INTELLIGENT VECHICLE DAMAGE ASSESSMENT AND COST ESTIMATOR FOR INSURANCE COMPANIES

NALAIYA THIRAN PROJECT BASED LEARNING

on

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

A PROJECT REPORT

BARATHRAJ R.S	19104034
BIBIN P	19104037
DANUSH HARI	19104042
DHAYABARAN	19104046
GOWTHAM P	19104063

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

HINDUSTHAN COLLEGE OF ENGINEERING AND TECHOLOGY

Approved by AICTE, New Delhi, Accredited with 'A' Grade by NAAC

(An Autonomous Institution, Affiliated to Anna University, Chennai)

COIMBATORE – 641 032

November 2022

TABLE OF CONTENTS

ABSTRACT

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.

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

1.INTRODUCTION

1. Overall description

At present, under the guidance of the new generation of information technology, the rapid accumulation of data, the continuous improvement of computing power, the continuous optimization of algorithm models, and the rapid rise of multi-scene applications have made profound changes in the development environment of artificial intelligence. In this Project, based on the demand of automobile insurance claims and intelligent transportation, combined with abundant basic data and advanced machine vision algorithm, an intelligent damage determination system of 'Artificial Intelligence + Vehicle Insurance' is constructed.

It helps the user and the insurance provider to quickly claim the insurance with the help of Simple Technology using Artificial Intelligence and web components, where the user can upload the images and claim his/her Insurance by the level of the damage

1.1 Problem Statement

- Takes more time to analyze the level of damage done to the vehicle
- Require many multiple level of processing to gain insurance amount
- Cost estimation also need multiple level of processing and require more time to process the whole estimation of the damage and its insurance amount
- There are many accidents happening worldwide and it is difficult to manage the details of them

1.2 Purpose

The project is based on the domain of Artificial Intelligence and powered by the IBM watson cloud. A responsive web application can be developed using the HTML and CSS which is connected to waston cloud. In the cloud, a database service by availing the service Instance of the IBM cloud and the database API key is collected and connected with the front-end using flash which is an python framework for desiging the backend. Pages such as index.html, login.html, logout.html and prediction.html are used to interact with the web application. The user can register and the data of the user is saved in the databse of the IBM cloud, during the time of login, the login ID is compared with the ID in the databse and allow the user to the next page. The Deep Learning model is build using the VGG16 which is present in the keras library and the model is trained with the images of mulitple car with various level cum types of damages. The model is deployed in the back-end using the flask and the prediction.html page is setted to collect the image from the user. The prediction algorithm is used treat the image and estimated the cost for the user. The project is based on the various components which helps to handle the back - end and Front - end. Then front - end is build using html and css which is connectedback - end which is build using the python and IBM cloud. The project is powered by the IBM Watson cloud and is based in the artificial intelligence field. With the use of HTML and CSS and the Waston Cloud, a responsive web application may be created. The database API key is gathered and connected with the front-end using flash, which is a python framework for designing the backend

2.PROJECT OBJECTIVES

- The user interacts with the UI (User Interface) to choose the image.
- The chosen image is analyzed by the model which is integrated with the flask application.
- VGG16 Model analyzes the image, then the prediction is showcased on the Flask UI.

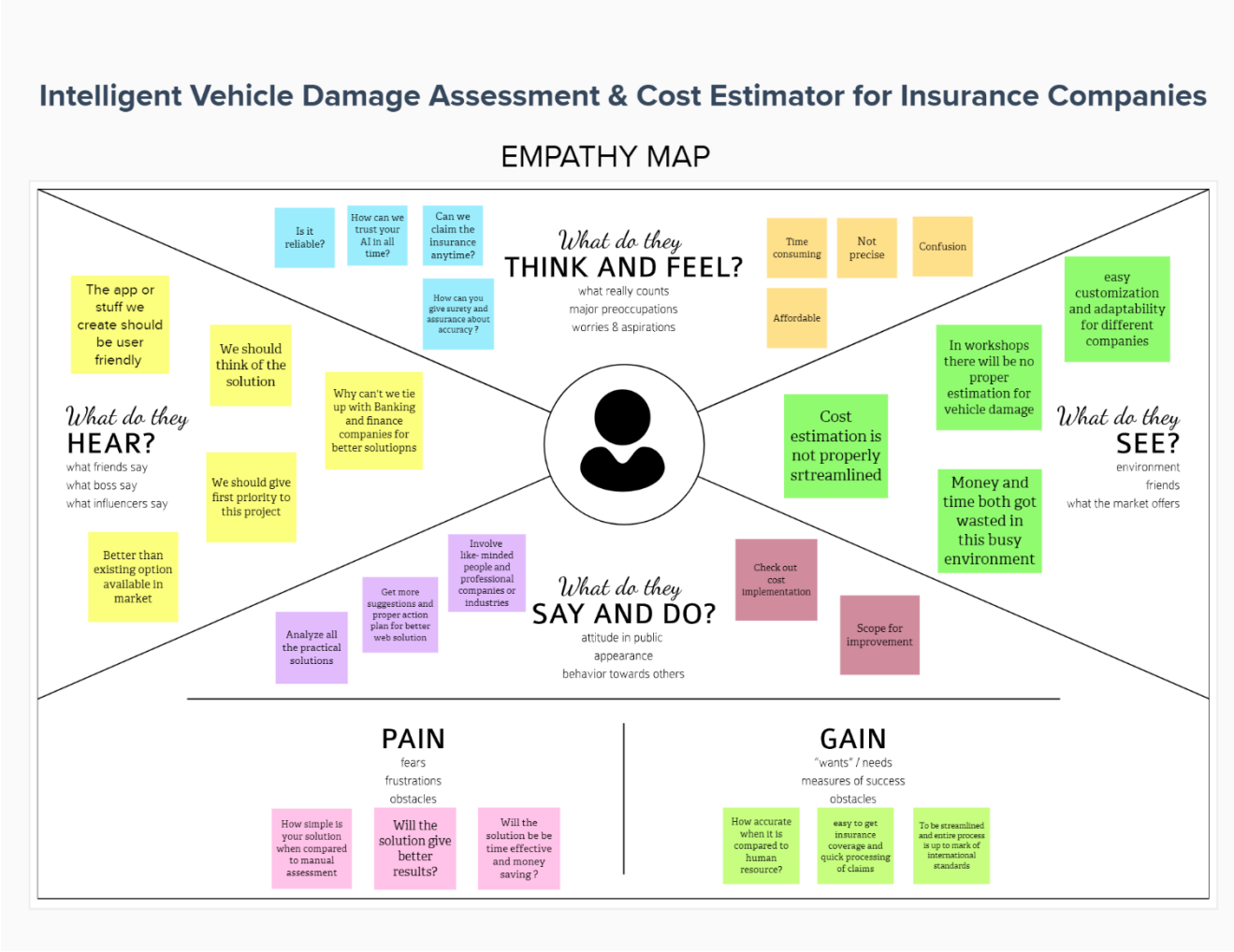
3. IDEATION PHASE

3.1 Literature Survey

1. Vaibhav Agarwal, Utsav Khandelwal, Shivam Kumar, Raja Kumar, Shilpa M, "Damage Assessment Of A Vehicle And Insurance Reclaim", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.10, Issue 4, pp.e197-e201, April 2022, Available at :http://www.ijcrt.org/papers/IJCRT2204483.pdf 2. R. E. van Ruitenbeek and S. Bhulai, “Convolutional Neural Networks for vehicle damage detection,” Machine Learning with Applications, vol. 9. Elsevier BV, p. 100332, Sep. 2022. doi: 10.1016/j.mlwa.2022.100332. 3. Mandara G S and Prashant Ankalkoti, “Car Damage Assessment for Insurance Companies,” International Journal of Advanced Research in Science, Communication and Technology. Naksh Solutions, pp. 431–436, Jun. 23, 2022. doi: 10.48175/ijarsct-5048. 4. H. Bandi, S. Joshi, S. Bhagat, and A. Deshpande, “Assessing Car Damage with Convolutional Neural Networks,” 2021 International Conference on Communication information and Computing Technology (ICCICT). IEEE, Jun. 25, 2021. doi: 10.1109/iccict50803.2021.9510069. 5. U. Waqas, N. Akram, S. Kim, D. Lee and J. Jeon, "Vehicle Damage Classification and Fraudulent Image Detection Including Moiré Effect Using Deep Learning," 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2020, pp. 1-5, doi: 10.1109/CCECE47787.2020.9255806. 6. M. Dwivedi et al., “Deep Learning-Based Car Damage Classification and Detection,” Advances in Intelligent Systems and Computing. Springer Singapore, pp. 207–221, Aug. 14, 2020. doi: 10.1007/978-981-15-3514-7_18. 7. P. M. Kyu and K. Woraratpanya, “Car Damage Detection and Classification,” Proceedings of the 11th International Conference on Advances in Information Technology. ACM, Jul. 2020. doi: 10.1145/3406601.3406651

