



**POC Documentation**

# **AInspecto**

**AI Vehicle Inspection Module**

## Table of contents

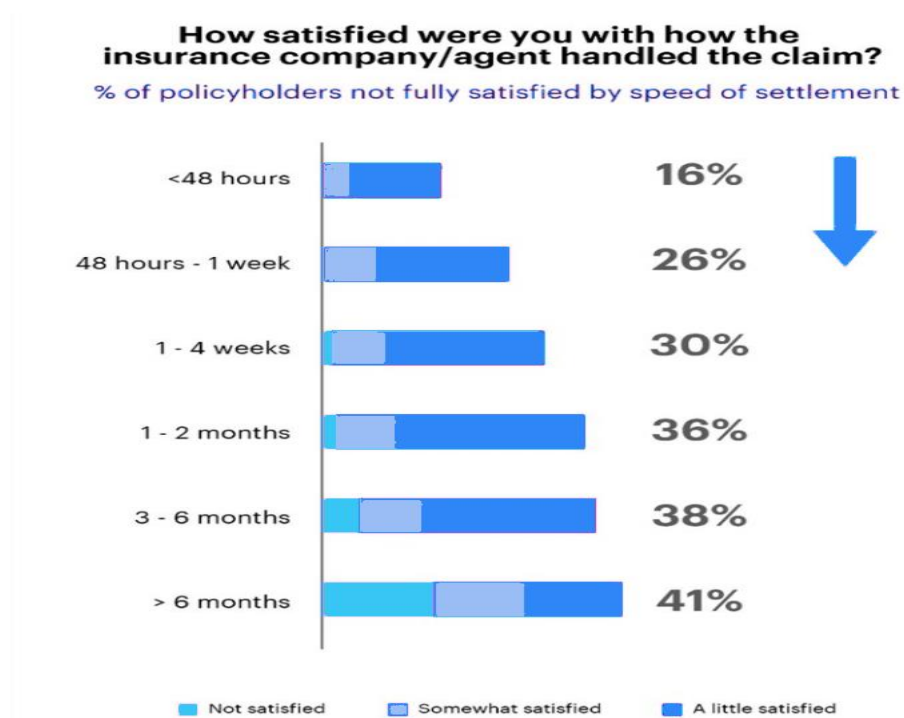
1. Problem statement.....	3
2. Abstract	
3. Methodology.....	5
4. Flow chart.....	5
5. Data collection and Data preparation .....	7
a. Annotations.....	7
6. Image Validations.....	8
a. Input image validation.....	8
b. 360 degrees image validation.....	9
7. License plate detection.....	10
8. Data Modelling.....	11
a. Image Validation.....	11
b. License plate recognition.....	11
i. License plate detection.....	11
ii. License plate recognition.....	11
c. Damage Assessment.....	12
d. Damage Severity.....	12
9. Streamlit Application.....	13
a. About.....	13
b. Setup.....	13
c. Example.....	13
10. API Integrations.....	15
a. Upload Image.....	15
b. Check User.....	16
c. Damage Assessment.....	17
d. Get Data Frame.....	18
e. Get Image.....	18
11. Future Scope.....	19
12. Challenges.....	19
13. Conclusion.....	20

## Problem Statement:

To Streamline vehicle pre inspection processes by making use of state of the art deep learning object detection models and neural network capabilities to ensure a seamless, cost effective, quick damage detection and report generation that helps in speeding up the renewal and claims process for any vehicle insurance client/provider.

## Why to streamline:

- **Vehicle insurance costs** are expected to continue rising through 2023; Third-party insurance for cars and bikes will become costlier from April 1, with the insurance regulator raising the premium rates by up to 40%. Every year, IRDAI revises premium rates, taking into account the number of claims made.
- **Customer satisfaction** is suffering due to slower car repairs and claims processing.



- Insurance companies have been accused of **pressuring repair facilities to cut corners**; Car repair shops say auto insurance companies are coercing them to use cheap parts and sometimes dangerous practices to fix vehicles involved in accidents

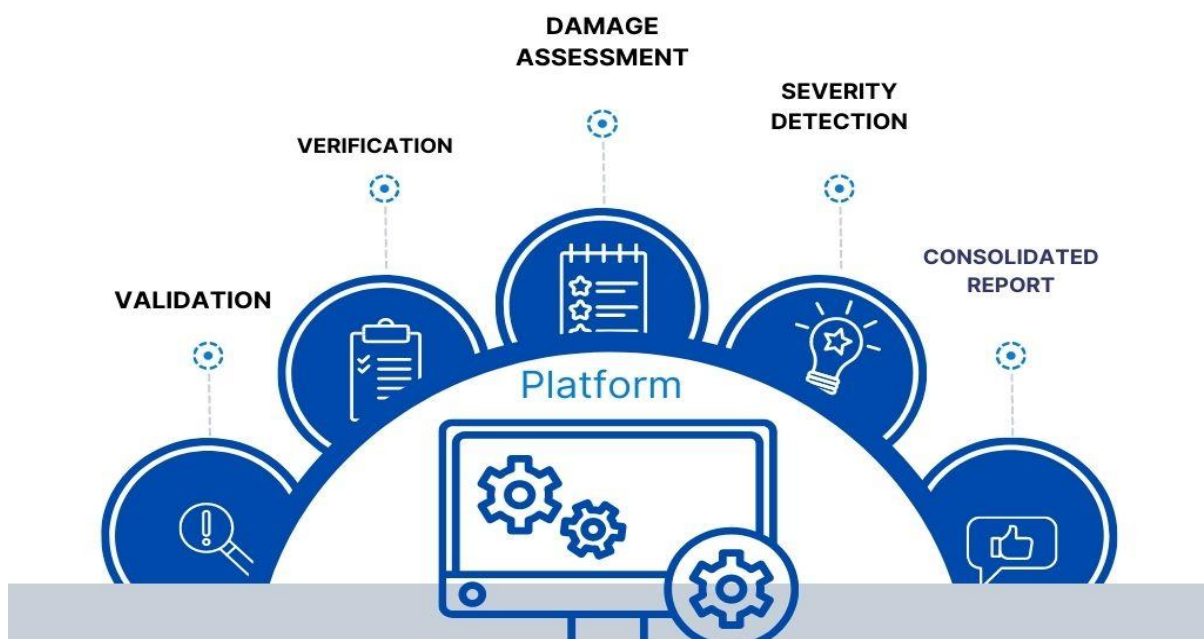
## Abstract:

