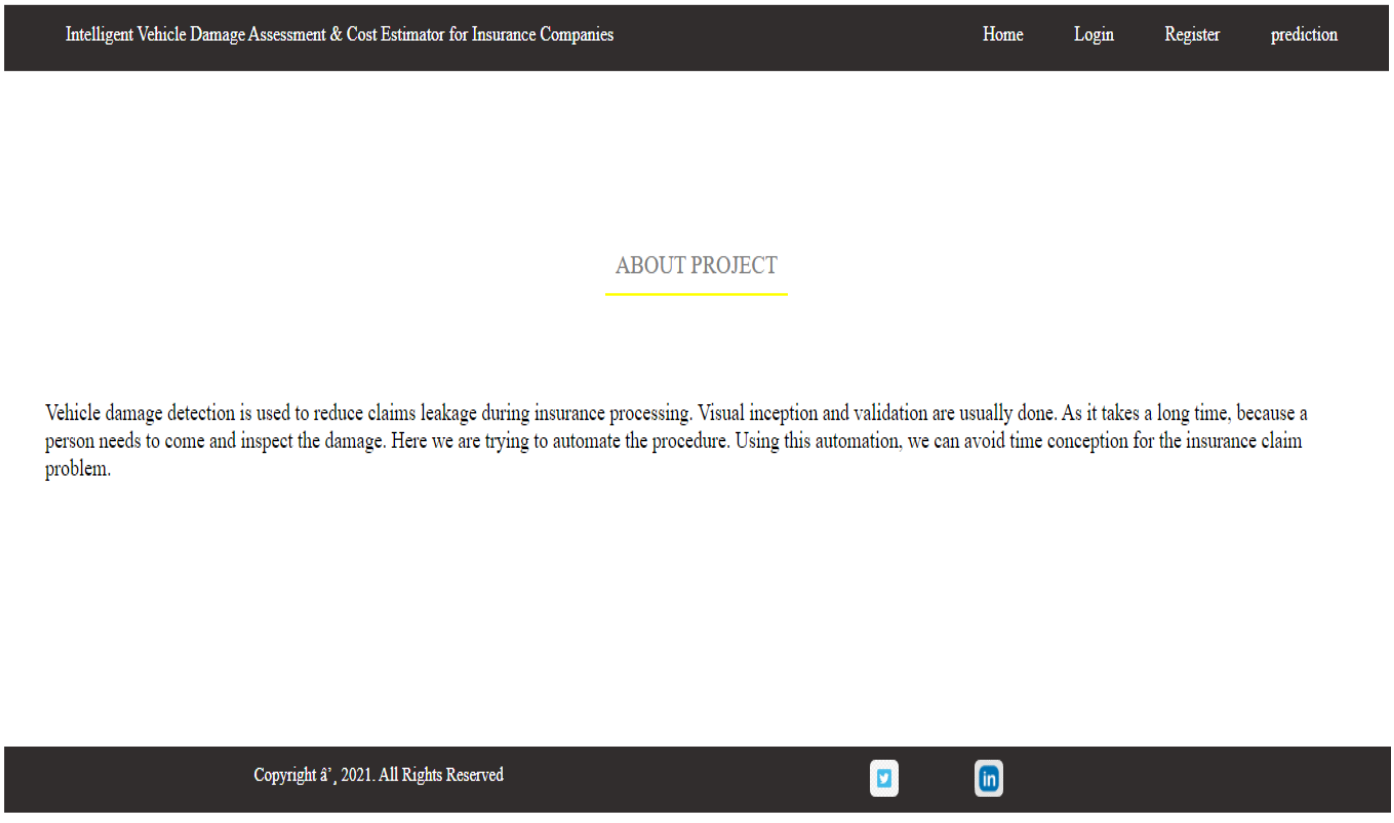


CODING & SOLUTIONING

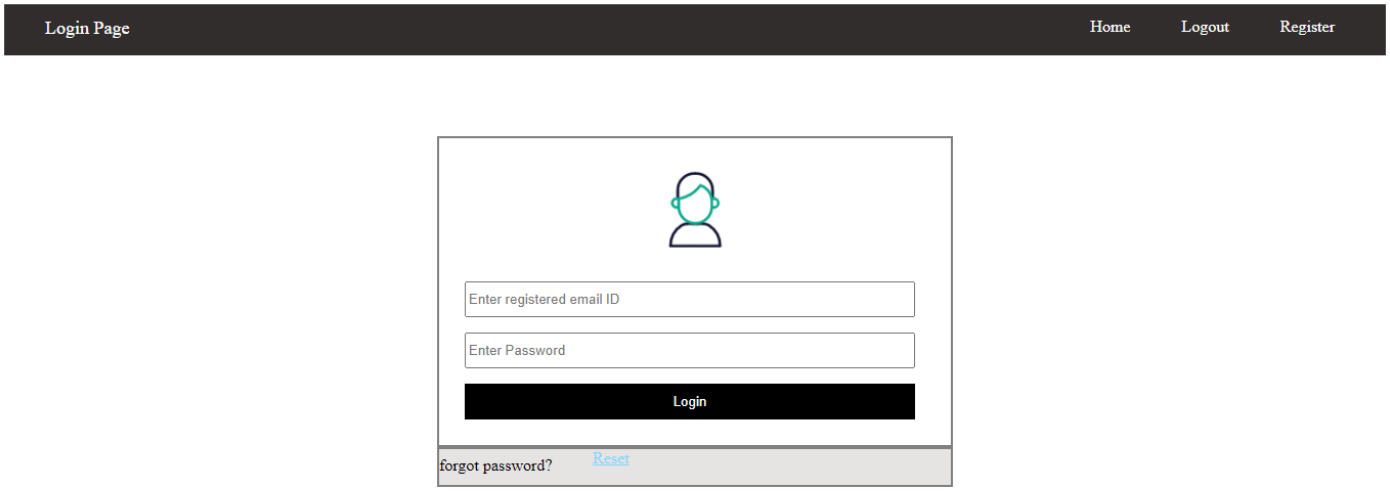
(Explain the featuresadded in the project along with code)

Feature 1:++

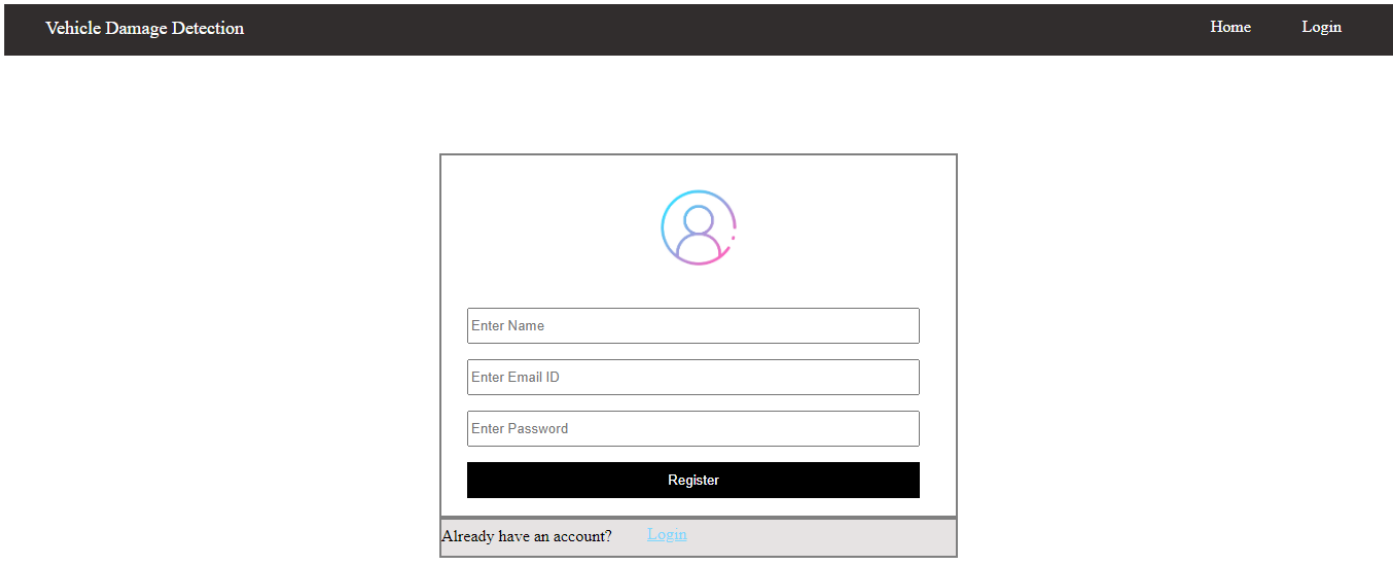
index Page:



