PROJECT BASED EXPERIENTIAL LEARNING PROGRAM (NALAIYATHIRAN)

Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

PROJECT REPORT

Submitted by

Shafiya Sidrath A (SSNCE193002096)

Nethra Prakash K (SSNCE193002067)

Sam Devavaram Jebaraj (SSNCE193002090)

Vikhas V (S193002119)

Team ID : PNT2022TMID52972

Department of Electronics and Communication Engineering

Sri Sivasubramaniya Nadar College of Engineering

CONTENTS

1.INTRODUCTION

1.1 Project Overview

Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies ,This project focuses on estimating the intensity and type of damage and based on that provides a estimated insurance amount to the end user.

The Prediction of the intensity of damage and type is done using VGG-16,Machine Learning Model.
VGG-16.In the publication "Very Deep Convolutional Networks for Large-Scale Image Recognition," K. Simonyan and A. Zisserman from the University of Oxford introduced the convolutional neural network model known as VGG16. The model had a accuracy of top-5 test accuracy in ImageNet, a dataset of more than 14 million images divided into 1000 classes, is 92.7%.

The Dataset we use has 2 folders organised as body and level with a total of 1150 images and they are divided into 979 training images and 181 testing images.With the level folder having minor ,moderate and severe images whereas the body has front,rear and side. Our Model has an accuracy of 99%. A web application in which the prediction is made is also part of the solution.

1.2 Purpose

To find the intensity of the vehicle damage and predict the estimate of the insurance amount it reduces the delay in obtaining claims from the insurance companies.It increases the ease of insurance claiming process.Sometimes car damage is excluded in the insurance policy all this cause stress among the customers.Automating the insurance claiming process will also reduce the man power that is needed to go examine the vehicle damage and verify it before giving the insurance .

2.LITERATURE SURVEY

2.1 Existing Problem

In current solutions 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 hence they aren't very accurate and reliable. However, they impose delays in the processing of claims.

2.2 References

1. Research on Intelligent Vehicle Damage Assessment System Based on Computer Vision
Zhu Qianqian, Guo Weiming, Shen Ying and Zhao Zihao
2020

2. Car Damage Detection and Classification
Phyu Mar Kyu, Kuntpong Woraratpanya
2020

3. Vehicle Damage Classification and Fraudulent Image Detection
Umer Waqas, Nimra Akram, Soohwa Kim, Donghun Lee, Jihoon Jeon

2020

2.3 Problem statement definition

1. This paper gives us an insight of four stages of our project:

   -> Accident investigation

   -> Intelligent image damage assessment

   -> Damage result output

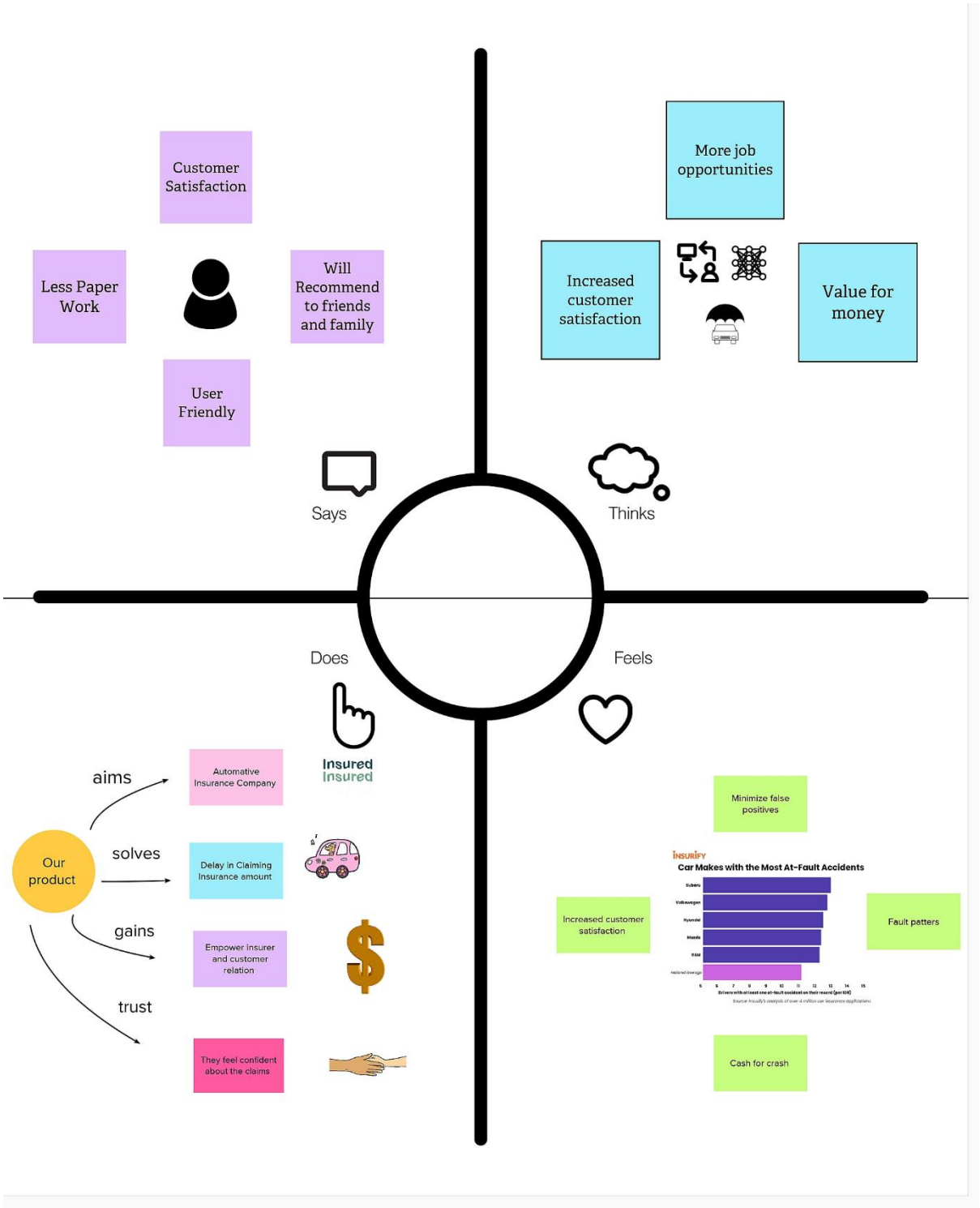
   -> Vehicle insurance anti-fraud

2. This paper discovers the effect of pre-trained CNN models, which are trained on an ImageNet dataset, and followed by fine-tuning, because some of the categories can be fine-granular to get our specific tasks.