2. Intelligent Vehicle Damage Assessment system based on computer vision :
Zhu Qianqian, Guo Weiming, Shen Ying, and Zhao Zihao: The system completes the whole process of survey and damage determination through four functions. They are: 1) Accident investigation, 2) Intelligent image damage assessment, 3) Damage result output, 4) Vehicle insurance anti- fraud, Resource Link: (PDF) Research on Intelligent Vehicle Damage Assessment System Based on Computer Vision (researchgate.net)

3.2 Empathy Map



3.4 Problem Statement

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	a car owner	claim an insurance	my money is wasted	of claim leakage	annoyed
PS-2	an insurer	be good with customers	they try to fool me	they try to get money from false claims	betrayal
PS-3	a policy holder	claim insurance amount	It's time consuming	insurer delay the process	more pressured
PS-4	a car owner	claim insurance for my damaged car	I didn't get appropriate money	of error in process	overstressed

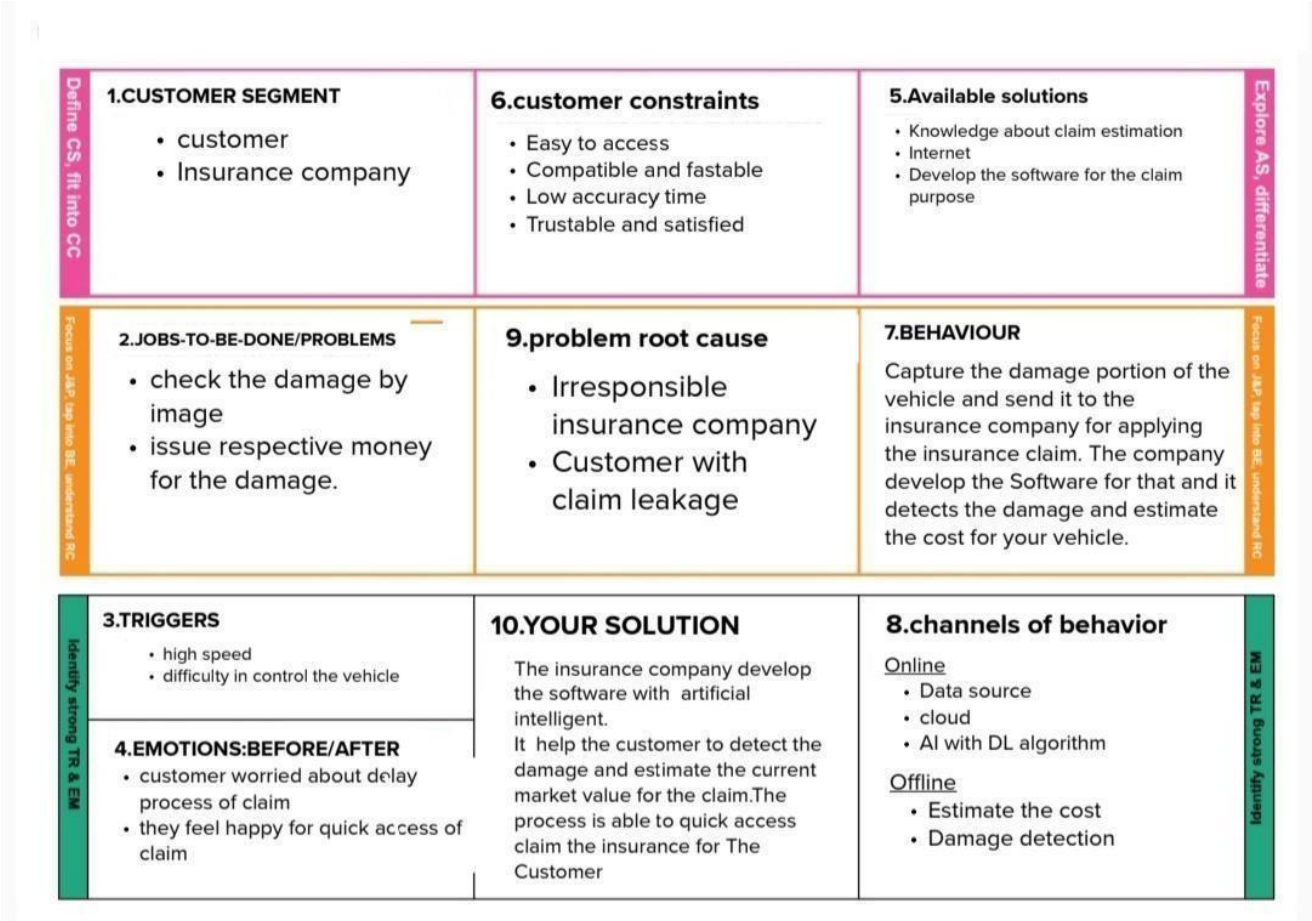
4.PROJECT DESIGN PHASE 1

4.1 Proposed Solution

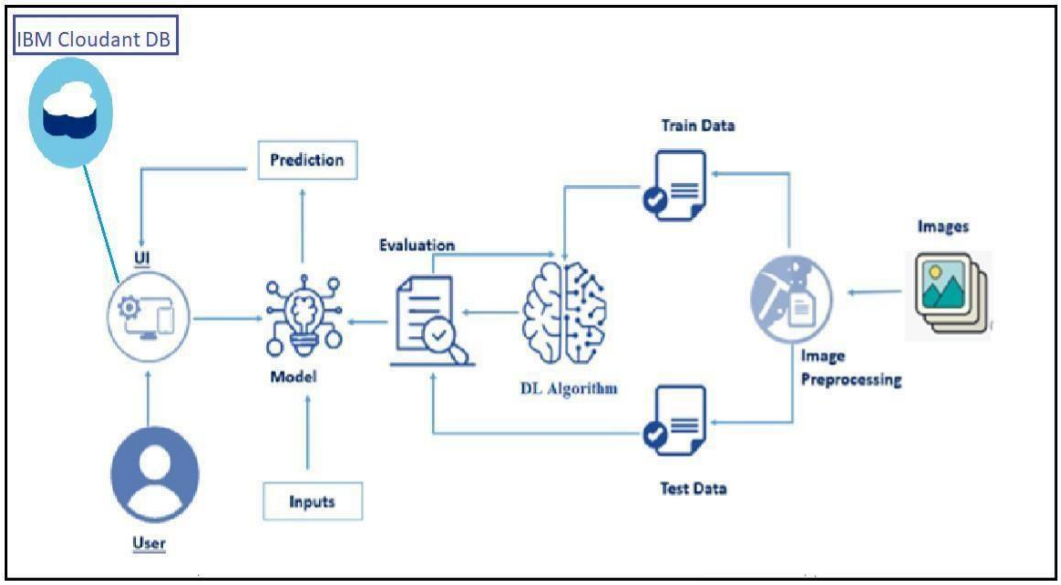
S.NO.	PARAMETER	DESCRIPTION
1.	Problem Statement (Problem to be solved)	Now a days, The Insurance Companies faced the greatest problem that is leakage of insurance claim. Some of the customer done the extra amount for their damage of vehicle through fake bill for the claim. So The Insurance Companies lost their most of the amount due to leakage of claim.
2.	Idea / Solution description	To solve this problem, we have to develop the software that helps to the insurance companies. The procedures we design involves creative initiative that will inspire the company has to believe in that software and also the customer.
3.	Novelty / Uniqueness	In our project we find out one solution for problems we used Artificial Intelligent .

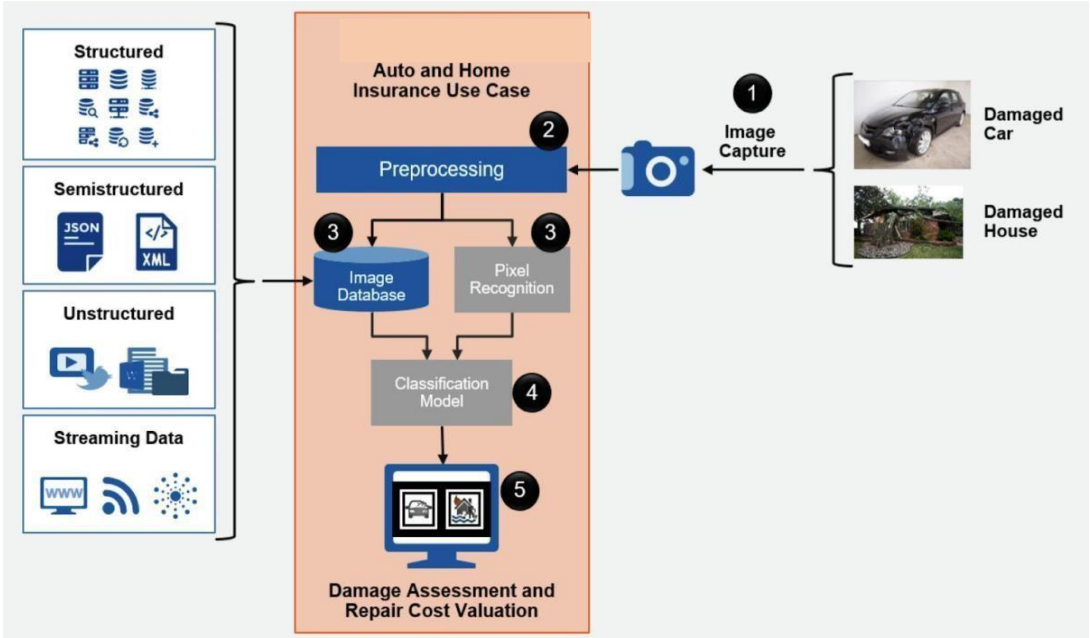
4.	Social Impact / Customer Satisfaction	In our projects , Take the photo of damaged portion of vehicle and send it to the company for apply the claim. The process is more quick to access the claim and estimate the cost .so we think it is one of the best satisfaction for customers
5.	Business Model (Revenue Model)	It is more effective than others. It reduces delay process of claim. This helps the customer to get the claim quickly. Reducing Time accuracy
6.	Scalability of the Solution	It is user friendly. It has more advantages.It is trustworthy.

4.2 Problem Solution Fit



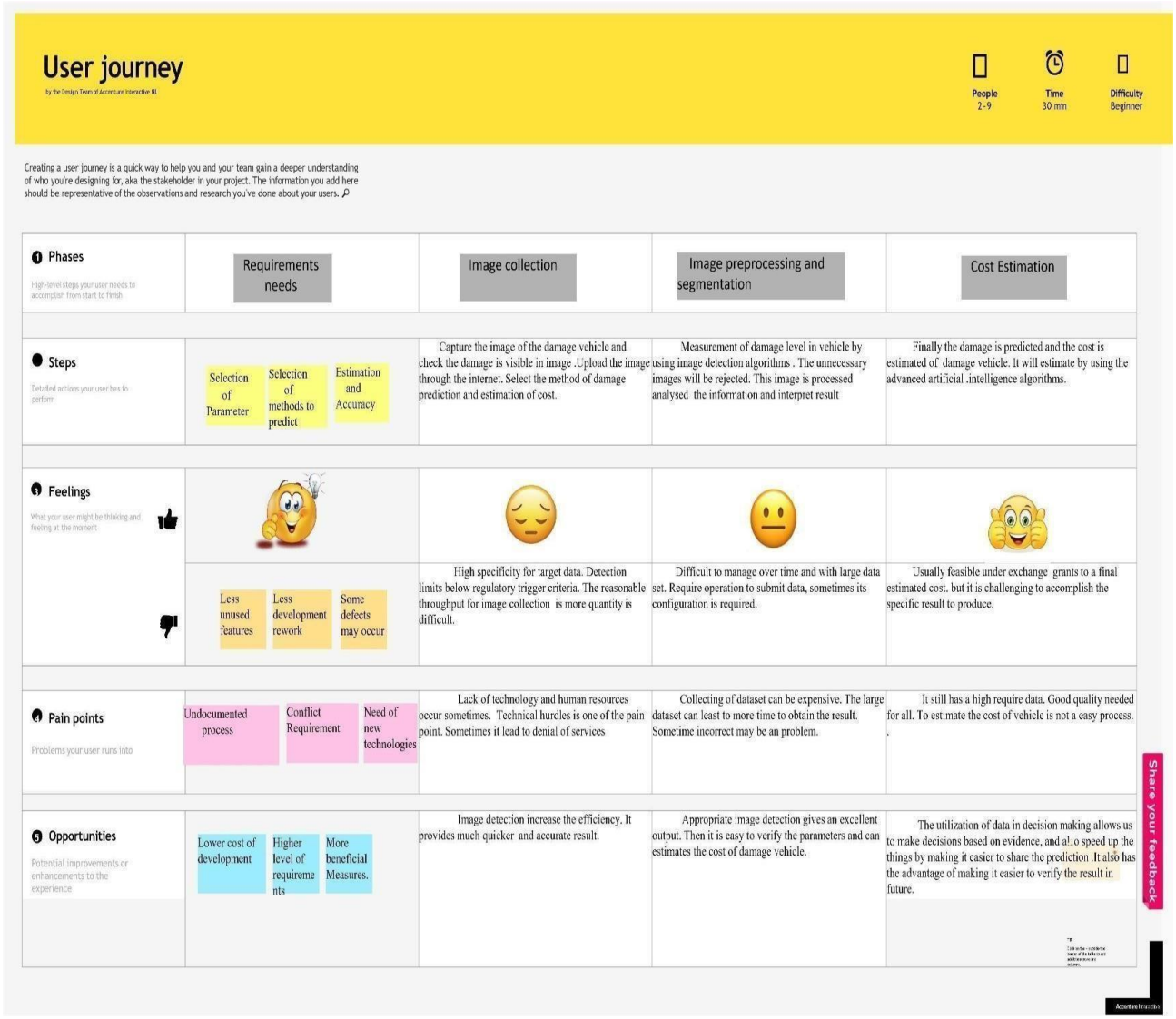
4.3 Solution Architecture





5.PROJECT DESIGN PHASE 2

5.1 Customer Journey Map



5.2 Solution Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User registeration	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 user to operate
FR-4	Accessing datasets	Details about user Details about vehicle Details about insurance companies
FR-5	Mobile application	AI and camera sensor in the field can be access by mobile application.

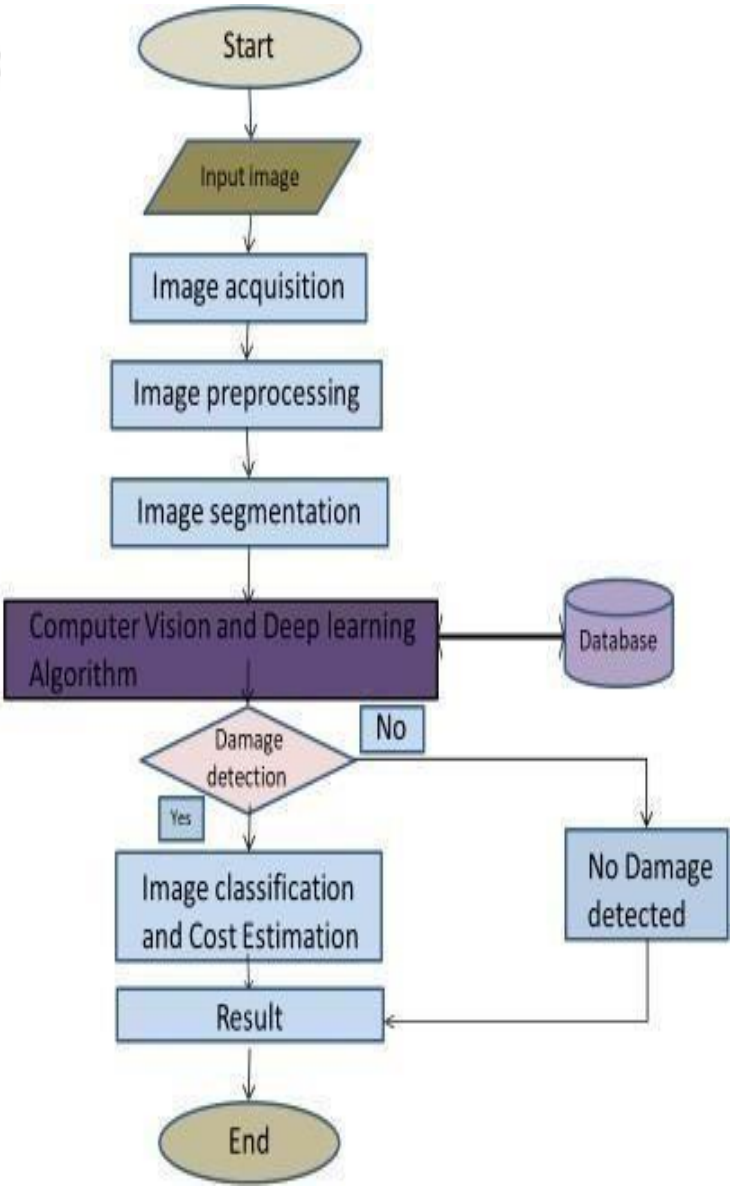
Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The smart claiming system for vehicle damage insurance in bank companies
NFR-2	Security	We have designed this project to user easy to claim the insurance.
NFR-3	Reliability	This project will help the user to claim the insurance cost based on vehicle damage. It gives the exact value to user. This helps user to get correct cost without any failure.
NFR-4	Performance	AI devices and sensors are used to indicate the user to estimated the cost of the vehicle.AI camera to scan the damaged vehicle and gives exact cost insurance to user.
NFR-5	Availability	This application is designed for all devices and also vialable in apk.
NFR-6	Scalability	This project is more scalability in our present and future uses to estimate the cost exactly to user.

5.3 Data Flow Diagrams

Data Flow Diagram:



5.4 Technology Stack

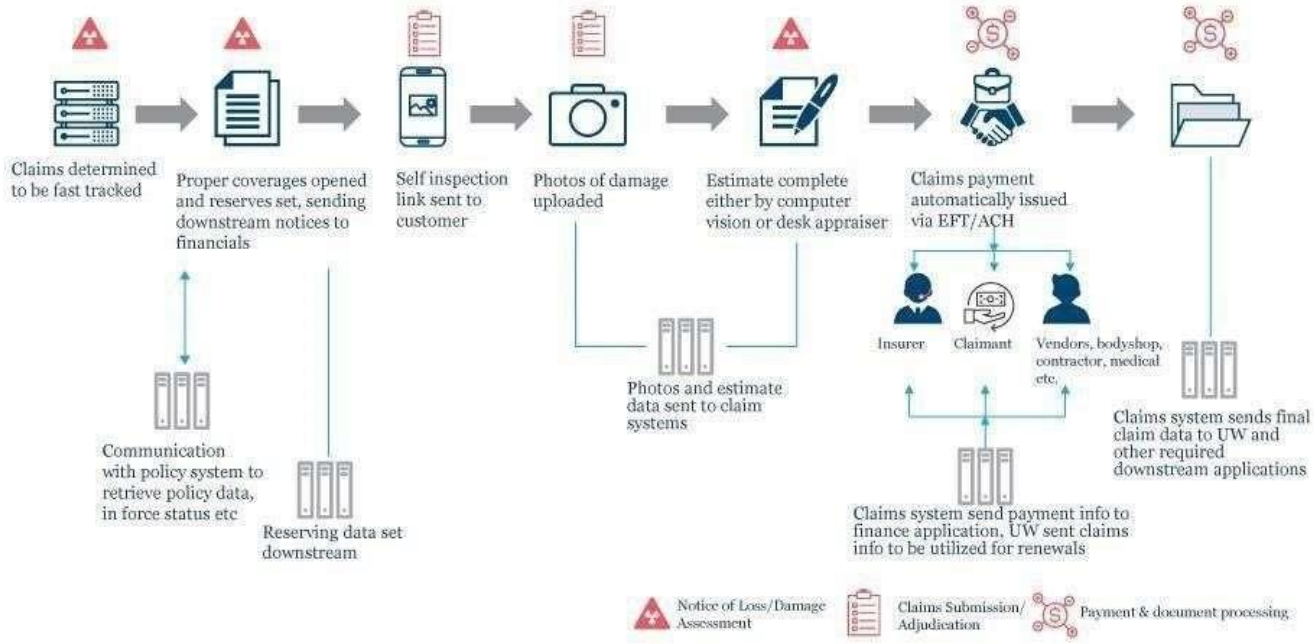


Table-1: Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g., Mobile Application	HTML, CSS, JavaScript.
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, Deep learning
6.	Cloud Database	Database Service on AI	IBM Cloudant DB
7.	File Storage	File storage is more.	IBM Block Storage or Other Storage Service or Local Filesystem
8.	AI Model	Purpose of AI Model is for estimating the cost of the damaged vehicle.	IBM AI Platform
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Cloud Foundry, etc.

6.PROJECT PLANNING PHASE

6.1 Prepare Milestone and Activity List

Sprint	Milestone
Sprint 1	**1.User can create a new account by providing user information such as name, email id and mobile number.** **2.User will get a confirmation message through E-mail.** **3.Upon confirmation, creating new password and account creation is done.** 4.**User login portal will be developed.** 5.**User can login and logout whenever required.**
Sprint 2	1. **Users can access their dashboard and other details can be updated.** 2. **User uploads the images of their vehicle and other relevant details**
Sprint 3	1.**Developing a Model and verifying the accuracy.** 2.**Creating a secure database to store and access user information and images.** 3.**User will get the detailed assessment of their damaged vehicle and estimated cost for insurance will be generated.**
Sprint 4	**1.Collection user feedback and queries are done.** **2.Constant Updation of the portal and its respective backend work is done.** **3.Additional login features will be implemented.**

6.2 Sprint Delivery Plan

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As an owner of a particular vehicle, I can log into the application by entering email & password.	2	High	Danush Hari Dhayabaran Bibin P Barathraj RS Gowtham P
Sprint-1	User Confirmation	USN-2	As an owner of a particular vehicle, I will receive confirmation email once I have registered for the application.	1	Medium	Danush Hari Dhayabaran Bibin P Barathraj RS Gowtham P
Sprint-1	Login	USN-3	As an owner of a particular vehicle, I can log into the application by entering email & password.	2	High	Danush Hari Dhayabaran Bibin P Barathraj RS Gowtham P
Sprint-2	Data Collection	USN-1	Download the dataset used in intelligent vehicle damage assessment & cost estimator for insurance companies.	2	High	Danush Hari Dhayabaran Bibin P Barathraj RS Gowtham P

Sprint-2	**Image Pre-Processing**	**USN-1**	**Improve the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, etc.**	**2**	**High**	**Danush Hari Dhayabaran Bibin P1` Barathraj RS Gowtham P**
Sprint-3	**Model Building**	**USN-1**	**Define the model architecture and adding CNN layer and testing ,saving the model.**	**2**	**High**	**Danush Hari Dhayabaran Bibin P Barathraj RS Gowtham P**
Sprint-3	**Cloud DB**	**USN-1**	**Below are steps that need to follow for creating and using cloudant service.** • **Register & login to IBM cloud** • **Create service instance** • **Creating service credentials** • **Launch cloudant DB** • **Create database**	**2**	**High**	**Danush Hari Dhayabaran Bibin P Barathraj RS Gowtham P**
Sprint-4	**Application Building**	**USN-1**	**Building a web application that is integrated into the model we built. A UI is provided to the user where he has uploaded the image. Based on the saved model, the uploaded image will be analysed and prediction is showcased on the UI.**	**2**	**High**	**Danush Hari Dhayabaran Bibin P Barathraj RS Gowtham P**
Sprint-4	**Train The Model On IBM**	**USN-1**	**Build Deep learning model and computer vision Using the IBM cloud.**	**2**	**High**	**Danush Hari Dhayabaran Bibin P Barathraj RS Gowtham P**

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	28 Oct 2022	03 Nov 2022	20	05 Nov 2022
Sprint-2	20	5 Days	03 Nov 2022	08 Nov 2022	20	07 Nov 2022
Sprint-3	20	8 Days	08 Nov 2022	16 Nov 2022	20	16 Nov 2022
Sprint-4	20	4 Days	16 Nov 2022	20 Nov 2022	20	20 Nov 2022

7. PROJECT DEVELOPMENT PHASE

7.1 Project Development - Delivery of Sprint – 1

Index.php

```
<?php
session_start();
if
(isset($_SESSION['SESSION_EM
AIL'])) {        header("Location:
welcome.php");        die();
}

include 'config.php';
$msg = "";

if (isset($_GET['verification'])) {
if (mysqli_num_rows(mysqli_query($conn, "SELECT * FROM users1 WHERE
```

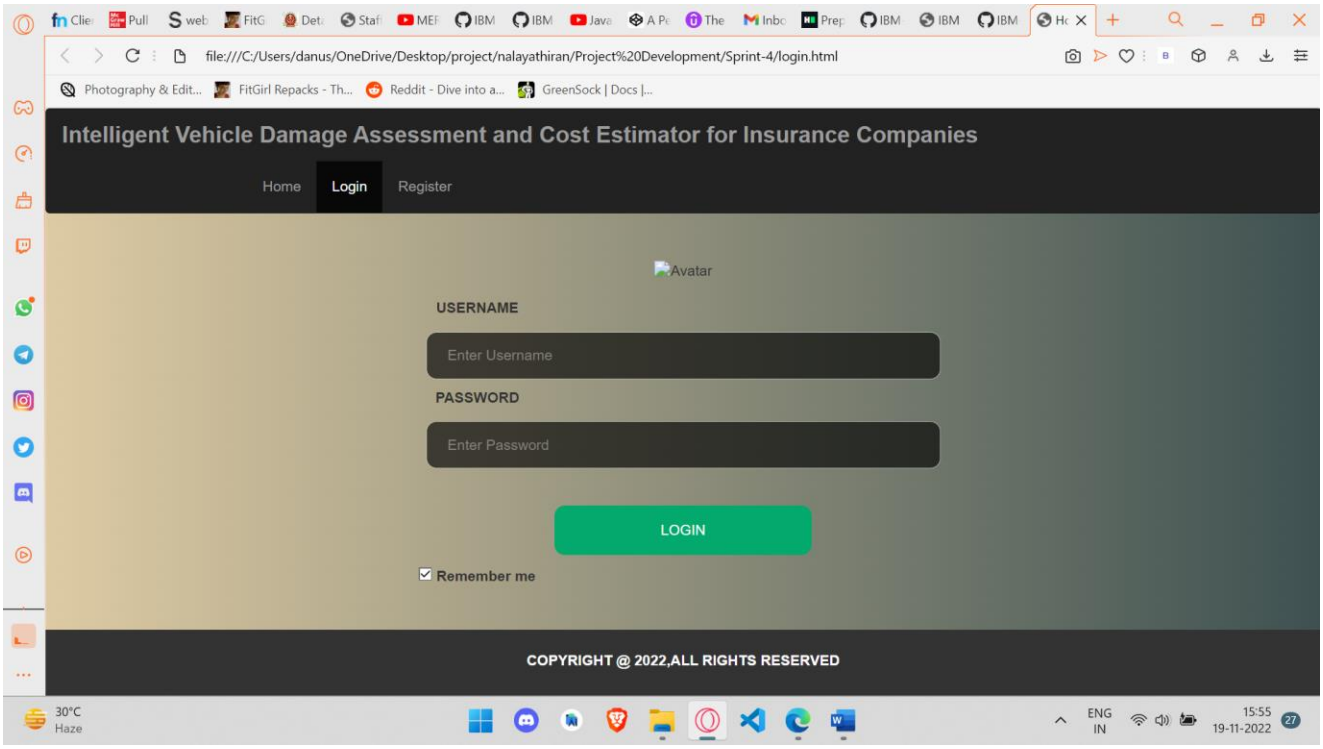
```
code='{$_GET['verification']}'")) > 0) {

    $query = mysqli_query($conn, "UPDATE users1 SET code=" WHERE
    code='{$_GET['verification']}'");


    if ($query) {

    $msg = "<div class='alert alert-success'>Account verification has been successfully
    completed.</div>";

    if (isset($_POST['submit'])) {

    $email = mysqli_real_escape_string($conn, $_POST['email']);

    $password = mysqli_real_escape_string($conn, md5($_POST['password']));


    $sql = "SELECT * FROM users1 WHERE email='{$email}' AND

    password='{$password}'";        $result = mysqli_query($conn, $sql);


    if (mysqli_num_rows($result) === 1) {

    $row = mysqli_fetch_assoc($result);


    if (empty($row['code'])) {

    $_SESSION['SESSION_EMAIL'] = $email;

    header("Location: welcome.php");

    } else {

    $msg = "<div class='alert alert-info'>First verify your account and try again.</div>";

    }

    } else {

    $msg = "<div class='alert alert-danger'>Email or password do not match.</div>";

    }

    }

    ?>
```

```
<!DOCTYPE html>
<html lang="zxx">

<head>
<title>Login Form - Damage Detection</title>
<!-- Meta tag Keywords -->
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta charset="UTF-8" />
<meta name="keywords"
content="Login Form" />
<!-- //Meta tag Keywords -->

<link
href="//fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600&display=swap"
rel="stylesheet">

<!--/Style-CSS -->
<link rel="stylesheet" href="css/style.css" type="text/css" media="all" />     <!--
//StyleCSS -->

<script src="https://kit.fontawesome.com/af562a2a63.js"
crossorigin="anonymous"></script>

</head>

<body>

<!-- form section start -->
<section class="w3l-mockup-form">
<div class="container">
```

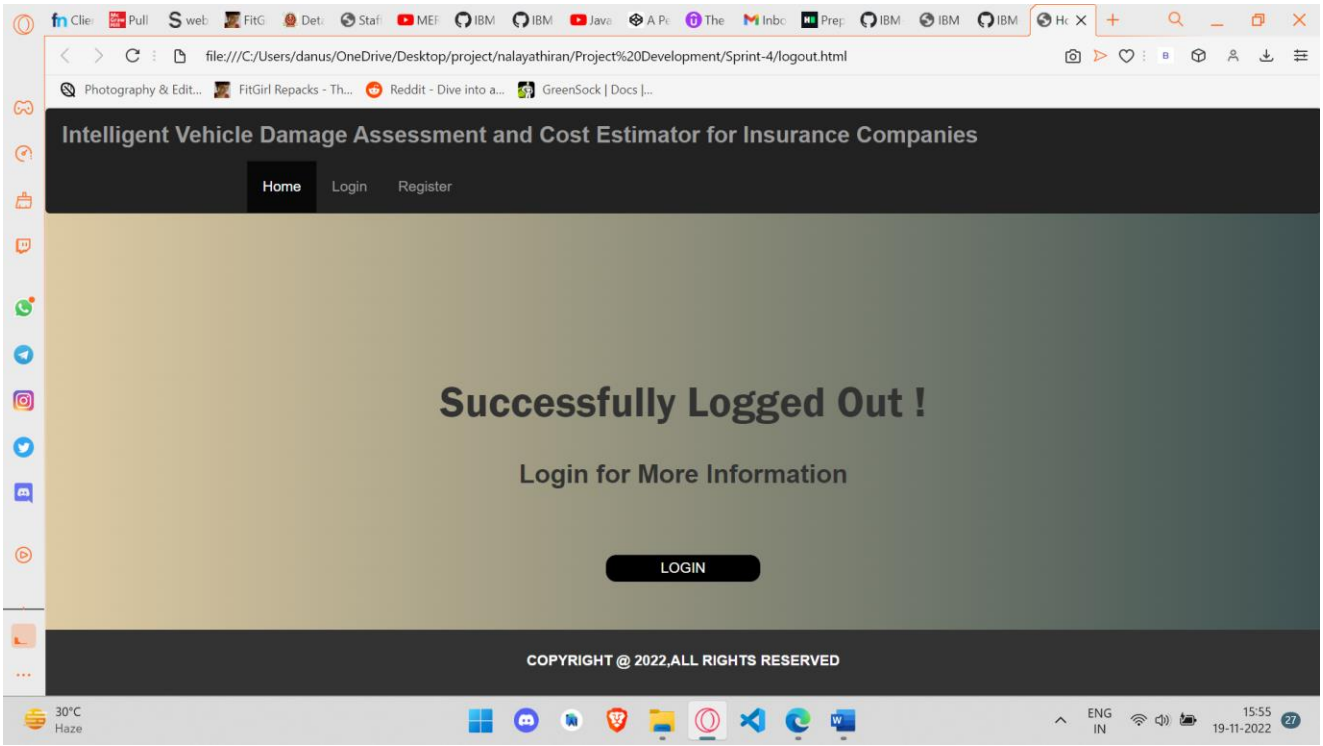
```
<!-- /form -->

<div class="workinghny-form-grid">

<div class="main-mockup">

<div class="alert-close">

<span class="fa fa-close"></span>

</div>

<div class="w3l_form align-self">

<div class="left_grid_info">
<img src="images/bsi1.png" alt="">

</div>
</div>

<div class="content-wthree">

<h2>Login Now</h2>

<?php echo $msg; ?>

<form action="" method="post">

<input type="email" class="email" name="email" placeholder="Enter Your Email" required>

<input type="password" class="password" name="password" placeholder="Enter Your Password" style="margin-bottom: 2px;" required>

<p><a href="forgot-password.php" style="margin-bottom: 15px; display: block; textalign: right;">Forgot Password?</a></p>

<button name="submit" name="submit" class="btn" type="submit">Login</button>

</form>

<div class="social-icons">

<p>Create Account! <a href="register.php">Register</a>.</p>

</div>

</div>

</div>

</div>

<!-- //form -->
```

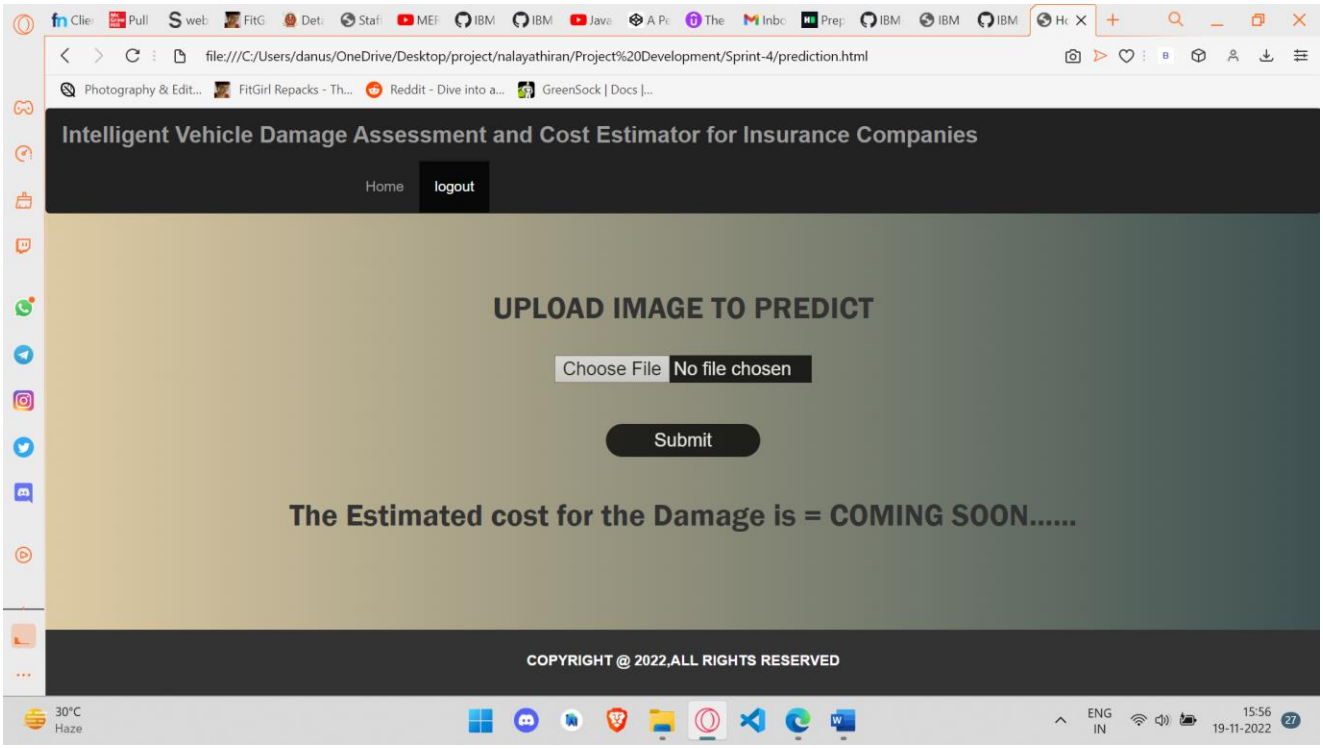
```
        </div>
        </section>
        <!-- //form section start -->

        <script src="js/jquery.min.js"></script>
        <script>
        $(document).ready(function (c) {
        $('.alert-close').on('click', function (c) {
        $('.main-mockup').fadeOut('slow', function (c) {
        $('.main-mockup').remove();
        });
        });
        });
        </script>

        </body>
</html>
```

Login.html



Logout.html





7.2 Project Development - Delivery of Sprint – 2

Image Pre-processing Click Here for Hyperlink

#Import The ImageDataGenerator Library:

```
# Import required lib
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

#Configure ImageDataGenerator Class :

```
#Creating augmentation on training variable
 train_datagen = ImageDataGenerator(rescale=1./255,
                    zoom_range=0.2,
                    horizontal_flip=True)

# Creating augmentation on testing variable test_datagen
= ImageDataGenerator(rescale=1./255)
```

#Apply ImageDataGenerator Functionality To Trainset And Testset :

For Body Damage:

```
# Passing training data to train variable for body
xtrain = train_datagen.flow_from_directory('/content/damage vehicle/body/training',
                                           target_size=(224,224),
                                           class_mode='categorical',
                                           batch_size=10)

# Passing testing data to test variable for body
xtest = test_datagen.flow_from_directory('/content/damage vehicle/body/validation',
                                         target_size=(224,224),
                                         class_mode='categorical', batch_size=10)
```

For Level Damage:

```
# Passing training data to train variable for body
x_train = train_datagen.flow_from_directory('/content/damage vehicle/level/training',
                                           target_size=(224,224),
                                           class_mode='categorical',
                                           batch_size=10)
# Passing training data to test variable for body
x_test = test_datagen.flow_from_directory('/content/damage vehicle/level/validation',
                                          target_size=(224,224),
                                          class_mode='categorical',
                                          batch_size=10)
```

7.3 Project Development - Delivery of Sprint – 3

Model Building :

Import and unzip the dataset

Team ID : PNT2022TMID09950

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#unzip the downloaded dataset
!unzip '/content/drive/MyDrive/damage vehicle.zip'
```

```
Archive:  /content/drive/MyDrive/damage vehicle.zip
creating:  damage  vehicle/       creating:  damage
vehicle/body/                  creating:    damage
vehicle/body/training/           creating:   damage
vehicle/body/training/00-front/   inflating: damage
```

```
vehicle/body/training/00-front/0001.jpeg
inflating:     damage     vehicle/body/training/00-
front/0002.JPEG                inflating:   damage
vehicle/body/training/00-front/0003.JPEG
inflating:     damage     vehicle/body/training/00-
front/0004.JPEG                inflating:   damage
vehicle/body/training/00-front/0005.JPEG
inflating:     damage     vehicle/body/training/00-
front/0006.JPEG                inflating:   damage
vehicle/body/training/00-front/0007.JPEG
inflating:     damage     vehicle/body/training/00-
front/0008.jpeg                inflating:   damage
vehicle/body/training/00-front/0009.JPEG
inflating:     damage     vehicle/body/training/00-
front/0010.JPEG                inflating:   damage
vehicle/body/training/00-front/0011.JPEG
inflating:     damage     vehicle/body/training/00-
front/0012.jpeg                inflating:   damage
vehicle/body/training/00-front/0013.JPEG
inflating:     damage     vehicle/body/training/00-
front/0014.JPEG                inflating:   damage
vehicle/body/training/00-front/0015.JPEG
inflating:     damage     vehicle/body/training/00-
front/0016.JPEG                inflating:   damage
vehicle/body/training/00-front/0017.JPEG
inflating:     damage     vehicle/body/training/00-
front/0018.JPEG                inflating:   damage
vehicle/body/training/00-front/0019.JPEG
inflating:     damage     vehicle/body/training/00-
front/0020.jpeg                inflating:   damage
vehicle/body/training/00-front/0021.JPEG
inflating:     damage     vehicle/body/training/00-
front/0022.JPEG                inflating:   damage
vehicle/body/training/00-front/0023.JPEG
inflating:     damage     vehicle/body/training/00-
front/0024.JPEG                inflating:   damage
vehicle/body/training/00-front/0025.jpeg
inflating:     damage     vehicle/body/training/00-
front/0026.JPEG                inflating:   damage
vehicle/body/training/00-front/0027.JPEG
inflating:     damage     vehicle/body/training/00-
front/0028.JPEG                inflating:   damage
vehicle/body/training/00-front/0029.JPEG
inflating:     damage     vehicle/body/training/00-
front/0030.JPEG                inflating:   damage
vehicle/body/training/00-front/0031.JPEG
inflating:     damage     vehicle/body/training/00-
front/0032.JPEG                inflating:   damage
vehicle/body/training/00-front/0033.JPEG
inflating:     damage     vehicle/body/training/00-
front/0034.JPEG                inflating:   damage
vehicle/body/training/00-front/0035.jpeg
inflating:     damage     vehicle/body/training/00-
front/0036.JPEG                inflating:   damage
vehicle/body/training/00-front/0037.JPEG
inflating:     damage     vehicle/body/training/00-
front/0038.JPEG                inflating:   damage
vehicle/body/training/00-front/0039.JPEG
inflating:     damage     vehicle/body/training/00-
front/0040.JPEG                inflating:   damage
vehicle/body/training/00-front/0041.JPEG
inflating:     damage     vehicle/body/training/00-
front/0042.JPEG                inflating:   damage
vehicle/body/training/00-front/0043.JPEG
inflating:     damage     vehicle/body/training/00-
front/0044.JPEG                inflating:   damage
vehicle/body/training/00-front/0045.JPEG
inflating:     damage     vehicle/body/training/00-
front/0046.jpeg                inflating:   damage
vehicle/body/training/00-front/0047.JPEG
inflating:     damage     vehicle/body/training/00-
front/0048.JPEG                inflating:   damage
vehicle/body/training/00-front/0049.JPEG
inflating:     damage     vehicle/body/training/00-
```

```
front/0050.JPEG                inflating:   damage
vehicle/body/training/00-front/0051.JPEG
inflating:     damage     vehicle/body/training/00-
front/0052.JPEG                inflating:   damage
vehicle/body/training/00-front/0053.JPEG   i fl ti
d hi l /b d /t i i /00 f t/00 j
```

Image Preprocessing

1. Import The ImageDataGenerator Library

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

2. Con gure ImageDataGenerator Class

```
#Creating augmentation on training variable
train_datagen =
ImageDataGenerator(rescale=1./255,
shear_range = 0.1,
zoom_range=0.1,
horizontal_flip=True)
```

```
# Creating augmentation on testing variable

test_datagen =

ImageDataGenerator(rescale=1./255)
```

3. Apply ImageDataGenerator Functionality To Trainset And Testset

```
# Passing training data to train variable for body

xtrain = train_datagen.flow_from_directory('/content/damage
vehicle/body/training',
target_size=(224,224),
class_mode='categorical',
batch_size=10)
```

```
Found 979 images belonging to 3 classes.
```

```
# Passing testing data to test variable for body

xtest = test_datagen.flow_from_directory('/content/damage
vehicle/body/validation',
target_size=(224,224),
class_mode='categorical',
batch_size=10)
```

```
Found 171 images belonging to 3 classes.
```

```
# Passing training data to train variable for level

x_train = train_datagen.flow_from_directory('/content/damage
vehicle/level/training',
target_size=(224,224),
class_mode='categorical',
batch_size=10)
```

```
Found 979 images belonging to 3 classes.
```

```
# Passing testing data to test variable for level

x_test = test_datagen.flow_from_directory('/content/damage
vehicle/level/validation',
```

```
target_size=(224,224),
class_mode='categorical',
batch_size=10)
```

```
Found 171 images belonging to 3 classes.
```

Model Building

For Body

1. Importing The Model Building Libraries

```
#Import the library
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 from glob import glob
```

```
import numpy as np
import matplotlib.pyplot as plt
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

```
IMAGE_SIZE = [224, 224]

train_path = '/content/damage vehicle/body/training' valid_path =
'/content/damage vehicle/body/validation'
```

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_ker 58889256/58889256 [==============================] - 3s
0us/step
```

3. Adding Flatten Layer

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

```
folders = glob('/content/damage vehicle/body/training/*')


folders
```

```
['/content/damage vehicle/body/training/00-front',
 '/content/damage vehicle/body/training/01-rear',
'/content/damage vehicle/body/training/02-side']
```

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

```
len(folders)
```

```
3
```

4. Adding Output Layer

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

4. Creating A Model Object

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

```
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

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

7. Train The Model

```
r = model.fit_generator(   xtrain,
validation_data=xtest,   epochs=25,
  steps_per_epoch=len(xtrain),
validation_steps=len(xtest)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit_generator` is deprecated
and will be


Epoch 1/25
98/98 [==============================] - 23s 146ms/step - loss: 1.2077 - accuracy: 0.5465 - val_loss: 1.2900 -
val_accuracy:
Epoch 2/25
98/98 [==============================] - 13s 128ms/step - loss: 0.8364 - accuracy: 0.7028 - val_loss: 0.8665 -
val_accuracy:
Epoch 3/25
98/98 [==============================] - 13s 128ms/step - loss: 0.5293 - accuracy: 0.7998 - val_loss: 1.3260 -
val_accuracy:
Epoch 4/25
98/98 [==============================] - 12s 127ms/step - loss: 0.3978 - accuracy: 0.8611 - val_loss: 0.9842 -
val_accuracy:
Epoch 5/25
98/98 [==============================] - 12s 127ms/step - loss: 0.2783 - accuracy: 0.9030 - val_loss: 0.9397 -
val_accuracy:
Epoch 6/25
98/98 [==============================] - 13s 128ms/step - loss: 0.2690 - accuracy: 0.9070 - val_loss: 0.9892 -
val_accuracy:
Epoch 7/25
98/98 [==============================] - 12s 127ms/step - loss: 0.1788 - accuracy: 0.9448 - val_loss: 1.0052 -
val_accuracy:
Epoch 8/25
98/98 [==============================] - 13s 129ms/step - loss: 0.1671 - accuracy: 0.9469 - val_loss: 1.1693 -
val_accuracy:
Epoch 9/25
98/98 [==============================] - 13s 129ms/step - loss: 0.1277 - accuracy: 0.9561 - val_loss: 1.0058 -
val_accuracy:
Epoch 10/25
98/98 [==============================] - 13s 128ms/step - loss: 0.1184 - accuracy: 0.9591 - val_loss: 1.0620 -
val_accuracy:
Epoch 11/25
98/98 [==============================] - 13s 130ms/step - loss: 0.0963 - accuracy: 0.9745 - val_loss: 1.1219 -
val_accuracy:
Epoch 12/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0857 - accuracy: 0.9765 - val_loss: 1.0284 -
val_accuracy:
Epoch 13/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0582 - accuracy: 0.9837 - val_loss: 1.1153 -
val_accuracy:
Epoch 14/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0688 - accuracy: 0.9877 - val_loss: 1.1033 -
val_accuracy:
Epoch 15/25
```

```
98/98 [==============================] - 13s 131ms/step - loss: 0.0709 - accuracy: 0.9867 - val_loss: 1.0730 -
val_accuracy:
Epoch 16/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0895 - accuracy: 0.9775 - val_loss: 1.1225 -
val_accuracy:
Epoch 17/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0609 - accuracy: 0.9918 - val_loss: 1.2937 -
val_accuracy:
Epoch 18/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0998 - accuracy: 0.9714 - val_loss: 1.1754 -
val_accuracy:
Epoch 19/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0728 - accuracy: 0.9847 - val_loss: 1.5074 -
val_accuracy:
Epoch 20/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0972 - accuracy: 0.9714 - val_loss: 1.4684 -
val_accuracy:
Epoch 21/25
98/98 [==============================] - 13s 131ms/step - loss: 0.0404 - accuracy: 0.9908 - val_loss: 1.4215 -
val_accuracy:
Epoch 22/25
98/98 [==============================] - 13s 131ms/step - loss: 0.0854 - accuracy: 0.9867 - val_loss: 1.4772 -
val_accuracy:
Epoch 23/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0399 - accuracy: 0.9918 - val_loss: 1.4306 -
val_accuracy:
Epoch 24/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0400 - accuracy: 0.9908 - val_loss: 1.4562 -
val_accuracy:
Epoch 25/25
98/98 [==============================] - 13s 129ms/step - loss: 0.1692 - accuracy: 0.9387 - val_loss: 1.6805 -
val_accuracy:
```

8. Save The Model

```
from tensorflow.keras.models import load_model

model.save('/content/damage vehicle/Model/body.h5')
```

9. Test The Model

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

```
model = load_model('/content/damage vehicle/Model/body.h5')
```

```
           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
import numpy as np
```

```
data = "/content/damage vehicle/body/training/00-
front/0002.JPEG" image = cv2.imread(data)
print(detect(image))
```

```
1/1 [==============================] - 0s
148ms/step front
```

Model Building

For Level

1. Importing The Model Building Libraries

```
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

```
IMAGE_SIZE = [224, 224]

train_path = '/content/damage vehicle/level/training' valid_path =
'/content/damage vehicle/level/validation'
```

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

3. Adding Flatten Layer

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

```
folders = glob('/content/damage vehicle/level/training/*')


folders
```

```
['/content/damage vehicle/level/training/03-severe',
 '/content/damage vehicle/level/training/02-moderate',
'/content/damage vehicle/level/training/01-minor']
```

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


len(folders)
```

```
3
```

4. Adding Output Layer

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

5. Creating A Model Object

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


model.summary()
```

```
Model: "model_1"
```

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param
#
=========================================================
=======  input_2 (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_1 (Flatten)          (None, 25088)             0
dense_1 (Dense)             (None, 3)                 75267
=========================================================
=== Total params: 14,789,955
Trainable params: 75,267
Non-trainable params: 14,714,688
_________________________________________________________________
```

6. Con gure The Learning Process

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

7. Train The Model

```
r = model.fit_generator(   x_train,
validation_data=x_test,   epochs=25,
  steps_per_epoch=len(x_train),
validation_steps=len(x_test)
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: `Model.fit_generator` is deprecated
and will be


Epoch 1/25
98/98 [==============================] - 14s 133ms/step - loss: 1.1629 - accuracy: 0.5495 - val_loss: 1.1559 -
val_accuracy:
Epoch 2/25
98/98 [==============================] - 13s 130ms/step - loss: 0.7157 - accuracy: 0.7089 - val_loss: 0.9643 -
val_accuracy:
Epoch 3/25
98/98 [==============================] - 13s 130ms/step - loss: 0.4978 - accuracy: 0.8161 - val_loss: 1.5663 -
val_accuracy:
Epoch 4/25
98/98 [==============================] - 13s 128ms/step - loss: 0.5277 - accuracy: 0.7865 - val_loss: 1.6003 -
val_accuracy:
Epoch 5/25
98/98 [==============================] - 13s 128ms/step - loss: 0.3763 - accuracy: 0.8468 - val_loss: 1.1925 -
val_accuracy:
Epoch 6/25
98/98 [==============================] - 13s 128ms/step - loss: 0.2445 - accuracy: 0.9203 - val_loss: 1.0354 -
val_accuracy:
Epoch 7/25
```

```
98/98 [==============================] - 13s 128ms/step - loss: 0.1902 - accuracy: 0.9346 - val_loss: 1.2155 -
val_accuracy:
Epoch 8/25
98/98 [==============================] - 13s 128ms/step - loss: 0.1327 - accuracy: 0.9571 - val_loss: 1.0902 -
val_accuracy:
Epoch 9/25
98/98 [==============================] - 13s 127ms/step - loss: 0.1206 - accuracy: 0.9540 - val_loss: 1.1282 -
val_accuracy:
Epoch 10/25
98/98 [==============================] - 13s 128ms/step - loss: 0.1181 - accuracy: 0.9591 - val_loss: 1.1311 -
val_accuracy:
Epoch 11/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0910 - accuracy: 0.9765 - val_loss: 1.1538 -
val_accuracy:
Epoch 12/25
98/98 [==============================] - 12s 127ms/step - loss: 0.0813 - accuracy: 0.9806 - val_loss: 1.2209 -
val_accuracy:
Epoch 13/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0603 - accuracy: 0.9857 - val_loss: 1.2545 -
val_accuracy:
Epoch 14/25
98/98 [==============================] - 12s 127ms/step - loss: 0.0474 - accuracy: 0.9949 - val_loss: 1.1609 -
val_accuracy:
Epoch 15/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0366 - accuracy: 0.9959 - val_loss: 1.1688 -
val_accuracy:
Epoch 16/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0493 - accuracy: 0.9888 - val_loss: 1.1850 -
val_accuracy:
Epoch 17/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0320 - accuracy: 0.9939 - val_loss: 1.1884 -
val_accuracy:
Epoch 18/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0363 - accuracy: 0.9939 - val_loss: 1.2897 -
val_accuracy:
Epoch 19/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0298 - accuracy: 0.9949 - val_loss: 1.2499 -
val_accuracy:
Epoch 20/25
98/98 [==============================] - 13s 130ms/step - loss: 0.0250 - accuracy: 0.9980 - val_loss: 1.2801 -
val_accuracy:
Epoch 21/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0329 - accuracy: 0.9959 - val_loss: 1.2366 -
val_accuracy:
Epoch 22/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0170 - accuracy: 1.0000 - val_loss: 1.2901 -
val_accuracy:
Epoch 23/25
98/98 [==============================] - 13s 130ms/step - loss: 0.0216 - accuracy: 1.0000 - val_loss: 1.2697 -
val_accuracy:
Epoch 24/25
98/98 [==============================] - 13s 128ms/step - loss: 0.0365 - accuracy: 0.9908 - val_loss: 1.4214 -
val_accuracy:
Epoch 25/25
98/98 [==============================] - 13s 129ms/step - loss: 0.0380 - accuracy: 0.9939 - val_loss: 1.4219 -
val_accuracy:
```

8. Save The Model

```
from tensorflow.keras.models import load_model

model.save('/content/damage vehicle/Model/level.h5')
```

9. Test The Model

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

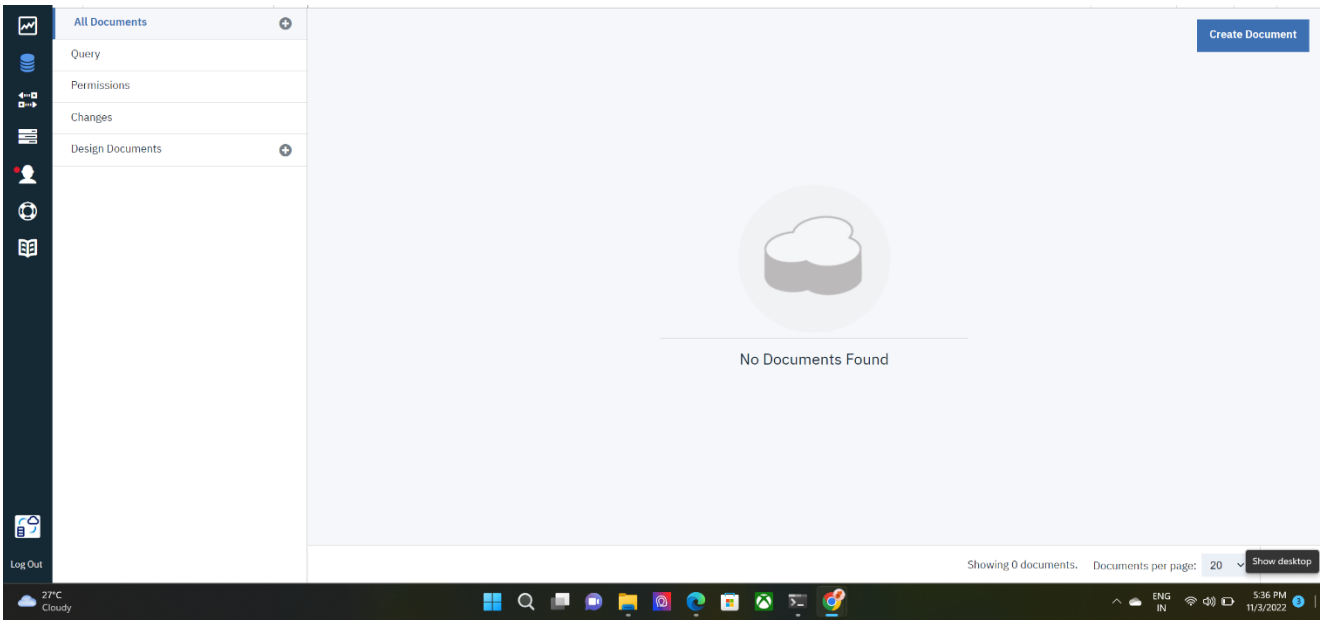
```
model = load_model('/content/damage vehicle/Model/level.h5')
```

```
            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 =
            ["minor","moderate","severe"]
            preds =
            label[np.argmax(prediction)]
            return preds
import numpy as np
```

```
data = "/content/damage vehicle/level/validation/01-
minor/0005.JPEG" image = cv2.imread(data)
print(detect(image))
```

```
     1/1 [==============================] - 0s
     142ms/step minor
```

Setting up cloud Database(IBM)



7.4. Project Development - Delivery of Sprint – 4

Connecting all the component to the Database:

App.py

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

```
render_template, redirect, url_for from cloudant.client
import Cloudant

#Create Database client = Cloudant.iam('00cba18f-2150-
4961-9102-f29b9aee35debluemix','ht_ByiEjrGeaitIZJTC-
ri5_8Oq-dxTNHLGho1mpt0d5', connect=True) my_database =
client.create_database('my_database')

#Loading the Model
model1 = load_model('Model/level.h5')
model2 = load_model('Model/body.h5')
 app =
Flask(__name__)

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

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

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

@app.route('/afterreg',
methods=['POST']) def afterreg():
x = [x for x in request.form.values()]
print(x)     data = {         '_id':
x[1],
        'name': x[0],
        'psw': x[2]
    }
print(data)
     query = {'_id': {'$eq':
data['_id']}}
```

```
     docs =
my_database.get_query_result(query)
print(docs)

print(len(docs.all()))

if(len(docs.all())==0):
        url = my_database.create_document(data)
response = request.get(url)          return
render_template('login.html', pred="Registration
Successful, Please login using your details")     else:
        return render_template('register.html', pred="You are
already a member, Please login using your details")

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

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

print(len(docs.all()))

if(len(docs.all())==0):
        return    render_template('login.html',    pred="The
Username is not found")     else:
        if((user==docs[0][0]['_id'] and
passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
else:
            print('Invalid User')
```

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

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

```
            value = "9000 - 11000 INR"
elif(result1 == "side" and result2 == "severe"):
value = "12000 - 15000 INR"         else:
            value = "16000 - 50000 INR"

return
render_template('prediction.html',prediction=value)
 if
__name__=="__main__"
:
    app.run(debug = False,port = 8080)
```

8.CONCLUSION

We conclude by suggesting this web application for damage assessment and cost estimation for the insurance companies. The web application is supported by the Deep Learning and IBM Watson cloud which stands for the complex image prediction and user information storage. The web application takes the user registration and login, The user can login into the prediction page using their ID and password. The prediction takes the image input and the model can predict the input based on the perviour knowledge about the damages. In future, The User Interface of the web application can be improved by updating the HTML and CSS codings. The improvement in UI can gives the better user exprience in future, The model's accuracy over various images can increased by trainning with various damaged images. The Image processing methods can be improved to achive higher performance of the model in the future.

9.REFERENCES

- https://www.youtube.com/watch?v=vI1PeKMwCZI
- https://www.researchgate.net/publication/341569399_Research_on_Intelligent_Vehicle_Damage_Assessment_System_Based_on_Computer_Vision
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwix6Kfzjbr7AhVm53MBHc1bAXYQFnoECB8QAQ&url=https%3A%2F%2Fwww.shaip.com%2Fsolutions%2Fvehicle-damage-assessment%2F&usg=AOvVaw3FbayHQUhfyt6NDo0QEY7_
- https://www.computerweekly.com/news/252480849/Insurance-company-uses-artificial-intelligence-to-assess-car-damage-when-claims-are-made