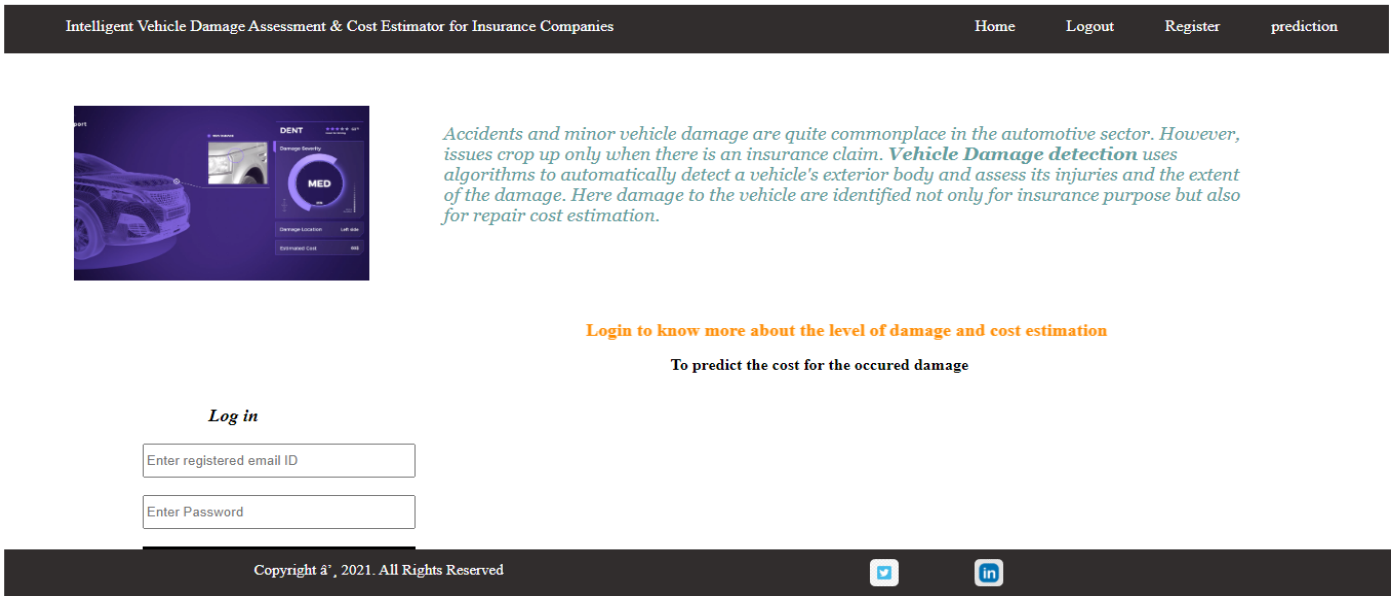
Login :



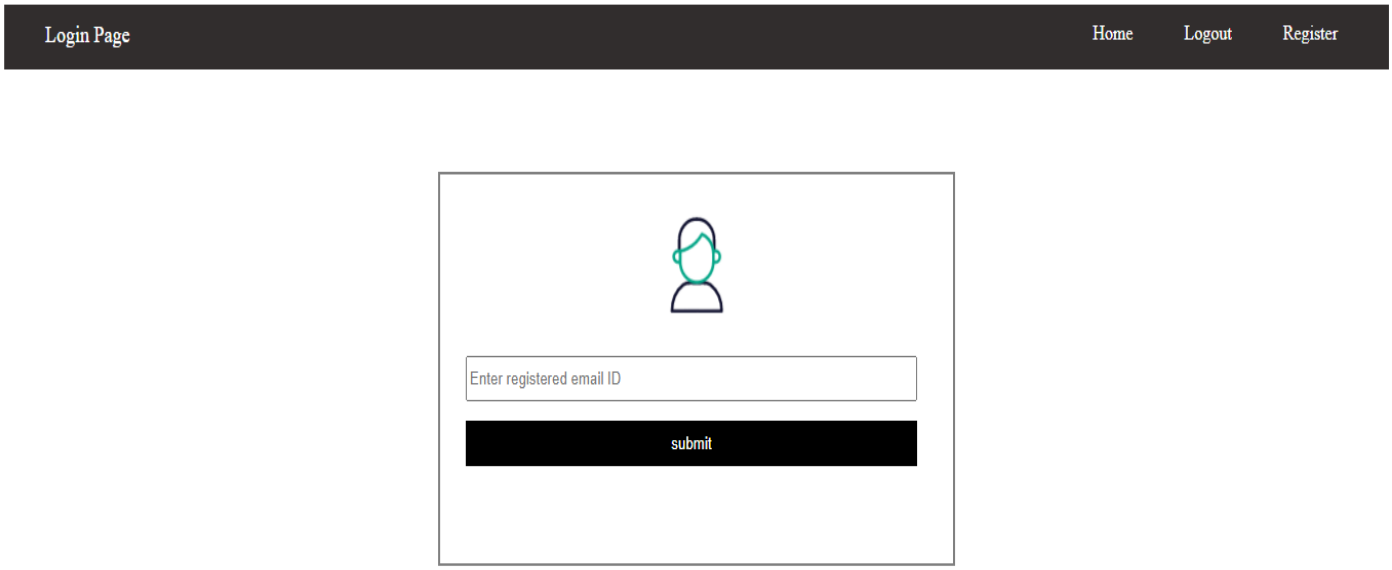
Register :



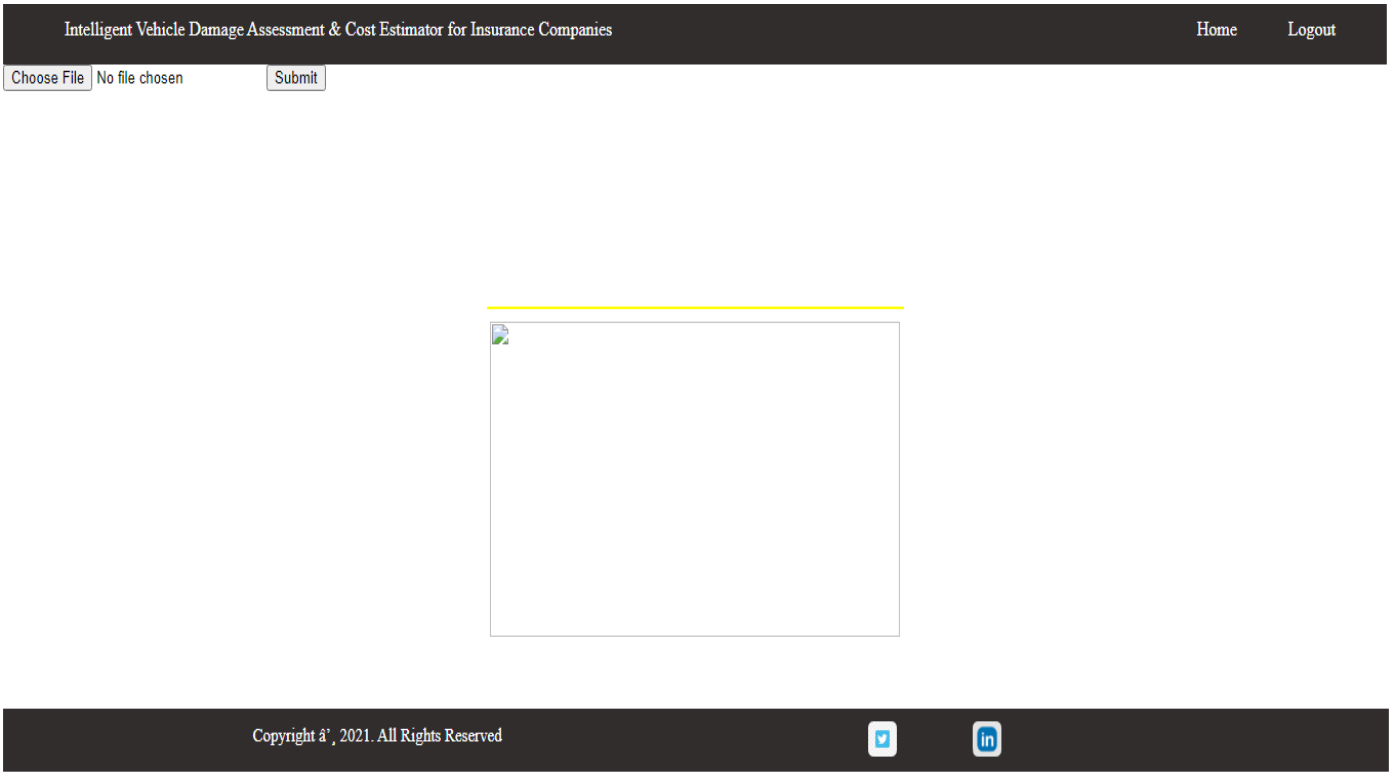
Dashboard :