A Vehicle Inspection is considered as an important aspect for insurance providers to calculate the annual premium and generate claim estimates in case of damage. This inspection is done in many insurance companies to process the insurance claims or renewals. Currently the inspection process is done manually which consumes a lot of time, effort and cost to the insurance providers. So to improve the inspection process and customer satisfaction **AInspecto** aims to streamline the inspection process by automatic damage and severity detection along with cost estimation of claim. These features are available in the form of services that can be used by any insurance provider from their native application used. This provides a seamless usage of the features to the clients without many changes to their current process followed. The main functionalities focused on are Damage detections and their severity on each side of the vehicle.

## Objectives:

- Enhancing the efficiency and reliability of vehicle inspections for insurance claims.
- It aims to automate the assessment of vehicle damages.
- Reduce human error, and expedite the claims settlement process.
- Ultimately, the goal is to improve customer satisfaction, reduce fraudulent claims, and provide a more seamless experience for insurance policyholders.

## Feature Mapping



## Methodology:

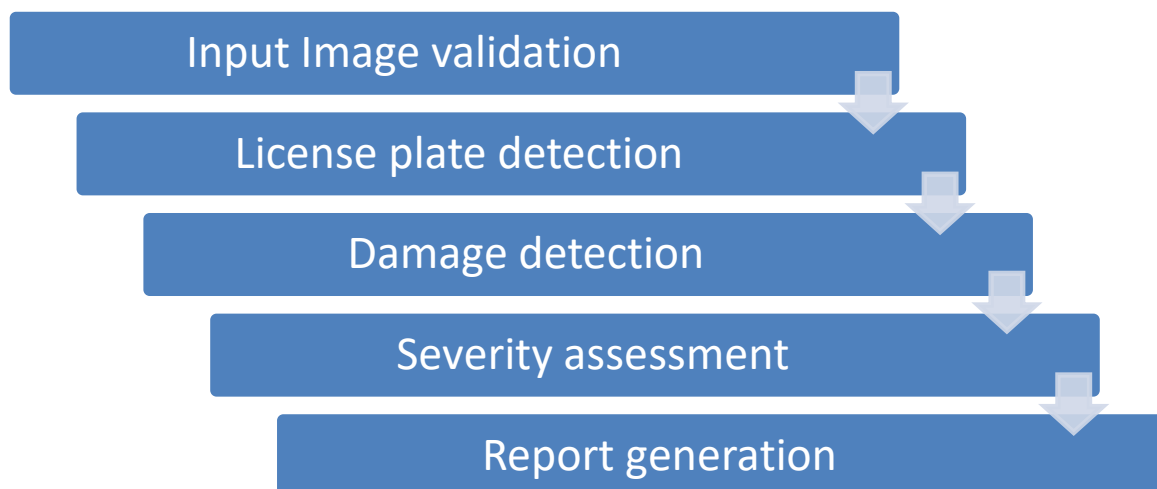
As stated in the abstract of the project, to streamline the inspection process, we are going to use deep learning models such as CNN, YOLO, and VGG etc. All the models are custom trained on datasets that were prepared based on our requirement to detect different damages such as dent, scratch and intense-damages. We have trained models on YOLO v8 for license plate detection and damage detections. We have trained CNN and VGG pre-trained models on “ImageNet” to train damage severity assessment and image validations.