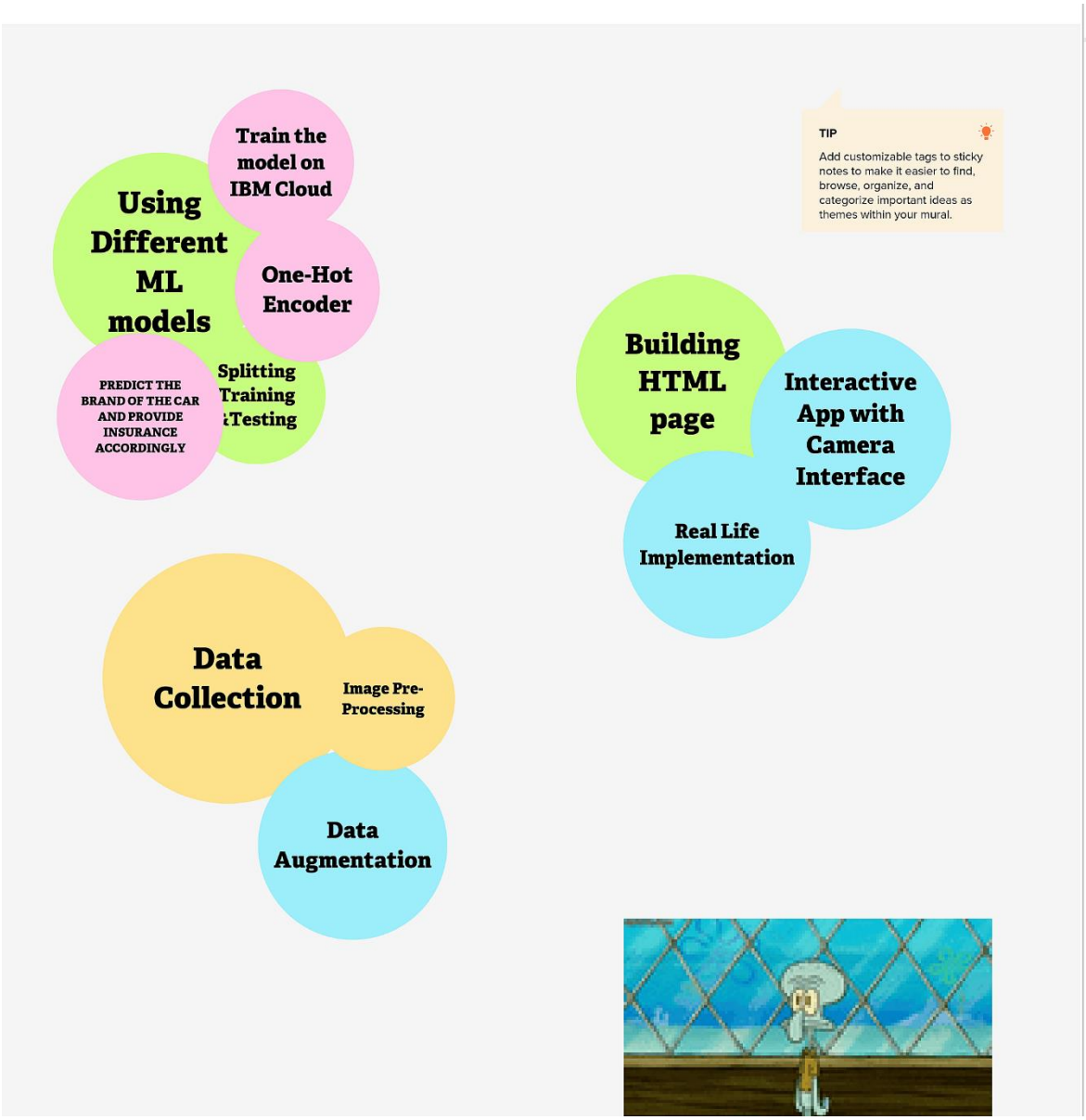
3. The paper shifts towards the same automation with diverse hurdles such as users can upload fake images like screenshots or taking pictures from computer screens, etc. To tackle the problem, a hybrid approach was proposed to provide only authentic images to algorithm for damage classification as input.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy map Canvas



3.2 Ideation & Brainstorming



3.3 Proposed Solution

The project that we propose can detect the type of damage in a car. The need for such a model is that it can be used by insurance companies for faster processing of claims. The damage extent is categorised into Low, Moderate and High and the location of damage is categorised to rear, front and side. When users uploads the pictures of damage , the model can assess damage and thereby estimate the cost of damage. This solution has several advantages such as, high accuracy in cost estimation and increased automation.
The users of the website will be the insurance companies , and their customers are our target audience. Due to this, the insurance company's workforce and the time taken to claim the insurance is reduced.

3.4 Problem Solution Fit

1. CUSTOMER SEGMENT(S)
   - Common people who own vehicles.
   - Insurance Companies.

2. JOBS TO BE DONE / PROBLEMS
- To find the intensity of the vehicle damage
- Predict the insurance based on damage

3. TRIGGERS
- The ease of the entire insurance claiming process

4. EMOTIONS: BEFORE / AFTER
- Before- Confused ,Took a long time to claim insurance.
- After-Faster,Hazel Free process.

5. AVAILABLE SOLUTIONS
- Vehicle damage is assessed by a person hence there might be human error
- Traditional Insurance claim is a complicated process

6. CUSTOMER CONSTRAINTS
- Delay in Claim Retention
- Car Damage Excluded in Policy

7. BEHAVIOUR
- Exploring the different options available for claiming the insurance

8. CHANNELS of BEHAVIOUR
- Uploading the picture of the damaged vehicle
- Getting to know the insurance amount

9. PROBLEM ROOT CAUSE
- Few customers don’t raise legitimate claims
- Car is repaired before the insurance company makes inspection

10. YOUR SOLUTION
- Keeping the customer in mind we would like to ensure that the website has a simple frontend and the ML model should be accurate so that the customer doesn’t lose on anything
- The insurance company can act fast with the help of the ML Model

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

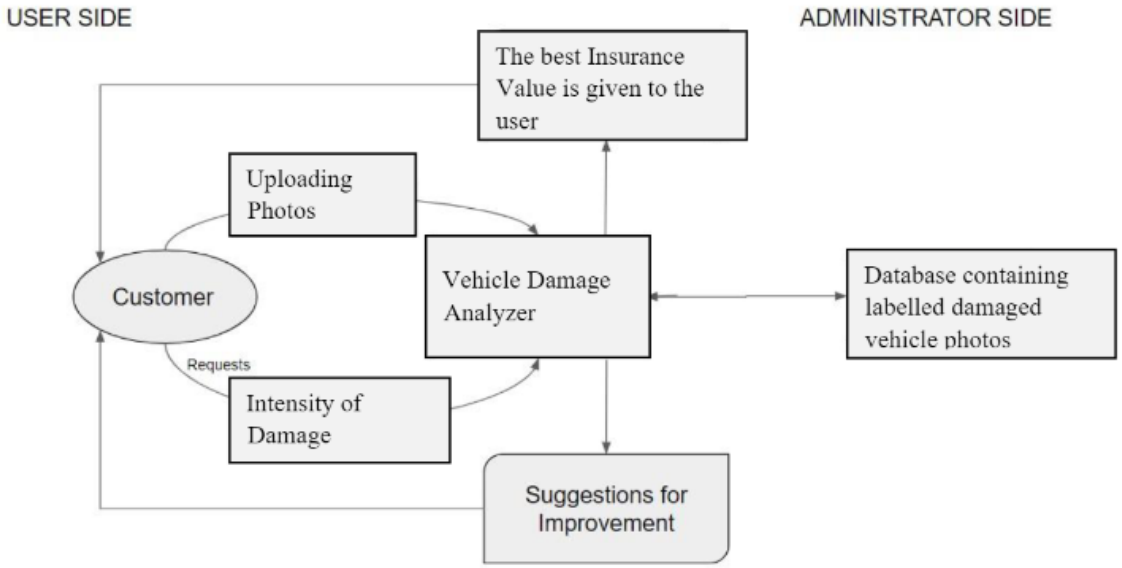
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Reset Password	Reset password through Gmail Reset password through Mobile number
FR-4	Uploading Images	The Images uploaded by the user should analysed accurately my Machine Learning Model. The user Interface of the webpage should be simple and easily useable by everyone
FR-4	Feedback	The user can submit the feedback through a contact form on the website or through Gmail.

4.2 Non Functional Requirements

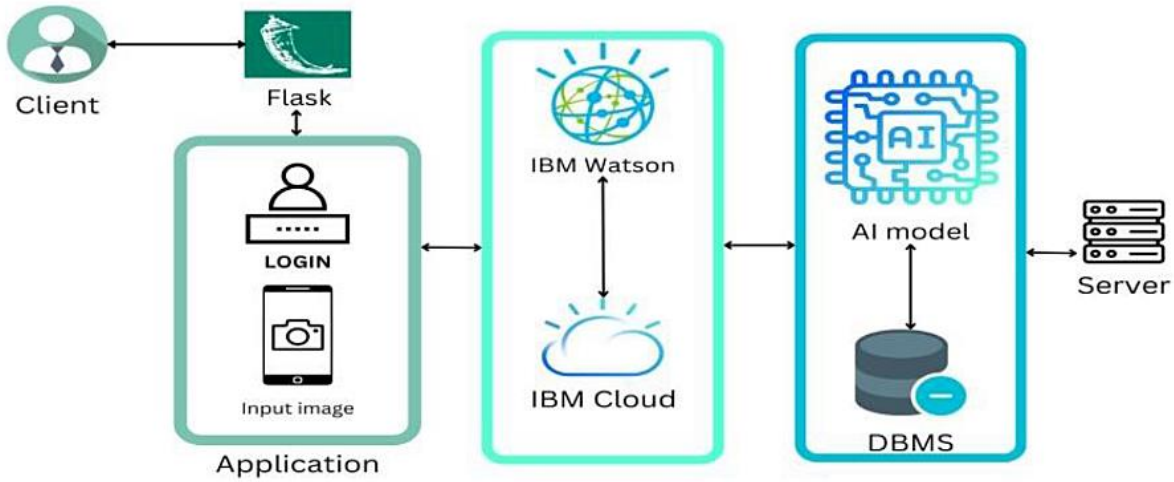
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Used to compare different car models under different metrics.
NFR-2	Security	The images uploaded by the user and the insurance amount received is secured and confidential
NFR-3	Reliability	The accuracy in predicted the intensity of the damage and the insurance amount should be agreed.
NFR-4	Performance	The real time images uploaded by the user will be analysed to give accurate results
NFR-5	Availability	This model should be made available to every common man owing a vehicle and the webpage should be compatible with all devices.
NFR-6	Scalability	High scalability. The model will help the customers to get the insurance amount quickly without any hassle