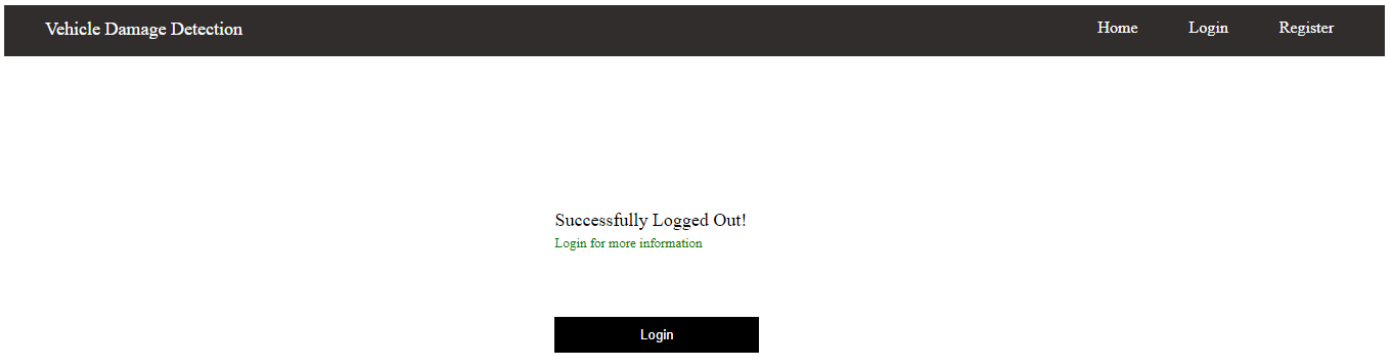
Forget Password :



Prediction:

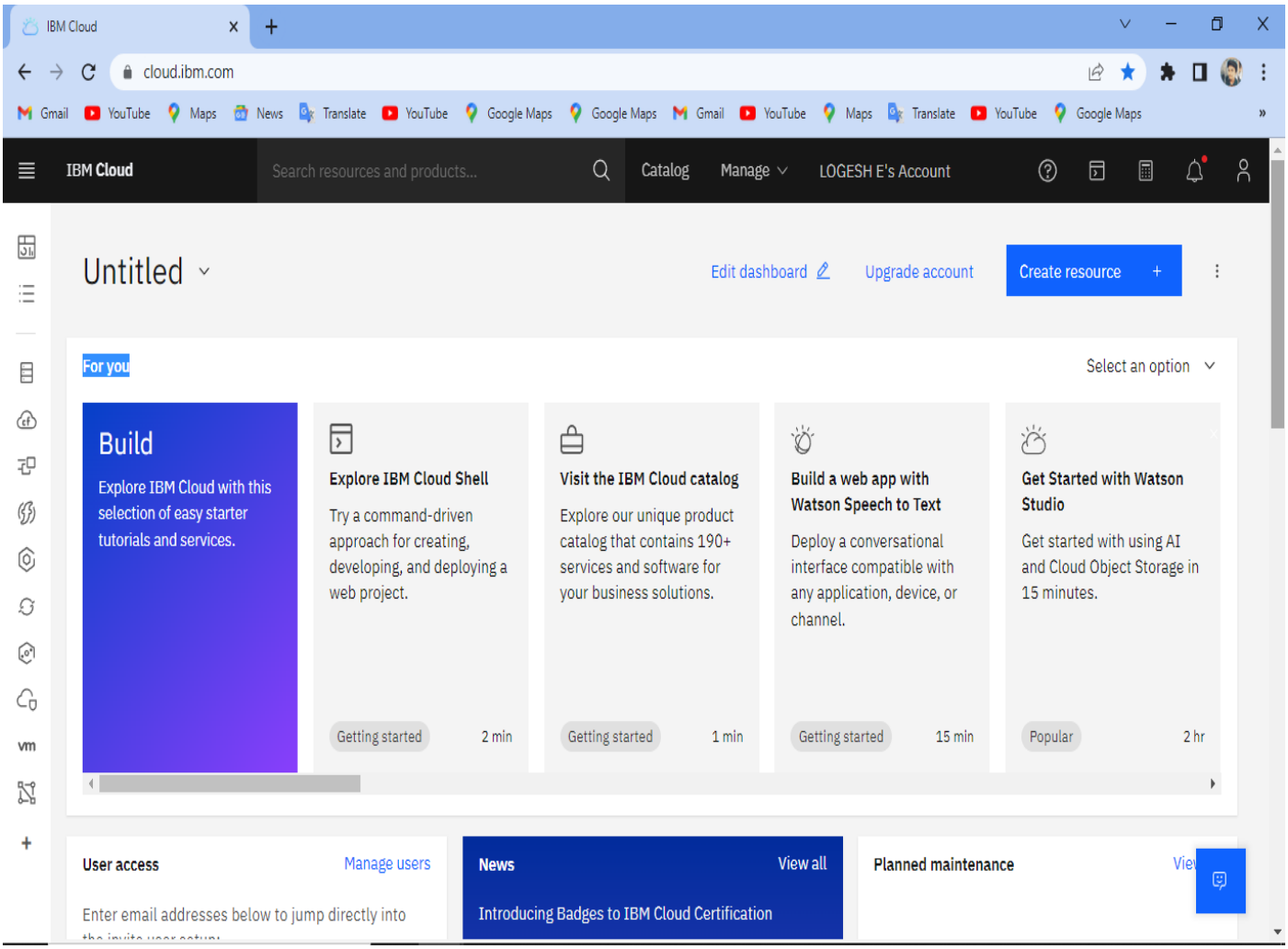


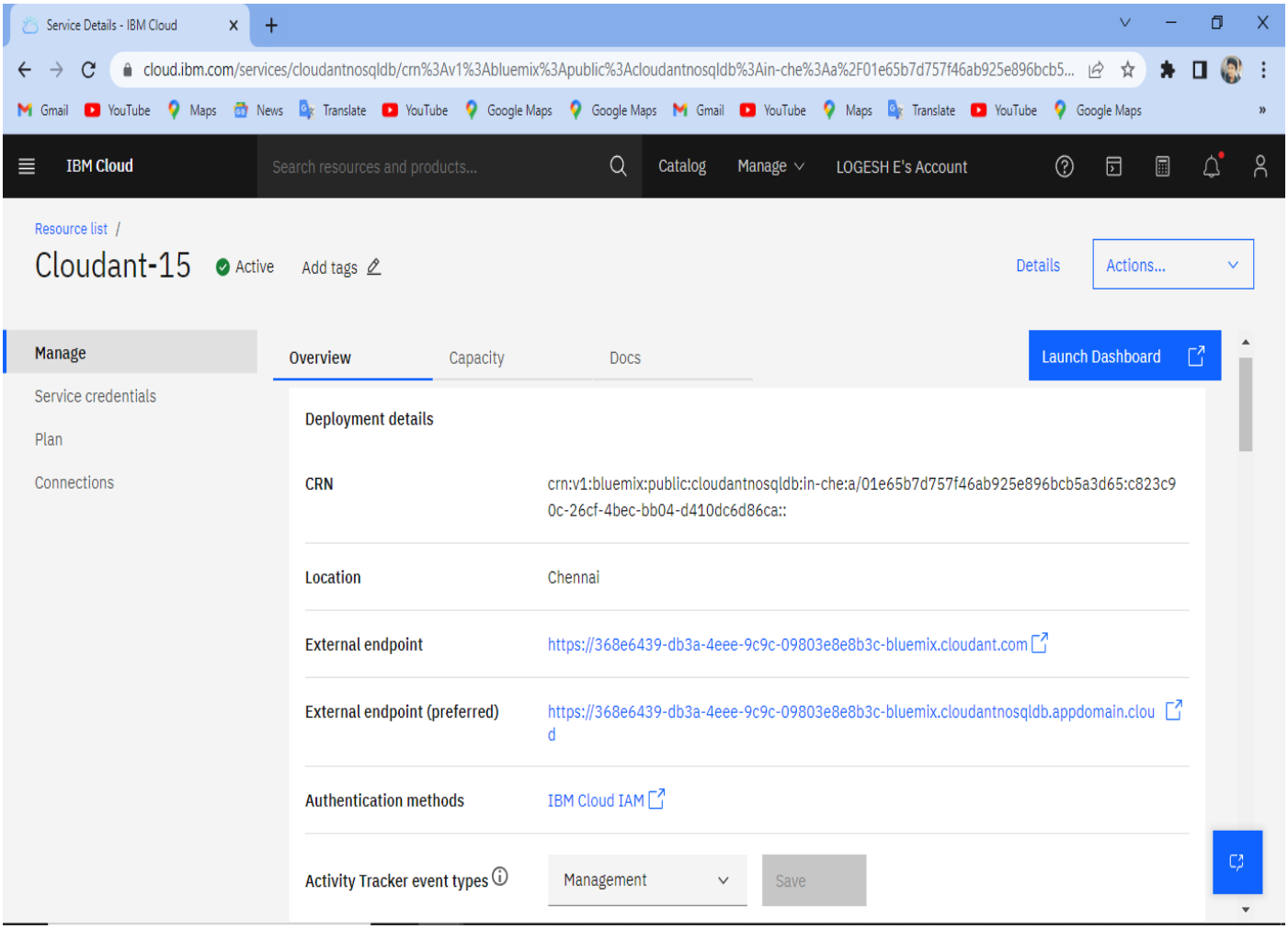
Log out :