Typically, the project flow consists of the following activities they are

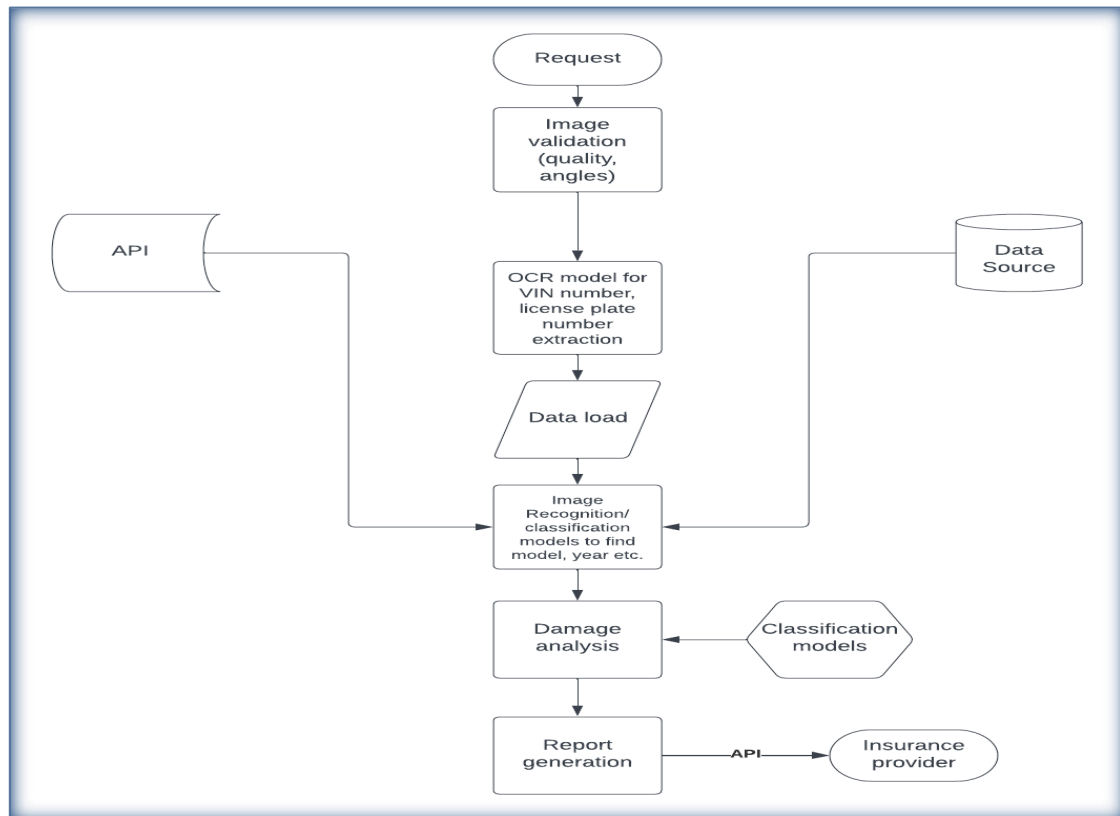
- Dataset collection
- Custom annotations
- Input image validations
- License plate detection and verification
- 360 degree image validations
- Damage detection
- Severity assessment
- Consolidated report generation
- API integrations to use all these services

## Flowchart:

Flow chart consists of graphical representation of the features mentioned and their connectivity with each other providing a streamlined flow of events that can ensure the working of project to inspect the vehicle damages and generate a report. The sequential processing of events are shown in the below graphic.



The technical flow chart of the project is shown below



This flow chart describes the features that will be added as API's which will be used by the insurance providers or intermediate platforms associated.

In brief, flow of the functionalities is explained as follows:

When there is a request from any insurance provider or its associated platforms for vehicle inspection, the first step is to validate the vehicle; so we request a back view image of the vehicle and detect the license plate with our model and read the license plate with easy OCR model, then verify this information with the insurance provider to validate the vehicle and owner details.

Then we request the 360 degree images of the vehicle from the user, further validations are performed on these images whether they belong to the same vehicle, quality of the images.

Once the validations are satisfied then these images are given to two different models to access damage severity and detect the damages. These results are tabulated and returned along with the damage detected images back to the insurance provider as API response.

These functionalities complete the scope of renewal inspection process, the report produced can be used to renew the policy. The claims process requires another module that compares the damages during renewal and during claim. This module will be added to scope for the next version of API's.

## Data collection and Data preparation:

In this project the required images are collected from four different sources. Only raw images were gathered from the data sources and dataset preparation was customized based on our requirement. The major data sources are

1. Google images
2. Roboflow
3. Kaggle
4. Open images Dataset



Open Images  
Dataset

kaggle



### Annotations:

As the requirement demands customized class labels to identify the damages, images of different damages are collected and annotated into three class labels namely

- Dent
- Scratch and
- Intense-damage



Dent



Scratch



Intense-damage

We have used **CVAT.ai** as our primary annotating tool and labelled around 3000+ images dataset with the three class labels decided. There are two versions of annotated datasets that are generated they are

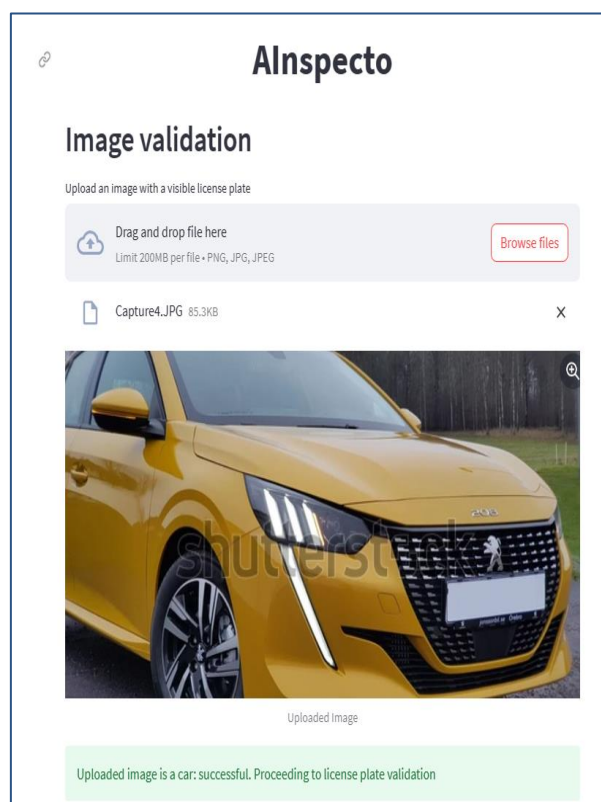
- **YOLO 1.1:** This format gives the image annotations in the form of text files that are used to train a YOLO model. Each image will have its respective .txt file having the class annotations.
- **Pascal VSAC:** This format gives the image annotations in the form of xml files that are used to train a CNN models. Each image will have its respective .xml file having the class annotations.

