# PROJECT REPORT

| | |
|---|---|
| Team ID | PNT2022TMID03587 |
| Project Name | Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies |
| Team Members | 1.Jithesh Kumar R - 212219060118<br>2.Shree varshan M - 212219060247<br>3.Subramaniyan V - 212219020261<br>4.Udaya Krishna S - 212219060283 |

## 1.INTRODUCTION

### 1.1 PROJECT OVERVIEW

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

Car insurance settlement claims require near-perfect accuracy to avoid deceiving the customer in the process. The task of manually approving a claim completely reside on the staff who must be both trained and be equipped to evalute considering all detailed metrics.

## 1.2 PURPOSE

Analysing user perspective towards the given situation, further taking into consideration of the market demands and effectively determine the damage incurred by the vehicle. Thereby determining the cost to be paid by the user.

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. This model can also be used by lenders if they are underwriting a car loan, especially for a used car.

## 2.LITERATURE SURVEY

### 2.1. EXISTING PROBLEM

| **Paper** | **Drawbacks** | **PROPOSED METHODOLOGIES** | **OUTCOMES** |
|---|---|---|---|
| Image Based Automatic Vehicle Damage Detection | This thesis proposes a solution which uses 3D Computer Aided Design for the discernment of car damage from the picture, the system only detects damage at the edge | Monocular 2D/3D pose estimation<br>3D model-assisted segmentation<br>Reflection detection<br>Obtain reliable point correspondences across | The project explores the problem of automatically detecting mild damage in vehicles using photographs taken at the scene of the accident. |

| | | | |
|---|---|---|---|
| | portion only. Detection of car damage through CAD software requires some knowledge about the software. | photographs with largely reflective and homogeneous regions | |
| Car Damage Assessment Based on VGG Models | Observed that training with a small dataset is insufficient to get the best accuracy based on the deep learning approach.<br>Persistence of overfitting problem in the model performance | Deep learning-based algorithms, VGG16 and VGG19, for car damage detection and assessment<br>Pre-trained CNN models trained on ImageNet dataset<br>YOLO object detection to train and detect damage region<br>Transfer learning in pre-trained VGG model | 94%, 71% and 61% in damage detection, damage location and damage severity in VGG16<br>Comparison of VGG16 and VGG19 model Precision, Recall, and F!-score |
| Convolutional Neural Networks for vehicle | Challenge in damage inspection is the robustness against different | A damage detection model is developed to locate vehicle damages and classify these into | A deep learning model that is able to accurately detect and classify vehicle damages. |

| | | | |
|---|---|---|---|
| damage detection | light conditions | twelve categories.<br>FSSD with Darknet-53 and YOLO v3 with Darknet-53 yield the best mAP on, respectively. | The model is evaluated in a specially designed light street, indicating that strong reflections complicate the detection performance.<br>The model outperforms in the classes Bend and Cover Damage |
| Damage Assessment of a vehicle and Insurance Reclaim. | The major drawback of the proposed model is that it only identifies the physical visible damage and not the internal or the interior damage. | A technique that compares before-and after-accident car images to automatically detect the damaged location.<br>The R-CNN network identifies the severity of damage and a report is filed and sent to the user and the insurance firm. | The proportion of damaged parts is categorized and determining whether they need to be replaced or repaired. the user is aided in expediting the process of filing an insurance claim for his vehicle |

| | | | |
|---|---|---|---|
| Car Damage Assessment for Insurance Companies | Less number of epochs with increasing validation loss<br>Image Net dataset used limiting the diversity in the possibilities of damage detection | The following methods are used in the proposed system.<br>Dataset Explanation.<br>Describing the level of damage.<br>CNN Model.<br>VGG16 Algorithm. | In this proposed project a neural network-based solution for car detection; manage the problem of car damage analysis, prediction of car damage location and severity of the damage.<br>By simply sending the image of the car, the system will analyze 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. |

## 2.2. REFERENCES

| PAPER TITLE | AUTHOR - PUBLICATION |
|---|---|
| Image Based Automatic Vehicle Damage Detection | Srimal Jayawardena<br>A thesis submitted for the degree of Doctor of Philosophy at The Australian National University |
| Car Damage Assessment Based on VGG Models | Phyu Mar Kyu and Kuntpong Woraratpanya - Institute of Electrical and Electronics Engineers (IEEE)<br>Conference: JSCI8 |
| Convolutional Neural Networks for vehicle damage detection | R.E. van Ruitenbeek, S. Bhulai<br>Machine Learning with Applications<br>Volume 9, 15 September 2022, 100332 |
| Damage Assessment of a vehicle and Insurance Reclaim. | Vaibhav Agarwal, Utsav Khandelwal, Shivam Kumar, Raja Kumar, Shilpa M<br>2022 IJCRT \| Volume 10, Issue 4 April 2022 \| ISSN: 2320-2882 |

| | |
|---|---|
| Car Damage Assessment for Insurance Companies | Mandara G S1 and Prashant Ankalkoti2<br>PG Student, Department of Master of Computer Application1 Assistant Professor, Department of Master of Computer Application2 J N N College of Engineering, Shimoga, India<br>International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) |

## 2.3 PROBLEM STATEMENT DEFINITION

- 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.
- Maintaining a large set of trained models on multiple devices is a costly endeavor, especially for insurance companies with small budgets. These challenges render traditional computer vision-based damage assessment systems unsuitable for use by insurance companies, forcing them to rely instead on less accurate and less accurate algorithms or to abstain from using such systems altogether.
- 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

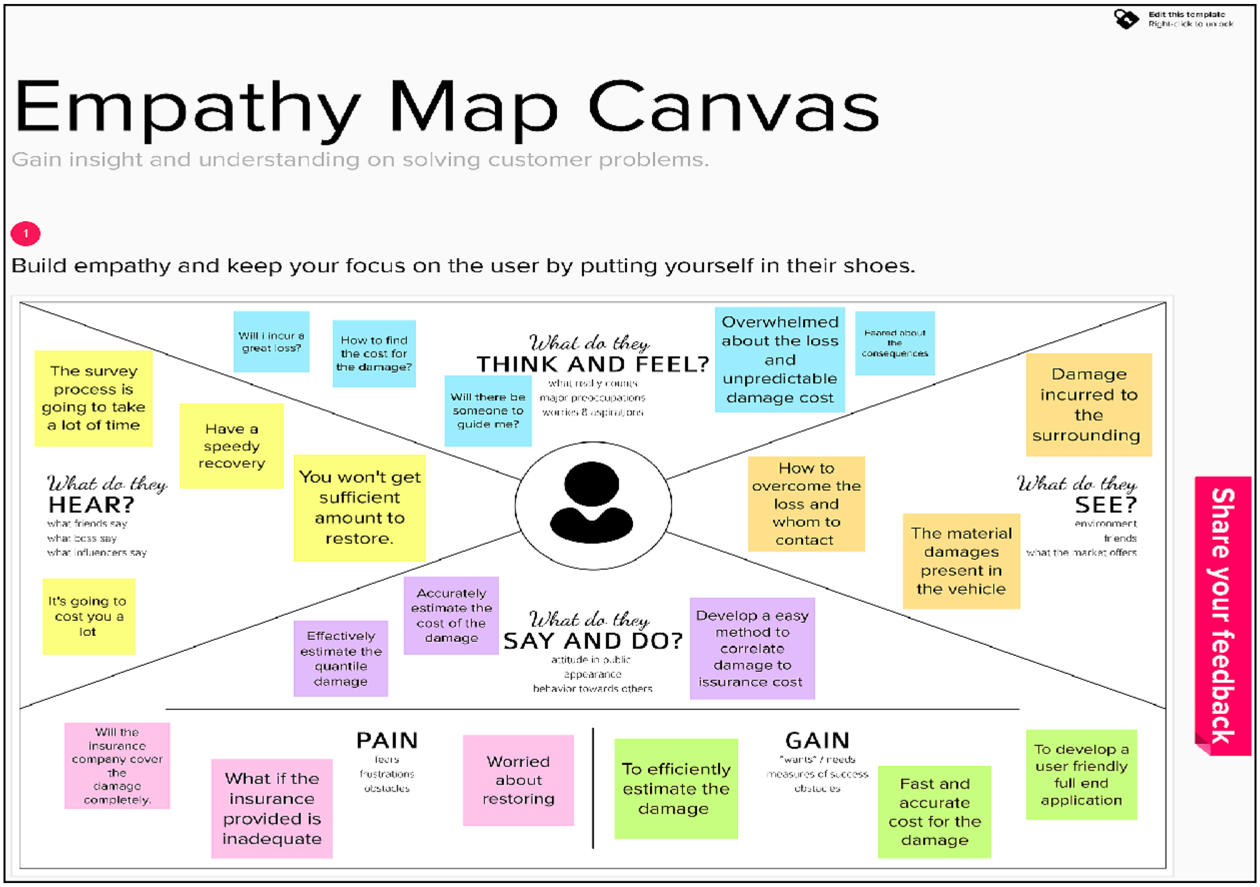
- While the computer can avoid human errors, there are often situations that would require such a model to flag for human assistance.
- The task of manually approving or disputing a claim falls on staff who must be both well-trained and well-equipped to deal with a variety of situations, both expected 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.
- As the volume of claims increases, the probability of mistakes increases as well.

## 3.IDEATION AND PROPOSED SOLUTION

### 3.1 EMPATHY MAP CANVAS

An empathy map is a collaborative tool team can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment.

The team discussed the pain and gains from various user scenarios and then the thoughts are listed in the canvas based on the pain ,gain,see,hear,say ,do, think and feel



Empathy Map Canvas

Gain insight and understanding on solving customer problems.

Build empathy and keep your focus on the user by putting yourself in their shoes.

What do they THINK AND FEEL?

what really counts
major preoccupations
worries & aspirations

Will i incur a great loss?

How to find the cost for the damage?

Will there be someone to guide me?

Overwhelmed about the loss and unpredictable damage cost

Feared about the consequences

What do they HEAR?

what friends say
what boss say
what influencers say

The survey process is going to take a lot of time

Have a speedy recovery

You won't get sufficient amount to restore.

It's going to cost you a lot

What do they SEE?

environment
friends
what the market offers

Damage incurred to the surrounding

How to overcome the loss and whom to contact

The material damages present in the vehicle

What do they SAY AND DO?

attitude in public
appearance
behavior towards others

Effectively estimate the quantile damage

Accurately estimate the cost of the damage

Develop a easy method to correlate damage to issurance cost

PAIN

fears
frustrations
obstacles

Will the insurance company cover the damage completely.

What if the insurance provided is inadequate

Worried about restoring

GAIN

"wants" / needs
measures of success
obstacles

To efficiently estimate the damage

Fast and accurate cost for the damage

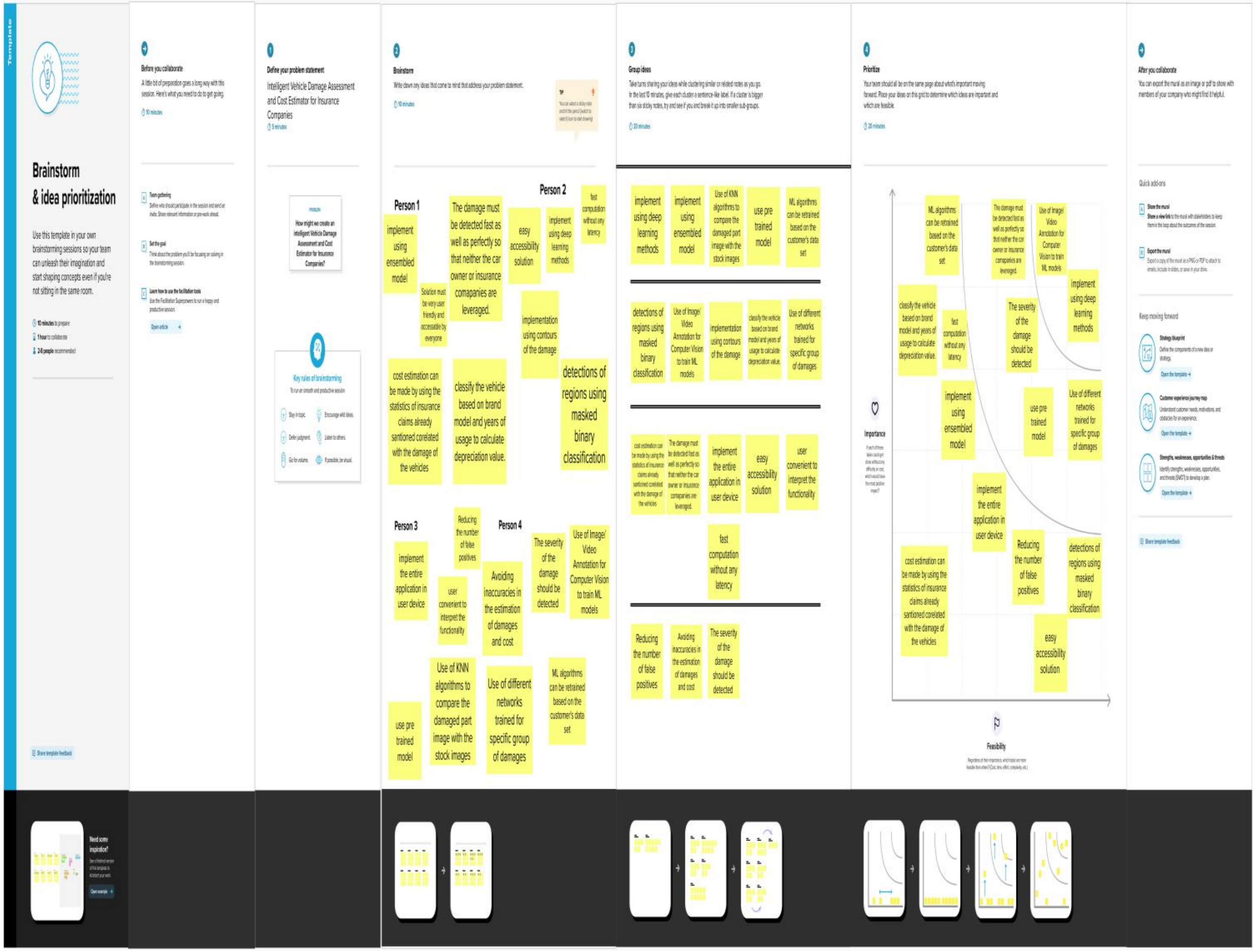
To develop a user friendly full end application

Figure 1:Empathy canvas map

## 3.2 IDEATION AND BRAINSTORMING :

Brainstorming is a great way to generate many ideas by leveraging the collective thinking of the group, engaging with each other, listening, and building on other ideas.

The team was gathered for a meeting and individual ideas about the project are collected , then the similar ideas are grouped,later based on the importance and feasibility of the idea, the ideas are posted in curve.



**Brainstorm & idea prioritization**

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

Need some inspiration?
See a finished version of this template to kickstart your work.
Open example →

**Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

**1 Define your problem statement**

Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies

5 minutes

PROBLEM
How might we create an Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies?

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.

**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!

Person 1

implement using ensembled model

Solution must be very user friendly and accessable by everyone

The damage must be detected fast as well as perfectly so that neither the car owner or insurance comapanies are leveraged.

cost estimation can be made by using the statistics of insurance claims already santioned corelated with the damage of the vehicles

classify the vehicle based on brand model and years of usage to calculate depreciation value.

Person 2

easy accessibility solution

implement using deep learning methods

fast computation without any latency

implementation using contours of the damage

detections of regions using masked binary classification

Person 3

implement the entire application in user device

Reducing the number of false positives

user convenient to interpret the functionality

use pre trained model

Use of KNN algorithms to compare the damaged part image with the stock images

Person 4

Avoiding inaccuracies in the estimation of damages and cost

The severity of the damage should be detected

Use of Image/ Video Annotation for Computer Vision to train ML models

Use of different networks trained for specific group of damages

ML algorithms can be retrained based on the customer's data set

**3 Group ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, 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

implement using deep learning methods

implement using ensembled model

Use of KNN algorithms to compare the damaged part image with the stock images

use pre trained model

ML algorithms can be retrained based on the customer's data set

detections of regions using masked binary classification

Use of Image/ Video Annotation for Computer Vision to train ML models

implementation using contours of the damage

classify the vehicle based on brand model and years of usage to calculate depreciation value.

Use of different networks trained for specific group of damages

cost estimation can be made by using the statistics of insurance claims already santioned corelated with the damage of the vehicles

The damage must be detected fast as well as perfectly so that neither the car owner or insurance comapanies are leveraged.

implement the entire application in user device

easy accessibility solution

user convenient to interpret the functionality

fast computation without any latency

Reducing the number of false positives

Avoiding inaccuracies in the estimation of damages and cost

The severity of the damage should be detected

**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

Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

ML algorithms can be retrained based on the customer's data set

The damage must be detected fast as well as perfectly so that neither the car owner or insurance comapanies are leveraged.

Use of Image/ Video Annotation for Computer Vision to train ML models

implement using deep learning methods

classify the vehicle based on brand model and years of usage to calculate depreciation value.

fast computation without any latency

The severity of the damage should be detected

implement using ensembled model

use pre trained model

Use of different networks trained for specific group of damages

implement the entire application in user device

cost estimation can be made by using the statistics of insurance claims already santioned corelated with the damage of the vehicles

Reducing the number of false positives

detections of regions using masked binary classification

easy accessibility solution

**After you collaborate**

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A Share the mural
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B Export the mural
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

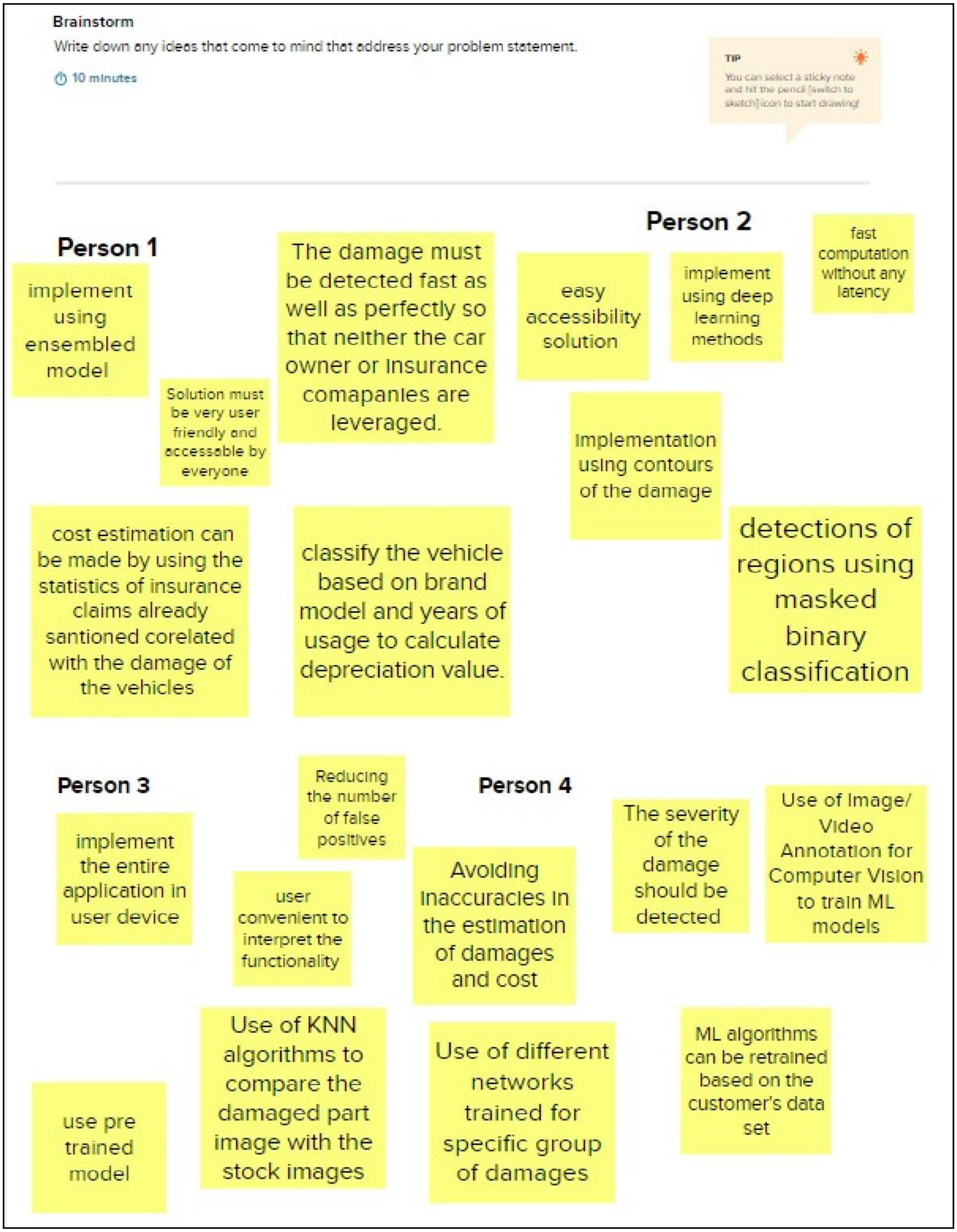
Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback

Figure 2: Ideation and Brainstroming

BRAINSTORM:



**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!

**Person 1**

implement using ensembled model

Solution must be very user friendly and accessable by everyone

The damage must be detected fast as well as perfectly so that neither the car owner or insurance comapanies are leveraged.

cost estimation can be made by using the statistics of insurance claims already santioned corelated with the damage of the vehicles

classify the vehicle based on brand model and years of usage to calculate depreciation value.