Feature 2

Database :





TESTING

Test Cases:

1. Verify user is able to see the Login/Signup popup when user clickson login button
2. Verifythe UI elements in Login/Signup popup
3. Verifythe UI elements in Register option for new user
4. Verifyuser is able to login into application with Valid Credentials
5. Verify user is not able to login into application with InvalidCredentials
6. Verify user is able to login into dashboard, if not user needs toregister
7. Verify user is able to register successfully and direct into loginpage
8. Verifythe user is able to upload the images
9. Verifythe user is able to view the predictedcost
10. Verify user is able to see dashboard, if not need to login with correc tlogin credentials
11. Verify user is able to logout properlyand whether able to login again

1. PurposeofDocument

The purpose of this documentis to briefly explain the test coverageand open issuesof theproject at the time of the release to User AcceptanceTesting (UAT).

2. DefectAnalysis

This report shows the number of resolved or closed bugs at each severity level, and how theywere resolved

Resolution	Severity1	Severity2	Severity3	Severity4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won'tFix	0	5	2	1	8
Totals	24	14	13	26	77

3. TestCaseAnalysis

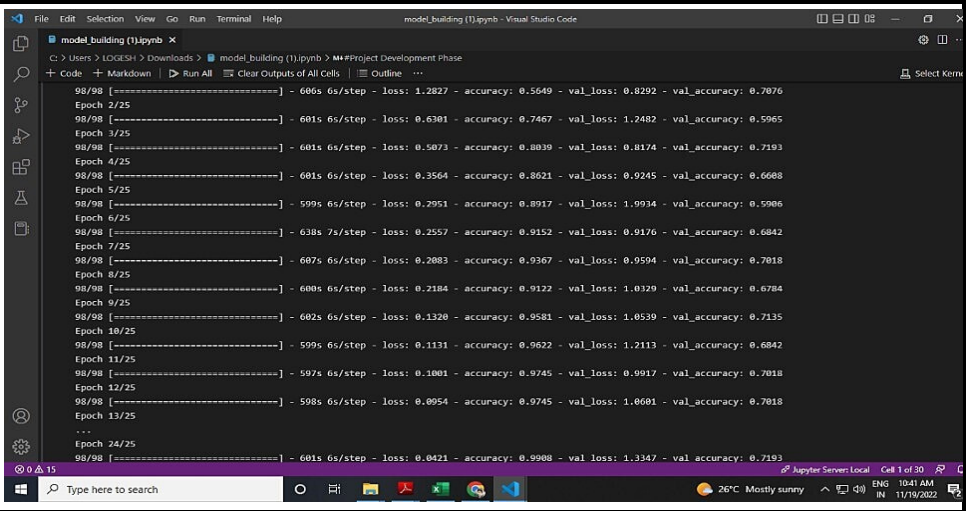
This report shows the number of test cases that have passed, failed,and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
OutsourceShipping	3	0	0	3
ExceptionReporting	9	0	0	9
FinalReportOutput	4	0	0	4
VersionControl	2	0	0	2

Project Development Phase
ModelPerformance Test

Model Performance Testing:

Project team shall fill the followinginformation in the model performance testing template.

S.No.	Parameter	Values
1.	Model Summary	
2.	Accuracy	Training Accuracy for body model- 98.6% Validation Accuracy for body model- 67% Training Accuracy for levelmodel 99.79% Validation Accuracy forlevel model - 62% -

ADVANTAGES & DISADVANTAGES:

- Advantages:
  - Insurance companies are using artificial intelligence (AI) toassess damage to vehicles through photographs and help inmaking claims.
  - This is being made possible through computer vision and machinelearning technology which look at digital photographsof damaged cars and quickly estimate the repair cost.
  - The AI calculates the full repair costs by identifying which partsof the vehicle have been affected and how. It will provide a detailed estimate, including recommended repair and paint, as well as costs and labor hours.

Disadvantages:

- The manualapproach for submitting an insurance claim will takelonger.
- The companyacts badly and currently doesn'tmake payments asa result of false accusations.
- May result in subparcustomer service.

CONCLUSION

1. In this research proposal, the problems of vehicle damage analysis and positionand severity prediction will be dealt with using a neural network-based automobile detecting system.
2. This projectdoes a number of tasks in one go. The technique will undoubtedly help the insurance companies undertake much more extensive and organized examinations of the car damage.
3. Simply providing a snapshot of the vehicle will allow the system to examineit, identify whetherany damage is present, where it islocated, and how bad it is.

FUTURE SCOPE

In future work, we will need to apply numerousregularization algorithms with a large dataset. If we have higher quality datasets that include the characteristics of a car (make, model, and year of production), location data, kind of damaged part, and repair cost, we can predict the cost of a broken automotive component more correctly and reliably. Together with a focus on the vehicle insurance sector, this study paves the way for future photo recognition efforts. By removing human bias, the study was able to accurately validate the existence of damage, its location, and its severity. By including the on-the-flydata augmentation methodologies, they can be further improved.

APPENDIX

Source Code:

Main :

```
from flask import Flask, app, request, render_template
import os
import flask
import re
import flask_login
import base64
from PIL import Image
from io import BytesIO
import datetime
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from cloudant.client import Cloudant
from cloudant.error import CloudantException
from cloudant.result import Result, ResultByKey
```

```
#os.chdir('Project Development Phase\Sprint-3')
model1 = load_model('Model/level.h5')
model2 = load_model('Model/body.h5')


def detect(frame,model1,f):
    img = cv2.resize(frame,(244,244))
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    if(np.max(img)>1):
        img=img/255.0
    img = np.array([img])

    prediction = model1.predict(img)
    if(f):
        label= ['front','rear','side']
    else:
        label =['minor','moderate','severe']
    preds = label[np.argmax(prediction)]
    return preds



client = Cloudant.iam(
    '6803cce5-6842-45c7-b5e2-65a08f2d80fa-bluemix','reiQA6VSK7w6yG9lKhhn5Vt-
lAWAzrrzZTV4ZnqNoaA_',connect=True)
name = 'name'
email = 'a@b.c'
password = '123'

user_database = client.create_database('user_database')
user_image_database = client.create_database('user_image_database')

def image_database_updation(name,email,imagestr):
    global user_image_database
    now = datetime.datetime.now()
    json_image_document={
        'name':name,
        'email':email,
        'image':imagestr,
        'datetime':now.strftime("%m/%d/%Y, %H:%M:%S")
    }
    new_image_document  = user_image_database.create_document(json_image_document)
    if(new_image_document.exists()):
        print('database updated')
    else:
        print('database couldn\'t be edited')
    return

def image_database_retrieval():
```

```
    global user_image_database
    image_result_retrieved = Result(user_image_database.all_docs,include_docs=True)
    image_result ={}
    for i in image_result_retrieved:
        if(i['doc']['email'] in image_result.keys()):
            # like current date> rx date('str')
            n = datetime.datetime.strptime(i['doc']['datetime'],'%m/%d/%Y, %H:%M:%S')
            o = datetime.datetime.strptime(image_result[i['doc']['email']]['date'],'%m/%d/%Y, %H:%M:%S')
            if(n>o):
       image_result[i['doc']['email']] = {'name':i['doc']['name'],'image':i['doc']['image'],'date':i['doc']['datetime']}
        else:
            image_result[i['doc']['email']] = {'name':i['doc']['name'],'image':i['doc']['image'],'date':i['doc']['datetime']}
    return(image_result)

def database_updation(name,email,password):
    global user_database
    jsonDocument = {
        'name':name,
        'email':email,
        'password':password
    }
    newDocument = user_database.create_document(jsonDocument)
    if(newDocument.exists()):
        print('database updated')
    else:
        print('database couldn\'t be edited')
    return
#database_updation(name,email,password)

def database_retrieval():
    global user_database
    result_retrieved = Result(user_database.all_docs,include_docs=True)
    #print(list(result_retrieved))
    result = {}
    for i in list(result_retrieved):
        result[i['doc']['email']]={'name':i['doc']['name'],'password':i['doc']['password']}
    return result
#print(database_retrieval())
app = Flask(__name__)
app.secret_key = 'apple'
login_manager = flask_login.LoginManager()

login_manager.init_app(app)
users = {'a@b.c': {'password': '123'}}
class User(flask_login.UserMixin):
    pass


@login_manager.user_loader
def user_loader(email):
```

```
    data = database_retrieval()
    if email not in data:
        return

    user = User()
    user.id = email
    user.name = data[email]['name']
    return user


@login_manager.request_loader
def request_loader(request):
    email = request.form.get('email')
    data = database_retrieval()
    if email not in data:
        return

    user = User()
    user.id = email
    user.name = data[email]['name']
    return user
@app.route('/')
def index():
    if(flask_login.current_user.is_authenticated):
        return render_template('dashboard.html')
    else:
        return flask.redirect(flask.url_for('login'))

@app.route('/register',methods = ['GET','POST'])
def register():
    data = database_retrieval()
    if(flask.request.method == 'GET'):
        return render_template('register.html')
    email = flask.request.form['email']
    if(email in data):
        return render_template('register.html',flash_message='True')
    else:
        database_updation(flask.request.form['name'],email,flask.request.form['password'])
        #users[email]={'password':flask.request.form['password']}
        user  = User()
        user.id = email
        user.name = flask.request.form['name']
        flask_login.login_user(user)
        return render_template('dashboard.html',flash_message='True')



@app.route('/login',methods =['GET','POST'])
def login():
    data = database_retrieval()
```

```
    if(flask.request.method == 'GET'):

        return render_template('login.html',flash_message='False')
    email = flask.request.form['email']
    if(email in data and flask.request.form['password']==data[email]['password']):
        user  = User()
        user.id = email
        flask_login.login_user(user)
        return render_template('dashboard.html',flash_message='Fal')
    #flask.flash('invalid credentials !!!')
    return render_template('login.html',flash_message="True")
    #error = 'inavlid credentials')


@app.route('/dashboard',methods = ['GET','POST'])
@flask_login.login_required
def dashboard():
    if(flask.request.method == 'GET'):
        return render_template('dashboard.html',flash_message='False')
    email = flask.request.form['email']
    if(email in users and flask.request.form['password']==users[email]['password']):
        user  = User()
        user.id = email
        flask_login.login_user(user)
        return render_template('dashboard.html',flash_message="Fal")
    return render_template('dashboard.html',flash_message="Fals")


@app.route('/logout')
@flask_login.login_required
def logout():
    flask_login.logout_user()
    return render_template('logout.html')


@app.route('/prediction',methods = ['GET','POST'])
@flask_login.login_required
def prediction():
    if(flask.request.method=='POST'):
        img = flask.request.files['myFile']
        try:
            os.remove('static\imagedata\save.png')
        except:
            pass
        imgstr = base64.b64encode(img.read()).decode('utf-8')
        image_database_updation(flask_login.current_user.name,flask_login.current_user.id,imgstr)
        data = image_database_retrieval()
        print(flask_login.current_user.id)
        #print(len(base64.b64decode(data[flask_login.current_user.id]['image'].strip())))
        img_retrived = np.asarray(bytearray(base64.b64decode(data[flask_login.current_user.id]['image'])))
```

```
        print(img_retrived.shape)
        print()
        '''img_retrived = np.frombuffer(
           BytesIO(
              base64.b64decode(data[flask_login.current_user.id]['image'])
              )
           )'''
        print('################################')
        result1=detect(img_retrived,model1=model2,f=True)
        result2 = detect(img_retrived,model1=model1,f=False)
        value=''
        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 = '900 - 11000 INR'
        elif(result1 == 'side' and result2 == 'severe'):
           value = '12000 - 15000 INR'
        else:
           value = '16000 - 50000 INR'
        print(result1,result2,value)
        print('################################')
        img_retrived = Image.fromarray(img_retrived)
        img_retrived.save('static\imagedata\save.png')
        print('image uploaded and retrieved')
        return render_template('prediction.html',flash_message='True')
        #,imag=img_retrived)

    return render_template('prediction.html',flash_message='Flase')

if __name__ == '__main__':
    app.run(debug=True)
```

HTML CODES :

Index:

```
<html>
  <head>
    <title>index</title>
    <style type="text/css">
      #topmenu {
        width: 100%;
        background-color: 312D2D;
        height: 50px;
      }
      #hedder {
        color: white;
        padding-top: 13px;
        padding-left: 60px;
      }

      #home {
        float: right;
        padding-top: 13px;
        padding-right: 50px;
        color: rgb(222, 216, 216);
        font-size: medium;
      }
      #login {
        float: right;
        padding-top: 13px;
        padding-right: 50px;
        color: rgb(222, 216, 216);
        font-size: medium;
      }
      #register {
        float: right;
        padding-top: 13px;
        padding-right: 50px;
        color: rgb(222, 216, 216);
        font-size: medium;
      }
      #prediction {
        float: right;
        padding-top: 13px;
        padding-right: 50px;
        color: rgb(222, 216, 216);
        font-size: medium;
      }
      #about {
        text-align: center;
        padding-top: 10%;
```

```
    color: gray;
    font-size: 20px;
   }
   #content {
    padding-top: 50px;
    padding-left: 40px;
    padding-right: 40px;
    font-size: large;
   }
   #footer {
    width: 99%;
    background-color: 312D2D;
    height: 50px;
    position: absolute;
    bottom: 1%;
   }
   #textcontent {
    color: white;
    font-size: 15px;
    padding-left: 18%;
    padding-top: 1%;
   }
   #logo {
    margin-top: -1.5%;
    margin-right: 28%;
    float: right;
   }
  </style>
</head>
<body>
  <div id="topmenu">
   <div id="prediction">
    <a href="{{ url_for('prediction') }}" style="color: white;text-decoration: none;">prediction</a>
   </div>
   <div id="register">
    <a href="{{ url_for('register') }}" style="color: white;text-decoration: none;">Register</a>
   </div>
   <div id="login">
    <a href="{{ url_for('login') }}" style="color: white;text-decoration: none;">Login</a>
   </div>
   <div id="home">
    <a href="{{ url_for('dashboard') }}" style="color: white;text-decoration: none;">Home</a>
   </div>
   <div id="hedder">
    Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance
    Companies
   </div>
  </div>
  <div id="about">
   ABOUT PROJECT
```

```
    <hr style="width: 13%" color="yellow" />
   </div>
   <div id="content">
    <p>
     Vehicle damage detection is used to reduce claims leakage during
     insurance processing. Visual inception and validation are usually done.
     As it takes a long time, because a person needs to come and inspect the
     damage. Here we are trying to automate the procedure. Using this
     automation, we can avoid time conception for the insurance claim
     problem.
    </p>
   </div>

   <div id="footer">
    <div id="textcontent">Copyright Ⓒ 2021. All Rights Reserved</div>
    <div id="logo">
     <img
      src="/static/images/twitter.jpg"
      height="28px"
      width="28px"
      style="border-radius: 18%; margin-right: 40px"
     />

     <img
      src="/static/images/linkedin.jpg"
      height="28px"
      width="28px"
      style="margin-left: 30px; border-radius: 18%"
     />
    </div>
   </div>
  </body>
</html>
<html>
 <head>
  <title>index</title>
  <style type="text/css">
   #topmenu {
    width: 100%;
    background-color: 312D2D;
    height: 50px;
   }
   #hedder {
    color: white;
    padding-top: 13px;
    padding-left: 60px;
   }

   #home {
    float: right;
```

```
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#prediction {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#about {
  text-align: center;
  padding-top: 10%;
  color: gray;
  font-size: 20px;
}
#content {
  padding-top: 50px;
  padding-left: 40px;
  padding-right: 40px;
  font-size: large;
}
#footer {
  width: 99%;
  background-color: 312D2D;
  height: 50px;
  position: absolute;
  bottom: 1%;
}
#textcontent {
  color: white;
  font-size: 15px;
  padding-left: 18%;
  padding-top: 1%;
```

```
    }
    #logo {
      margin-top: -1.5%;
      margin-right: 28%;
      float: right;
    }
  </style>
</head>
<body>
  <div id="topmenu">
    <div id="prediction">
      <a href="{{ url_for('prediction') }}" style="color: white;text-decoration: none;">prediction</a>
    </div>
    <div id="register">
      <a href="{{ url_for('register') }}" style="color: white;text-decoration: none;">Register</a>
    </div>
    <div id="login">
      <a href="{{ url_for('login') }}" style="color: white;text-decoration: none;">Login</a>
    </div>
    <div id="home">
      <a href="{{ url_for('dashboard') }}" style="color: white;text-decoration: none;">Home</a>
    </div>
    <div id="hedder">
      Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance
      Companies
    </div>
  </div>
  <div id="about">
    ABOUT PROJECT
    <hr style="width: 13%" color="yellow" />
  </div>
  <div id="content">
    <p>
      Vehicle damage detection is used to reduce claims leakage during
      insurance processing. Visual inception and validation are usually done.
      As it takes a long time, because a person needs to come and inspect the
      damage. Here we are trying to automate the procedure. Using this
      automation, we can avoid time conception for the insurance claim
      problem.
    </p>
  </div>
  <div id="footer">
    <div id="textcontent">Copyright Ⓒ 2021. All Rights Reserved</div>
    <div id="logo">
      <img
        src="/static/images/twitter.jpg"
        height="28px"
        width="28px"
        style="border-radius: 18%; margin-right: 40px"
      />
```

```
      <img
        src="/static/images/linkedin.jpg"
        height="28px"
        width="28px"
        style="margin-left: 30px; border-radius: 18%"
      />
    </div>
  </div>
 </body>
</html>
```