## Image Validations:

Image validations play a major role in successful functioning of this product, as all the major functionalities are performed on vehicle images. We have performed two different image validations in this version they are

### Input image validation:

Initially when the back view image of the vehicle is requested for vehicle detail validation, there is a chance of uploading any random image not necessarily of a vehicle. So we are using a pre trained **VGG16** model that is trained on '**imagenet**' to verify whether the uploaded image is a car or not. If the uploaded image is not a car, user is requested to re-upload the image to continue. If the validation is satisfied then control moves to the next validation that is license plate detection.



Success Scenario

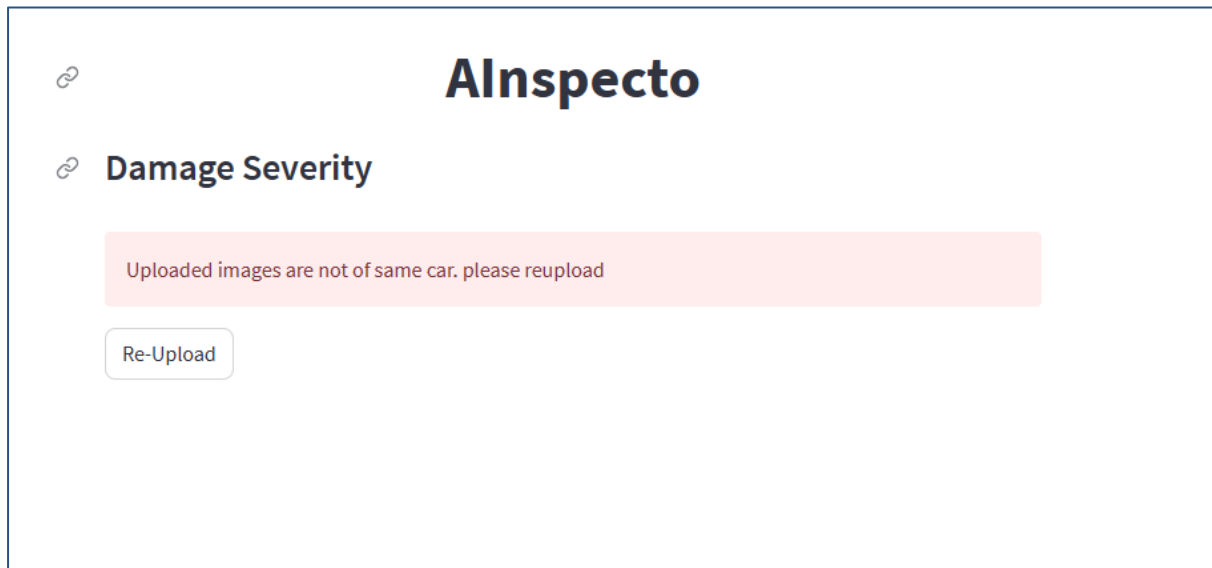
In the success scenario the uploaded image is a car, so it will be proceeding to license plate detection and validations.

In failure scenario the uploaded image might not be a car, so there will be an error message to re upload the image to continue to license plate detection.



### 360 degree image validation:

While uploading images of four sides of the vehicle, there can be a chance of uploading different car images. This issue has been addressed where we validated the given images with a pre-trained model and validate whether they are from the same car or not. If they are not of the same car then user has to re-upload the images to continue to damage assessment. If the images are of same car then it will proceed to damage assessment.



Here the images uploaded were not of the same car, so we have an error message to re-upload the images of same car to continue the damage assessment.

## License plate detection:

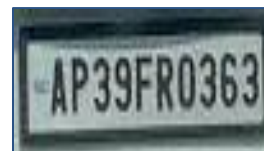
One of the important modules in this project is the license plate detection and validation. After the input validation is satisfied image is fed into a YOLO v8 model which was custom trained to detect the license plates from any image having a visible license plate.

This detected license plate is cropped from the original image and will go through a series of image processing functions such as gray scaling, thresholding, resizing etc.

Then this is fed into an OCR model to read the plate number from the processed license plate. We have used Easy OCR pre-trained model to read the license plate and send the result to the insurance provider for vehicle validation and verification before moving to damage detection and assessment module.

This will return all the details of the vehicle, owner, make, model, year, insurance policy, insurance expiry etc. These are used to validate the user on authenticating the assessment module of the owner's car registered with the insurance company or not. Any fraudulent activity can be detected in this phase before moving forward to the assessment module.

Here are the images of the input image, cropped detected license plate and the details corresponding to the license plate.