5. PROJECT DESIGN

5.1 Data flow Diagram



5.2 Solution & Technical Architecture



Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user interacts with the web UI application.	HTML, CSS, python
2.	Application-Level 1	Getting user input image	Python
3.	Application-Level 2	Getting model output for damage prediction	IBM Watson STT service
4.	Application-Level 3	Getting model output for cost estimation	IBM Watson Assistant
5.	Database	Data Type – Images and user inputs details are stored	MySQL
6.	Cloud Database	Database Service on Cloud	IBM Cloud
7.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
8.	Machine Learning Model	Purpose of the AI Model is for estimating the cost of the damaged vehicle.	Object Recognition Model
9.	Infrastructure (Server / Cloud)	On cloud server we will be deploying the AI Model using flask in the web page	Python flask.

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Registration	1	As a user, I can register for the insurance query by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint - 1
		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 - 2
		3	As a user, I can register for the application through Gmail	I can register via Gmail	Medium	Sprint - 1
	Login	4	As a user, I can log into the application by entering email & password	I am able to login	High	Sprint - 2
	Dashboard	5	As a user, I can access all the facilities by the website	I am able to access all the facilities	High	Sprint - 1
	Webpage	6	The Webpage should be user friendly and the customers need to upload the pictures without any glitch	The users are able to upload the pictures and the Machine Learning model can be applied.	High	Sprint-2
	Report	7	The customers should get a detailed report about the damage and estimated cost	Able to generate Report	Moderate	Sprint-1
Administrator	Database	8	As an admin, I can manage the database		High	Sprint - 1

6. PROJECT PLANNING & SCHEDULING

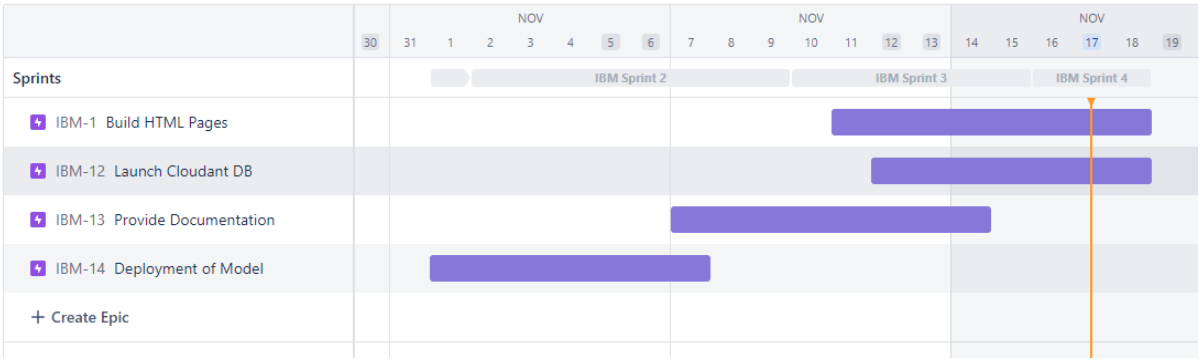
6.1 Sprint Planning & Estimation

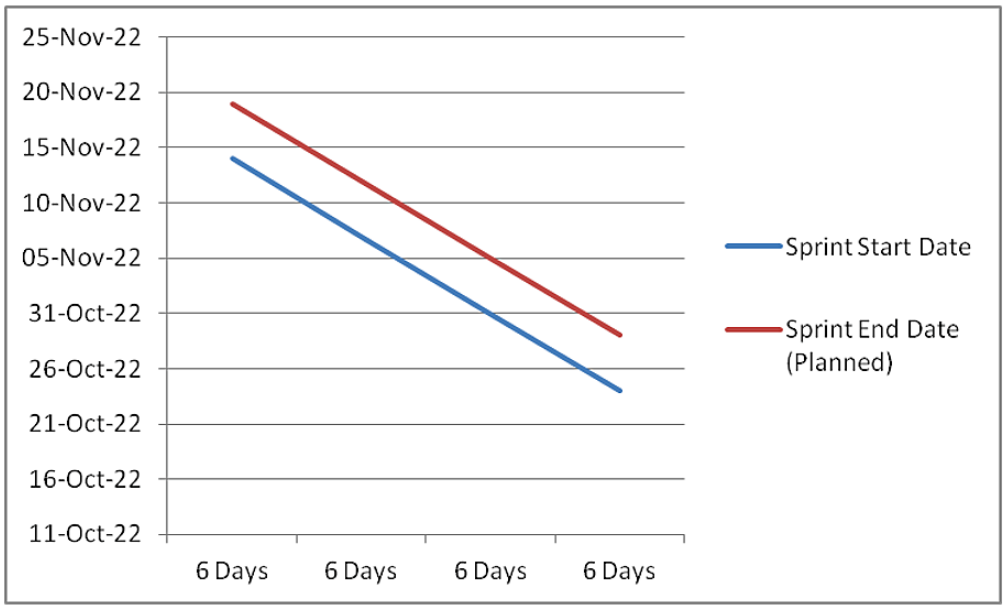
Sprint	User Story / Task	Story Points	Priority	Team Members
Sprint-1	As a user, I can login, register and sign out from the website created.	5	High	Shafiya Sidrath
Sprint-2	As a user, I will be using the cloud to access the model which has been deployed.	5	High	Nethra Prakash
Sprint-3	As a user, I can use the AI model built to estimate the extent of damage in the vehicle.	5	High	Sam Devavaram Jebaraj
Sprint-4	As a user, I can understand and learn about the project using the documentation prepared.	5	Medium	Vikhas V

6.2 Sprint Delivery Schedule

Sprint	Duration	Total Story Points	Sprint Start Date	Sprint End Date (Planned)	Sprint Release Date (Actual)
Sprint-1	6 Days	6	24 Oct 2022	29 Oct 2022	29 Oct 2022
Sprint-2	6 Days	6	31 Oct 2022	05 Nov 2022	5 Nov 2022
Sprint-3	6 Days	6	07 Nov 2022	12 Nov 2022	13 Nov 2022
Sprint-4	6 Days	6	14 Nov 2022	19 Nov 2022	18 Oct 2022

6.3 Reports from JIRA





7.CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

Image Pre Processing

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./
255,shear_range=0.1,zoom_range=0.1,horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale = 1.1255

from google.colab import drive
drive.mount('/content/drive')
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
force_remount=True).