Login :

```
<html>
 <head>
  <title>Login</title>
  <script src="https://cdn.lordicon.com/qjzruarw.js"></script>
  <style type="text/css">
   #topmenu {
    width: 100%;
    background-color: 312D2D;
    height: 50px;
   }
   #hedder {
    color: white;
    font-size: large;
    padding-top: 13px;
    padding-left: 40px;
   }
   #home {
    float: right;
    padding-top: 13px;
    padding-right: 50px;
    color: rgb(222, 216, 216);
    font-size: medium;
   }
   #login {
    float: right;
    padding-top: 13px;
    padding-right: 50px;
    color: rgb(222, 216, 216);
    font-size: medium;
   }
   #register {
    float: right;
    padding-top: 13px;
    padding-right: 50px;
    color: rgb(222, 216, 216);
```

```
    font-size: medium;
   }
   #box {
    height: 300px;
    width: 500px;
    background-color: antiquewhite;
    margin: 10px;
    border-color: black;
    border-width: 25px;
   }
   div.background {
    border: 2px solid gray;
    height: 300px;
    width: 500px;
    margin: auto;
    margin-top: 7%;
   }
   #loginlogo {
    text-align: center;
    margin-top: 20px;
   }
   #textcontent {
    margin-top: 10px;
    margin-left: 25px;
    margin-top: 20px;
   }
   div.choice {
    border: 2px solid gray;
    height: 35px;
    width: 500px;
    background-color: rgb(230, 227, 227);
    margin: auto;
    margin-top: 0%;
   }

   #question {
    margin-top: 7px;
   }
   #choice-login {
    color: rgb(67, 64, 247);
    text-decoration: underline;
    margin-left: 150px;
    margin-top: -25px;
   }
  </style>
</head>
<body onload="flashMessage()">
  <div id="topmenu">
   <div id="register">
    <a href="{{ url_for('register') }}" style="color: white;text-decoration: none;">Register</a>
```

```
    </div>
    <div id="login">
      <a href="{{ url_for('logout') }}" style="color: white;text-decoration: none;">Logout</a>
    </div>
    <div id="home">
      <a href="{{ url_for('dashboard') }}" style="color: white;text-decoration: none;">Home</a>
    </div>
    <div id="hedder">Login Page</div>
   <!--</div>
   {% with messages = get_flashed_messages() %}
  {% if messages %}
   <ul class=flashes>
   {% for message in messages %}
    <p><strong>Error:</strong> {{ message }}
   {% endfor %}
   </ul>
  {% endif %}
{% endwith %}-->

   <div class="background">
    <div id="loginlogo">
      <lord-icon
      src="https://cdn.lordicon.com/imamsnbq.json"
      trigger="hover"
      style="width:100px;height:100px">
      </lord-icon>
      <!-- <img
        src= "/static/images/login icon.png"
        alt="login logo"
        style="width: 100px; height: 100px; border-radius: 50%"
      /> -->
    </div>
    <div id="textcontent">

      <form action="login" method="POST">
        <script>
          function flashMessage(){
            if("{{flash_message}}" == "True"){
              alert("invalid credentials")
            }
          }
        </script>
        <input
          type="text"
          name="email"
          id="email"
          placeholder="Enter registered email ID"
          style="width: 440px; height: 35px; margin-bottom: 15px"
        />
        <input
```

```
          type="password"
          name="password"
          id="password"
          placeholder="Enter Password"
          style="width: 440px; height: 35px; margin-bottom: 15px"
        />

        <input
          type="submit"
          name="submit"
          value="Login"
          style="
            width: 440px;
            height: 35px;
            text-align: center;
            background-color: black;
            color: white;
          "
        />
      </form>

    </div>
   </div>
   <div class="choice">
    <div id="question">forgot password?</div>
    <div id="choice-login">
     <a href="{{ url_for('forgotpassword') }}" style="color: #7ed8ff;">Reset</a>
    </div>
   </div>
  </body>
</html>
```

Register :

```
<html>
  <head>
   <title>Register</title>
   <style type="text/css">
    #topmenu {
     width: 100%;
     background-color: 312D2D;
     height: 50px;
    }
    #hedder {
     color: white;
     font-size: large;
     padding-top: 13px;
     padding-left: 40px;
```

```
}
#home {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#box {
  height: 300px;
  width: 500px;
  background-color: antiquewhite;
  margin: 10px;
  border-color: black;
  border-width: 25px;
}
div.background {
  border: 2px solid gray;
  height: 350px;
  width: 500px;
  margin: auto;
  margin-top: 7%;
}
#registerlogo {
  text-align: center;
  margin-top: 20px;
}
#textcontent {
  margin-top: 28px;
  margin-left: 25px;
}
div.choice {
  border: 2px solid gray;
  height: 35px;
  width: 500px;
  background-color: rgb(230, 227, 227);
```

```
    margin: auto;
    margin-top: 0%;
   }

   #question {
    margin-top: 7px;
   }
   #choice-login {
    color: rgb(67, 64, 247);
    text-decoration: underline;
    margin-left: 200px;
    margin-top: -20px;
   }
  </style>
</head>
<body onload="flashMessage()">
  <div id="topmenu">
   <div id="login">
    <a href="{{ url_for('login') }}" style="color: white;text-decoration: none;">Login</a>
   </div>
   <div id="home">
    <a href="{{ url_for('dashboard') }}" style="color: white;text-decoration: none;">Home</a>
   </div>
   <div id="hedder">Vehicle Damage Detection</div>
  </div>
  <div class="background">
   <div id="registerlogo">
    <img
     src="/static/images/login icon.png"
     alt="login logo"
     style="width: 100px; height: 100px; border-radius: 50%"
    />
   </div>
   <div id="textcontent">
    <form action="register" method="POST">
     <script>
      function flashMessage(){
       if("{{flash_message}}" == "True"){
        alert("account with this email id already exist")
       }

      }
     </script>
     <input
      type="text"
      name="name"
      id="name"
      placeholder="Enter Name"
      style="width: 440px; height: 35px; margin-bottom: 15px"
     />
```

```
        <input
          type="text"
          name="email"
          id="email"
          placeholder="Enter Email ID"
          style="width: 440px; height: 35px; margin-bottom: 15px"
        />
        <input
          type="password"
          name="password"
          id="password"
          placeholder="Enter Password"
          style="width: 440px; height: 35px; margin-bottom: 15px"
        />
        <input
          type="submit"
          value="Register"
          name="submit"
          style="
            width: 440px;
            height: 35px;
            text-align: center;
            background-color: black;
            color: white;
          "
        />
      </form>
    </div>
  </div>
  <div class="choice">
    <div id="question">Already have an account?</div>
    <div id="choice-login">
      <a href="{{ url_for('login') }}" style="color: #7ed8ff;">Login</a>
    </div>
  </div>
 </body>
</html>
```