### License plate detection

License plate detection successful

License plate number:

AP39FR0363

Please confirm to move to vehicle detail verification

Confirm

## AInspecto

### Vehicle Details

	Field	Value
0	License plate	AP39FR0363
1	Model	Model XYX
2	Year	2020
3	Owner name	Popoye
4	Gender	Male
5	Country	India
6	License Expiry	01-01-2028
7	Insurance policy	Gold plus
8	Insurance Expiry	03-06-2024

Continue

## Data Modelling:

In this project we have used models for image validations, license plate detections and damage assessments.

We have trained the following models to achieve the inscribed scope

### Image Validation:

#### Model used: VGG-16 model

VGG-16 model is used for image validation. Here the user needs to upload an image with a visible license plate. The uploaded image is validated to determine if it is a car image or not. If the validation is successful, the step is updated to proceed to next step. So in this process we use VGG-16 model to validated whether the uploaded image is car or not. VGG-16 is a convolutional neural network that is 16 layers deep which is used for image recognition.

### License plate recognition:

#### Model used: YOLO model

YOLO model is used for License plate recognition. This Section is divided into 2 parts:-

1. To get the cropped license plate image from the uploaded car image.
2. To extract the text from the cropped license plate image.

- **License plate detection:**

So initially our first aim is to detect the license plate from the given image. So here we use YOLO model for license plate detection. YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

So once the license plate is cropped it is saved to a particular path.

- **License plate recognition:**

Now we read the cropped image from the saved image path. Now we apply OCR (optical character recognition) model to recognize the text from the image. Here we are using EasyOCR for text extraction. EasyOCR is actually a python package. It detects the text from images. EasyOCR supports 42+ languages for detection purposes. EasyOCR is a powerful and user-friendly OCR library that can detect and extract text from various image formats. Once we get the License plate number we ask the user to confirm whether the extracted number is correct or not and then we can proceed to next step.

## Damage Assessment:

### Model used: YOLO Model

After getting the vehicle details our next step is damage assessment for which user is allowed to upload images from different angles (left, right, front, and back). Now the uploaded images are saved. Now user clicks on access damage now it moves to next step.

In this stage, we are able to detect the damages on the four sides of the car and label them into the three classes annotated; dent, scratch and intense-damage.





## Damage Severity:

### Model used: CNN Model

Once the damages are identified then the images are processed through this model to identify the severity of the damage on each side. These results are merged with damage detection results and the final results are shown/send back to the user/insurance provider.

### AIInspecto

Damage Severity



left

right

front

back

	Vehicle side	Result
0	left	minor
1	right	minor
2	front	minor
3	back	minor

## Streamlit Application:

We have used streamlit library to integrate the python functionality in the front end.

### About:

Streamlit is an open-source Python library that simplifies the process of creating and deploying interactive web applications for data exploration and visualization. It allows data scientists and developers to turn data scripts into shareable web apps with just a few lines of Python code. With Streamlit, users can create web applications to showcase data visualizations, machine learning models, and other data-driven insights, making it an excellent tool for communicating data analysis to a broader audience.

### Setup:

To get started with Streamlit, you first need to install the library using pip. You can create a virtual environment to keep dependencies separate from other projects:

**pip install streamlit**

Once installed, you can create a new Python script and import Streamlit to start building your interactive apps.

### Example:

```
# test.py
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

# Load data
data = pd.read_csv("data.csv")

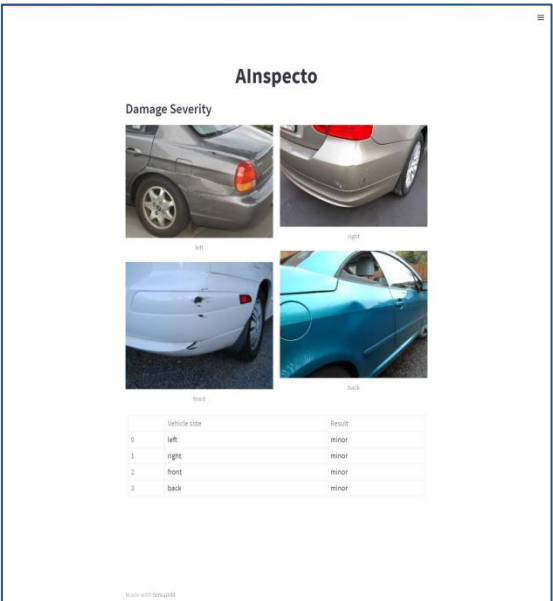
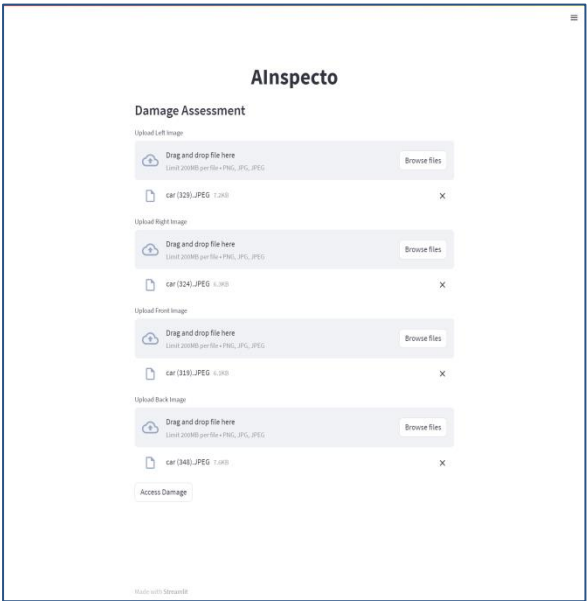
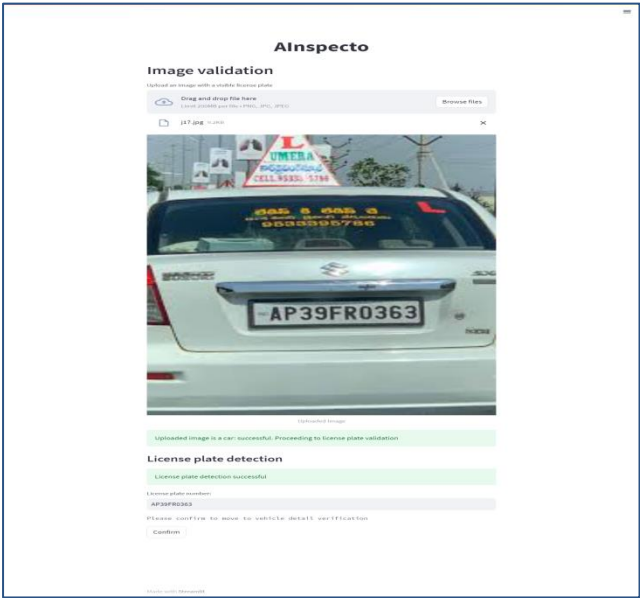
# Create a sidebar for user input
st.sidebar.title("Data Visualization App")
x_axis = st.sidebar.selectbox("Select X-Axis", data.columns)
y_axis = st.sidebar.selectbox("Select Y-Axis", data.columns)

# Create plot
fig, ax = plt.subplots()
ax.scatter(data[x_axis], data[y_axis])
ax.set_xlabel(x_axis)
ax.set_ylabel(y_axis)
st.pyplot(fig)
```

To run the app, open a terminal and navigate to the project folder. Then, execute the following command:

**streamlit run test.py**

Streamlit Screenshots:



## API Integrations:

All the functionalities have been integrated into API's, we currently have five API's covering all the validations and damage detections decided in the scope.

### 1. Upload Image:

**Name of the API:** upload\_image

**Route:** /upload (HTTP method: POST)

**Input:** This API accepts an image file as input, which should be sent using a 'multipart/form-data' request. The image should be passed with the key 'image' in the request body.

**Description:** This API checks if an image file was uploaded and if it is a valid image. If the file is a valid image, it saves the image to a specified path and performs various operations, such as checking if the uploaded image is of a car and performing license plate detection.

**Output:** The output of the API depends on the result of the image analysis. If the uploaded image is not a car, it returns a message stating that the image is not a car. If the analysis is successful, it returns the license plate text as a response.

AI / input upload

POST ⌵ http://127.0.0.1:5000/upload

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	image	j17.jpg ×
	Key	Value

Body Cookies Headers (6) Test Results 🌐

Pretty Raw Preview Visualize **HTML** ⌵ 🔧

1 AP39FR0363

## 2. Check User:

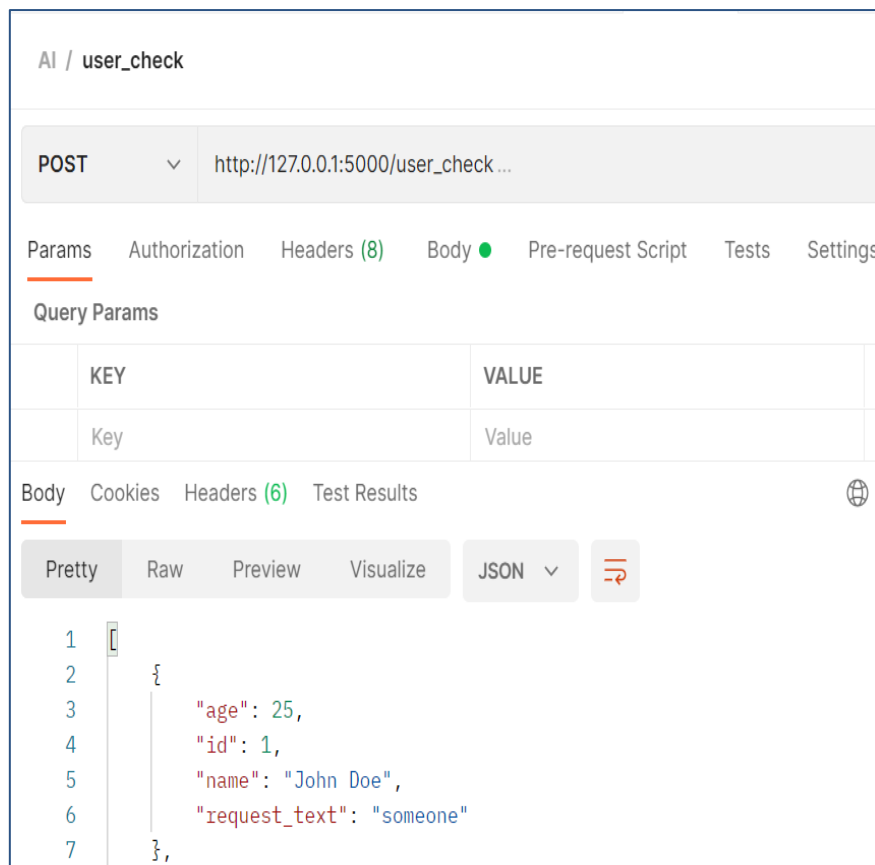
**Name of the API:** check\_user

**Route:** /user\_check (HTTP method: POST)

**Input:** This API accepts a JSON request containing the key 'request\_text'.

**Description:** This API receives the 'request\_text' from the JSON request and filters a sample data containing information about different users based on the input 'request\_text'.

**Output:** The output of the API is a JSON response containing the filtered data of users matching the 'request\_text'.





### 3. Damage Assessment:

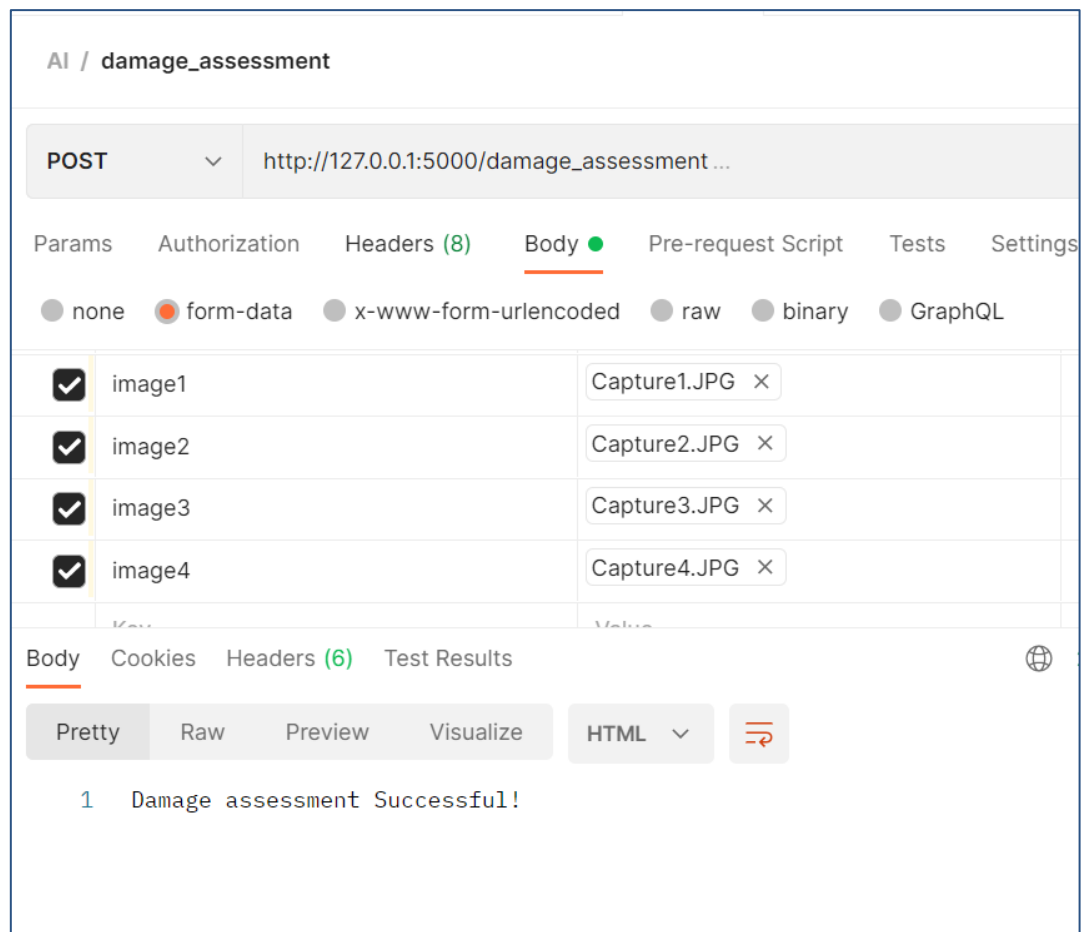
**Name of the API:** damage\_assessment

**Route:** /damage\_assessment (HTTP method: POST)

**Input:** This API accepts four images, one for each side of a vehicle, using 'multipart/form-data' request. The images should be sent with keys 'image1', 'image2', 'image3', and 'image4'.

**Description:** The API performs damage assessment on the vehicle based on the uploaded images. It validates if all images are of the same car, and if so, proceeds to assess the damages using machine learning models.

**Output:** The output of the API is a JSON response containing a DataFrame with information about the detected damages and their severity on each side of the vehicle. It also saves the DataFrame as a temporary JSON file.



#### 4. Get Data Frame:

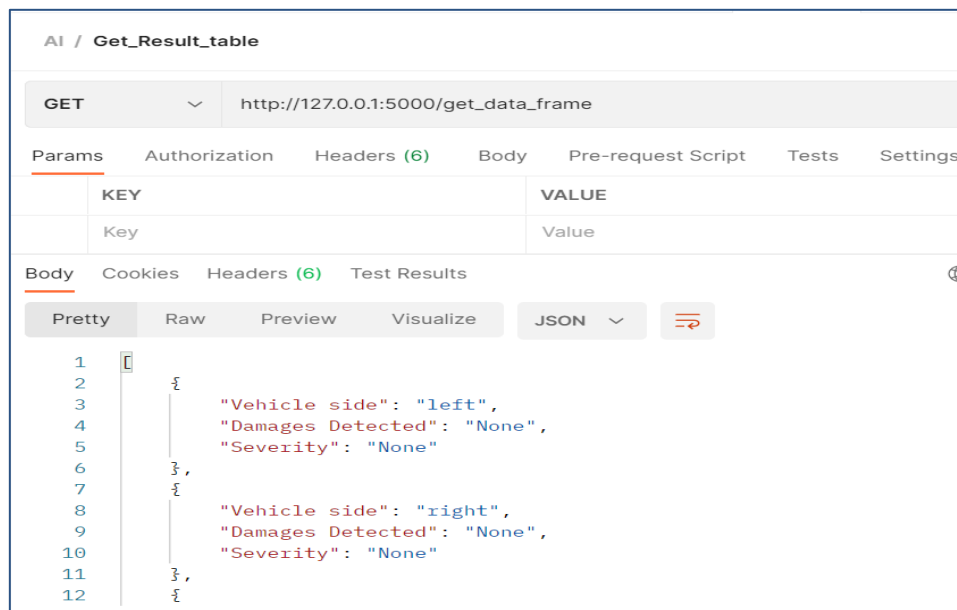
**Name of the API:** get\_data\_frame

**Route:** /get\_data\_frame (HTTP method: GET)

**Input:** This API does not require any input.

**Description:** This API retrieves the DataFrame containing information about the detected damages and their severity on each side of the vehicle.

**Output:** The output of the API is the JSON representation of the DataFrame containing the assessment results.



#### 4. Get Image:

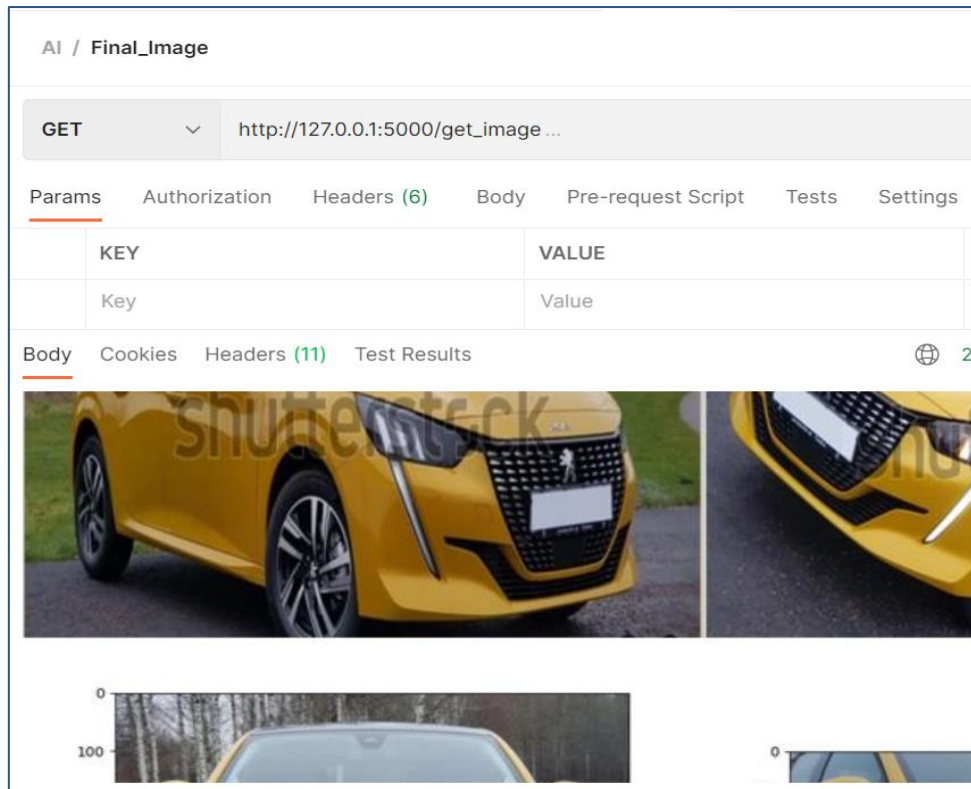
**Name of the API:** get\_image

**Route:** /get\_image (HTTP method: GET)

**Input:** This API does not require any input.

**Description:** This API retrieves the images of the vehicle showing the detected damages on each side.

**Output:** The output of the API is a collage image that combines the images of the vehicle's sides, with detected damages marked on each side. If a specific result image is not found, the input image for that particular side is returned instead. The collage image is saved as 'collage.jpg'.



## Future Scope:

- Increase model accuracy by training with more rich data.
- Include damage comparison module for claims process.
- Include location of the damage on the vehicle (ex: left door, bumper, hood etc.)
- Increase annotations in the dataset if required.
- Include cost estimate module for claims process.

## Challenges:

1. **Internal challenges:** Only the external damages can be detected. Medium and high severity accidents cause internal and structural damages and some electrical damages which cannot be detected.
2. **Repair estimation complexities:** Repair estimates are subject to parts cost, labour cost and negotiations.
3. **Subjectivity:** A single assessor can give two different estimates for the same car at two different times.
4. **Customer incentive to hide damages:** Customer can hide the damages during the self-inspection.
5. **Micro-damages:** Internal damages cannot be captured.
6. **Engine and repainting evaluation:** Engine health and re painting play a major role in evaluation and cannot be evaluated.

**Conclusion:**

As inscribed in the initial scope of the project, all the modules required to process inspection for renewal process was achieved and this being the basic version will be fine-tuned and perform more accurately in coming future. The inclusion of future scope into the project would give a promising perception from a business end.