#Body Images Path
trainpathB = "/content/drive/MyDrive/IBM Dataset/body/training"
testpathB = "/content/drive/MyDrive/IBM Dataset/body/validation"

#Level Images Path
trainpathL = "/content/drive/MyDrive/IBM Dataset/level/training";
testpathL = "/content/drive/MyDrive/IBM Dataset/level/validation"

#Using the same Imagegenerator for both body and severity level
trainB = train_datagen.flow_from_directory(trainpathB,
                                           target_size = (224, 224),
                                           batch_size = 10,
                                           class_mode = 'categorical')

testB = test_datagen.flow_from_directory(trainpathB,
                                         target_size = (224, 224),
                                         batch_size = 10,
                                         class_mode ='categorical' )
Found 979 images belonging to 3 classes.
Found 979 images belonging to 3 classes.

trainL = train_datagen.flow_from_directory(trainpathL,
                                           target_size = (224, 224),
                                           batch_size = 10,
                                           class_mode = 'categorical')

testL = test_datagen.flow_from_directory(trainpathL,
                                         target_size = (224, 224),
                                         batch_size = 10,
                                         class_mode ='categorical' )
Found 979 images belonging to 3 classes.
Found 979 images belonging to 3 classes.
```

7.2 Feature 2

```
import numpy as np
import matplotlib.pyplot as plt
from glob import glob
import tensorflow as tf
```

```
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input

#Adding preprocessed layers to the Transfer Model
vgg=VGG16(weights='imagenet',include_top=False,input_tensor=Input(shape=(224,224,3)))
vgg1=VGG16(weights='imagenet',include_top=False,input_tensor=Input(shape=(224,224,3)))
for layer in vgg.layers:
   layer.trainable=False
x=Flatten()(vgg.outputfor layer in vgg1.layers:
   layer.trainable=False
y=Flatten()(vgg1.output)

pred = Dense(3,activation='softmax')(x)
pred1 = Dense(3,activation='softmax')(y)

model =  Model(inputs=vgg.input,outputs=pred)
model1 = Model(inputs=vgg1.input,outputs=pred1)

model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['acc'])
model1.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['acc'])
```

Model for Body regions of the Car

```
r=model.fit_generator(trainB,
                      validation_data=testB,
                      epochs=25,
                      steps_per_epoch=979//10,
                      validation_steps=171//10)
model.save('BodyModel.h5')
```

Model for Damage severity level

```
r1= model1.fit_generator(trainL,
                       validation_data=testL,
                       epochs=25,
                       steps_per_epoch=979//10,
                       validation_steps=171//10)

model1.save('/content/drive/MyDrive/models/level.h5')
```

:

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

body_model=load_model('/content/BodyModel.h5')
```

```
from tensorflow.keras.models import load_model
import cv2
from skimage.transform import resize
                                                                                                    :
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 =body_model.predict(img)
    #print(prediction)
    label=["front","rear","side"]
    preds=label[np.argmax(prediction)]
    return preds

import numpy as np
data="/content/drive/MyDrive/IBM Dataset/body/validation/01-rear/0002.JPEG"
image=cv2.imread(data)
print(detect(image))
1/1 [==============================] - 1s 810ms/step
rear

level_model = load_model('/content/drive/MyDrive/models/level.h5')


def detect1(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 = level_model.predict(img)
    print(prediction)
    label = ["minor", "moderate", "severe"]
    preds = label[np.argmax(prediction)]
    return preds

data = "/content/drive/MyDrive/IBM Dataset/level/validation/03-severe/0004.JPEG"
image = cv2.imread(data)
print(detect1(image))

1/1 [==============================] - 0s 16ms/step
[[1.8254417e-05 2.8804177e-04 9.9969375e-01]]
severe
```

Code for the application:

- **Index.html**

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
  margin: 0;
```

```
  font-family:  Helvetica sans-serif;
}

.topnav {
  overflow: hidden;
  background-color: #DE3163;
}

.topnav a {
  float: right;
  color: #DFE2EE;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

#logo{
  float: left;
  font-size: 20px;
  text-decoration: none;
}


.content{
  text-align: center;
}
.content-button{
  margin-left: 50%;
  background-color: #333;
  padding: 5px 10px;
  color: white;
}

.para{
  padding-top: 25px;
  text-align:justify;
  font-size: 22px;
}

</style>
</head>
<body>

<div class="topnav">
  <a  id="logo" href="#">Vechile Damage Detection</a>
  <a href="{{ url_for('register') }}">Register</a>
  <a href="{{ url_for('login') }}">Login</a>
  <a href="{{ url_for('index') }}">Home</a>
</div>



<div>
  <h2 class="content"><u>ABOUT PROJECT</u></h2>
  <div class="para">
    <p class="contentPara">
```

Vechile detection is used to reduce claims leakage during insurance processing.Visual inception and validation are usually done.As it takes 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 fot the insurance claim process.</p>

</body>
</html>

- **Login.html**

```
<!DOCTYPE html>
<html lang="en" >
<head>
  <meta charset="UTF-8">
  <title>Login</title>
  <!-- <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/normalize/5.0.0/normalize.min.css">
<link rel="stylesheet" href="E:\html\login.css"> -->
<style>
  body {
    /* background-color: #9f9da7; */
    font-size: 1.6rem;
    font-family: "Open Sans", sans-serif;
    color: #2b3e51;
  }

  .topnav {
  overflow: hidden;
  background-color: #DE3163;
}

.topnav a {
  float: right;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

#logo{
  float: left;
  font-size: 20px;
}
  h2 {
    font-weight: 300;
    text-align: center;
  }
  p {
    position: relative;
  }
  a,
  a:link,
  a:visited,
  a:active {
```

```
  color: white;
  -webkit-transition: all 0.2s ease;
  transition: all 0.2s ease;
}
a:focus, a:hover,
a:link:focus,
a:link:hover,
a:visited:focus,
a:visited:hover,
a:active:focus,
a:active:hover {
  color: white;
  -webkit-transition: all 0.2s ease;
  transition: all 0.2s ease;
}
#login-form-wrap {
  background-color: #fff;
  width: 35%;
  margin: 100px auto;
  text-align: center;
  padding: 20px 0 0 0;
  border-radius: 4px;
  box-shadow: 0px 30px 50px 0px rgba(0, 0, 0, 0.2);
}
#login-form {
  padding: 0 60px;
}
input {
  display: block;
  box-sizing: border-box;
  width: 100%;
  outline: none;
  height: 60px;
  line-height: 60px;
  border-radius: 4px;
}
input[type="email"],
input[type="password"] {
  width: 100%;
  padding: 0 0 0 10px;
  margin: 0;
  color: #8a8b8e;
  border: 1px solid #c2c0ca;
  font-style: normal;
  font-size: 16px;
  -webkit-appearance: none;
     -moz-appearance: none;
          appearance: none;
  position: relative;
  display: inline-block;
  background: none;
}
input[type="email"]:focus,
input[type="password"]:focus {
  border-color: #3ca9e2;
}
input[type="email"]:focus:invalid,
```

```
input[type="password"]:focus:invalid {
  color: #cc1e2b;
  border-color: #cc1e2b;
}
input[type="email"]:valid ~ .validation,
input[type="password"]:valid ~ .validation {
  display: block;
  border-color: #0C0;
}
input[type="email"]:valid ~ .validation span,
input[type="password"]:valid ~ .validation span {
  background: #0C0;
  position: absolute;
  border-radius: 6px;
}
input[type="email"]:valid ~ .validation span:first-child,
input[type="password"]:valid ~ .validation span:first-child {
  top: 30px;
  left: 14px;
  width: 20px;
  height: 3px;
  -webkit-transform: rotate(-45deg);
          transform: rotate(-45deg);
}
input[type="email"]:valid ~ .validation span:last-child,
input[type="password"]:valid ~ .validation span:last-child {
  top: 35px;
  left: 8px;
  width: 11px;
  height: 3px;
  -webkit-transform: rotate(45deg);
          transform: rotate(45deg);
}
.validation {
  display: none;
  position: absolute;
  content: " ";
  height: 60px;
  width: 30px;
  right: 15px;
  top: 0px;
}
input[type="submit"] {
  border: none;
  display: block;
  background-color: #3ca9e2;
  color: #fff;
  font-weight: bold;
  text-transform: uppercase;
  cursor: pointer;
  -webkit-transition: all 0.2s ease;
  transition: all 0.2s ease;
  font-size: 18px;
  position: relative;
  display: inline-block;
  cursor: pointer;
  text-align: center;
```

```
  }
  input[type="submit"]:hover {
    background-color: #329dd5;
    -webkit-transition: all 0.2s ease;
    transition: all 0.2s ease;
  }
  #create-account-wrap {
    background-color: #eeedf1;
    color: #8a8b8e;
    font-size: 14px;
    width: 100%;
    padding: 10px 0;
    border-radius: 0 0 4px 4px;
  }
</style>

</head>
<body>
  <div class="topnav">
    <a  id="logo" href="#">Vechile Damage Detection</a>
    <a href="{{ url_for('register') }}">Register</a>
    <a href="{{ url_for('login') }}">Login</a>
    <a href="{{ url_for('index') }}">Home</a>
  </div>
    <div id="login-form-wrap">
      <h2>Login</h2>
      <form id="login-form" method="POST" action="/afterlogin">
        <p>
           <input type="email" id="email" name="_id" placeholder="Email Address" required><i
class="validation"><span></span><span></span></i>
        </p>
        <p>
           <input type="password" id="password" name="psw" placeholder="Password" required><i
class="validation"><span></span><span></span></i>
        </p>
        <p>
        <input type="submit" id="login" value="Login">
        </p>
      </form>
      <div id="create-account-wrap">
    </div>
</body>
</html>
```

- **Logout.html**

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
}

.topnav {
  overflow: hidden;
```

```
  background-color: #DE3163;
}

.topnav a {
  float: right;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

#logo{
  float: left;
  font-size: 20px;
  text-decoration: none;
}


.content{
  text-align: center;
  padding-left: 37px;
}
.content-button{
  margin-left: 50%;
  background-color: #DE3163;
  padding: 5px 10px;
  color: white;
  text-decoration: none;
}
</style>
</head>
<body>

<div class="topnav">
  <a  id="logo" href="#">Vechile Damage Detection</a>
  <a href="{{ url_for('register') }}">Register</a>
  <a href="{{ url_for('login') }}">Login</a>
  <a href="{{ url_for('index') }}">Home</a>
</div>



<div>
  <h2 class="content">Successfully Logged out</h2>
  <p class="content">Login for more information</p>
  <a class="content-button" href="{{ url_for('login') }}">Login</a>
</div>

</body>
</html>
```

- **Predict.html**

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<style>
body {
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
}

.topnav {
  overflow: hidden;
  background-color:#DE3163;
}

.topnav a {
  float: right;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

#logo{
  float: left;
  font-size: 20px;
  text-decoration: none;
}

.footer {
  position: fixed;
  left: 0;
  bottom: 0;
  width: 100%;
  font-weight: bold;
  background-color: #DE3163;
  color: white;
  text-align: center;
}
</style>
</head>
<body>

  <form id="" method="POST" action="/result" enctype="multipart/form-data">
  <div class="topnav">
        <a  id="logo" href="#">Vechile Damage Detection</a>
        <a href="{{ url_for('logout') }}">Logout</a>
        <a href="{{ url_for('index') }}">Home</a>
      </div>
   <div style="padding-top: 20px;">
    <input type="file" name="image" id=""/>
    <input type="submit"/>
   </div>
   <div>
    {% if value %}
      <h3 style="text-align: center ;">The Estimated cost For The Damage Is: {{value}} </h3>
    {% endif %}
  </div>
  </form>
      <div class="footer">
```

```
      <p>Copyright@2021.All Rights Reserved</p>
    </div>
</body>
</html>
```

- **Register.html**

```
<!DOCTYPE html>
<html lang="en" >
<head>
  <meta charset="UTF-8">
  <title>Register</title>
  <!-- <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/normalize/5.0.0/normalize.min.css">
<link rel="stylesheet" href="E:\IBM\static\styles\Register.css"> -->
<style>
  body {
    /* background-color: #9f9da7; */
    font-size: 1.6rem;
    font-family: "Open Sans", sans-serif;
    color: #2b3e51;
  }

  .topnav {
  overflow: hidden;
  background-color: #DE3163;
}

.topnav a {
  float: right;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

#logo{
  float: left;
  font-size: 20px;
}
  h2 {
    font-weight: 300;
    text-align: center;
  }
  p {
    position: relative;
  }
  a,
  a:link,
  a:visited,
  a:active {
    color: white;
    -webkit-transition: all 0.2s ease;
    transition: all 0.2s ease;
  }
  a:focus, a:hover,
  a:link:focus,
```

```
a:link:hover,
a:visited:focus,
a:visited:hover,
a:active:focus,
a:active:hover {
  color: white;
  -webkit-transition: all 0.2s ease;
  transition: all 0.2s ease;
}
#login-form-wrap {
  background-color: #fff;
  width: 35%;
  margin: 100px auto;
  text-align: center;
  padding: 20px 0 0 0;
  border-radius: 4px;
  box-shadow: 0px 30px 50px 0px rgba(0, 0, 0, 0.2);
}
#login-form {
  padding: 0 60px;
}
input {
  display: block;
  box-sizing: border-box;
  width: 100%;
  outline: none;
  height: 60px;
  line-height: 60px;
  border-radius: 4px;
}
input[type="text"],
input[type="email"],
input[type="password"] {
  width: 100%;
  padding: 0 0 0 10px;
  margin: 0;
  color: #8a8b8e;
  border: 1px solid #c2c0ca;
  font-style: normal;
  font-size: 16px;
  -webkit-appearance: none;
     -moz-appearance: none;
          appearance: none;
  position: relative;
  display: inline-block;
  background: none;
}

input[type="text"]:focus,
input[type="email"]:focus,
input[type="password"]:focus {
  border-color: #3ca9e2;
}
input[type="text"]:focus:invalid,
input[type="email"]:focus:invalid,
input[type="password"]:focus:invalid {
  color: #cc1e2b;
```

```
  border-color: #cc1e2b;
}
input[type="text"]:valid ~ .validation,
input[type="email"]:valid ~ .validation,
input[type="password"]:valid ~ .validation {
  display: block;
  border-color: #0C0;
}
input[type="text"]:valid ~ .validation span,
input[type="email"]:valid ~ .validation span,
input[type="password"]:valid ~ .validation span {
  background: #0C0;
  position: absolute;
  border-radius: 6px;
}
input[type="text"]:valid ~ .validation span:first-child,
input[type="email"]:valid ~ .validation span:first-child,
input[type="password"]:valid ~ .validation span:first-child {
  top: 30px;
  left: 14px;
  width: 20px;
  height: 3px;
  -webkit-transform: rotate(-45deg);
          transform: rotate(-45deg);
}
input[type="text"]:valid ~ .validation span:last-child,
input[type="email"]:valid ~ .validation span:last-child,
input[type="password"]:valid ~ .validation span:last-child {
  top: 35px;
  left: 8px;
  width: 11px;
  height: 3px;
  -webkit-transform: rotate(45deg);
          transform: rotate(45deg);
}
.validation {
  display: none;
  position: absolute;
  content: " ";
  height: 60px;
  width: 30px;
  right: 15px;
  top: 0px;
}
input[type="submit"] {
  border: none;
  display: block;
  background-color: #3ca9e2;
  color: #fff;
  font-weight: bold;
  text-transform: uppercase;
  cursor: pointer;
  -webkit-transition: all 0.2s ease;
  transition: all 0.2s ease;
  font-size: 18px;
  position: relative;
  display: inline-block;
```

```
    cursor: pointer;
    text-align: center;
  }
  input[type="submit"]:hover {
    background-color: #329dd5;
    -webkit-transition: all 0.2s ease;
    transition: all 0.2s ease;
  }
  #create-account-wrap {
    background-color: #eeedf1;
    color: #8a8b8e;
    font-size: 14px;
    width: 100%;
    padding: 10px 0;
    border-radius: 0 0 4px 4px;
  }

</style>
</head>
<body>
  <div class="topnav">
    <a  id="logo" href="#">Vechile Damage Detection</a>
    <a href="{{ url_for('register') }}">Register</a>
    <a href="{{ url_for('login') }}">Login</a>
    <a href="{{ url_for('index') }}">Home</a>
  </div>
<div id="login-form-wrap">
  <h2>Register</h2>
  <form id="login-form" method="POST" action="/afterreg">
    <p>
       <input type="text" id="text" name="name" placeholder="Username" required><i
class="validation"><span></span><span></span></i>
    </p>
    <p>
       <input type="email" id="email" name="email" placeholder="Email Address" required><i
class="validation"><span></span><span></span></i>
    </p>
    <p>
       <input type="password" id="password" name="psw" placeholder="Password" required=""><i
class="validation"><span></span><span></span></i>
    </p>
    <p>
    <input type="submit" id="login" value="Register">
    </p>
  </form>
  <div id="create-account-wrap">
    <!-- <p>Not a member? <a href="#">Create Account</a><p> -->
  </div><!--create-account-wrap-->
</div><!--login-form-wrap-->
<!-- partial -->
</body>
</html>
```

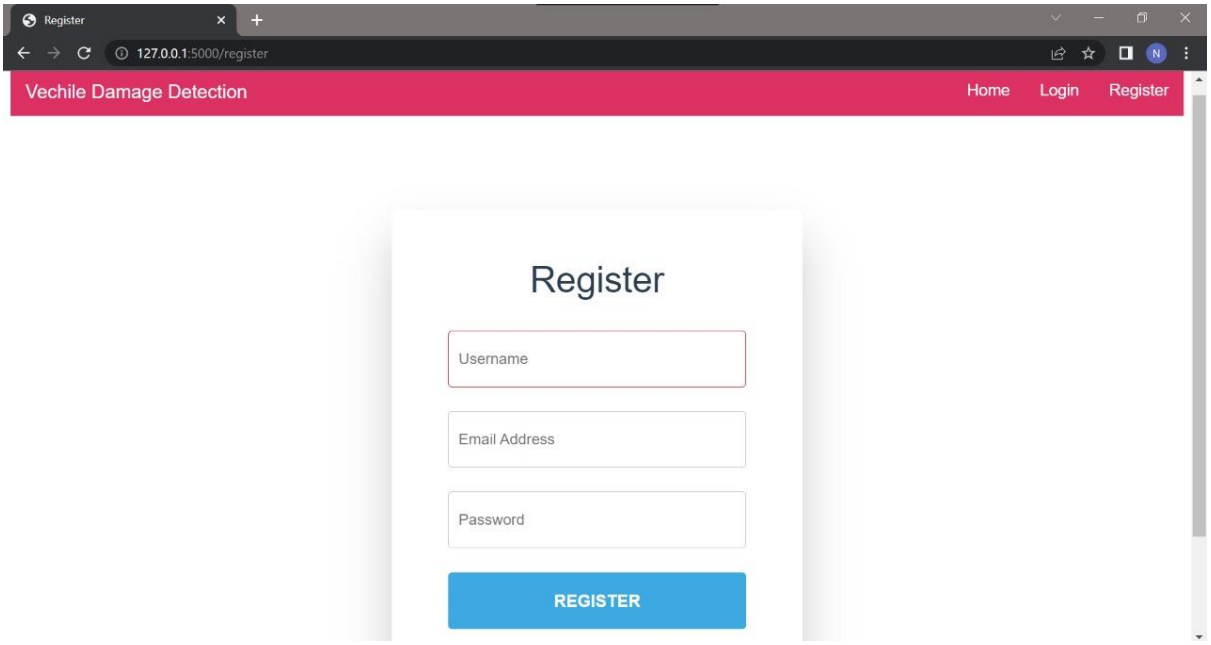
HTML Pages and cloud

Home page:

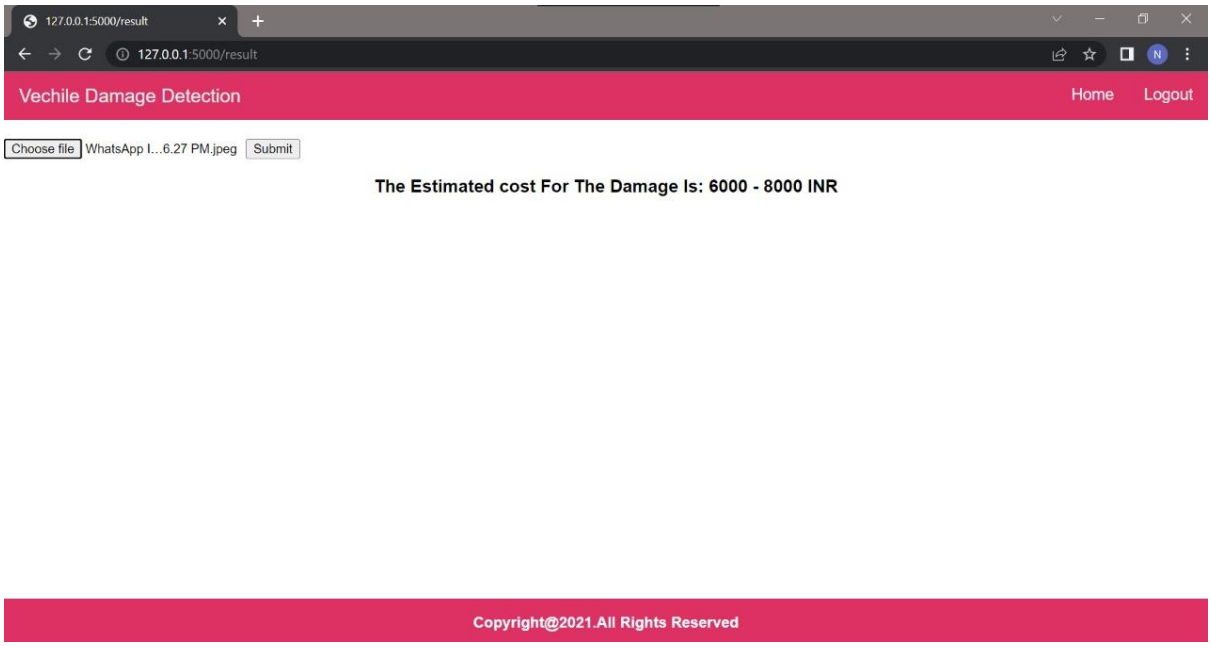
ABOUT PROJECT

Vechile detection is used to reduce claims leakage during insurance processing.Visual inception and validation are usually done.As it takes 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 fot the insurance claim process.

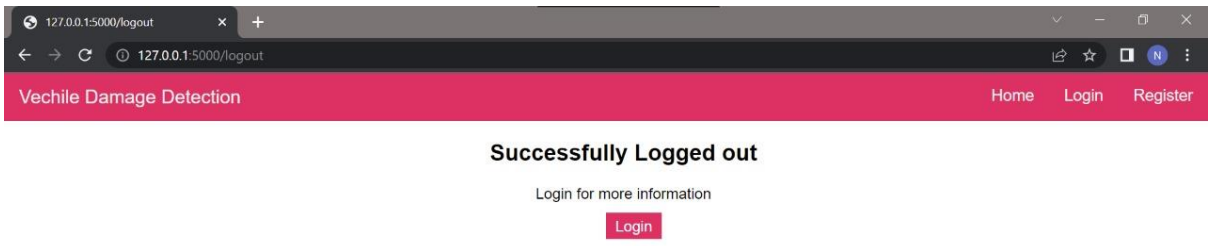
Register page:



Prediction page:

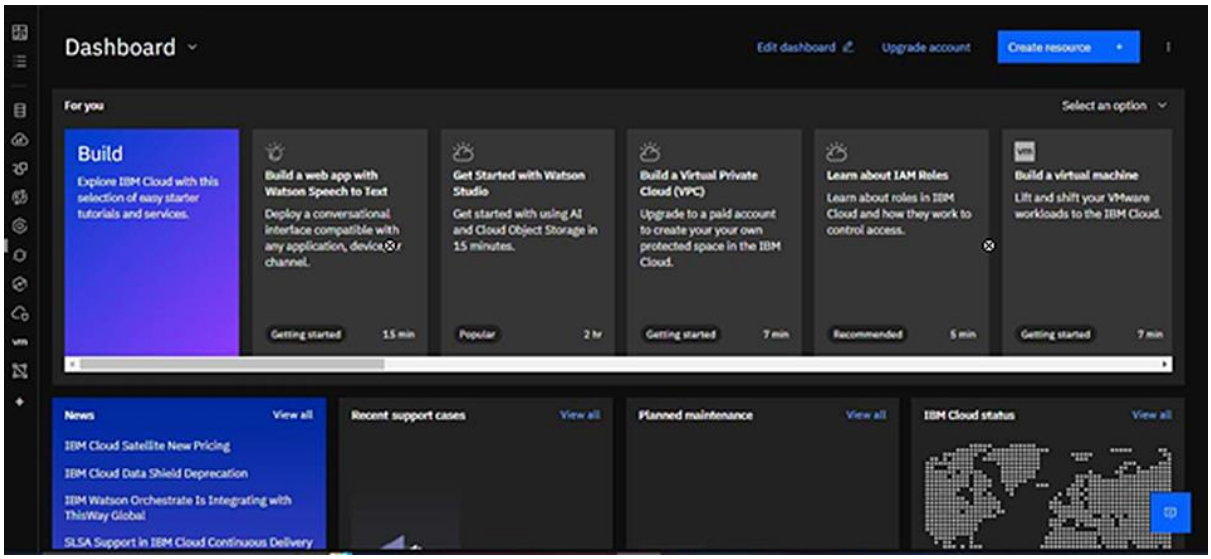


Logout page:

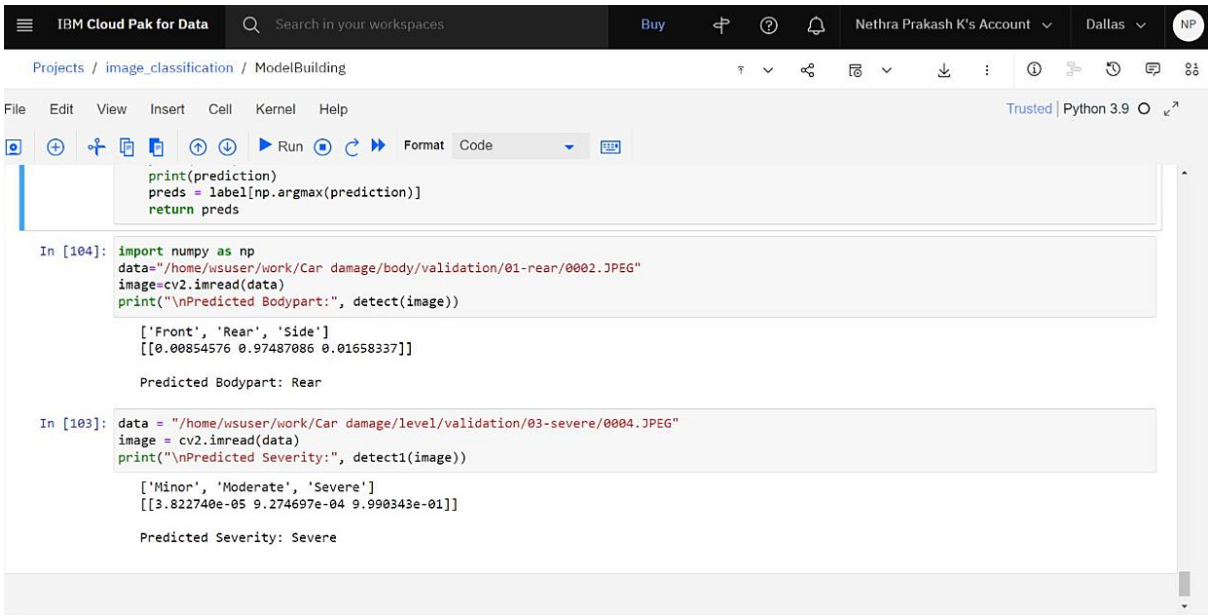


Additional Features:

Cloud Database:



Model trained on cloud:



8 TESTING

8.1 Test cases

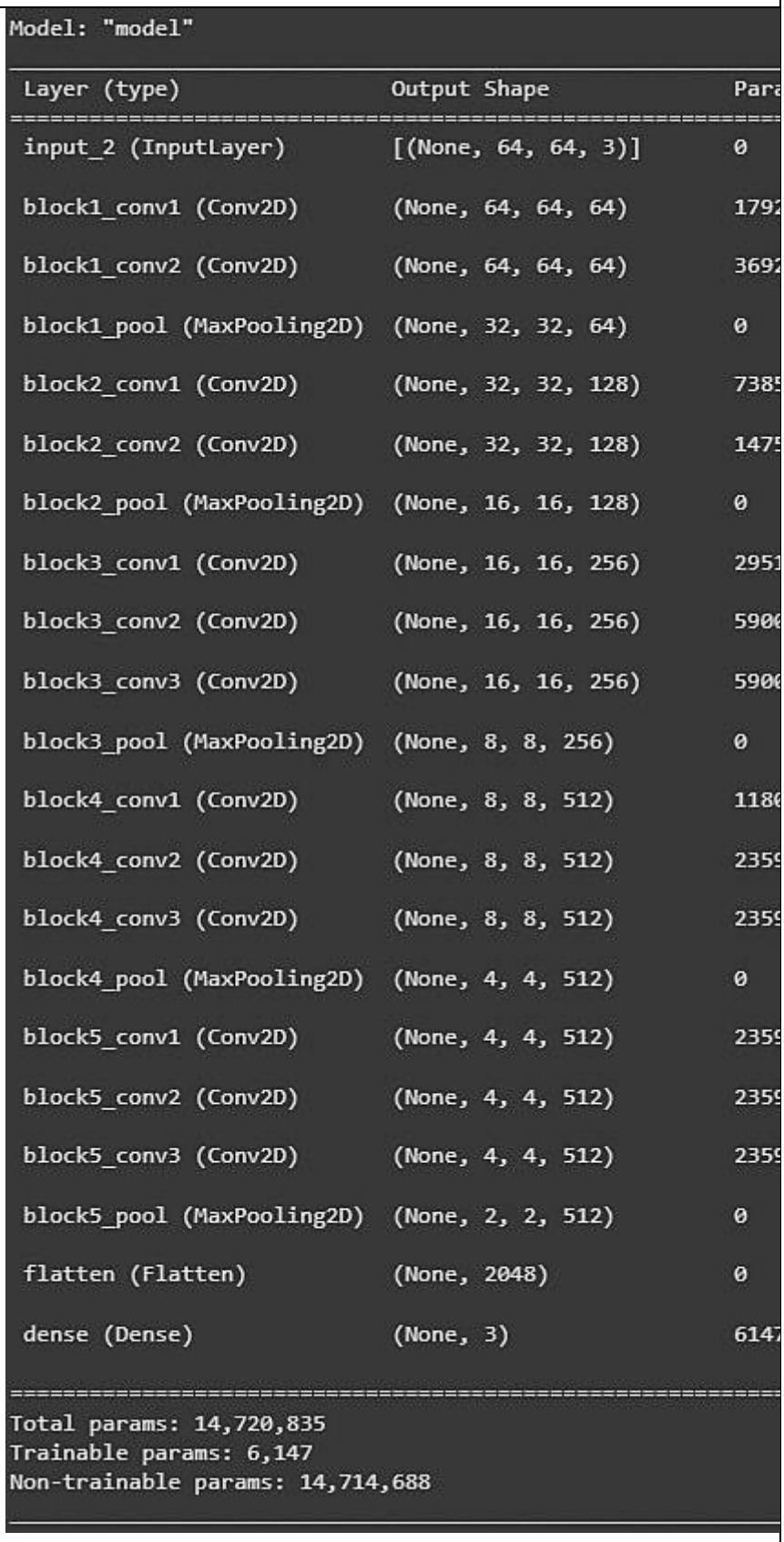
S. No.	Test Scenarios
1.	Verify if user is able to see the login popup whenever he/she clicks on the login button.
2.	Verify if user is able to log into the application with invalid credentials.
3.	Verify if the user is able to upload images in the jpeg format to the application.
4.	Verify if the user is able to find the location of damage and extent of severity of damage.
5.	Verify if the user is able to login again after logging out of the application successfully.
6.	Verify the UI in the register page of the application.
7.	Verify user information is stored in databse for further communications.
8.	Verify the UI elements in the register page of the application.

8.2 User Acceptance Testing

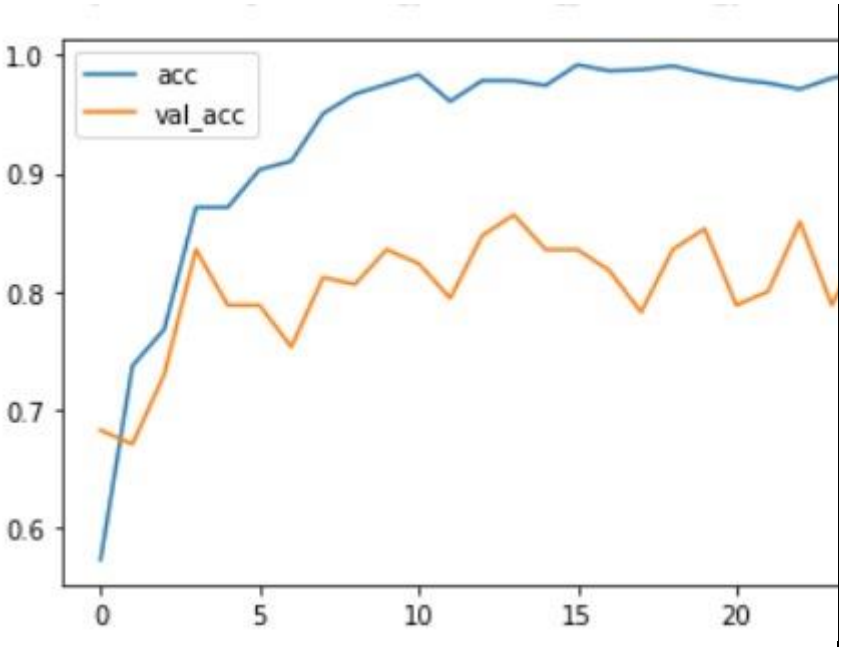
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

9. RESULTS

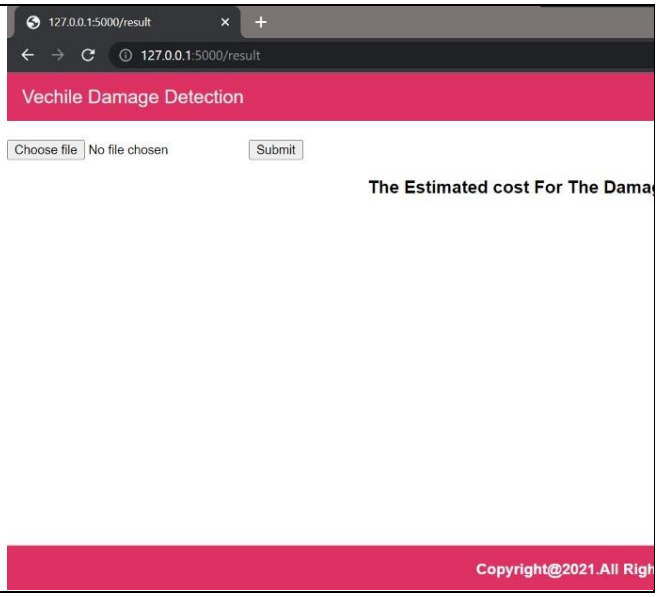
9.1 Performance Metrics

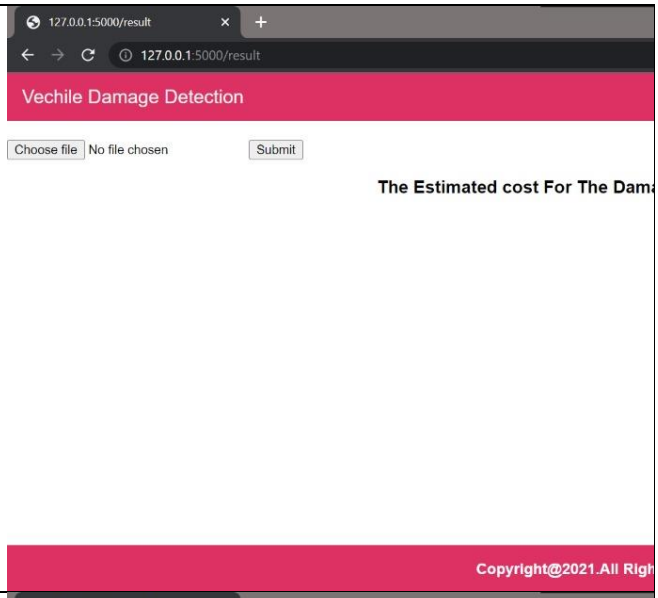
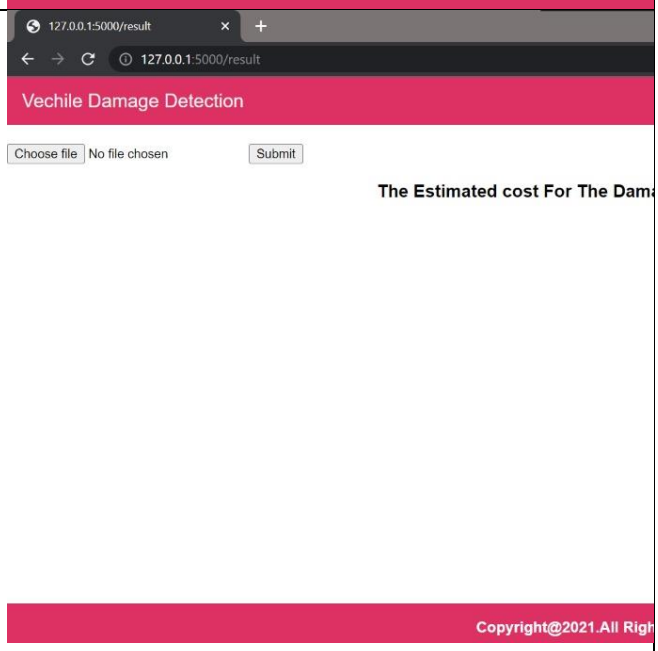
S.No.	Parameter	Values	Screenshot
1.	Model Summary (VGG 19)	Total params: 14,720,835 Trainable params: 6,147 Non-trainable params: 14,714,688	

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Para
=================================================================
 input_2 (InputLayer)        [(None, 64, 64, 3)]       0

 block1_conv1 (Conv2D)       (None, 64, 64, 64)        179

 block1_conv2 (Conv2D)       (None, 64, 64, 64)        369

 block1_pool (MaxPooling2D)  (None, 32, 32, 64)        0

 block2_conv1 (Conv2D)       (None, 32, 32, 128)       738

 block2_conv2 (Conv2D)       (None, 32, 32, 128)       147

 block2_pool (MaxPooling2D)  (None, 16, 16, 128)       0

 block3_conv1 (Conv2D)       (None, 16, 16, 256)       295

 block3_conv2 (Conv2D)       (None, 16, 16, 256)       590

 block3_conv3 (Conv2D)       (None, 16, 16, 256)       590

 block3_pool (MaxPooling2D)  (None, 8, 8, 256)         0

 block4_conv1 (Conv2D)       (None, 8, 8, 512)         118

 block4_conv2 (Conv2D)       (None, 8, 8, 512)         235

 block4_conv3 (Conv2D)       (None, 8, 8, 512)         235

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         235

 block5_conv2 (Conv2D)       (None, 4, 4, 512)         235

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         235

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 3)                 614

=================================================================
Total params: 14,720,835
Trainable params: 6,147
Non-trainable params: 14,714,688
_________________________________________________________________
```

2.	Accuracy	Training Accuracy - 98.56% Validation Accuracy -84.12%	
3.	Confidence Score (Only Yolo Projects)	Class Detected - NIL Confidence Score - NIL	NIL

MODEL RESULTS:



10. ADVANTAGES & DISADVANTAGES

Advantages	Disadvantages
The time taken to process such claims is reduced drastically.	The policies of different insurance companies are not similar to each other and hence the model parameters have to changes accordingly.

There is no need of a physical examination of the damaged vehicle and hence reduces labour as well.	The model predicts only if the damage has occurred in the side, rear and front and thus ignores the cost of the components individually.
The accuracy of the model is also good and thus gives us more precise claim amount and not a biased value.	

11. CONCLUSION

In this project we've built a neural network based solution for car damage detection; manage the problem of car damage analysis, prediction of car damage location and severity of the damage. This project carries out lot of functions in a one package. The system will definitely help the insurance companies to analyse the car damage a lot more successful and well organized. Simply by sending the image of the car, the system will analyse the given image and show if there is any kind of damage to the car along with the location of the damage and also the severity of the damage. This method provides a much better and efficient solution to the insurance companies hence cutting down the leakage in claim amounts.

12. FUTURE SCOPE

- This solution can be used by any insurance company be it private or public/ government companies to provide insurance claim to all kinds of vehicles ranging from two-wheelers to heavy load vehicles.
- It can also be used to provide the cost for that particular equipment which is damaged.
- The model that is deployed in this project can be personalized to suit the policies and needs of a particular Insurance company.
- On the other hand the cost of an individual component can be predicted and thus an insurance amount can be claimed, further reducing the leakage in claims.

13. APPENDIX

GitHub & Project Demo Link ~ https://github.com/IBM-EPBL/IBM-Project-26129-_1660016950
VideoLink~https://drive.google.com/file/d/11TrpMjwhWyOxqLBkmrfR_0Tjfsu2D9O/view?usp=sharing