reset password :

```
<html>
 <head>
  <title>Login</title>
  <script src="https://cdn.lordicon.com/qjzruarw.js"></script>
  <style type="text/css">
   #topmenu {
    width: 100%;
    background-color: 312D2D;
```

```
  height: 50px;
}
#hedder {
  color: white;
  font-size: large;
  padding-top: 13px;
  padding-left: 40px;
}
#home {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#box {
  height: 300px;
  width: 500px;
  background-color: antiquewhite;
  margin: 10px;
  border-color: black;
  border-width: 25px;
}
div.background {
  border: 2px solid gray;
  height: 300px;
  width: 500px;
  margin: auto;
  margin-top: 7%;
}
#loginlogo {
  text-align: center;
  margin-top: 20px;
}
#textcontent {
  margin-top: 10px;
```

```
      margin-left: 25px;
      margin-top: 20px;
    }
  </style>
 </head>
 <body onload="flashMessage()">
  <div id="topmenu">
   <div id="register">
    <a href="{{ url_for('register') }}" style="color: white;text-decoration: none;">Register</a>
   </div>
   <div id="login">
    <a href="{{ url_for('logout') }}" style="color: white;text-decoration: none;">Logout</a>
   </div>
   <div id="home">
    <a href="{{ url_for('dashboard') }}" style="color: white;text-decoration: none;">Home</a>
   </div>
   <div id="hedder">Login Page</div>
  <!--</div>
  {% with messages = get_flashed_messages() %}
 {% if messages %}
  <ul class=flashes>
  {% for message in messages %}
   <p><strong>Error:</strong> {{ message }}
  {% endfor %}
  </ul>
 {% endif %}
{% endwith %}-->

  <div class="background">
   <div id="loginlogo">
    <lord-icon
    src="https://cdn.lordicon.com/imamsnbq.json"
    trigger="hover"
    style="width:100px;height:100px">
    </lord-icon>
    <!-- <img
     src= "/static/images/login icon.png"
     alt="login logo"
     style="width: 100px; height: 100px; border-radius: 50%"
    /> -->
   </div>
   <div id="textcontent">

    <form action="resetpassword" method="POST">
     <script>
      function flashMessage(){
       if("{{flash_message}}" == "True"){
        alert("invalid credentials")
       }
      }
```

```
        </script>
        <input
          type="password"
          name="password"
          id="password"
          placeholder="Enter new password"
          style="width: 440px; height: 35px; margin-bottom: 15px"
        />
        <input
          type="submit"
          name="submit"
          value="submit"
          style="
            width: 440px;
            height: 35px;
            text-align: center;
            background-color: black;
            color: white;
          "
        />
      </form>

    </div>
   </div>
  </body>
</html>
```

Prediction :

```
  <html>
 <head>
   <title>index</title>
   <style type="text/css">
     #topmenu {
       width: 100%;
       background-color: 312D2D;
       height: 50px;
     }
     #hedder {
       color: white;
       padding-top: 13px;
       padding-left: 60px;
     }

     #home {
       float: right;
       padding-top: 13px;
       padding-right: 50px;
```

```
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#prediction {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#about {
  text-align: center;
  padding-top: 10%;
  color: gray;
  font-size: 20px;
}
#content {
  padding-top: 50px;
  padding-left: 40px;
  padding-right: 40px;
  font-size: large;
}
#footer {
  width: 99%;
  background-color: 312D2D;
  height: 50px;
  position: absolute;
  bottom: 1%;
}
#textcontent {
  color: white;
  font-size: 15px;
  padding-left: 18%;
  padding-top: 1%;
}
#logo {
```

```
    margin-top: -1.5%;
    margin-right: 28%;
    float: right;
   }
  </style>
</head>
<body onload="flashMessage()">
  <div id="topmenu">
   <div id="login">
    <a href="{{ url_for('logout') }}" style="color: white;text-decoration: none;">Logout</a>
   </div>
   <div id="home">
    <a href="{{ url_for('dashboard') }}" style="color: white;text-decoration: none;">Home</a>
   </div>
   <div id="hedder">
    Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance
    Companies
   </div>
  </div>
  <form action="prediction" method="POST"  enctype="multipart/form-data">
   <input type="file" id="myFile" name="myFile">
   <input type="submit">
   <script>
    function flashMessage(){
     if("{{flash_message}}" == "True"){
     //   alert("invalid credentials")
     //   const im = document.createElement('img');
     // im.src = "{{url_for('static', filename='imagedata/save.png')}}";
     // im.height = "200px";
     // im.width = '200px';
     // im.alt = 'hello world'
     // document.getElementById('about').appendChild(im);
      document.getElementById('image').src = 'static/imagedata/save.png';
      const e = document.getElementById("qwerty");
      const para = document.createElement("p");
      const node = document.createTextNode("The estimated cost for the damage is : | {{value}} |");
      para.appendChild(node);
      e.appendChild(para);
     }
    }
   </script>
  </form>
  <!-- <script>
   function flashMessage(){
    if("{{ flash_message }}"=='True'){
     const im = document.createElement('img');
     im.src = "{{url_for('static', filename='imagedata/save.png')}}";
     im.height = "200px";
     im.width = '200px';
     im.alt = 'hello world'
```

```
      }
     }
    </script> -->
  <!-- <img src="{{url_for('static', filename='imagedata/save.png')}}" alt=""
    height="200px"
    width="200px"
    /> -->
    <div id="about">
     <div id="qwerty">
      <p></p>
     </div>
     <hr style="width: 30%" color="yellow" />
     <img src="static/images/damage 1.png" height="250px" width="400px" alt="" id="image">
    </div>

    <div id="footer">
     <div id="textcontent">Copyright Ⓒ 2021. All Rights Reserved</div>
     <div id="logo">
      <img
       src="/static/images/twitter.jpg"
       height="28px"
       width="28px"
       style="border-radius: 18%; margin-right: 40px"
      />

      <img
       src="/static/images/linkedin.jpg"
       height="28px"
       width="28px"
       style="margin-left: 30px; border-radius: 18%"
      />
     </div>
    </div>
  </body>
</html>
```

Logout :

```
<html>
  <head>
    <title>Logout</title>
    <style type="text/css">
      #topmenu {
        width: 100%;
        background-color: 312D2D;
        height: 50px;
      }
      #hedder {
```

```
  color: white;
  font-size: large;
  padding-top: 13px;
  padding-left: 40px;
}
#home {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#login {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#register {
  float: right;
  padding-top: 13px;
  padding-right: 50px;
  color: rgb(222, 216, 216);
  font-size: medium;
}
#loggedout {
  color: black;
  font-size: large;
  text-align: center;
  justify-content: center;
  position: absolute;
  top: 50%;
  left: 40%;
  transform: translateY(-500%);
}
#info {
  color: green;
  font-size: small;
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
  position: absolute;
  top: 50%;
  left: 40%;
  transform: translateY(-500%);
}
#login-button {
  margin: 0%;
```

```
      display: flex;
      align-items: center;
      justify-content: center;
      text-align: center;
      position: absolute;
      top: 50%;
      left: 40%;
      transform: translateY(-500%0);
    }
  </style>
 </head>
 <body>
  <div id="topmenu">
  <div id="register">
   <a href="{{ url_for('register') }}" style="color: white;text-decoration: none;">Register</a>
  </div>
  <div id="login">
   <a href="{{ url_for('login') }}" style="color: white;text-decoration: none;">Login</a>
  </div>
  <div id="home">
   <a href="{{ url_for('dashboard') }}" style="color: white;text-decoration: none;">Home</a>
  </div>

   <div id="hedder">Vehicle Damage Detection</div>
  </div>
  <div id="loggedout" style="vertical-align: middle">
   Successfully Logged Out!
  </div>
  <div id="info">Login for more information</div>
  <div id="login-button">
   <form action="login">
    <input
     type="submit"
     value="Login"
     style="
      background-color: black;
      color: white;
      width: 200px;
      height: 35px;
     "
    />
   </form>
  </div>
 </body>
</html>
```