**Person 2**

easy accessibility solution

implement using deep learning methods

fast computation without any latency

implementation using contours of the damage

detections of regions using masked binary classification

**Person 3**

implement the entire application in user device

Reducing the number of false positives

user convenient to interpret the functionality

use pre trained model

Use of KNN algorithms to compare the damaged part image with the stock images

**Person 4**

Avoiding inaccuracies in the estimation of damages and cost

The severity of the damage should be detected

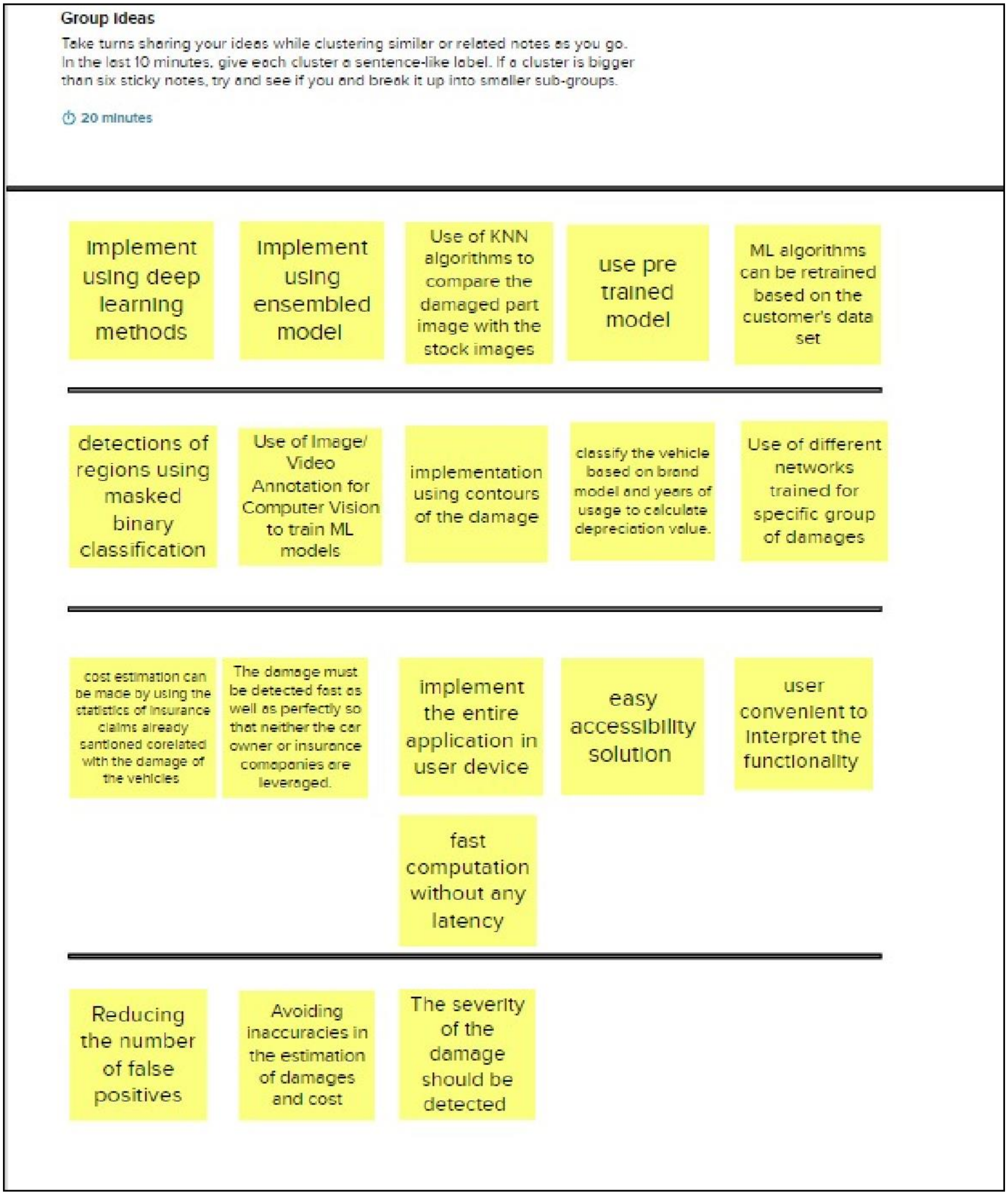
Use of Image/ Video Annotation for Computer Vision to train ML models

Use of different networks trained for specific group of damages

ML algorithms can be retrained based on the customer's data set

Figure 3:Brainstorming

## GROUP IDEAS:



**Group Ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, 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

Implement using deep learning methods

Implement using ensembled model

Use of KNN algorithms to compare the damaged part image with the stock images

use pre trained model

ML algorithms can be retrained based on the customer's data set

detections of regions using masked binary classification

Use of Image/ Video Annotation for Computer Vision to train ML models

implementation using contours of the damage

classify the vehicle based on brand model and years of usage to calculate depreciation value.

Use of different networks trained for specific group of damages

cost estimation can be made by using the statistics of insurance claims already santioned corelated with the damage of the vehicles

The damage must be detected fast as well as perfectly so that neither the car owner or insurance comapanies are leveraged.

implement the entire application in user device

easy accessibility solution

user convenient to interpret the functionality

fast computation without any latency

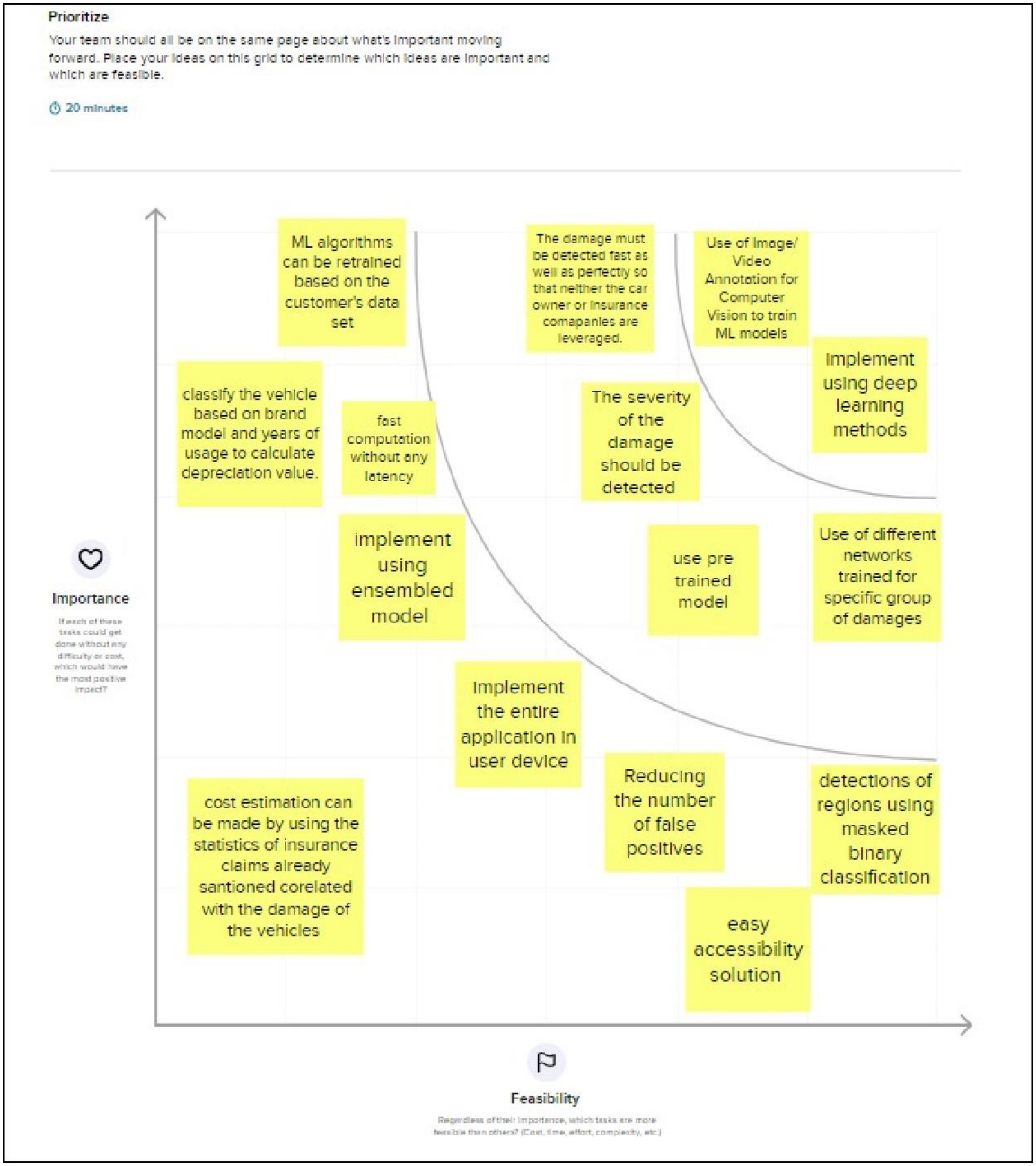
Reducing the number of false positives

Avoiding inaccuracies in the estimation of damages and cost

The severity of the damage should be detected

Figure 4:Grouping of ideas

PRIORITIZE:



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

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

ML algorithms can be retrained based on the customer's data set

The damage must be detected fast as well as perfectly so that neither the car owner or insurance comapanies are leveraged.

Use of Image/ Video Annotation for Computer Vision to train ML models

Implement using deep learning methods

classify the vehicle based on brand model and years of usage to calculate depreciation value.

fast computation without any latency

The severity of the damage should be detected

implement using ensembled model

use pre trained model

Use of different networks trained for specific group of damages

Implement the entire application in user device

Reducing the number of false positives

detections of regions using masked binary classification

cost estimation can be made by using the statistics of insurance claims already santioned corelated with the damage of the vehicles

easy accessibility solution

Figure 5: Prioritizing of ideas

## 3.3 PROPOSED SOLUTION:

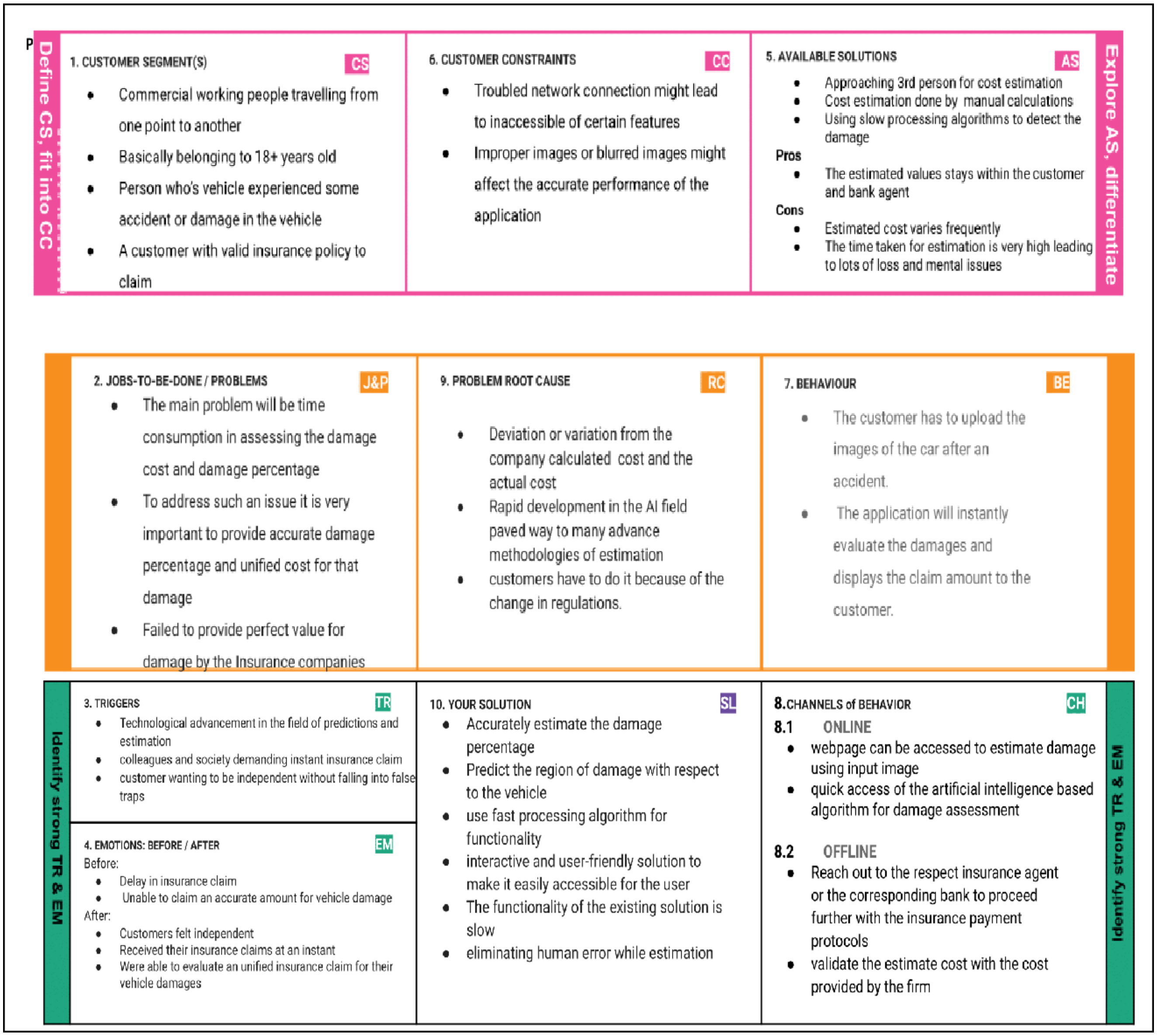
| S.No. | Parameter | Description |
|---|---|---|
| 1. | Problem Statement (Problem to be solved) | Every asset has a value attached to it that is primarily economic in nature. There is always a risk of these assets being destroyed due to incidents beyond human control. They also may not work due to such events. Depending on the asset class, the type and weight of risk also vary. This is where insurance policies are useful. The problem that might arise is that the claimant may not know the amount of coverage that he/she has. |
| 2. | Idea / Solution description | 1. To develop an optimized and accurate deep learning architecture to detect the damage percentage and location of the damage with respect to the vehicle<br>2. Implementing classification algorithms to classify damaged regions and implementing the model in web based application<br>3. Create a user accessible portal and securely store the data provided by the user<br>4. Compare the obtained damage percentage with the statistical cost estimation value to predict the cost. |

| | | |
|---|---|---|
| 3. | Novelty / Uniqueness | 1. The deep learning algorithm will analyse images in real time and identifies the presence of any damage.<br>2. Even in the presence of minute damages, artificial intelligence can detect the dents and marks on the car’s body.<br>3. With a lot of training, Artificial intelligence will able to distinguish simple stain from a scratch and effectively estimate the respective damage cost |
| 4. | Social Impact / Customer Satisfaction | 1. All the features of this project will be made easily accessible to the customers.<br>2. The web-app is intuitive, easy to use, simple and that the customer can rely on the product. It is easy to start with the app and understand how to use it, high complexity is not valuable for the user.<br>3. All the uploaded images will be and the personal information of the customer will be secured in cloud data security.<br>4. The cost estimation for damages that the web-app provides to the customer will be legitimate and exact to what a normal insurance company offers. |

|  |  |  |
|---|---|---|
| 5. | Business Model (Revenue Model) | 1. The business model will be a freemium model providing the prediction of damage intensity which will be useful for the vehicle owners to keep track of their vehicle damage and the credentials to access the webpage can be provided on the purchase of the vehicle insurance.<br>2. The add-on subscription model can be initiated for the user where the damage cost is evaluated and provided to the users.<br>3. The further revenue can be generated by tying up with the automobile parts manufacturers and distributors by promoting their products to the vehicle that has specified parts damaged. |
| 6. | Scalability of the Solution | 1. The damage detection can be provided to all the insured clients to reach the stable base and then extend the service of cost estimation to the insurers.<br>2. Make use of advanced machine learning techniques to analyse the damaged vehicle with high accuracy levels and keep on improving the learning ability of the model.<br>3. In addition to the webpage a mobile application can be created where the real time images and videos of the vehicle can be extracted and insurance cost can be estimated. |

## 3.4. PROBLEM SOLUTION FIT:

Problem-solution fit is a term used to describe the point validating that the base problem resulting in a business idea really exists and the proposed solution actually solves that problem



P

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- 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 estimation is very high leading to lots of loss and mental issues

Explore AS, differentiate

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 an issue it is very important to provide accurate damage percentage and unified cost for that damage
- Failed to provide perfect value for damage by the Insurance 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 application will instantly evaluate the damages and displays the claim amount to the customer.

Identify strong TR & EM

3. TRIGGERS TR

- Technological advancement in the field of predictions and estimation
- 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 their vehicle damages

10. YOUR SOLUTION SL

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

8.CHANNELS of BEHAVIOR CH

8.1 ONLINE

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

8.2 OFFLINE

- Reach out to the respect insurance agent or 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

Figure 6:Problem Solution Fit

## 4. REQUIREMENT ANALYSIS:

### 4.1. FUNCTIONAL REQUIREMENTS:

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | User Registration | Registration through Form<br>Registration through Gmail<br>Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email<br>Confirmation via OTP |
| FR-3 | User Interface | User friendly and simple website |
| FR-4 | Collection of datasets | Information about the user and their vehicle.<br>Information about Insurance plans. |
| FR-5 | Results | Model should be trained with high accuracy.<br>Results obtained from the model should be displayed to the user with easy interpretability. |

### 4.2 NON FUNCTIONAL REQUIREMENTS:

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | Usability | Intelligent model to assess the damage in the vehicle and estimate the cost to be provided by the insurance company. |
| NFR-2 | Security | The authenticity of the user and the confidentiality of the user details about their vehicle should be maintained. |

| | | |
|---|---|---|
| NFR-3 | Reliability | This project should be able to achieve good accuracy in damaging assessment as well in cost estimation so that the user is provided with the accurate and unbiased insurance amount. |
| NFR-4 | Performance | The real time images should be captured and uploaded into the website where the proposed model will carry out the damage assessment and give the cost of insurance accordingly. |
| NFR-5 | Availability | The webpage should be compatible for the web browsers in both mobile phones and computers. |
| NFR-6 | Scalability | The proposed solution will be scalable in future because of the efficient and quicker analysis and exact cost prediction |

## 5.PROJECT DESIGN:

### 5.1 DATA FLOW DIAGRAMS:

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement

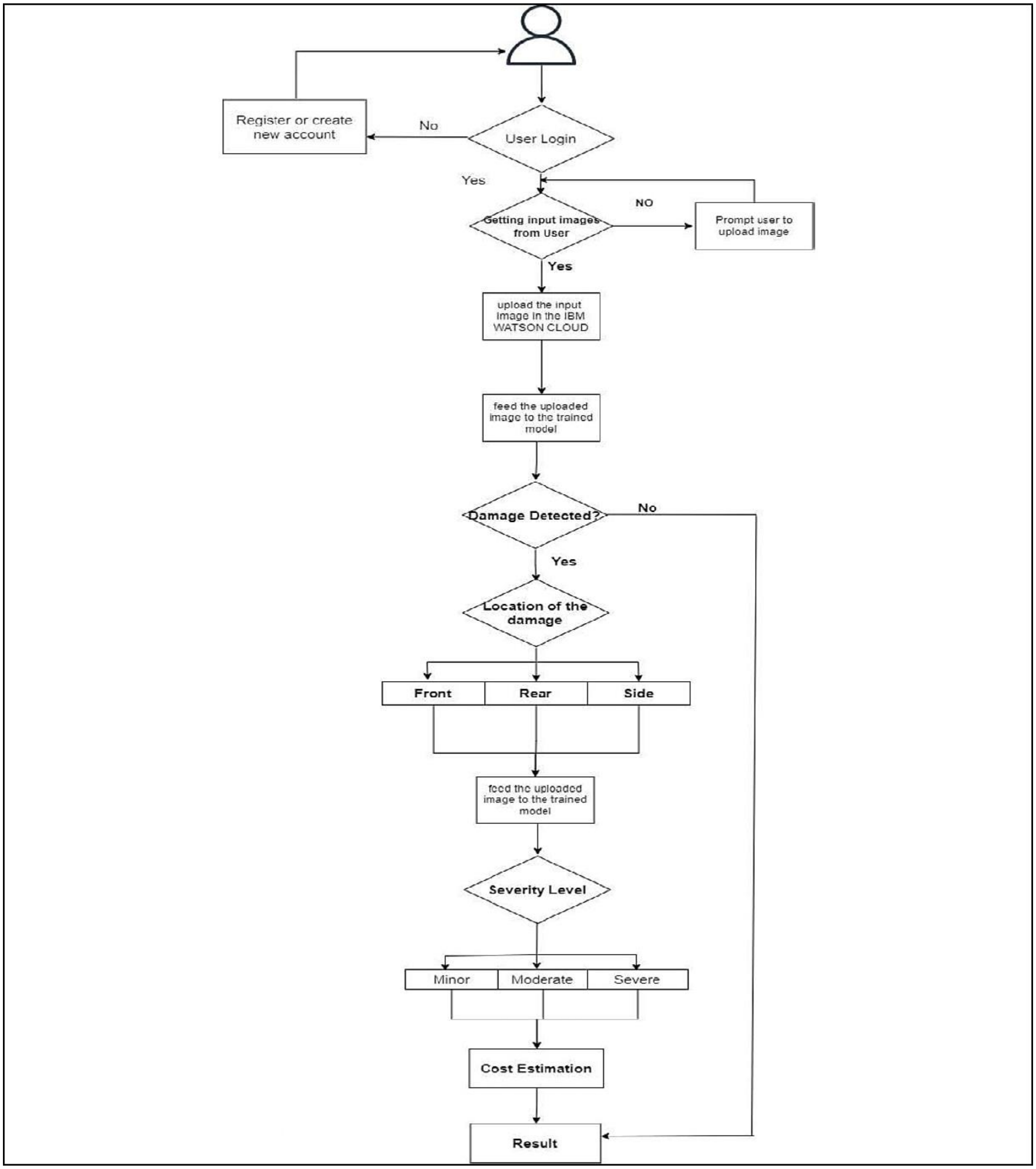

Register or create new account
No
User Login
Yes
NO
Getting input images from User
Prompt user to upload image
Yes
upload the input image in the IBM WATSON CLOUD
feed the uploaded image to the trained model
Damage Detected?
No
Yes
Location of the damage
Front
Rear
Side
feed the uploaded image to the trained model
Severity Level
Minor
Moderate
Severe
Cost Estimation
Result

Figure 7:Data flow diagram

## 5.2 SOLUTION AND TECHNICAL ARCHITECTURE:

SOLUTION ARCHITECTURE:

An architecture diagram depicts the conceptual model for a system; in our case, the system is the interconnected technology components creating a solution architecture. A conceptual model's primary objective is to convey the fundamental principles and basic functionality of the system which it represents.

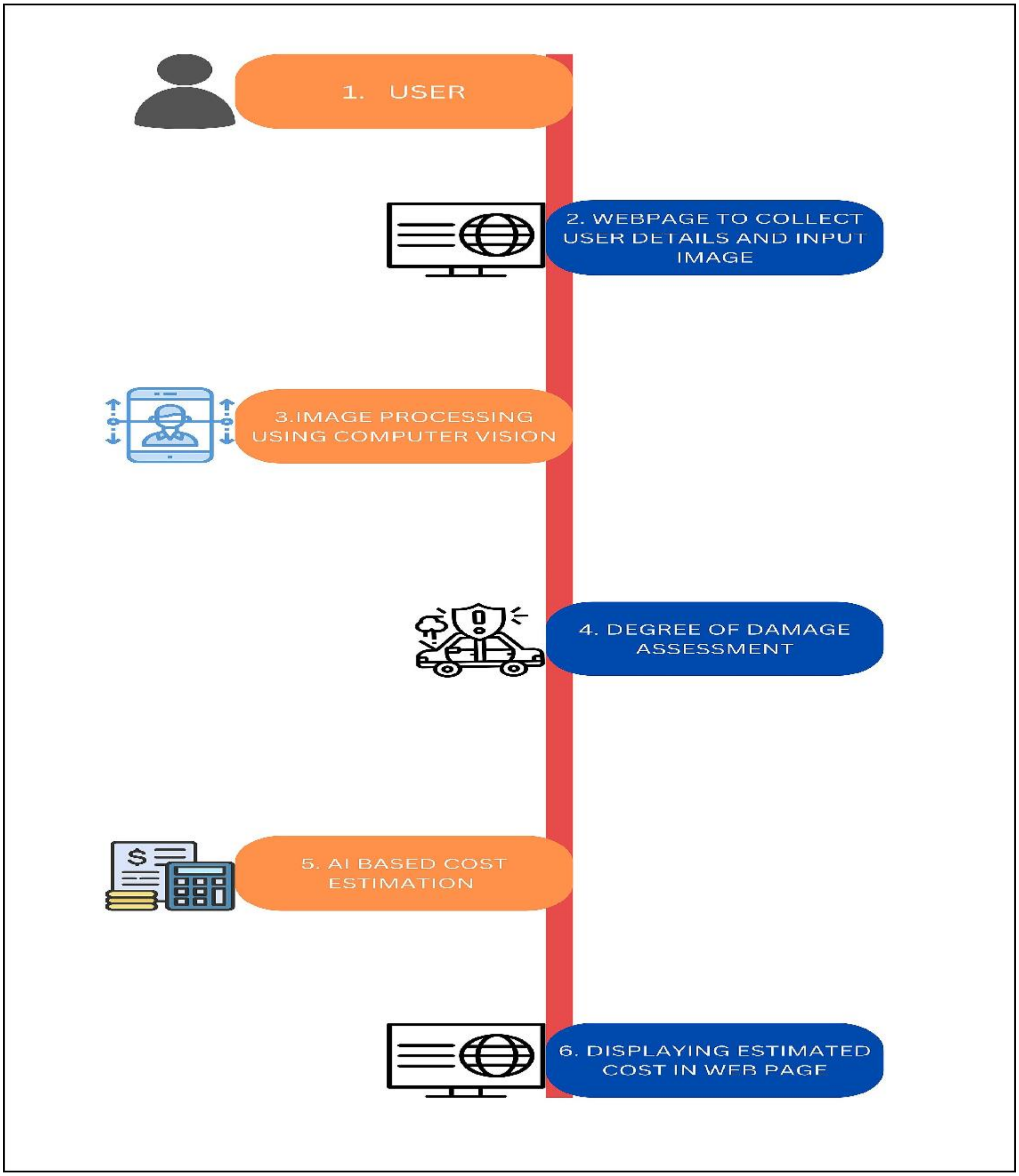

1. USER
2. WEBPAGE TO COLLECT USER DETAILS AND INPUT IMAGE
3.IMAGE PROCESSING USING COMPUTER VISION
4. DEGREE OF DAMAGE ASSESSMENT
5. AI BASED COST ESTIMATION
6. DISPLAYING ESTIMATED COST IN WEB PAGE

Figure 8:Solution Architecture

## TECHNICAL ARCHITECTURE:

Technical architecture includes the major components of the system, their relationships, and the contracts that define the interactions between the components. The goal of technical architects is to achieve all the business needs with an application that is optimized for both performance and security.

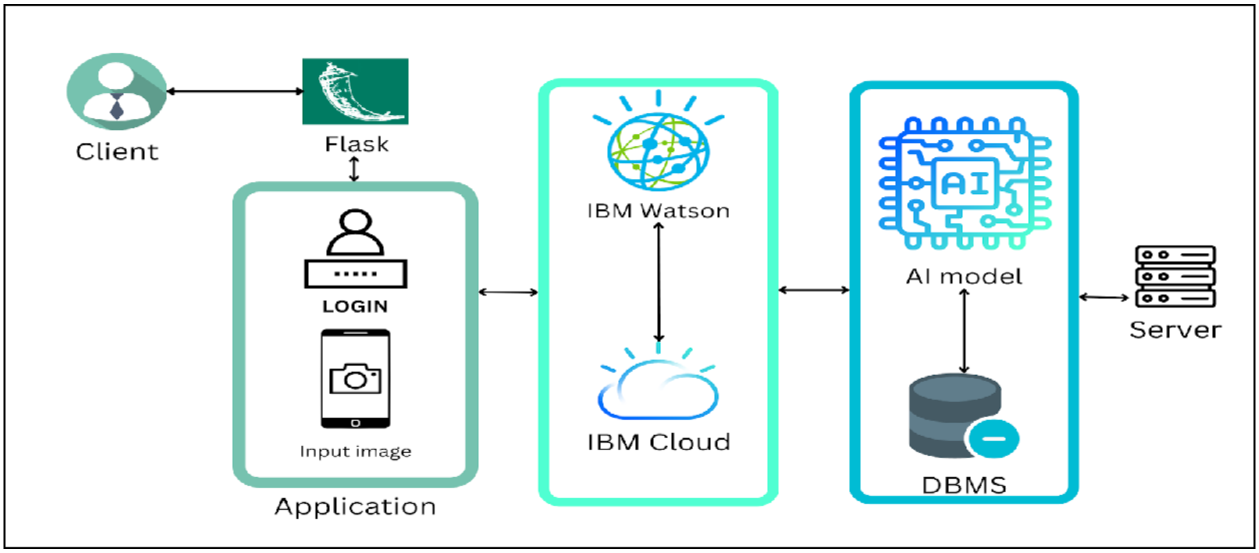


Figure 9:Technical Architecture

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 user input image | Python |
| 3. | Application Logic-2 | Getting model output for damage prediction | IBM Watson, Python |
| 4. | Application Logic-3 | Getting model output for cost estimation | IBM Watson, Python |

| | | | |
|---|---|---|---|
| 5. | Database | Data Type – Images and user inputs details are stored | MySQL, Js, IBM DB2 |
| 6. | Cloud Database | Database Service on Cloud | IBM DB2, IBM Cloudant etc. |
| 7. | File Storage | Received user details and received user input images of the vehicle is stored in cloud | IBM Block Storage, IBM cloud |
| 8. | Machine Learning Model | Purpose of the AI Model is for estimating the cost of the damaged vehicle. | Object Recognition Model, and CNN based model for damage estimation |
| 9. | Infrastructure (Server / Cloud) | On cloud server we will be deploying the AI Model using flask in the web page | Python Flask |

Table-2: Application Characteristics:

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1. | Open-Source Frameworks | Open-source frameworks used is IBM Watson | Technology of Open Source framework-IBM Watson |
| 2. | Security Implementations | IBM Cloud | Certified Watson assistant for Encrypted file systems, Encrypted storage systems, Key management systems. |

| | | | |
|---|---|---|---|
| 3. | Scalable Architecture | Web server - static and dynamic website content present in the website will be update based upon user demands and suggestion<br>Application server - updation of the basic functionality of the website and integration of new logic within the website can be done.Database server - based upon the varying inputs given 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 Cloud assistance |
| 5. | Performance | IBM Watson –automate processes, The deep learning model is trained using IBM Watson studio for better performance and quick accessibility . | IBM Watson Assistant |

## 5.3 USER STORIES

Use the below template to list all the user stories for the product.

| 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, and password, and confirming my password. | I can access my account/dashboard by entering valid credentials | High | Sprint-1 |
| Customer Details | Login | USN-2 | As a user, I will receive a confirmation email once I have registered for the application | I can receive a 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-4 |
| Customer Options | Details about insura | USN-4 | As a user, I can register for the application through | I can register & access the dashboard | Medium | Sprint-1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | nce companies | | Gmail | with Facebook Gmail | | |
| Customer usage | Login and repeated usage | USN-5 | As a user, I can log into the application by entering email & password | I can log in and view my dashboard at my demand on any time | High | Sprint-1 |
| Customer needs to do | web page | USN-6 | As a user I must capture images of my vehicle and upload it into the web portal. | I can capture the entire vehicle and upload | High | Sprint-2 |
| Customer (Web user) value | Details about estimated cost based on damage | USN-7 | As a user I must receive a detailed report of the damages present in the vehicle and the cost estimated | I can get the estimated insurance cost | High | Sprint-3 |
| Customer Care Executive | Provide friendly and efficient customer support and | USN-8 | As a user, I need to get support from developers in case of queries and failure of service provided | I can have smooth user experiences and all the issues raised is sorted | Medium | Sprint-4 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | sort out the queries | | | | | |
| Administrator | Overview the entire process and act as a bridge between user and developers | USN-9 | We need to satisfy the customer needs in an efficient way and make sure any sort of errors are fixed | I can finish the work without any problems | High | Sprint-4 |

# 6.PROJECT PLANNING & SCHEDULING

## 6.1 SPRINT PLANNING & ESTIMATION

Product Backlog, Sprint Schedule, and Estimation

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, and password, and confirming my password. | 2 | High | team member 2, team member 4 |
| Sprint-1 | Login | USN-2 | As a user, I will receive a confirmation email once I have registered for the application | 1 | High | team member 3 and team member 4 |
| Sprint-1 | Dashboard | USN-3 | As a user, I can register for the application through Facebook | 1 | High | team member 1 and team member 3 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Sprint-2 | Details about insurance company | USN-4 | As a user, I can register for the application through Gmail |  | low | team member 3 |
| Sprint-1 | repeated logins and logout | USN-5 | As a user, I can log into the application by entering email & password |  | medium | team member 1 and team member 2 |
| Sprint-2 | Webpage | USN-6 | As a user I must capture images of my vehicle and upload it into the web portal. |  | high | team member 4 |
| Sprint-3 | Details about estimated cost based on damage | USN-7 | As a user I must receive a detailed report of the damages present in the vehicle and the cost estimated |  | high | team member 1, team member 2, team member 3 and team member 4 |
| Sprint-4 | Provide friendly and efficient customer | USN-8 | As a user, I need to get support from developers in case of queries |  | high | team member 1 and team member |

| | | | | | | |
|---|---|---|---|---|---|---|
| | support and sort out the queries. | | and failure of service provided | | | 3 |
| Sprint-4 | overview the entire process and act as a bridge between user and developer | USN-9 | We need to satisfy the customer needs in an efficient way and make sure any sort of errors are fixed | | high | team member 2 and team member 4 |

Project Tracker, Velocity & Burn down 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 | 24 Oct 2022 | 29 Oct 2022 | 20 | 29 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 20 | 05 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 12 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 20 | 19 Nov 2022 |

## 6.2 SPRINT DELIVERY SCHEDULE:

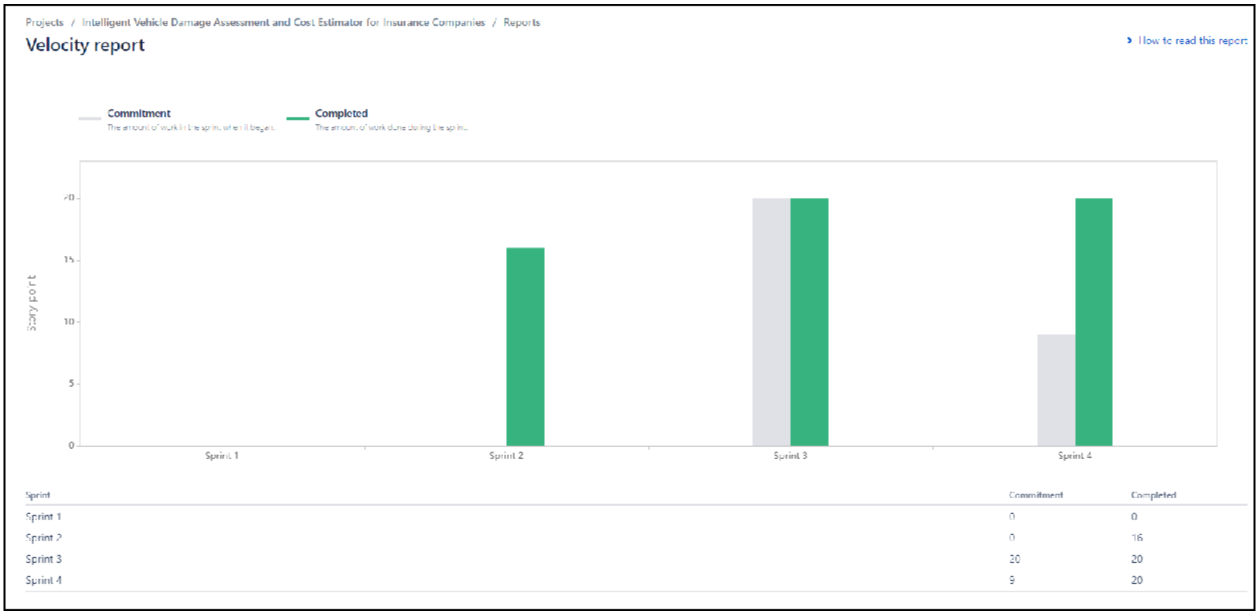
| Sprint | Milestone |
|---|---|
| Sprint 1 | 1. The user Registers into the application by entering their Email Id Password and Re-entering the Password for confirmation.<br>2. User Receives a confirmation mail for their registered Email.<br>3. User can also register for the application through a Mobile number.<br>4. User logs in to the website using Email Id password or through Gmail |
| Sprint 2 | 1. User can access the dashboard<br>2. User uploads the images of their vehicle and other relevant details |
| Sprint 3 | 1. Application should carry out the damage assessment and produce an estimated cost for insurance.<br>2. The data stored should be secure. |
| Sprint 4 | 1. Administrator should properly maintain the website and update it when required.<br>2.When any issues are raised it should be sorted. |

## 6.3 REPORTS FROM JIRA

**Velocity:**

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let’s calculate the team’s average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$


Projects / Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies / Reports

Velocity report

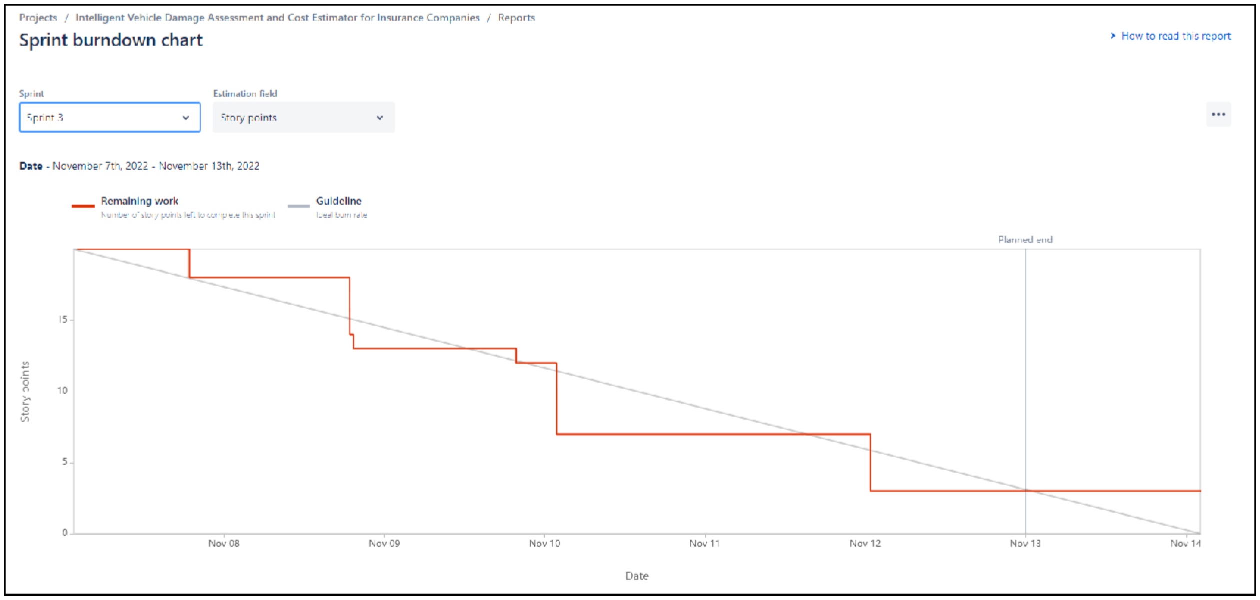
› How to read this report
Commitment
The amount of work in the sprint when it began.
Completed
The amount of work done during the sprint.
Story point
20
15
10
5
0
Sprint 1
Sprint 2
Sprint 3
Sprint 4

| Sprint | Commitment | Completed |
|---|---|---|
| Sprint 1 | 0 | 0 |
| Sprint 2 | 0 | 16 |
| Sprint 3 | 20 | 20 |
| Sprint 4 | 9 | 20 |

Figure 10: Velocity report from JIRA

## 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 charts can be applied to any project containing measurable progress over time.



Projects / Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies / Reports

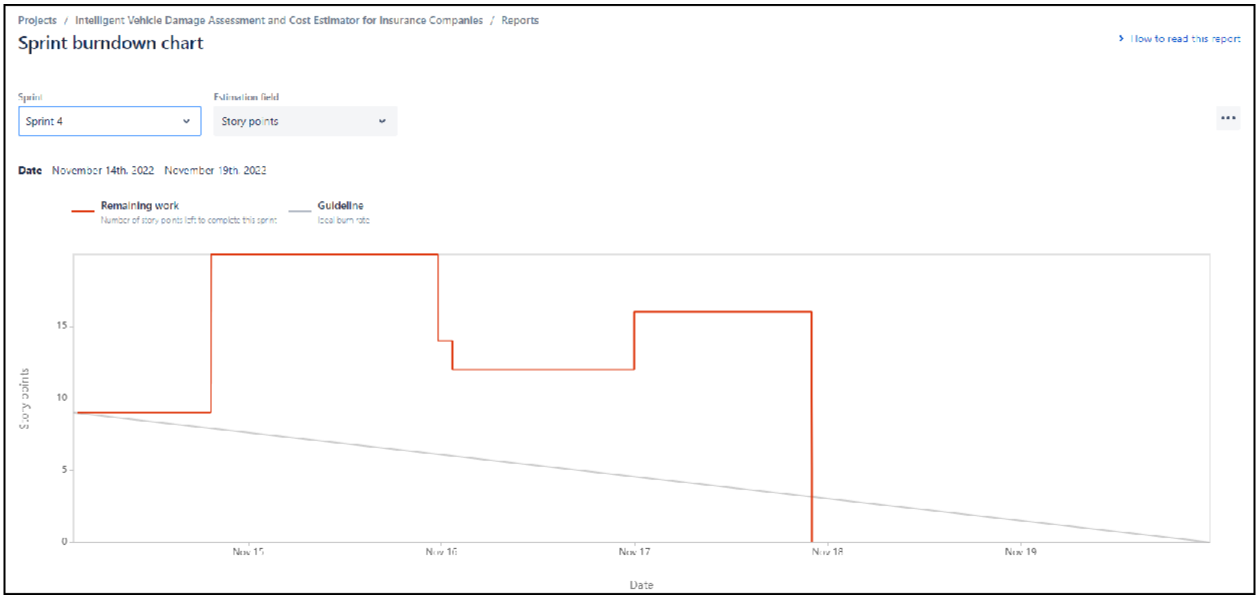
Sprint burndown chart

Sprint: Sprint 3

Estimation field: Story points

**Date** - November 7th, 2022 - November 13th, 2022

Figure 11 a. Burndown Chart from JIRA-Sprint 3



Projects / Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies / Reports

Sprint burndown chart

Sprint: Sprint 4

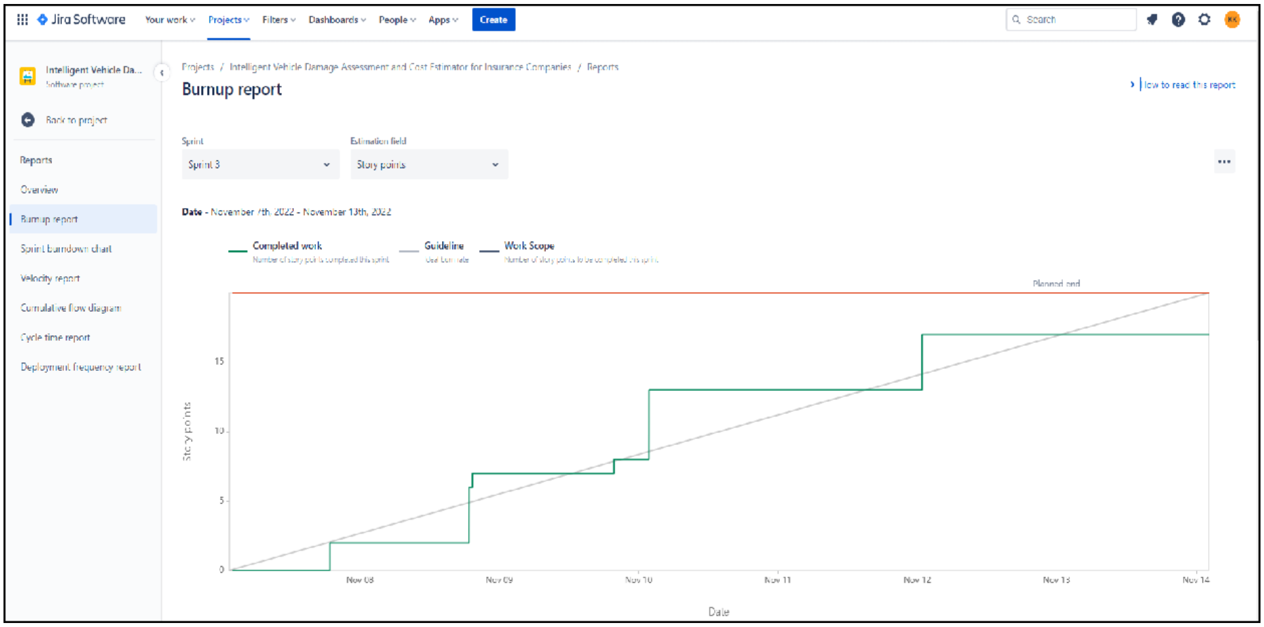
Estimation field: Story points

**Date** November 14th, 2022 November 19th, 2022

Figure 11 b. Burndown Chart from JIRA-Sprint 4

## Burnup report:

A burn up chart is a visual diagram commonly used on Agile projects to help measure progress. Agile burn up charts allow project managers and teams to quickly see how their workload is progressing and whether project completion is on schedule.



Projects / Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies / Reports

Burnup report

Sprint: Sprint 3

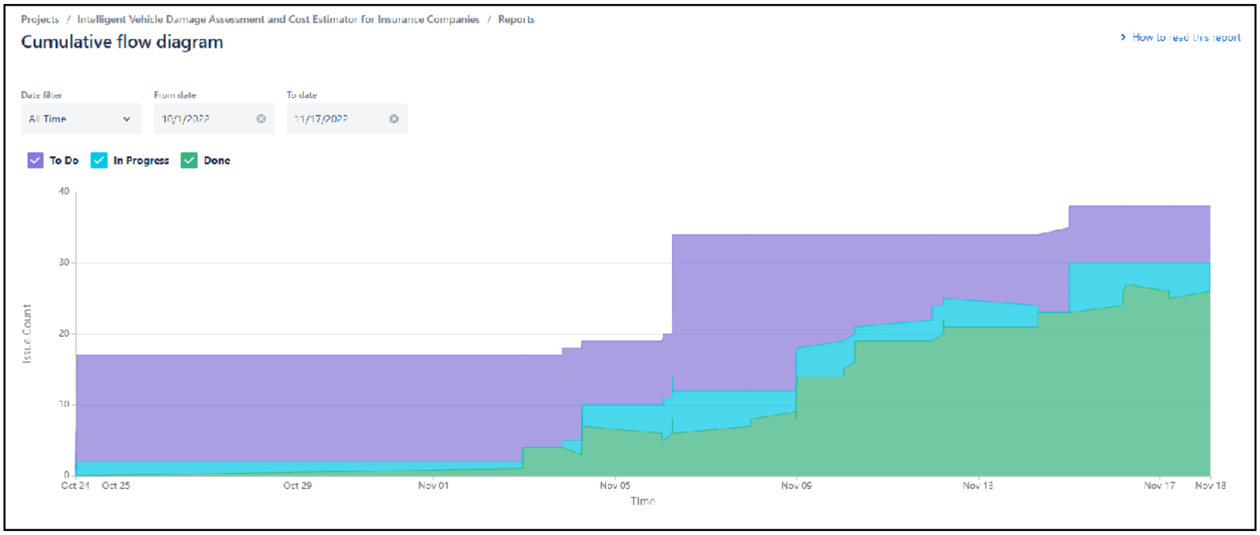
Estimation field: Story points

**Date** - November 7th, 2022 - November 13th, 2022

Figure 12 : Burn Up Chart

## Cumulative flow diagram

Cumulative flow diagram shows the statuses of issues over time. This helps you identify potential bottlenecks that need to be investigated.The distance between each column lines shows you how long issues take to get from one state to another. Look for points where one band is growing at a faster rate than another to find bottlenecks.



Projects / Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies / Reports

Cumulative flow diagram

Date filter: All Time

From date: 10/1/2022

To date: 11/17/2022

FIgure 13 : Cumulative flow diagram

# 7.CODING & SOLUTIONING

## 7.1. DATA COLLECTION:

The dataset consists of training and validation images splitted into two subdivisions based on level of damage(minor,moderate,severe) and based on part of the vehicle damaged(front,side,rear).

file directory:

Dataset

- BODY
    - training
        - 00-front
        - 01-rear
        - 02-side
    - validation
- LEVEL
    - training
        - 01-minor
        - 02-moderate
        - 03-severe

  - validation
    - 01-minor
    - 02-moderate
    - 03-severe

### 7.2.1 IMPORT THE IMAGE DATA GENERATOR LIBRARY:

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class

***PYTHON CODE:***

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

### 7.2.2 CONFIGURE IMAGE DATA GENERATOR:

Image Data Generator class is instantiated and the configuration for the types of data augmentation.

***PYTHON CODE:***

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

### 7.2.3 APPLY IMAGE DATA GENERATOR FUNCTIONALITY TO TRAINSET AND TESTSET:

Applying ImageDataGenerator functionality to Trainset and Testset by using the following code.For Training set using flow_from_directory function.

This function will return batches of images from the subdirectories

**Arguments:**

- directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch_size: Size of the batches of data which is 64.
- target_size: Size to resize images after they are read from disk.
- class_mode:

  - ‘int': means that the labels are encoded as integers (e.g. for sparse_categorical_crossentropy loss).
  - 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical_crossentropy loss).
  - 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary_crossentropy).
  - None (no labels).

***PYTHON CODE:***

```
#body part
trainPath = '/home/wsuser/work/Dataset/body/training'
testPath = '/home/wsuser/work/Dataset/body/validation'
```

```
training_set =
train_datagen.flow_from_directory(trainPath,target_size=(244,244),batch_size=1
0,class_mode='categorical')
test_set =
train_datagen.flow_from_directory(testPath,target_size=(244,244),batch_size=10,
class_mode='categorical')

#level part
trainPath = '/home/wsuser/work/Dataset/level/training'
testPath = '/home/wsuser/work/Dataset/level/validation'

training_set =
train_datagen.flow_from_directory(trainPath,target_size=(244,244),batch_size=1
0,class_mode='categorical')
test_set =
train_datagen.flow_from_directory(testPath,target_size=(244,244),batch_size=10,
class_mode='categorical')
```

Loading our data and performing Data Augmentation

## 7.3. MODEL BUILDING:

The following procedures are carried in order to build the model:

1.**Importing The Model Building Libraries**

***PYTHON CODE:***

```
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
```

## 2.Loading The Model

***PYTHON CODE:***

```
vgg=VGG16(input_shape=(244,244,3),weights='imagenet',include_top=False)
```

## 3.Adding Flatten Layer

***PYTHON CODE:***

```
for layer in vgg.layers:
        layer.trainable=False
x=Flatten()(vgg.output)
```

## 4.Adding Output Layer

***PYTHON CODE:***

```
prediction=Dense(3,activation='softmax')(x)
```

## 5.Creating A Model Object

***PYTHON CODE:***

```
model=Model(inputs=vgg.input,outputs=prediction)
model1=Model(inputs=vgg.input,outputs=prediction)

model.summary()
```

## 6.Configure The Learning Process

***PYTHON CODE:***

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

## 7.Train The Model

***PYTHON CODE:***

```
# body
r = model.fit_generator(
    training_set,
    validation_data = test_set,
    epochs = 25,
    steps_per_epoch=979//10,
    validation_steps = 171//10
)

#level
y = model1.fit_generator(
    training_set,
    validation_data = test_set,
    epochs = 25,
    steps_per_epoch=979//10,
    validation_steps = 171//10
)
```

### 8.Save The Model

***PYTHON CODE:***

```
model.save('body.h5')

model1.save('level.h5')
```

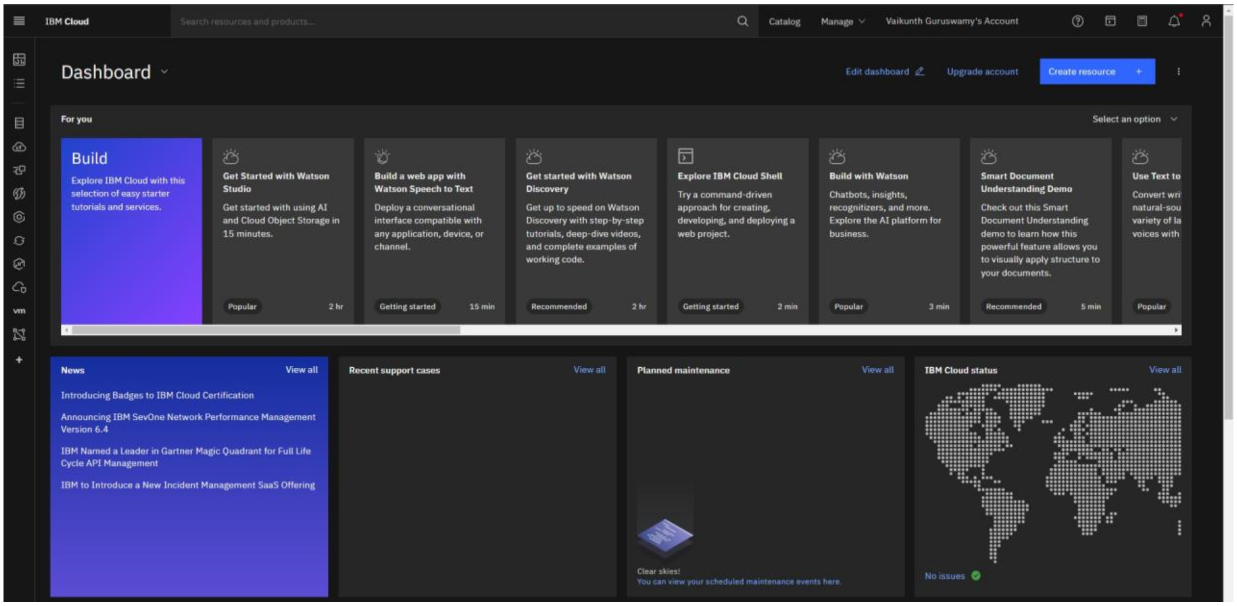
### 9.Test The Model

***PYTHON CODE:***

```
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

from tensorflow.keras.models import load_model

model1 = load_model('Model/level.h5')
model2 = load_model('Model/body.h5')
```

## 7.4 CLOUDANT DB:

Register & Login To IBM Cloud



Build

Explore IBM Cloud with this selection of easy starter tutorials and services.

Get Started with Watson Studio

Get started with using AI and Cloud Object Storage in 15 minutes.

Build a web app with Watson Speech to Text

Deploy a conversational interface compatible with any application, device, or channel.

Get started with Watson Discovery

Get up to speed on Watson Discovery with step-by-step tutorials, deep-dive videos, and complete examples of working code.

Explore IBM Cloud Shell

Try a command-driven approach for creating, developing, and deploying a web project.

Build with Watson

Chatbots, insights, recognitizers, and more. Explore the AI platform for business.

Smart Document Understanding Demo

Check out this Smart Document Understanding demo to learn how this powerful feature allows you to visually apply structure to your documents.

Use Text to

Convert wri natural-sou variety of la voices with

Introducing Badges to IBM Cloud Certification

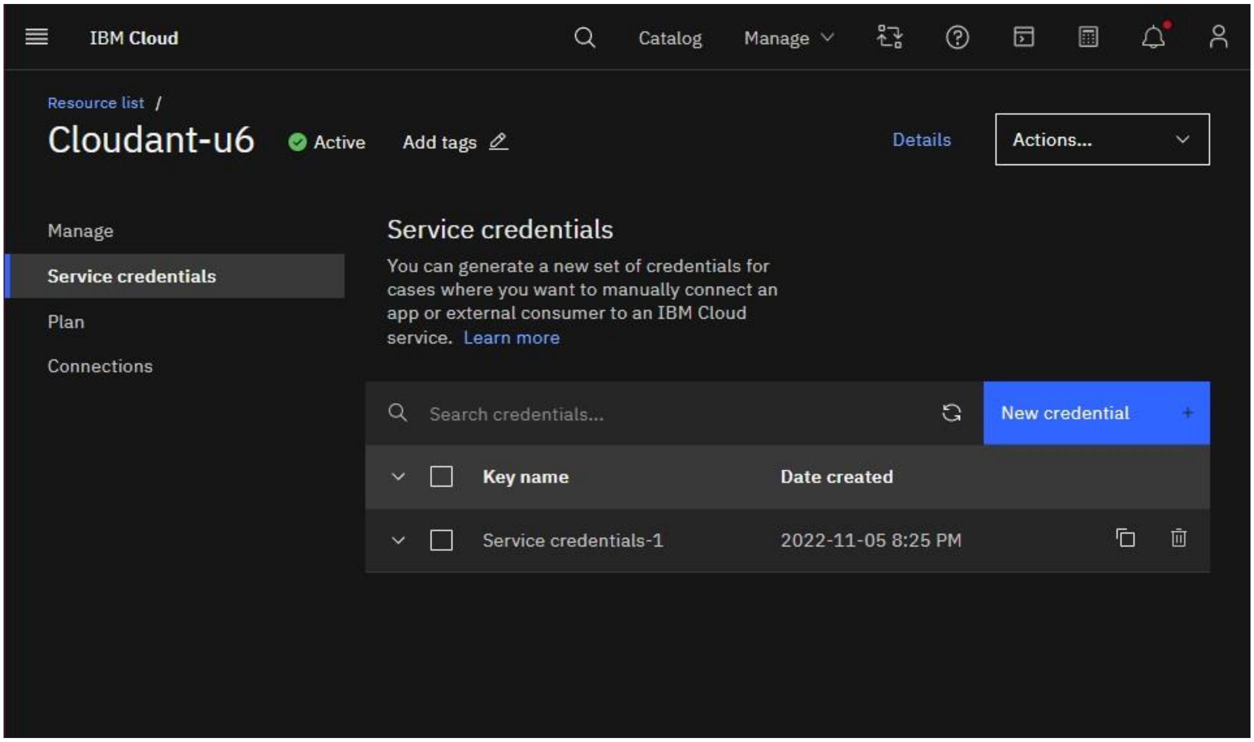
Announcing IBM SevOne Network Performance Management Version 6.4

IBM Named a Leader in Gartner Magic Quadrant for Full Life Cycle API Management

IBM to Introduce a New Incident Management SaaS Offering

Clear skies!
You can view your scheduled maintenance events here.

Figure 14:Creating Service Credentials



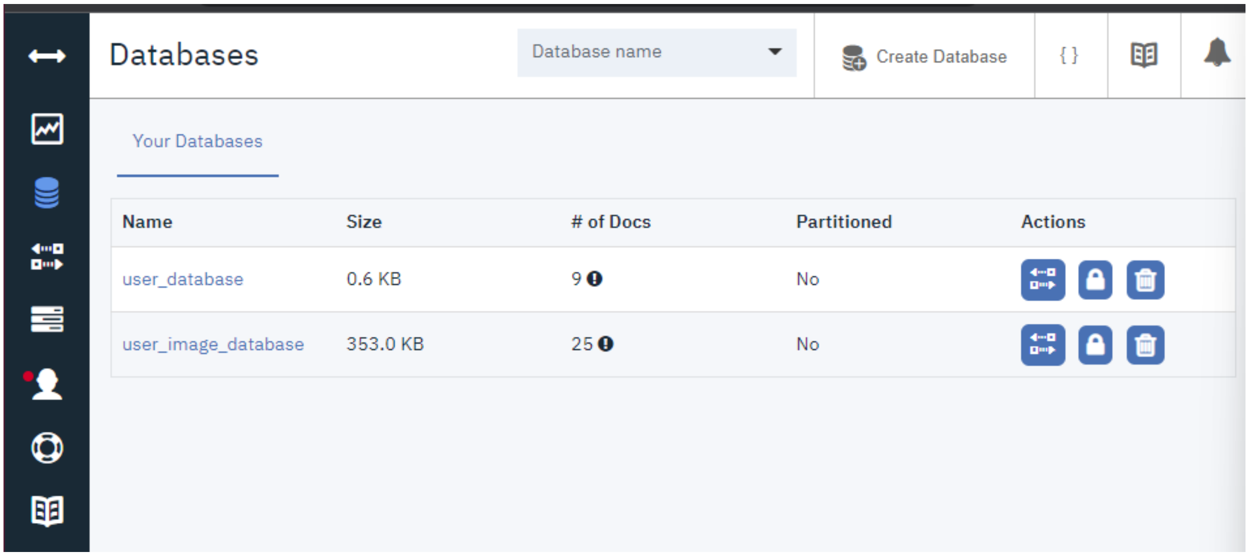
Cloudant-u6 Active

Service credentials

You can generate a new set of credentials for cases where you want to manually connect an app or external consumer to an IBM Cloud service. Learn more

| Key name | Date created |
|---|---|
| Service credentials-1 | 2022-11-05 8:25 PM |

Figure 15:Launch Cloudant DB



| Name | Size | # of Docs | Partitioned | Actions |
|---|---|---|---|---|
| user_database | 0.6 KB | 9 | No | |
| user_image_database | 353.0 KB | 25 | No | |

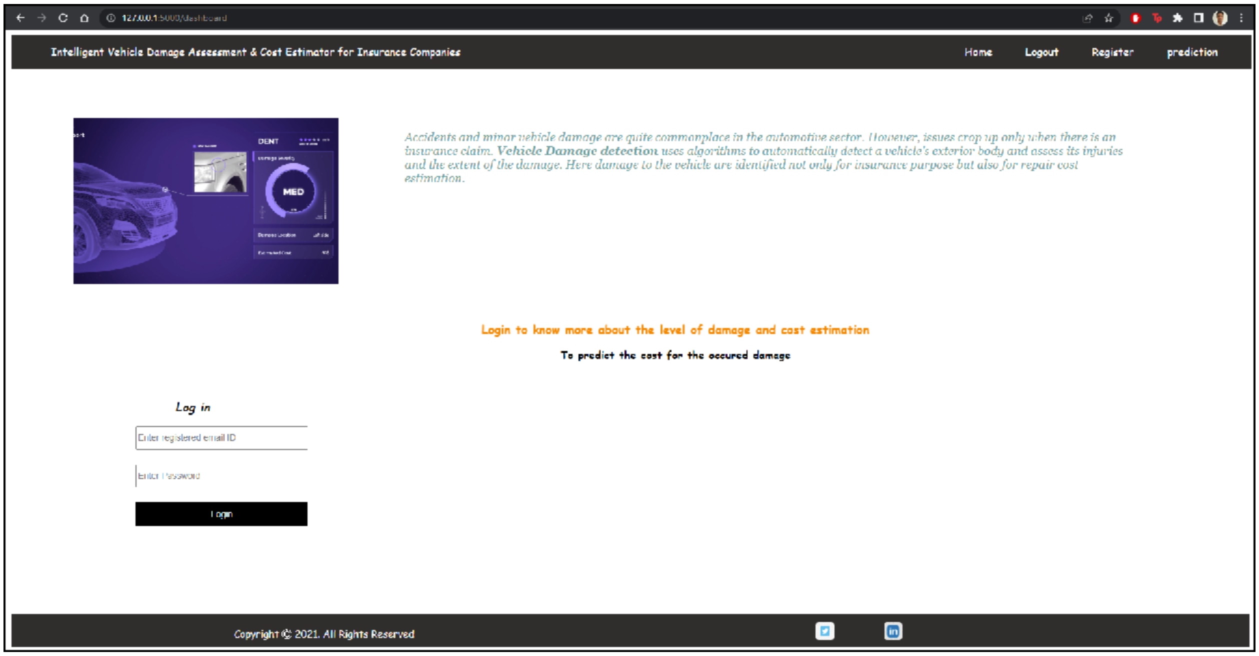
Figure 16:Create Database

(Creation and accessing cloud database has been carried out and recorded as meeting and uploaded in the drive link provided in Appendix )

## 7.5 APPLICATION BUILDING

Built a web application that is integrated into the model developed in the above section. A UI is provided to the user where they can upload the image. Based on the saved model, the uploaded image will be analyzed and prediction is showcased on the UI.

### 7.5.1 BUILDING HTML PAGES:



Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies

*Accidents and minor vehicle damage are quite commonplace in the automotive sector. However, issues crop up only when there is an insurance claim.* ***Vehicle Damage detection*** *uses algorithms to automatically detect a vehicle's exterior body and assess its injuries and the extent of the damage. Here damage to the vehicle are identified not only for insurance purpose but also for repair cost estimation.*

Login to know more about the level of damage and cost estimation

To predict the cost for the occured damage

Copyright © 2021. All Rights Reserved

Figure 17:Dashboard

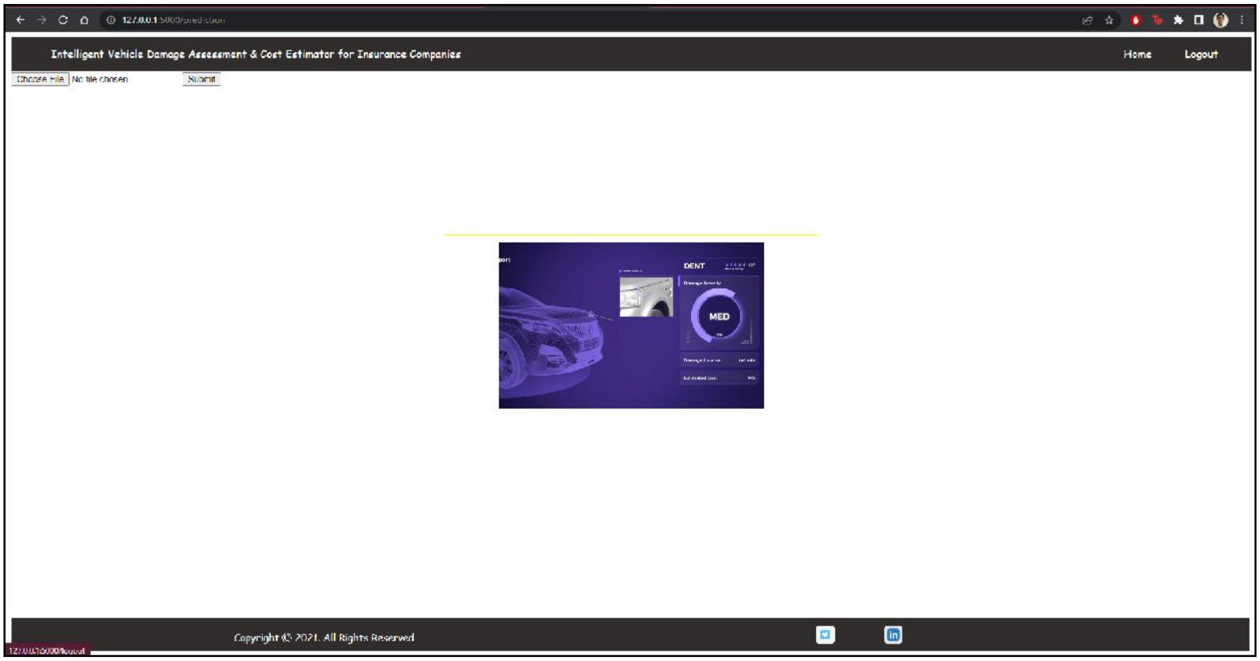

127.0.0.1:5000/prediction
Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Companies
Home
Logout
Choose File
No file chosen
Submit
DENT
MED
Copyright © 2021. All Rights Reserved

Figure 18:Prediction

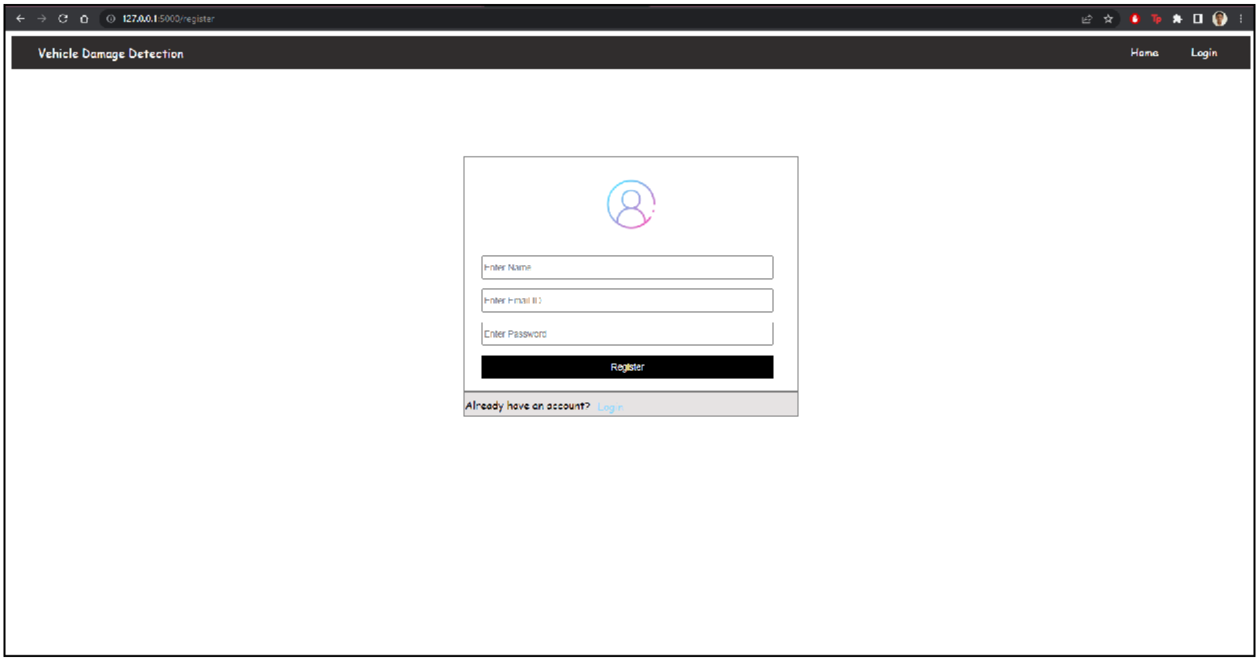


Figure 19:Registration

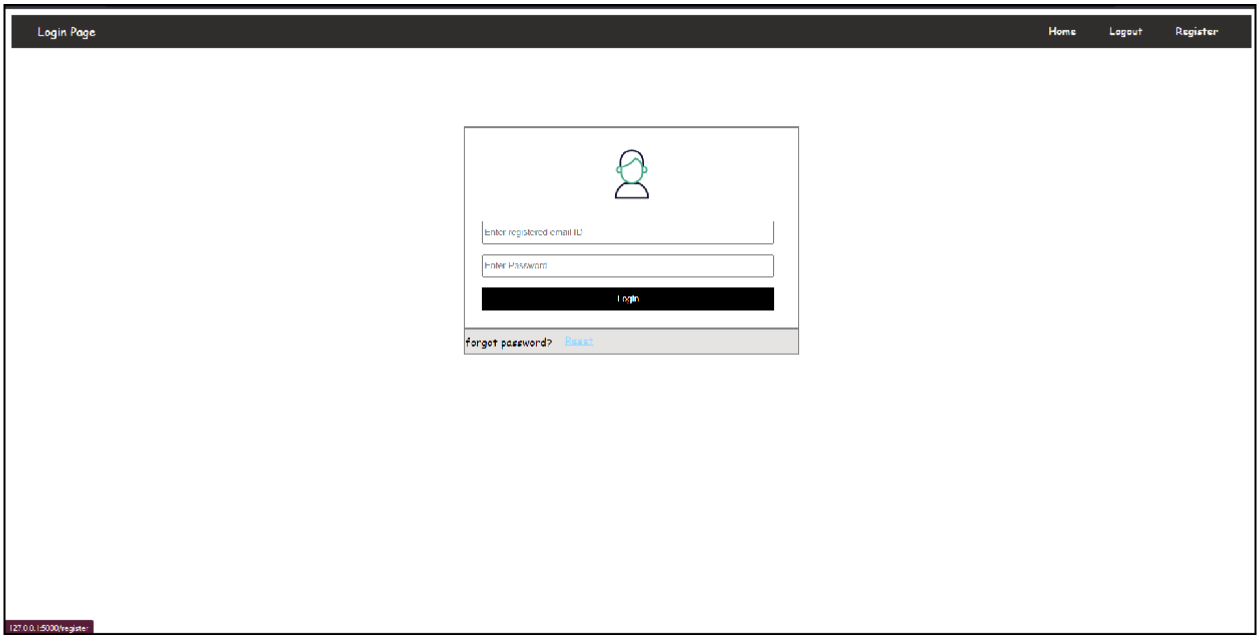
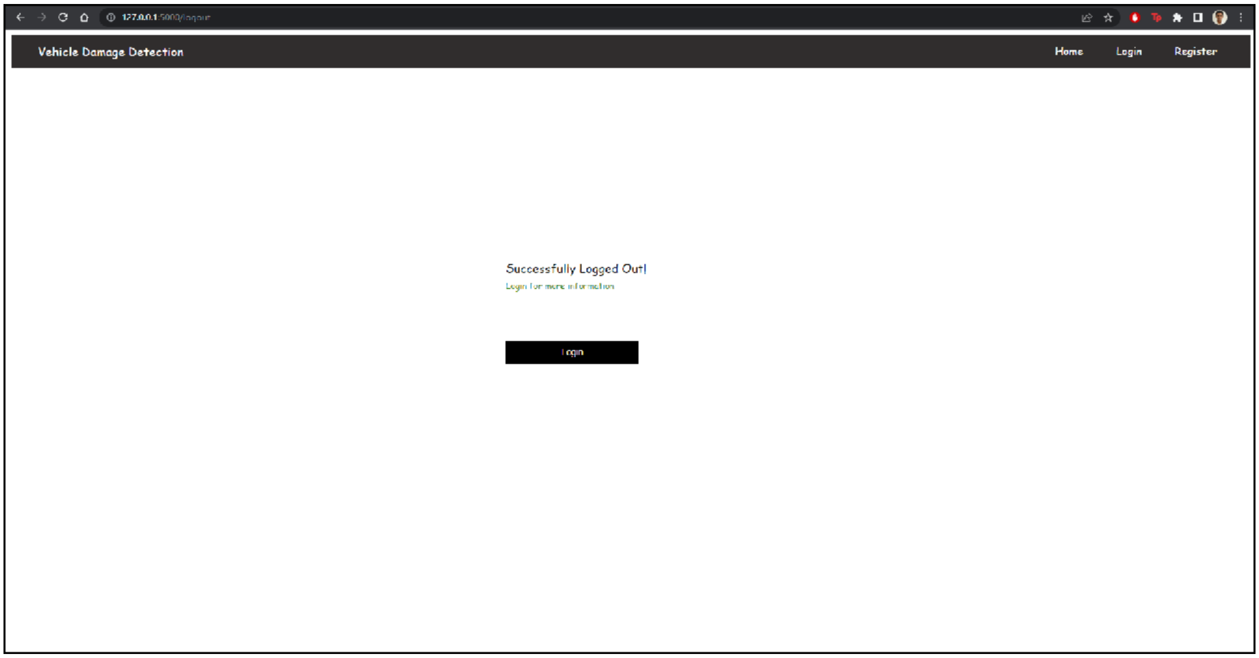


Figure 20:Login



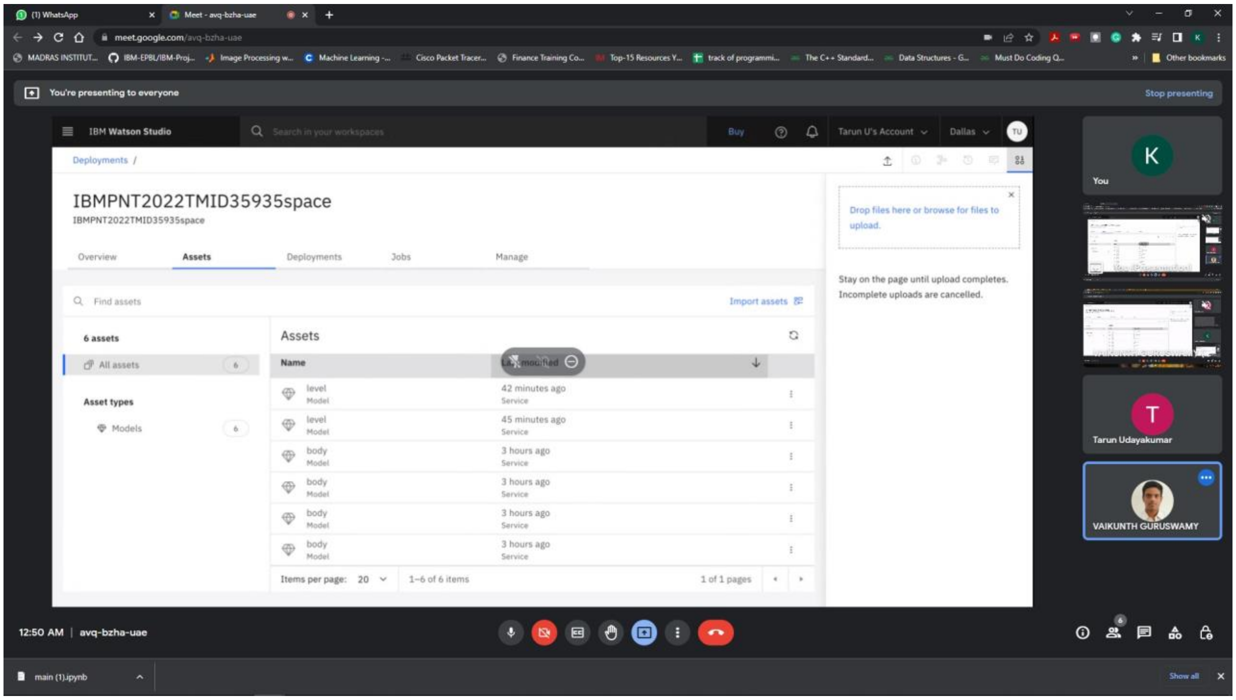
Successfully Logged Out!

Login for more information

fIGUE

# 8. TRAIN THE MODEL ON IBM:

## 8.1. REGISTER FOR IBM CLOUD



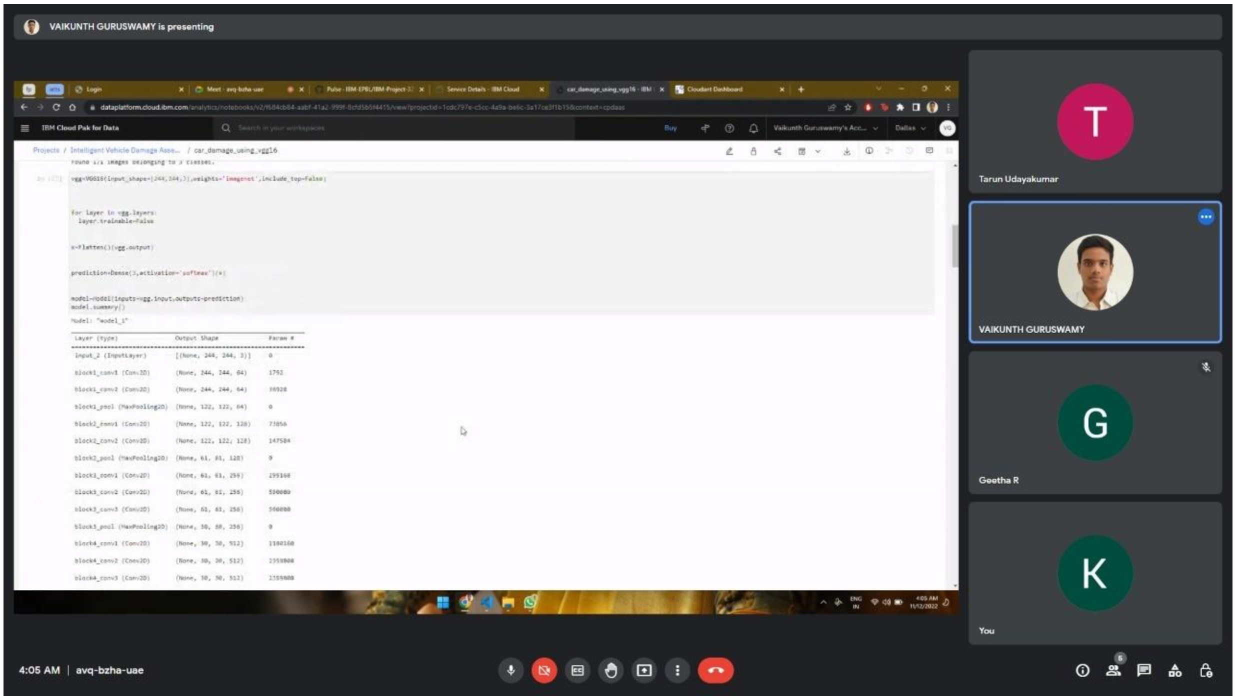
IBMPNT2022TMID35935space

IBMPNT2022TMID35935space

Assets

| Name | Last modified |
|---|---|
| level<br>Model | 42 minutes ago<br>Service |
| level<br>Model | 45 minutes ago<br>Service |
| body<br>Model | 3 hours ago<br>Service |
| body<br>Model | 3 hours ago<br>Service |
| body<br>Model | 3 hours ago<br>Service |
| body<br>Model | 3 hours ago<br>Service |

## 8.2. TRAIN MODEL ON IBM



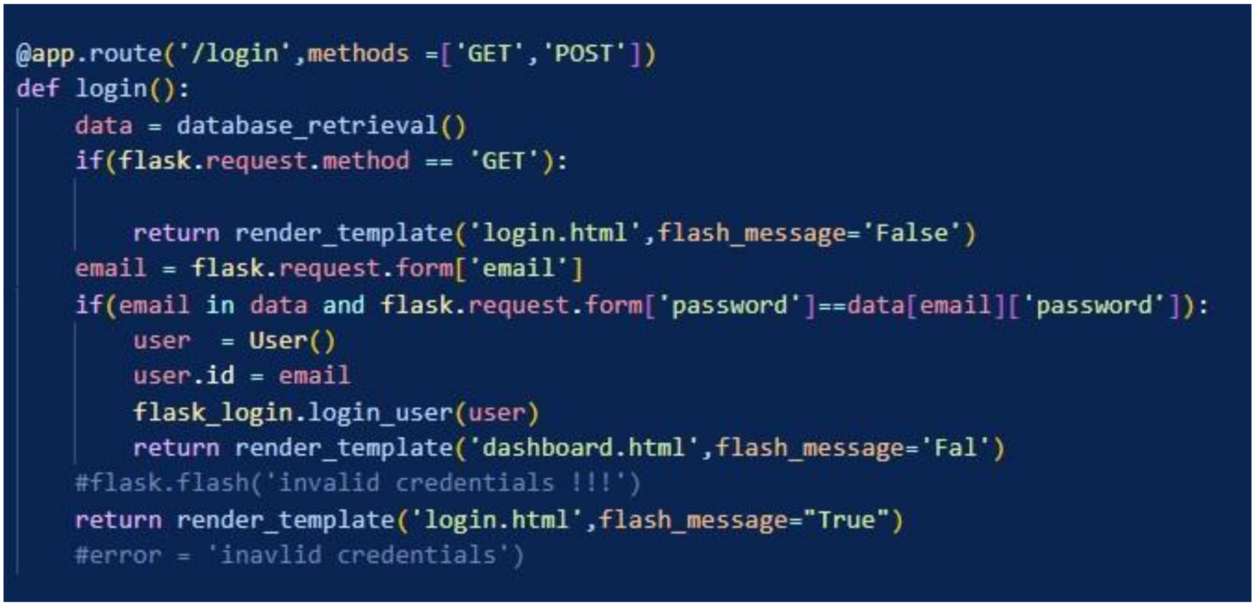
VAIKUNTH GURUSWAMY is presenting
IBM Cloud Pak for Data
Projects / Intelligent Vehicle Damage Asse... / car_damage_using_vgg16
Tarun Udayakumar
VAIKUNTH GURUSWAMY
Geetha R
You
4:05 AM | avq-bzha-uae

```
Found 171 images belonging to 3 classes.

vgg=VGG16(input_shape=[244,244,3],weights='imagenet',include_top=False)


for layer in vgg.layers:
  layer.trainable=False


x=Flatten()(vgg.output)


prediction=Dense(3,activation='softmax')(x)


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

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 244, 244, 3)]     0

 block1_conv1 (Conv2D)       (None, 244, 244, 64)      1792

 block1_conv2 (Conv2D)       (None, 244, 244, 64)      36928

 block1_pool (MaxPooling2D)  (None, 122, 122, 64)      0

 block2_conv1 (Conv2D)       (None, 122, 122, 128)     73856

 block2_conv2 (Conv2D)       (None, 122, 122, 128)     147584

 block2_pool (MaxPooling2D)  (None, 61, 61, 128)       0

 block3_conv1 (Conv2D)       (None, 61, 61, 256)       295168

 block3_conv2 (Conv2D)       (None, 61, 61, 256)       590080

 block3_conv3 (Conv2D)       (None, 61, 61, 256)       590080

 block3_pool (MaxPooling2D)  (None, 30, 30, 256)       0

 block4_conv1 (Conv2D)       (None, 30, 30, 512)       1180160

 block4_conv2 (Conv2D)       (None, 30, 30, 512)       2359808

 block4_conv3 (Conv2D)       (None, 30, 30, 512)       2359808
```

## 7.1 USER LOGIN AND REGISTRATION

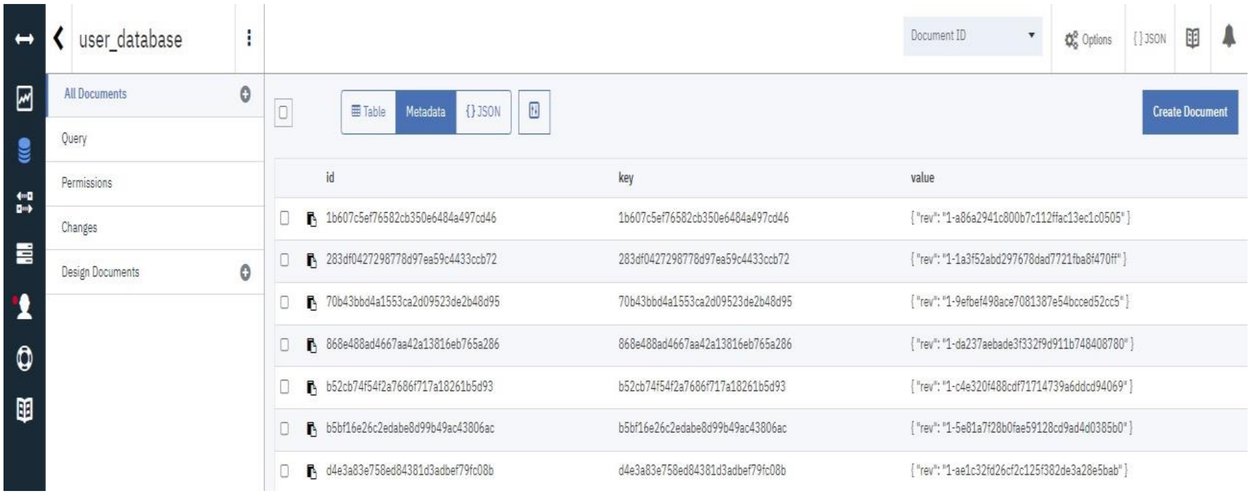
- Successfully created user login and user registration features
- user can create an account in the website user their personal email id
- confirmation mail will be send to the respective user once new account is created



```
@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')
```

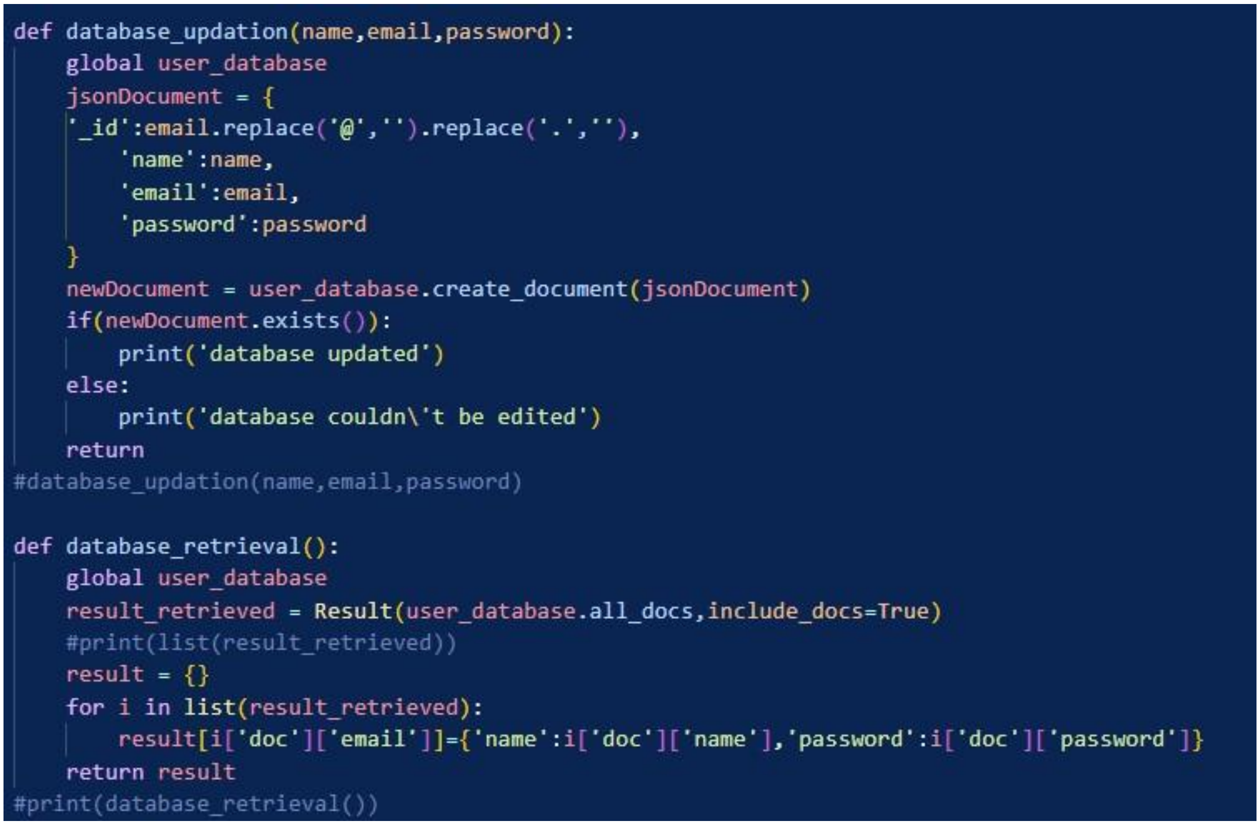
## 7.2 CLOUD STORAGE

- in order to store the user data all the user details along with their credentials are securely stored in the IBM CLOUDANT database and are retrieved whenever required
- user details such as name, email id, password are stored in a separate database



| id | key | value |
|---|---|---|
| 1b607c5ef76582cb350e6484a497cd46 | 1b607c5ef76582cb350e6484a497cd46 | { "rev": "1-a86a2941c800b7c112ffac13ec1c0505" } |
| 283df0427298778d97ea59c4433ccb72 | 283df0427298778d97ea59c4433ccb72 | { "rev": "1-1a3f52abd297678dad7721fba8f470ff" } |
| 70b43bbd4a1553ca2d09523de2b48d95 | 70b43bbd4a1553ca2d09523de2b48d95 | { "rev": "1-9efbef498ace7081387e54bcced52cc5" } |
| 868e488ad4667aa42a13816eb765a286 | 868e488ad4667aa42a13816eb765a286 | { "rev": "1-da237aebade3f332f9d911b748408780" } |
| b52cb74f54f2a7686f717a18261b5d93 | b52cb74f54f2a7686f717a18261b5d93 | { "rev": "1-c4e320f488cdf71714739a6ddcd94069" } |
| b5bf16e26c2edabe8d99b49ac43806ac | b5bf16e26c2edabe8d99b49ac43806ac | { "rev": "1-5e81a7f28b0fae59128cd9ad4d0385b0" } |
| d4e3a83e758ed84381d3adbef79fc08b | d4e3a83e758ed84381d3adbef79fc08b | { "rev": "1-ae1c32fd26cf2c125f382de3a28e5bab" } |

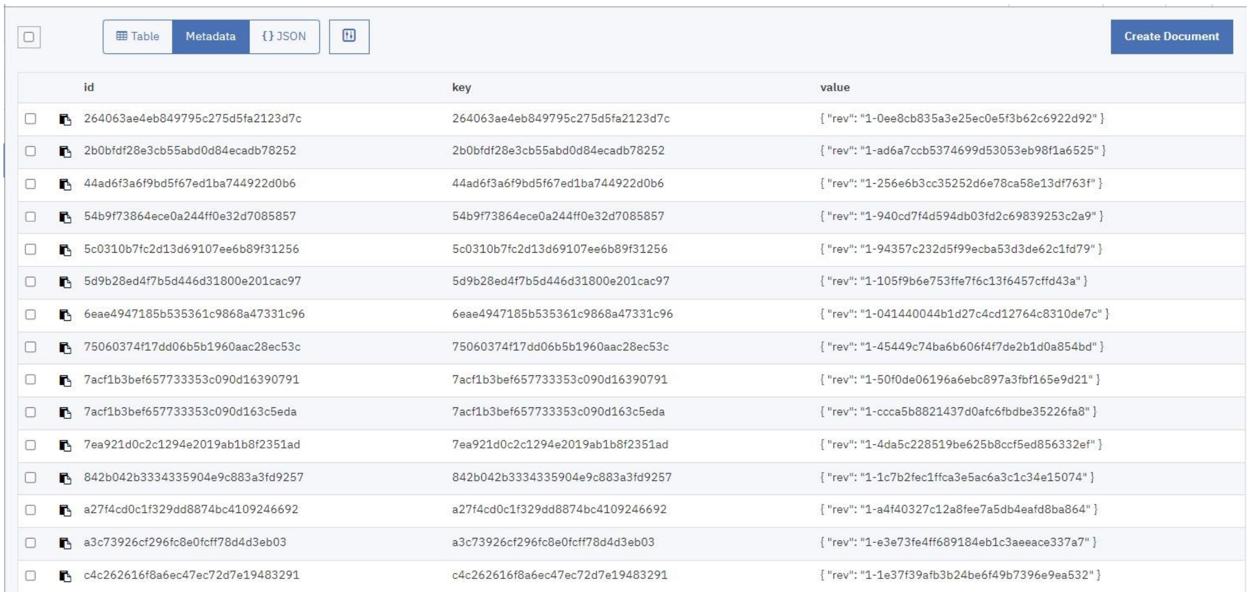
python script for user data updation:



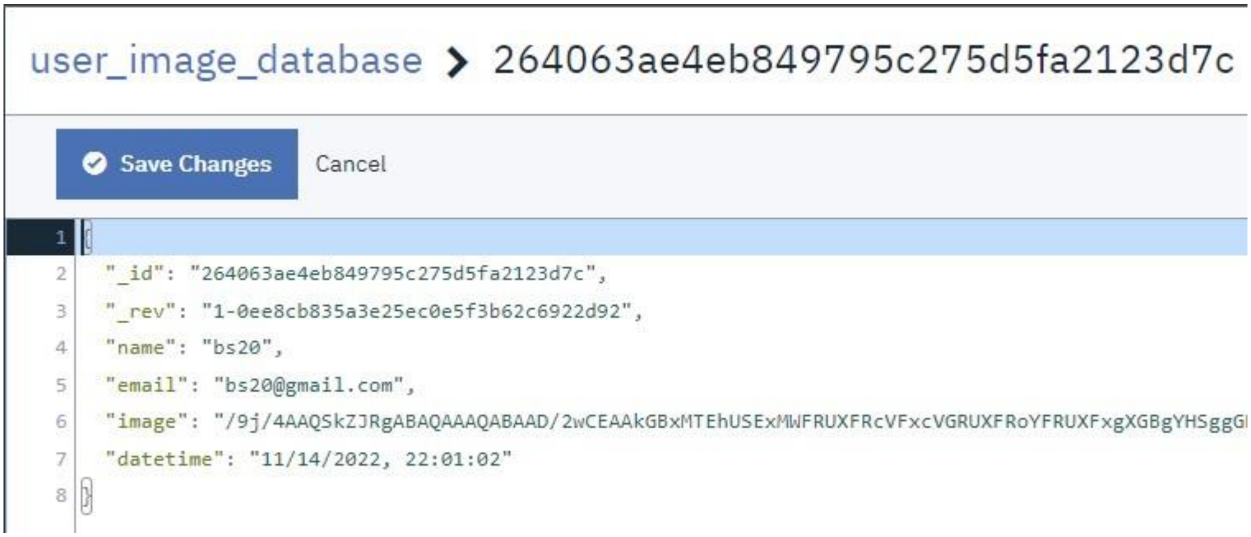
```
def database_updation(name,email,password):
    global user_database
    jsonDocument = {
    '_id':email.replace('@','').replace('.',''),
        'name':name,
        'email':email,
        'password':password
    }
    newDocument = user_database.create_document(jsonDocument)
    if(newDocument.exists()):
        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())
```

- Further to maintain log file user uploaded images are also being updated in the cloud in a seperate database within IBM CLOUDANT

- The input image is encoded and converted to base 64 string format and then uploaded as individual documents in the database along with user details such as email id and name and date and time of request is being updated in cloud



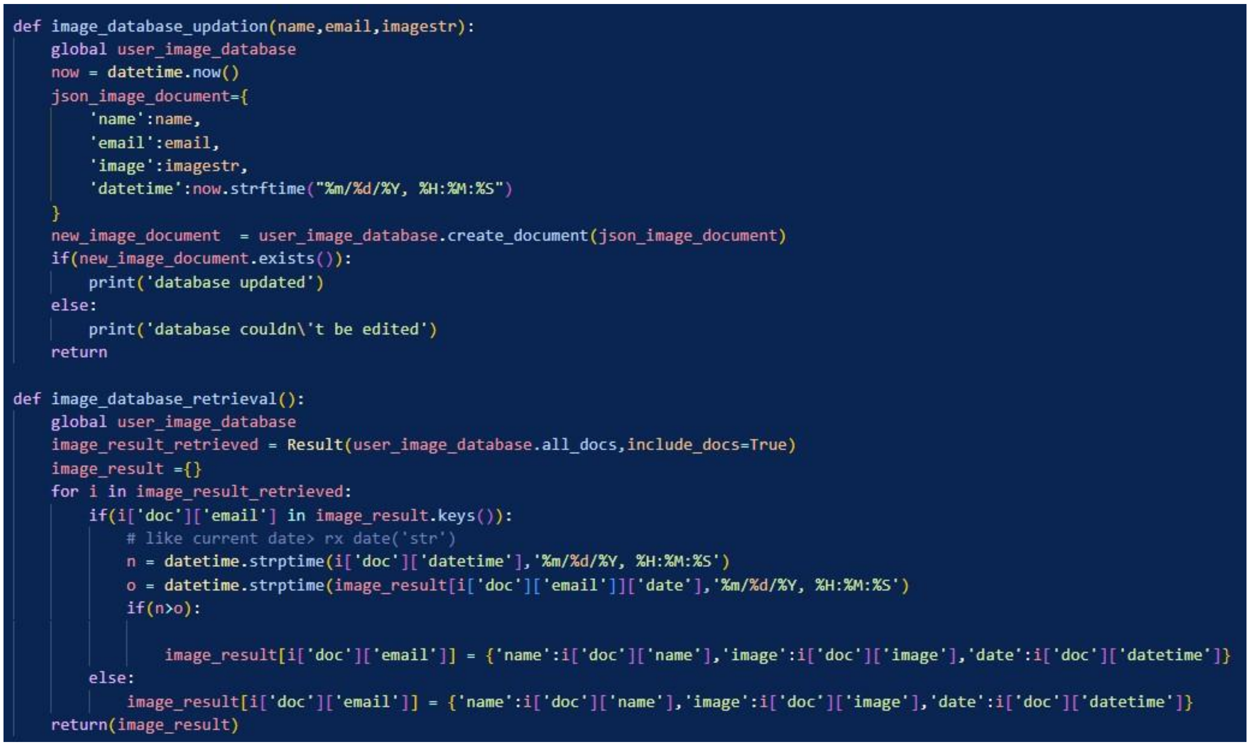
| id | key | value |
|---|---|---|
| 264063ae4eb849795c275d5fa2123d7c | 264063ae4eb849795c275d5fa2123d7c | { "rev": "1-0ee8cb835a3e25ec0e5f3b62c6922d92" } |
| 2b0bfdf28e3cb55abd0d84ecadb78252 | 2b0bfdf28e3cb55abd0d84ecadb78252 | { "rev": "1-ad6a7ccb5374699d53053eb98f1a6525" } |
| 44ad6f3a6f9bd5f67ed1ba744922d0b6 | 44ad6f3a6f9bd5f67ed1ba744922d0b6 | { "rev": "1-256e6b3cc35252d6e78ca58e13df763f" } |
| 54b9f73864ece0a244ff0e32d7085857 | 54b9f73864ece0a244ff0e32d7085857 | { "rev": "1-940cd7f4d594db03fd2c69839253c2a9" } |
| 5c0310b7fc2d13d69107ee6b89f31256 | 5c0310b7fc2d13d69107ee6b89f31256 | { "rev": "1-94357c232d5f99ecba53d3de62c1fd79" } |
| 5d9b28ed4f7b5d446d31800e201cac97 | 5d9b28ed4f7b5d446d31800e201cac97 | { "rev": "1-105f9b6e753ffe7f6c13f6457cffd43a" } |
| 6eae4947185b535361c9868a47331c96 | 6eae4947185b535361c9868a47331c96 | { "rev": "1-041440044b1d27c4cd12764c8310de7c" } |
| 75060374f17dd06b5b1960aac28ec53c | 75060374f17dd06b5b1960aac28ec53c | { "rev": "1-45449c74ba6b606f4f7de2b1d0a854bd" } |
| 7acf1b3bef657733353c090d16390791 | 7acf1b3bef657733353c090d16390791 | { "rev": "1-50f0de06196a6ebc897a3fbf165e9d21" } |
| 7acf1b3bef657733353c090d163c5eda | 7acf1b3bef657733353c090d163c5eda | { "rev": "1-ccca5b8821437d0afc6fbdbe35226fa8" } |
| 7ea921d0c2c1294e2019ab1b8f2351ad | 7ea921d0c2c1294e2019ab1b8f2351ad | { "rev": "1-4da5c228519be625b8ccf5ed856332ef" } |
| 842b042b3334335904e9c883a3fd9257 | 842b042b3334335904e9c883a3fd9257 | { "rev": "1-1c7b2fec1ffca3e5ac6a3c1c34e15074" } |
| a27f4cd0c1f329dd8874bc4109246692 | a27f4cd0c1f329dd8874bc4109246692 | { "rev": "1-a4f40327c12a8fee7a5db4eafd8ba864" } |
| a3c73926cf296fc8e0fcff78d4d3eb03 | a3c73926cf296fc8e0fcff78d4d3eb03 | { "rev": "1-e3e73fe4ff689184eb1c3aeeace337a7" } |
| c4c262616f8a6ec47ec72d7e19483291 | c4c262616f8a6ec47ec72d7e19483291 | { "rev": "1-1e37f39afb3b24be6f49b7396e9ea532" } |



user_image_database > 264063ae4eb849795c275d5fa2123d7c

```
{
  "_id": "264063ae4eb849795c275d5fa2123d7c",
  "_rev": "1-0ee8cb835a3e25ec0e5f3b62c6922d92",
  "name": "bs20",
  "email": "bs20@gmail.com",
  "image": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAkGBxMTEhUSExMWFRUXFRcVFxcVGRUXFRoYFRUXFxgXGBgYHSggG
  "datetime": "11/14/2022, 22:01:02"
}
```

Python script for image updation

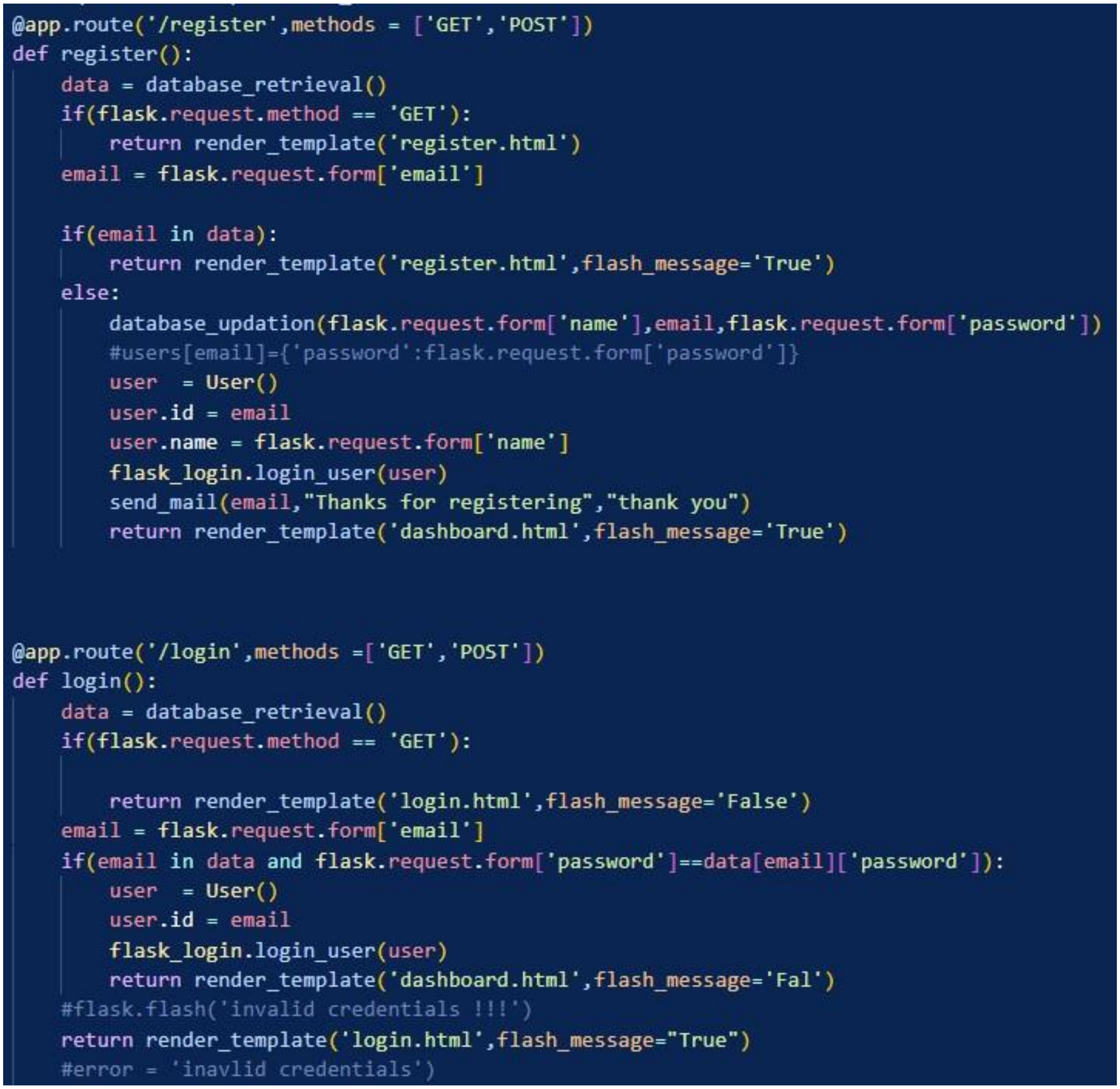


```
def image_database_updation(name,email,imagestr):
    global user_image_database
    now = datetime.now()
    json_image_document={
        'name':name,
        'email':email,
        'image':imagestr,
        'datetime':now.strftime("%m/%d/%Y, %H:%M:%S")
    }
    new_image_document  = user_image_database.create_document(json_image_document)
    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.strptime(i['doc']['datetime'],'%m/%d/%Y, %H:%M:%S')
            o = 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)
```

## 7.3 LOGIN MANAGER

- an effective deployment of login manager user FLASK has been developed to ease user navigation within tabs of the applications
- Futher to aauthenticate user access to various features provided by the system
- with user being logged in the user will have full access to all the routes provided by the application whereas without logging in restrictions to dashboard page and prediction page has be included to create a stablized log report and secure communication between the server and the client

Python script for register and login



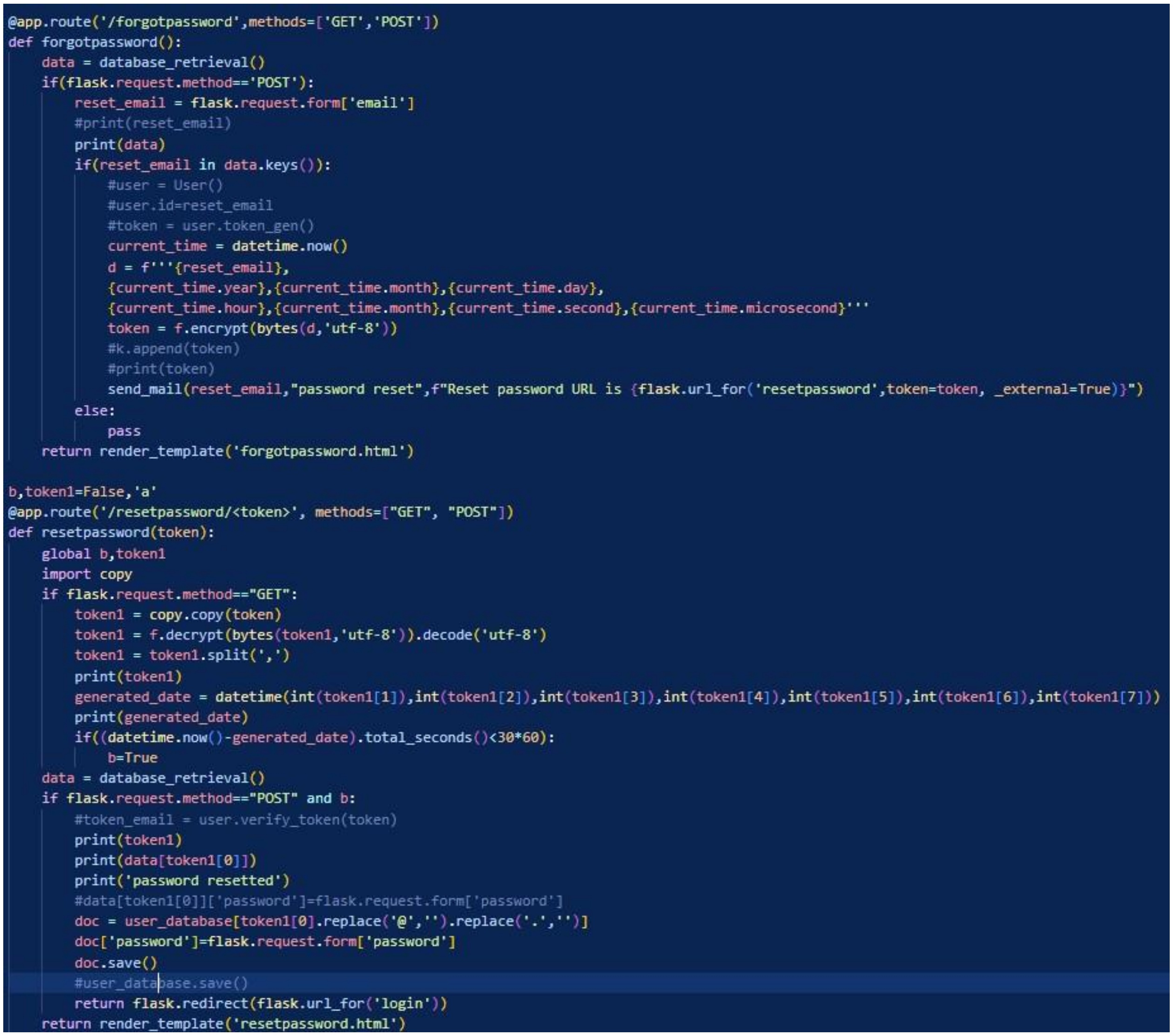
```
@app.route('/register',methods = ['GET','POST'])
def register():
    data = database_retrieval()
    if(flask.request.method == 'GET'):
        return render_template('register.html')
    email = flask.request.form['email']

    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)
        send_mail(email,"Thanks for registering","thank you")
        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')
```

### **7.4** FEATURE 1: FORGOT PASSWORD

- Inorder to ease user accessibility forgot password feature has being implemented for registered user
- User can avail the forgot password/ procedure to reset password anytime using their registered email id
- An highly secure password reset URL is sent to the user via their mail id
- On entering new password the modifications will be updated in the cloud immediately

- Authentication is ensure by genrating URL route using FERNET Encryption with base 64 format
- Verfication for the link is done through the decryption of the FERNET cipher text

Python script for forgot password and reset



```python
@app.route('/forgotpassword',methods=['GET','POST'])
def forgotpassword():
    data = database_retrieval()
    if(flask.request.method=='POST'):
        reset_email = flask.request.form['email']
        #print(reset_email)
        print(data)
        if(reset_email in data.keys()):
            #user = User()
            #user.id=reset_email
            #token = user.token_gen()
            current_time = datetime.now()
            d = f'''{reset_email},
            {current_time.year},{current_time.month},{current_time.day},
            {current_time.hour},{current_time.month},{current_time.second},{current_time.microsecond}'''
            token = f.encrypt(bytes(d,'utf-8'))
            #k.append(token)
            #print(token)
            send_mail(reset_email,"password reset",f"Reset password URL is {flask.url_for('resetpassword',token=token, _external=True)}")
        else:
            pass
    return render_template('forgotpassword.html')

b,token1=False,'a'
@app.route('/resetpassword/<token>', methods=["GET", "POST"])
def resetpassword(token):
    global b,token1
    import copy
    if flask.request.method=="GET":
        token1 = copy.copy(token)
        token1 = f.decrypt(bytes(token1,'utf-8')).decode('utf-8')
        token1 = token1.split(',')
        print(token1)
        generated_date = datetime(int(token1[1]),int(token1[2]),int(token1[3]),int(token1[4]),int(token1[5]),int(token1[6]),int(token1[7]))
        print(generated_date)
        if((datetime.now()-generated_date).total_seconds()<30*60):
            b=True
    data = database_retrieval()
    if flask.request.method=="POST" and b:
        #token_email = user.verify_token(token)
        print(token1)
        print(data[token1[0]])
        print('password resetted')
        #data[token1[0]]['password']=flask.request.form['password']
        doc = user_database[token1[0].replace('@','').replace('.','')]
        doc['password']=flask.request.form['password']
        doc.save()
        #user_database.save()
        return flask.redirect(flask.url_for('login'))
    return render_template('resetpassword.html')
```

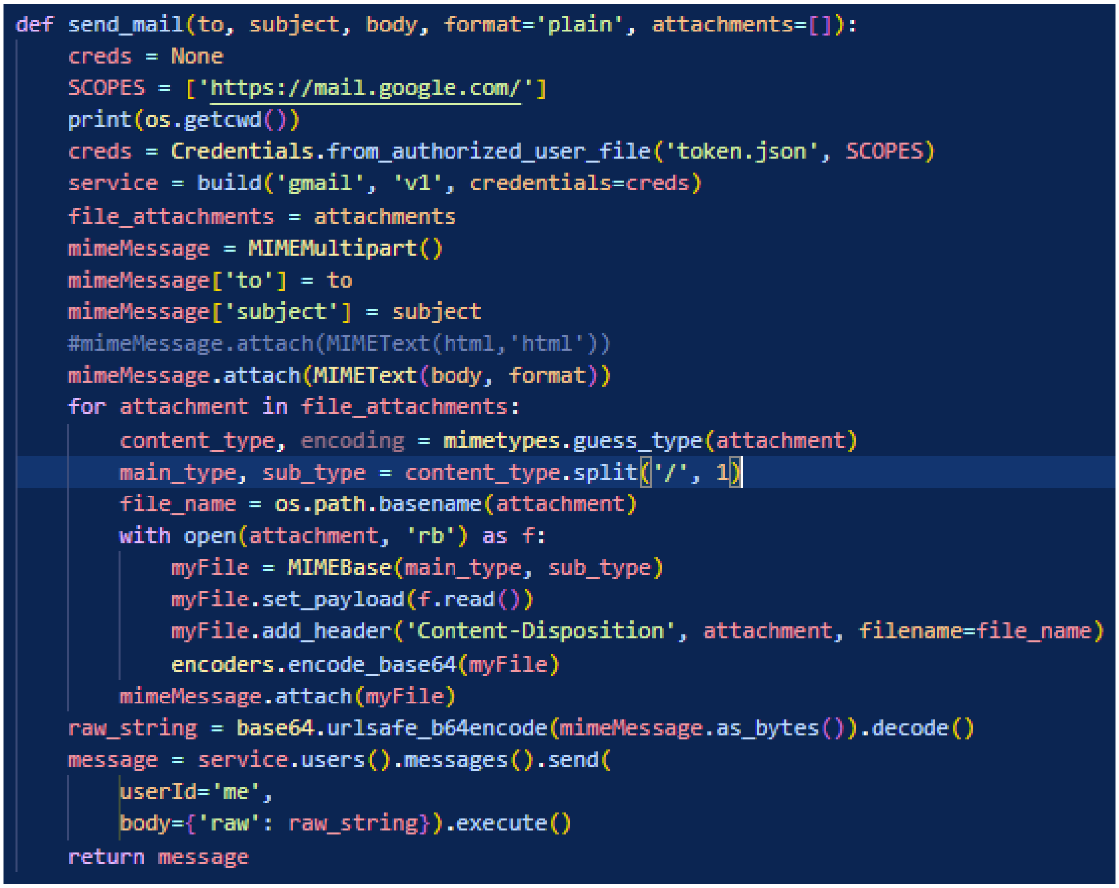
- The password reset URL will be valid on till 30 mins after generation of URL

**7.5** FEATURE 2:SENDING EMAIL

- An dedicated gmail account has be created to handle the mail request from the server
- The NO-REPLY mail sends mail using MIME protocol

- MIME (Multipurpose Internet Mail Extensions) is an extension of Simple Mail Transport Protocol (SMTP) protocol. It allows users to exchange data files, including audio, video, images and application programs, over email.

Python script to send mail



```
def send_mail(to, subject, body, format='plain', attachments=[]):
    creds = None
    SCOPES = ['https://mail.google.com/']
    print(os.getcwd())
    creds = Credentials.from_authorized_user_file('token.json', SCOPES)
    service = build('gmail', 'v1', credentials=creds)
    file_attachments = attachments
    mimeMessage = MIMEMultipart()
    mimeMessage['to'] = to
    mimeMessage['subject'] = subject
    #mimeMessage.attach(MIMEText(html,'html'))
    mimeMessage.attach(MIMEText(body, format))
    for attachment in file_attachments:
        content_type, encoding = mimetypes.guess_type(attachment)
        main_type, sub_type = content_type.split('/', 1)
        file_name = os.path.basename(attachment)
        with open(attachment, 'rb') as f:
            myFile = MIMEBase(main_type, sub_type)
            myFile.set_payload(f.read())
            myFile.add_header('Content-Disposition', attachment, filename=file_name)
            encoders.encode_base64(myFile)
        mimeMessage.attach(myFile)
    raw_string = base64.urlsafe_b64encode(mimeMessage.as_bytes()).decode()
    message = service.users().messages().send(
        userId='me',
        body={'raw': raw_string}).execute()
    return message
```

## **7.6** DATABASE SCHEME

USER_DATABASE :

- "_id": "UNIQUE ID FOR USER",
- "_rev": "REFERENCE NUMBER",
- "name": "USER GIVEN NAME",
- "email": "USER EMAIL ID",
- "password": "USER REGISTERED PASSWORD"

USER_IMAGE_DATABASE

- "_id": "UNIQUE ID FOR EACH IMAGE",
- "_rev": "REFERENCE NUMBER FOR EACH DOCUMENT",
- "name": "USER NAME",
- "email": "USER REGISTERED EMAIL ID",
- "image": "IMAGE IN BASE64 FORMAT",
- "datetime": "DATE AND TIME OF CREATION OF DOCUMENT"

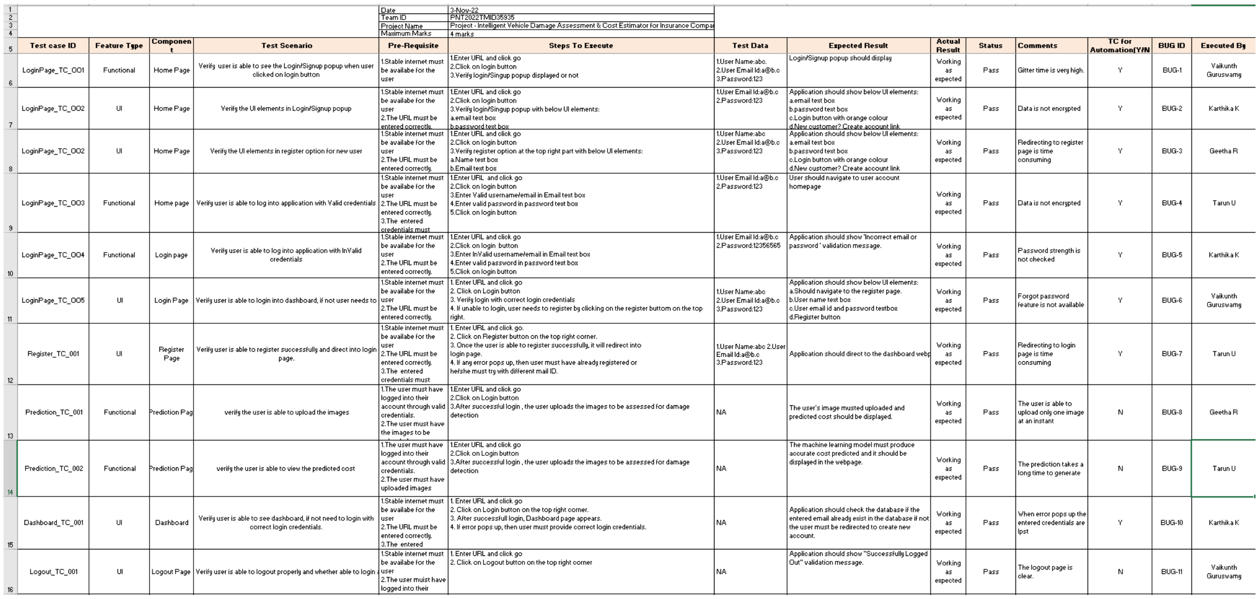
# 8.TESTING

## 8.1 TEST CASES:

| S. NO. | TEST CASES |
|---|---|
| 1. | Verify  user is able to see the Login/Signup popup when user clicked on login button |
| 2. | Verify the UI elements in Login/Signup popup |
| 3. | Verify the UI elements in register option for new user |
| 4. | Verify user is able to log into application with Valid credentials |
| 5. | Verify user is able to log into application with InValid credentials |
| 6. | Verify user can login into dashboard, if not user needs to register. |
| 7. | Verify user can register successfully and direct into login page. |

| | |
|---|---|
| 8. | verify the user is able to upload the images |
| 9. | Verify the user is able to view the predicted cost |
| 10. | Verify user can see dashboard, if not need to login with correct login credentials. |
| 11. | Verify user can logout properly and whether able to login again. |

## 8.2 USER ACCEPTANCE TESTING



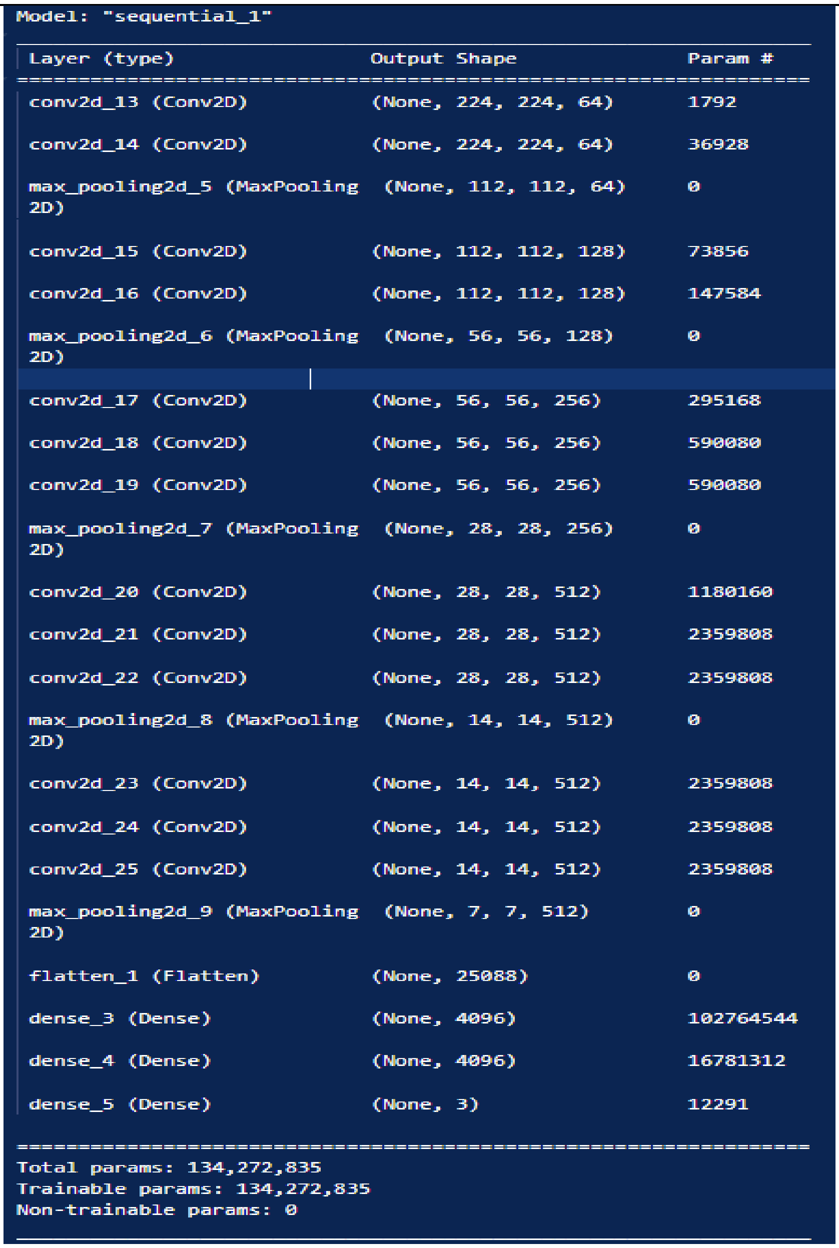
| | |
|---|---|
| Date | 3-Nov-22 |
| Team ID | PNT2022TMID35935 |
| Project Name | Project - Intelligent Vehicle Damage Assessment & Cost Estimator for Insurance Compa |
| Maximum Marks | 4 marks |

| Test case ID | Feature Type | Component | Test Scenario | Pre-Requisite | Steps To Execute | Test Data | Expected Result | Actual Result | Status | Comments | TC for Automation(Y/N | BUG ID | Executed By |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LoginPage_TC_OO1 | Functional | Home Page | Verify user is able to see the Login/Signup popup when user clicked on login button | 1.Stable internet must be availabe for the user | 1.Enter URL and click go<br>2.Click on login button<br>3.Verify login/Singup popup displayed or not | 1.User Name:abc.<br>2.User Email Id:a@b.c<br>3.Password:123 | Login/Signup popup should display | Working as expected | Pass | Gitter time is very high. | Y | BUG-1 | Vaikunth Guruswamy |
| LoginPage_TC_OO2 | UI | Home Page | Verify the UI elements in Login/Signup popup | 1.Stable internet must be availabe for the user<br>2.The URL must be entered correctly. | 1.Enter URL and click go<br>2.Click on login button<br>3.Verify login/Singup popup with below UI elements:<br>a.email text box<br>b.password text box | 1.User Email Id:a@b.c<br>2.Password:123 | Application should show below UI elements:<br>a.email text box<br>b.password text box<br>c.Login button with orange colour<br>d.New customer? Create account link | Working as expected | Pass | Data is not encrypted | Y | BUG-2 | Karthika K |
| LoginPage_TC_OO2 | UI | Home Page | Verify the UI elements in register option for new user | 1.Stable internet must be availabe for the user<br>2.The URL must be entered correctly. | 1.Enter URL and click go<br>2.Click on login button<br>3.Verify register option at the top right part with below UI elements:<br>a.Name text box<br>b.Email text box | 1.User Name:abc<br>2.User Email Id:a@b.c<br>3.Password:123 | Application should show below UI elements:<br>a.email text box<br>b.password text box<br>c.Login button with orange colour<br>d.New customer? Create account link | Working as expected | Pass | Redirecting to register page is time consuming | Y | BUG-3 | Geetha R |
| LoginPage_TC_OO3 | Functional | Home page | Verify user is able to log into application with Valid credentials | 1.Stable internet must be availabe for the user<br>2.The URL must be entered correctly.<br>3.The entered credentials must | 1.Enter URL and click go<br>2.Click on login button<br>3.Enter Valid username/email in Email text box<br>4.Enter valid password in password text box<br>5.Click on login button | 1.User Email Id:a@b.c<br>2.Password:123 | User should navigate to user account homepage | Working as expected | Pass | Data is not encrypted | Y | BUG-4 | Tarun U |
| LoginPage_TC_OO4 | Functional | Login page | Verify user is able to log into application with InValid credentials | 1.Stable internet must be availabe for the user<br>2.The URL must be entered correctly. | 1.Enter URL and click go<br>2.Click on login button<br>3.Enter InValid username/email in Email text box<br>4.Enter valid password in password text box<br>5.Click on login button | 1.User Email Id:a@b.c<br>2.Password:12356565 | Application should show 'Incorrect email or password ' validation message. | Working as expected | Pass | Password strength is not checked | Y | BUG-5 | Karthika K |
| LoginPage_TC_OO5 | UI | Login Page | Verify user is able to login into dashboard, if not user needs to | 1.Stable internet must be availabe for the user<br>2.The URL must be entered correctly. | 1. Enter URL and click go<br>2. Click on Login button<br>3. Verify login with correct login credentials<br>4. If unable to login, user needs to register by clicking on the register buttom on the top right. | 1.User Name:abc<br>2.User Email Id:a@b.c<br>3.Password:123 | Application should show below UI elements:<br>a.Should navigate to the register page.<br>b.User name text box<br>c.User email id and password textbox<br>d.Register button | Working as expected | Pass | Forgot password feature is not available | Y | BUG-6 | Vaikunth Guruswamy |
| Register_TC_001 | UI | Register Page | Verify user is able to register successfully and direct into login page. | 1.Stable internet must be availabe for the user<br>2.The URL must be entered correctly.<br>3.The entered credentials must | 1. Enter URL and click go.<br>2. Click on Register button on the top right corner.<br>3. Once the user is able to register successfully, it will redirect into login page.<br>4. If any error pops up, then user must have already registered or he/she must try with different mail ID. | 1.User Name:abc 2.User Email Id:a@b.c<br>3.Password:123 | Application should direct to the dashboard webp | Working as expected | Pass | Redirecting to login page is time consuming | Y | BUG-7 | Tarun U |
| Prediction_TC_001 | Functional | Prediction Pag | verify the user is able to upload the images | 1.The user must have logged into their account through valid credentials.<br>2.The user must have the images to be | 1.Enter URL and click go<br>2.Click on Login button<br>3.After successful login , the user uploads the images to be assessed for damage detection | NA | The user's image musted uploaded and predicted cost should be displayed. | Working as expected | Pass | The user is able to upload only one image at an instant | N | BUG-8 | Geetha R |
| Prediction_TC_002 | Functional | Prediction Pag | verify the user is able to view the predicted cost | 1.The user must have logged into their account through valid credentials.<br>2.The user must have uploaded images | 1.Enter URL and click go<br>2.Click on Login button<br>3.After successful login , the user uploads the images to be assessed for damage detection | NA | The machine learning model must produce accurate cost predicted and it should be displayed in the webpage. | Working as expected | Pass | The prediction takes a long time to generate | N | BUG-9 | Tarun U |
| Dashboard_TC_001 | UI | Dashboard | Verify user is able to see dashboard, if not need to login with correct login credentials. | 1.Stable internet must be availabe for the user<br>2.The URL must be entered correctly.<br>3.The entered | 1. Enter URL and click go<br>2. Click on Login button on the top right corner.<br>3. After successfull login, Dashboard page appears.<br>4. If error pops up, then user must provide correct login credentials. | NA | Application should check the database if the entered email already exist in the database if not the user must be redirected to create new account. | Working as expected | Pass | When error pops up the entered credentials are lpst | Y | BUG-10 | Karthika K |
| Logout_TC_001 | UI | Logout Page | Verify user is able to logout properly and whether able to login a | 1.Stable internet must be availabe for the user<br>2.The user muist have logged into their | 1. Enter URL and click go<br>2. Click on Logout button on the top right corner | NA | Application should show "Successfully Logged Out" validation message. | Working as expected | Pass | The logout page is clear. | N | BUG-11 | Vaikunth Guruswamy |

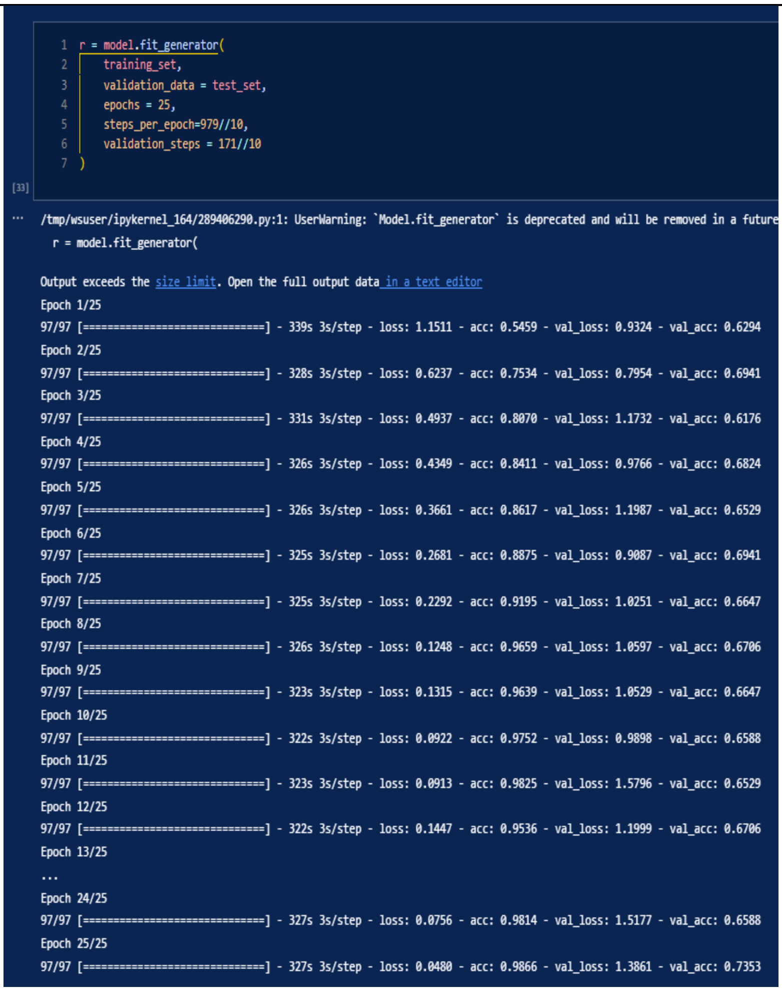
# 9. RESULTS

## 9.1 Performance Metrics

| S.No. | Parameter | Values | Screenshot |
|---|---|---|---|

|  |  |  |  |
|---|---|---|---|
| 1. | Model Summary |  |  |

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_13 (Conv2D)          (None, 224, 224, 64)      1792

 conv2d_14 (Conv2D)          (None, 224, 224, 64)      36928

 max_pooling2d_5 (MaxPooling  (None, 112, 112, 64)     0
 2D)

 conv2d_15 (Conv2D)          (None, 112, 112, 128)     73856

 conv2d_16 (Conv2D)          (None, 112, 112, 128)     147584

 max_pooling2d_6 (MaxPooling  (None, 56, 56, 128)      0
 2D)

 conv2d_17 (Conv2D)          (None, 56, 56, 256)       295168

 conv2d_18 (Conv2D)          (None, 56, 56, 256)       590080

 conv2d_19 (Conv2D)          (None, 56, 56, 256)       590080

 max_pooling2d_7 (MaxPooling  (None, 28, 28, 256)      0
 2D)

 conv2d_20 (Conv2D)          (None, 28, 28, 512)       1180160

 conv2d_21 (Conv2D)          (None, 28, 28, 512)       2359808

 conv2d_22 (Conv2D)          (None, 28, 28, 512)       2359808

 max_pooling2d_8 (MaxPooling  (None, 14, 14, 512)      0
 2D)

 conv2d_23 (Conv2D)          (None, 14, 14, 512)       2359808

 conv2d_24 (Conv2D)          (None, 14, 14, 512)       2359808

 conv2d_25 (Conv2D)          (None, 14, 14, 512)       2359808

 max_pooling2d_9 (MaxPooling  (None, 7, 7, 512)        0
 2D)

 flatten_1 (Flatten)         (None, 25088)             0

 dense_3 (Dense)             (None, 4096)              102764544

 dense_4 (Dense)             (None, 4096)              16781312

 dense_5 (Dense)             (None, 3)                 12291

=================================================================
Total params: 134,272,835
Trainable params: 134,272,835
Non-trainable params: 0
_________________________________________________________________
```

|  |  |  |  |
|---|---|---|---|
| 2. | Accuracy | Training Accuracy - 98.66%<br><br>Validation Accuracy - 73.53% |  |

```
r = model.fit_generator(
    training_set,
    validation_data = test_set,
    epochs = 25,
    steps_per_epoch=979//10,
    validation_steps = 171//10
)
```

```
[33]

/tmp/wsuser/ipykernel_164/289406290.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future
  r = model.fit_generator(

Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/25
97/97 [==============================] - 339s 3s/step - loss: 1.1511 - acc: 0.5459 - val_loss: 0.9324 - val_acc: 0.6294
Epoch 2/25
97/97 [==============================] - 328s 3s/step - loss: 0.6237 - acc: 0.7534 - val_loss: 0.7954 - val_acc: 0.6941
Epoch 3/25
97/97 [==============================] - 331s 3s/step - loss: 0.4937 - acc: 0.8070 - val_loss: 1.1732 - val_acc: 0.6176
Epoch 4/25
97/97 [==============================] - 326s 3s/step - loss: 0.4349 - acc: 0.8411 - val_loss: 0.9766 - val_acc: 0.6824
Epoch 5/25
97/97 [==============================] - 326s 3s/step - loss: 0.3661 - acc: 0.8617 - val_loss: 1.1987 - val_acc: 0.6529
Epoch 6/25
97/97 [==============================] - 325s 3s/step - loss: 0.2681 - acc: 0.8875 - val_loss: 0.9087 - val_acc: 0.6941
Epoch 7/25
97/97 [==============================] - 325s 3s/step - loss: 0.2292 - acc: 0.9195 - val_loss: 1.0251 - val_acc: 0.6647
Epoch 8/25
97/97 [==============================] - 326s 3s/step - loss: 0.1248 - acc: 0.9659 - val_loss: 1.0597 - val_acc: 0.6706
Epoch 9/25
97/97 [==============================] - 323s 3s/step - loss: 0.1315 - acc: 0.9639 - val_loss: 1.0529 - val_acc: 0.6647
Epoch 10/25
97/97 [==============================] - 322s 3s/step - loss: 0.0922 - acc: 0.9752 - val_loss: 0.9898 - val_acc: 0.6588
Epoch 11/25
97/97 [==============================] - 323s 3s/step - loss: 0.0913 - acc: 0.9825 - val_loss: 1.5796 - val_acc: 0.6529
Epoch 12/25
97/97 [==============================] - 322s 3s/step - loss: 0.1447 - acc: 0.9536 - val_loss: 1.1999 - val_acc: 0.6706
Epoch 13/25
...
Epoch 24/25
97/97 [==============================] - 327s 3s/step - loss: 0.0756 - acc: 0.9814 - val_loss: 1.5177 - val_acc: 0.6588
Epoch 25/25
97/97 [==============================] - 327s 3s/step - loss: 0.0480 - acc: 0.9866 - val_loss: 1.3861 - val_acc: 0.7353
```

## 10. ADVANTAGES & DISADVANTAGES

### 1. ADVANTAGES

- ➤ The deep learning algorithm will analyse images in real time and identifies the presence of any damage

- ➤ Cost estimation by the application is accurate
- ➤ all uploaded images and user details will be stored securely in cloud
- ➤ user friendly and easy understandability in accessing the function is deployed
- ➤ effective and fast computation in detecting damage
- ➤ images with various orientation of vehicle will also be analysed

2. DISADVANTAGES

- ➤ user cannot access the portal in remote places where internet connectivity is not present.
- ➤ Blurred image or image with poor lighting will not yield accurate results
- ➤ application file size is large ,there by leading to longer computation time

## 11.CONCLUSION:

the artificial intelligence and automation are places everywhere, manual error need to reduced both in marketing and financial sector as well.this application will pave path for users to effectively determine the degree of damage and cost incurred without any human interactions.this software as a service (SaaS)will be a stepping stone to establish and server to client authenticated interaction to yield accurate results.Data security plays a vital role in the information era,aligning with CIA principle of cryptography user data is stored securely in cloud .Aligning with the demands this web application will precisely solve user queries.

## 12.FUTURE SCOPE:

1.The damage detection can be provided to all the insured clients to reach the stable base and then extend the service of cost estimation to the insurers.

2.Make use of advanced machine learning techniques to analyse the damaged vehicle with high accuracy levels and keep on improving the learning ability of the model.

3.In addition to the webpage a mobile application can be created where the real time images and videos of the vehicle can be extracted and insurance cost can be

estimated.
4.Implementing further features to address user friendly demands.
5.Implementing better customer service by sorting out user issues.

## 13.APPENDIX:

main python file:

```
import base64
import datetime
import os
import re
from io import BytesIO

import cv2
import flask
import flask_login
import numpy as np

from cryptography.fernet import Fernet
key = Fernet.generate_key()
f= Fernet(key)
from datetime import datetime

from cloudant.client import Cloudant
from cloudant.error import CloudantException
from cloudant.result import Result, ResultByKey
from flask import Flask, app, render_template, request
from PIL import Image


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(
    'API KEY','API_KEY',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.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.strptime(i['doc']['datetime'],'%m/%d/%Y, %H:%M:%S')
            o = 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 = {
                                                                        '_id':email.replace('@',").replace('.',"),
        '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'))

from quickstart import send_mail
```

```
@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)
        send_mail(email,"Thanks for registering","thank you")
        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('/forgotpassword',methods=['GET','POST'])
def forgotpassword():
    data = database_retrieval()

    #flask.flash('23232','info')
    #flask_login.logout_user()

    if(flask.request.method=='POST'):
        reset_email = flask.request.form['email']
        #print(reset_email)
        print(data)
        if(reset_email in data.keys()):
            #user = User()
            #user.id=reset_email
            #token = user.token_gen()
            current_time = datetime.now()
            d =
f'{reset_email},{current_time.year},{current_time.month},{current_time.day},{cu
rrent_time.hour},{current_time.month},{current_time.second},{current_time.micr
osecond}'
            token = f.encrypt(bytes(d,'utf-8'))
            #k.append(token)
            #print(token)
            send_mail(reset_email,"password reset",f"Reset password URL is
{flask.url_for('resetpassword',token=token, _external=True)}")
        else:
            print('#########################')
            pass

    return render_template('forgotpassword.html')

b,token1=False,'a'
@app.route('/resetpassword/<token>', methods=["GET", "POST"])
def resetpassword(token):
    global b,token1
    import copy
    if flask.request.method=="GET":
        token1 = copy.copy(token)
        #print("^^^^^^^^^^^^^^^^^^^^^^^")
        #print(token1)

        token1 = f.decrypt(bytes(token1,'utf-8')).decode('utf-8')
        token1 = token1.split(',')
        print(token1)
        generated_date = datetime(int(token1[1]),int(token1[2]),int(token1[3]),int(token1[4]),int(token1[5]),int(token1[6]),int(token1[7]))
        print(generated_date)
        if((datetime.now()-generated_date).total_seconds()<30*60):
            b=True

    data = database_retrieval()

    if flask.request.method=="POST" and b:

        #token_email = user.verify_token(token)
        print(token1)
        print(data[token1[0]])
        print('password resetted 3333333333333333333333333333333333333333333333333333333333333333333333333')
        #data[token1[0]]['password']=flask.request.form['password']
        doc = user_database[token1[0].replace('@',").replace('.',")]
        doc['password']=flask.request.form['password']
        doc.save()

        #user_database.save()

        return flask.redirect(flask.url_for('login'))
    return render_template('resetpassword.html')




@app.route('/prediction',methods = ['GET','POST'])
@flask_login.login_required
def prediction():
    from tensorflow.keras.models import load_model

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

    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.i
d,imgstr)
        data = image_database_retrieval()
        print(flask_login.current_user.id)

#print(len(base64.b64decode(data[flask_login.current_user.id]['image'].strip())))
        image =
Image.open(BytesIO(base64.b64decode(data[flask_login.current_user.id]['image'])
))
        img_retrived = np.array(image)
        '''img_retrived =
np.asarray(base64.b64decode(data[flask_login.current_user.id]['image']))
        print(data[flask_login.current_user.id]['image'])
        print(img_retrived.shape)'''
        #img_retrived = np.resize(img_retrived,(244,244))
        img_retrive = Image.fromarray(img_retrived)
        img_retrive.save('static\imagedata\sae.png')
        '''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',value =
result1+' '+result2+' '+value)
        #,imag=img_retrived)

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

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

**SENDING MAIL PYTHON CODE:**

```
from__future__import print_function

import os.path

from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

from googleapiclient.discovery import build
from google.oauth2.credentials import Credentials
```

```
import base64

from email import encoders
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
import mimetypes

import os
def verification():
# If modifying these scopes, delete the file token.json.
    SCOPES = ['https://mail.google.com/']



    def main():
        """Shows basic usage of the Gmail API.
        Lists the user's Gmail labels.
        """
        creds = None
        # The file token.json stores the user's access and refresh tokens, and is
        # created automatically when the authorization flow completes for the first
        # time.
        if os.path.exists('token.json'):
            creds = Credentials.from_authorized_user_file('token.json', SCOPES)
        # If there are no (valid) credentials available, let the user log in.
        if not creds or not creds.valid:
            if creds and creds.expired and creds.refresh_token:
                creds.refresh(Request())
            else:
                flow = InstalledAppFlow.from_client_secrets_file(
                    'credentials.json', SCOPES)
                creds = flow.run_local_server(port=0)
            # Save the credentials for the next run
            with open('token.json', 'w') as token:
                token.write(creds.to_json())

        try:
            # Call the Gmail API
            service = build('gmail', 'v1', credentials=creds)
            results = service.users().labels().list(userId='me').execute()
            labels = results.get('labels', [])

            if not labels:
                print('No labels found.')
                return
            print('Labels:')
            for label in labels:
                print(label['name'])

        except HttpError as error:
            # TODO(developer) - Handle errors from gmail API.
            print(f'An error occurred: {error}')



    main()
if(not 'token.json' in os.listdir('.')):
    verification()
print('user verified. token is existing ')
def send_mail(to, subject, body, format='plain', attachments=[]):
    creds = None
    SCOPES = ['https://mail.google.com/']
    print(os.getcwd())
    creds = Credentials.from_authorized_user_file('token.json', SCOPES)
```

```
service = build('gmail', 'v1', credentials=creds)

file_attachments = attachments

#html = ''
#with open('message.html') as msg:
#    html += msg.read()

#create email
mimeMessage = MIMEMultipart()
mimeMessage['to'] = to
mimeMessage['subject'] = subject
#mimeMessage.attach(MIMEText(html,'html'))
mimeMessage.attach(MIMEText(body, format))

for attachment in file_attachments:
    content_type, encoding = mimetypes.guess_type(attachment)
    main_type, sub_type = content_type.split('/', 1)
    file_name = os.path.basename(attachment)

    with open(attachment, 'rb') as f:
        myFile = MIMEBase(main_type, sub_type)
        myFile.set_payload(f.read())
        myFile.add_header('Content-Disposition', attachment, filename=file_name)
        encoders.encode_base64(myFile)

    mimeMessage.attach(myFile)


raw_string = base64.urlsafe_b64encode(mimeMessage.as_bytes()).decode()
```

```
    message = service.users().messages().send(
        userId='me',
        body={'raw': raw_string}).execute()

    return message
```

dashboard.html

```
<html>
  <head>
    <title>
      Intelligent Vehicle Damage Assessment and Cost Estimator for insurance
      Companies
    </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;
}
#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;
}
.container {
  display: flex;
}
#vehicle_img {
  margin-top: 4%;
  margin-left: 5%;
}
#topic_content {
  font-family: Georgia;
  font-size: large;
  padding-top: 4%;
  color: cadetblue;
  padding-right: 10%;
}
.pname1 {
  margin-top: 3%;
  font-weight: 600 !important;
  font-size: large;
  color: darkorange !important;
```

```
    }
    .login_prediction {
      display: flex;
    }
    #login_details {
      padding-left: 10%;
    }
    #signin {
      text-align: center;
      padding-bottom: 10%;
      font-size: large;
    }
    #predict {
      text-align: right;
    }
    #blink {
      color: red;
      animation: blinker 0.9s linear infinite;
      font-weight: bold;
    }
    @keyframes blinker {
      50% {
        opacity: 0;
      }
    }
  </style>
</head>
<body onload="flashMessage()">
  <script>
    function flashMessage(){
      if("{{flash_message}}" == "True"){
      alert("account created successfully")
```

```
        }
        if("{{flash_message}}" == "Fals"){
                alert("invalid credentials")
        }
        if("{{flash_message}}" == "Fal"){
          alert("Logged in successfully")
        }
      }
    </script>
    <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('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>
    <div class="container">
```

```
  <div id="vehicle_img">
    <img
      src="/static/images/damage 1.png"
      alt="Damaged Vehicle"
      width="80%"
      height="auto"
    />
  </div>
  <div id="topic_content">
    <p>
      <i>
        Accidents and minor vehicle damage are quite commonplace in the
        automotive sector. However, issues crop up only when there is an
        insurance claim.
        <b>Vehicle Damage detection </b> uses algorithms to automatically
        detect a vehicle's exterior body and assess its injuries and the
        extent of the damage. Here damage to the vehicle are identified not
        only for insurance purpose but also for repair cost estimation.
      </i>
    </p>
  </div>
</div>
<div id="slider_text">
  <marquee
    class="pname1"
    direction="left"
    behavior="scroll"
    scrollamount="10"
    >Login to know more about the level of damage and cost
    estimation</marquee
  >
</div>
```

```
<div class="login_prediction">
  <div id="login_details" style="padding-top: 5%">
    <div id="signin">
      <b><i>Log in</i></b>
    </div>
    <form action="dashboard" method="POST">
      <input
        type="text"
        name="email"
        id="email"
        placeholder="Enter registered email ID"
        style="width: 150%; height: 35px"
      /><br />
      <br />
      <input
        type="password"
        name="password"
        id="password"
        placeholder="Enter Password"
        style="width: 150%; height: 35px"
      /><br />
      <br />
      <input
        type="submit"
        name="submit"
        id="submit"
        value="Login"
        style="
          width: 150%;
          height: 35px;
          text-align: center;
          background-color: black;
```

```
            color: white;
          "
        />
      </form>
    </div>
    <div id="predict" style="text-align: center;margin-left: 25%;">
      <p>
        <b>To predict the cost for the occured damage </b>
      </p>
      <!--<p id="blink">Click Here!</p>-->
    </div>
  </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>
```

FORGOTPASSWORD.HTML

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

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

INDEX.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

```
<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>
```

LOGOUT.HTML

```
<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>
```

PREDICTION.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 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>
```

REGISTER.HTML

```
<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>
```

RESETPASSWORD.HTML

```
<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>
```

MACHINE LEARNING SEGMENT

```
# Importing Libraries
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import sys
from tensorflow.keras.models import load_model
import cv2
from skimage.transform import resize
# Importing Dataset

import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def___iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='ibm_api_key_id',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'Bucket'
object_key = 'Dataset.zip'
```

```
streaming_body_1 = cos_client.get_object(Bucket=bucket,
Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the
possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/

# Unzipping Dataset
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()), 'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
import os
os.listdir('.')
train_datagen =
ImageDataGenerator(rescale=1./255,shear_range=0.1,zoom_range=0.1,horizontal_
flip=True)
val_datagen = ImageDataGenerator(rescale=1./255)
# MODEL FOR BODY TYPE DETECTION
trainPath = '/home/wsuser/work/Dataset/body/training'
testPath = '/home/wsuser/work/Dataset/body/validation'
training_set =
train_datagen.flow_from_directory(trainPath,target_size=(244,244),batch_size=10,
class_mode='categorical')
test_set =
train_datagen.flow_from_directory(testPath,target_size=(244,244),batch_size=10,c
lass_mode='categorical')
```

```
training_set.class_indices
# Declaring Model Variable
vgg=VGG16(input_shape=(244,244,3),weights='imagenet',include_top=False)



for layer in vgg.layers:
  layer.trainable=False


x=Flatten()(vgg.output)


prediction=Dense(3,activation='softmax')(x)


model=Model(inputs=vgg.input,outputs=prediction)
model.summary()
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['acc'])
# Training model
r = model.fit_generator(
    training_set,
    validation_data = test_set,
    epochs = 25,
    steps_per_epoch=979//10,
    validation_steps = 171//10
)
model.save('body.h5')
!tar -zcvf body.tgz body.h5
ls -1
!pip install watson-machine-learning-client --upgrade
```

```
# Connecting with IBM CLOUD
from ibm_watson_machine_learning import APIClient
wml_credentials = {"url":"https://us-south.ml.cloud.ibm.com", "apikey":"apikey"}
client = APIClient(wml_credentials)
def guid_from_space_name(client,space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if
item['entity']["name"]==space_name)['metadata']['id'])
space_uid = guid_from_space_name(client, 'spacename')
#space_uid
client.set.default_space(space_uid)
software_spec_uid =
client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
#software_spec_uid
client.software_specifications.list()
model_details = client.repository.store_model(model = 'body.tgz' , meta_props = {
    client.repository.ModelMetaNames.NAME : "body",
    client.repository.ModelMetaNames.TYPE : "tensorflow_rt22.1",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid
})
model_id = client.repository.get_model_id(model_details)

client.repository.download(model_id, 'body_cloud.tar.gz')

model_body = load_model('body.h5')
# MODEL FOR LEVEL TYPE DETECTION
trainPath = '/home/wsuser/work/Dataset/level/training'
testPath = '/home/wsuser/work/Dataset/level/validation'
training_set =
train_datagen.flow_from_directory(trainPath,target_size=(244,244),batch_size=10,
class_mode='categorical')
```

```
test_set =
train_datagen.flow_from_directory(testPath,target_size=(244,244),batch_size=10,class_mode='categorical')
training_set.class_indices
# Declaring Model Variable
vgg=VGG16(input_shape=(244,244,3),weights='imagenet',include_top=False)



for layer in vgg.layers:
  layer.trainable=False


x=Flatten()(vgg.output)


prediction=Dense(3,activation='softmax')(x)


model1=Model(inputs=vgg.input,outputs=prediction)
model1.summary()
model1.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['acc'])
# Training model
y = model1.fit_generator(
    training_set,
    validation_data = test_set,
    epochs = 25,
    steps_per_epoch=979//10,
    validation_steps = 171//10
)
model1.save('level.h5')
!tar -zcvf level.tgz level.h5
```

```
ls -1
model_details = client.repository.store_model(model = 'level.tgz' , meta_props = {
    client.repository.ModelMetaNames.NAME : "level",
    client.repository.ModelMetaNames.TYPE : "tensorflow_rt22.1",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid
})
model_id = client.repository.get_model_id(model_details)

client.repository.download(model_id, 'level_cloud.tar.gz')

model_body = load_model('level.h5')
os.listdir('.')
client.repository.download('model_id','body_cloud.tar.gz')
client.repository.download('model1_id','level_cloud.tar.gz')
```

DEMO LINK:

https://drive.google.com/file/d/14y-PeMgXgOGKnZlrTsYeTFNnWOyOcL0s/view?usp=share_link