Model Building ;

1. Importing The Model Building Libraries

In [ ]:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

2. Loading The Model

In [ ]:

```
IMAGE_SIZE = [224, 224]

train_path = '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost
Estimator For Insurance Companies/Dataset/body/training'
valid_path = '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost
Estimator For Insurance Companies/Dataset/body/validation'
```

In [ ]:

```
vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

3. Adding Flatten Layer

In [ ]:

```
for layer in vgg16.layers:
    layer.trainable = False
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
 in
----> 1 for layer in vgg16.layers:
      2     layer.trainable = False

NameError: name 'vgg16' is not defined
```

In [ ]:

```
folders = glob('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost
Estimator For Insurance Companies/Dataset/body/training/*')
```

In [ ]:

```
folders
```

Out[ ]:

```
['/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For
Insurance Companies/Dataset/body/training/02-side',
 '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For
Insurance Companies/Dataset/body/training/00-front',
 '/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost Estimator For
```

```
Insurance Companies/Dataset/body/training/01-rear']
```

In [ ]:

```
x = Flatten()(vgg16.output)
```

In [ ]:

```
len(folders)
```

Out[ ]:

4. Adding Output Layer

In [ ]:

```
prediction = Dense(len(folders), activation='softmax')(x)
```

5. Creating A Model Object

In [ ]:

```
model = Model(inputs=vgg16.input, outputs=prediction)
```

In [ ]:

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
```

```
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 3)                 75267

=================================================================
Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
_________________________________________________________________
```

6. Configure The Learning Process

In [ ]:

```
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

7. Train The Model

In [ ]:

```
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=25,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version. Please
use `Model.fit`, which supports generators.

Epoch 1/25
98/98 [==============================] - 606s 6s/step - loss: 1.2827 - accuracy:
0.5649 - val_loss: 0.8292 - val_accuracy: 0.7076
Epoch 2/25
98/98 [==============================] - 601s 6s/step - loss: 0.6301 - accuracy:
0.7467 - val_loss: 1.2482 - val_accuracy: 0.5965
Epoch 3/25
98/98 [==============================] - 601s 6s/step - loss: 0.5073 - accuracy:
0.8039 - val_loss: 0.8174 - val_accuracy: 0.7193
Epoch 4/25
98/98 [==============================] - 601s 6s/step - loss: 0.3564 - accuracy:
0.8621 - val_loss: 0.9245 - val_accuracy: 0.6608
Epoch 5/25
```

```
98/98 [==============================] - 599s 6s/step - loss: 0.2951 - accuracy:
0.8917 - val_loss: 1.9934 - val_accuracy: 0.5906
Epoch 6/25
98/98 [==============================] - 638s 7s/step - loss: 0.2557 - accuracy:
0.9152 - val_loss: 0.9176 - val_accuracy: 0.6842
Epoch 7/25
98/98 [==============================] - 607s 6s/step - loss: 0.2083 - accuracy:
0.9367 - val_loss: 0.9594 - val_accuracy: 0.7018
Epoch 8/25
98/98 [==============================] - 600s 6s/step - loss: 0.2184 - accuracy:
0.9122 - val_loss: 1.0329 - val_accuracy: 0.6784
Epoch 9/25
98/98 [==============================] - 602s 6s/step - loss: 0.1320 - accuracy:
0.9581 - val_loss: 1.0539 - val_accuracy: 0.7135
Epoch 10/25
98/98 [==============================] - 599s 6s/step - loss: 0.1131 - accuracy:
0.9622 - val_loss: 1.2113 - val_accuracy: 0.6842
Epoch 11/25
98/98 [==============================] - 597s 6s/step - loss: 0.1001 - accuracy:
0.9745 - val_loss: 0.9917 - val_accuracy: 0.7018
Epoch 12/25
98/98 [==============================] - 598s 6s/step - loss: 0.0954 - accuracy:
0.9745 - val_loss: 1.0601 - val_accuracy: 0.7018
Epoch 13/25
98/98 [==============================] - 594s 6s/step - loss: 0.0695 - accuracy:
0.9816 - val_loss: 1.3700 - val_accuracy: 0.6433
Epoch 14/25
98/98 [==============================] - 599s 6s/step - loss: 0.1414 - accuracy:
0.9653 - val_loss: 1.1607 - val_accuracy: 0.6667
Epoch 15/25
98/98 [==============================] - 600s 6s/step - loss: 0.0905 - accuracy:
0.9796 - val_loss: 1.4014 - val_accuracy: 0.6667
Epoch 16/25
98/98 [==============================] - 601s 6s/step - loss: 0.0797 - accuracy:
0.9775 - val_loss: 1.6741 - val_accuracy: 0.6491
Epoch 17/25
98/98 [==============================] - 602s 6s/step - loss: 0.1042 - accuracy:
0.9745 - val_loss: 1.2824 - val_accuracy: 0.6959
Epoch 18/25
98/98 [==============================] - 600s 6s/step - loss: 0.0831 - accuracy:
0.9785 - val_loss: 1.1667 - val_accuracy: 0.6901
Epoch 19/25
98/98 [==============================] - 603s 6s/step - loss: 0.0826 - accuracy:
0.9704 - val_loss: 1.3747 - val_accuracy: 0.6374
Epoch 20/25
98/98 [==============================] - 600s 6s/step - loss: 0.0536 - accuracy:
0.9837 - val_loss: 1.2074 - val_accuracy: 0.6550
Epoch 21/25
98/98 [==============================] - 597s 6s/step - loss: 0.0716 - accuracy:
0.9796 - val_loss: 1.5491 - val_accuracy: 0.6725
Epoch 22/25
98/98 [==============================] - 599s 6s/step - loss: 0.0457 - accuracy:
```

```
0.9918 - val_loss: 1.2930 - val_accuracy: 0.7135
Epoch 23/25
98/98 [==============================] - 601s 6s/step - loss: 0.0526 - accuracy:
0.9928 - val_loss: 1.2576 - val_accuracy: 0.6959
Epoch 24/25
98/98 [==============================] - 601s 6s/step - loss: 0.0421 - accuracy:
0.9908 - val_loss: 1.3347 - val_accuracy: 0.7193
Epoch 25/25
98/98 [==============================] - 597s 6s/step - loss: 0.0597 - accuracy:
0.9826 - val_loss: 1.4728 - val_accuracy: 0.6725
```

8. Save The Model

In [ ]:

```
from tensorflow.keras.models import load_model

model.save('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost
Estimator For Insurance Companies/Model/body.h5')
```

9. Test The Model

In [ ]:

```
from tensorflow.keras.models import load_model
import cv2
from skimage.transform import resize
```

In [ ]:

```
model = load_model('/content/drive/MyDrive/Intelligent Vehicle Damage Assessment &
Cost Estimator For Insurance Companies/Model/body.h5')
```

In [ ]:

```
def detect(frame):
  img = cv2.resize(frame,(224,224))
  img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

  if(np.max(img)>1):
    img = img/255.0
  img = np.array([img])
  prediction = model.predict(img)
  label = ["front","rear","side"]
  preds = label[np.argmax(prediction)]
  return preds
```

In [ ]:

```
import numpy as np
```

In [ ]:

```
data = "/content/drive/MyDrive/Intelligent Vehicle Damage Assessment & Cost
Estimator For Insurance Companies/Dataset/body/training/00-front/0005.JPEG"
image = cv2.imread(data)
print(detect(image))
1/1 [==============================] - 1s 638ms/step
front
```

GitHub & Project Demo Link:

GitHub repository: https://github.com/IBM-EPBL/IBM-Project-40427-1660629312

Demo Link: