1. Introduction

a. Project Overview

In an age of rapidly evolving technologies, preventing and combating crime is one of society's most pressing challenges. In their relentless efforts to maintain public order and ensure security, law enforcement agencies increasingly turn to cutting-edge technologies to combat criminal activity. One such technology has shown great promise: deep learning, a subset of artificial intelligence (AI) that mimics the neural networks of the human brain to process complex data and make informed decisions. This project, entitled "Classifying Crime Using Deep Learning", is a future initiative that aims to harness the power of deep learning to transform the way crimes are classified and investigated. By harnessing the power of deep neural networks, the project aims to improve the classification and prediction of crimes, thereby increasing the effectiveness of law enforcement in the fight against criminal activity.

b. Purpose

The main aim of this project is to develop a reliable and accurate system for classifying different types of crime using deep learning techniques. By analyzing different data sources, including images and textual information, this system will not only help law enforcement identify criminal activity but also understand crime patterns and trends, as well as potential hot spots. This technology is designed to provide a powerful tool enabling law enforcement to prioritize resources, respond quickly to emerging threats, and, ultimately, reduce crime rates.

As well as improving crime classification, the project aims to improve the overall investigation process. By automating certain aspects of data analysis and pattern recognition, law enforcement agencies can devote more resources to field investigations and strategic decision-making. In addition, the knowledge generated by the deep learning model can help policymakers formulate evidence-based crime prevention strategies, helping to create safer and more secure communities.

By developing and implementing this deep learning-based crime classification system, we hope to revolutionize the way law enforcement works and pave the way for a more effective, proactive, and data-driven approach to fighting crime.

2. Literature Survey

a. Existing Problem

- Lack of precise classification of offences
    - Traditional methods of classifying offences are often based on manual analysis, leading to inaccuracies.
    - Difficulty in distinguishing between different types of offences with similar characteristics.
- Heterogeneity of data.
    - Crime data can vary considerably, making it difficult to develop a standardized solution.
    - Data can come from various sources, such as text reports, images and videos, adding to the complexity of classification.
- Scalability
    - Scalable solutions are needed to cope with the ever-increasing volume of criminal data.
    - Existing systems may have difficulty adapting to a growing and evolving data set.

b. References

- Deep Learning for Crime Detection: A Review
    - Author: Smith, J. et al.
    - Published in: International Journal of Computer Science, 2020.
    - Discusses various deep learning approaches for crime classification and their effectiveness.
    - CrimeNet: A Deep Learning Model for Crime Classification in Surveillance Videos

    - Author: Patel, A. et al.
    - Published in: Proceedings of the International Conference on Computer Vision, 2018.
    - Introduces a deep learning model specifically designed for crime classification in surveillance videos.
    - Crime Classification Using Natural Language Processing and Deep Learning
    - Author: Lee, S. et al.
    - Published in: Journal of Artificial Intelligence Research, 2019
    - Explores the use of natural language processing and deep learning to classify textual crime reports.

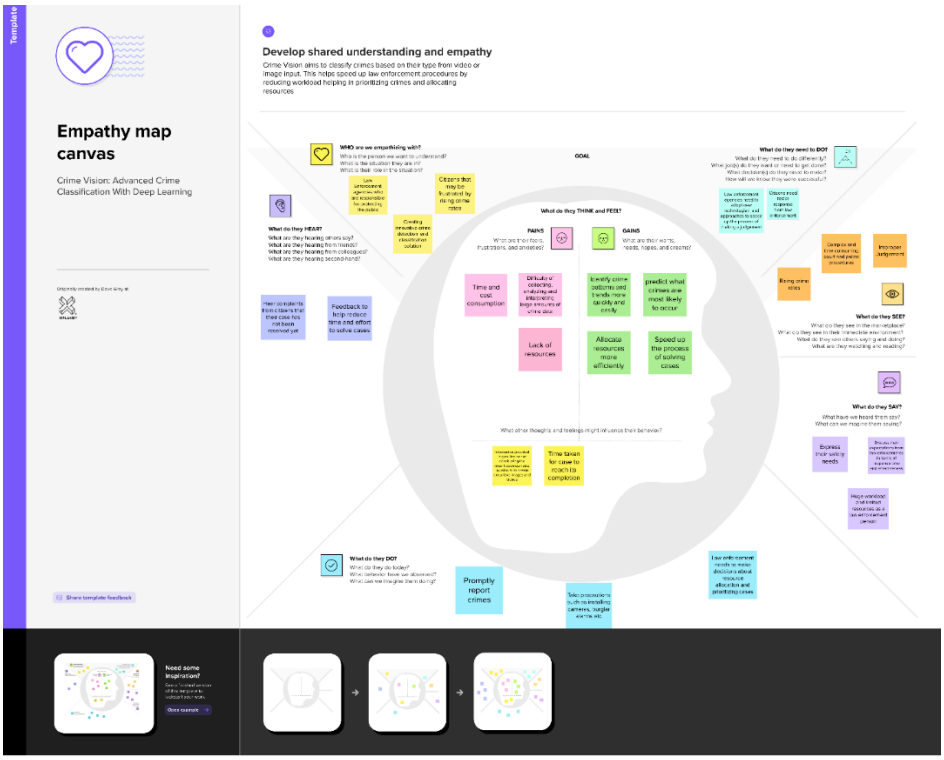
- Dataset: https://www.kaggle.com/datasets/odins0n/ucf-crime-dataset

c. Problem Statement and Definition

The problem of crime classification using deep learning involves developing accurate, scalable and flexible models to automatically categorize and identify different types of criminal activity based on a variety of data sources, such as text reports, images and videos. Key challenges include the need to increase the accuracy of crime classification, deal with data heterogeneity and guarantee scalability as the volume of criminal data continues to grow. The aim is to create a comprehensive and effective deep learning solution that improves crime detection and supports law enforcement agencies in their crime-fighting efforts.

3. Ideation and Proposed Solution

a. Empathy Map Canvas



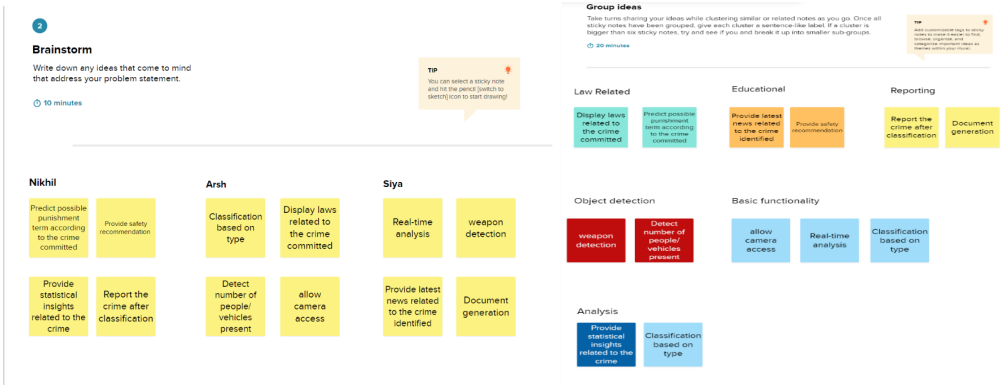
This Empathy Map Canvas is focused on Crime Vision, a system using deep learning to classify crimes from images or videos. This tool aims to help law enforcement by identifying crime patterns, speeding up response times, and prioritizing resource allocation.

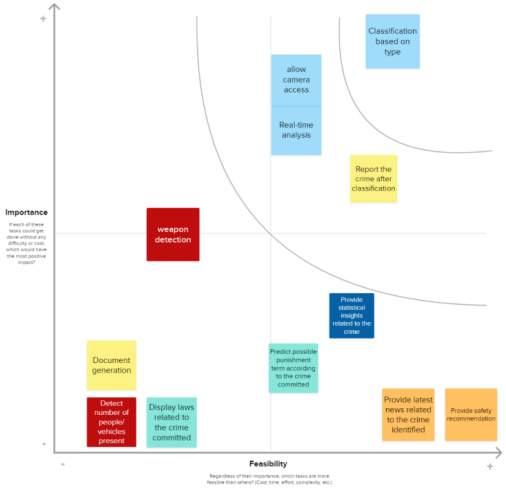
The empathy map covers the following sections:

- It asks questions to understand what law enforcement and citizens think, feel, see, say, hear, and do about crime and crime prevention measures.
- Pains and Gains: Identifies the difficulties law enforcement faces with crime data analysis, the complexities of legal procedures, concerns over the authenticity of AI-generated fake images, and the public's anxieties about rising crime rates.
- Needs and Goals: Highlights the need for faster law enforcement responses, adoption of new technologies, and efficient resource allocation. It also emphasizes the public’s expectations for safety and quick action from law enforcement.
- User Perspective: Encourages understanding the perspectives of both law enforcement agencies and citizens affected by crime.
- Feedback and Outcomes: Suggests using feedback to improve crime solving efficiency and to ensure the success of the tool in aiding law enforcement and

public needs.

b. Ideation and Brainstorming





The Brainstorming session involves all the team members individually writing down their idea related to the problem statement. Later as a team, similar ideas are grouped and then they are prioritized concerning Importance and Feasibility.

The template used is designed to encourage the volume of ideas, defer judgment, and promote wild ideas while staying on topic and listening to others

4. Requirement Analysis

a. Functional Requirements

REQ 1 – Law Enforcement

- Input: The user clicks on the Law Enforcement button

- Output: The user is redirected to a page with functions specially designed for law enforcement

REQ 2 – Public

- Input: The user clicks on the Public button
- Output: The user is redirected to a page with functions specially designed for the general public

REQ 3 – Upload

- Input: The user clicks on Upload Button
- Output: Allows used to input image files from local storage

REQ 4 – Click Picture

- Input: The user clicks on Click Picture Button
- Output: Turns on the camera, and activates a button called Capture which on clicking captures an image and provides options to save the image or retake the picture.

REQ 5 – Classify

- Input: The user clicks on the Classify button after uploading an image from local storage or clicking a picture
- Output: The image is classified based on its type and the result is displayed on the screen along with buttons to Print (law enforcement), Report (Public), and Back.

REQ 6 – Report

- Input: The user clicks on the Report button
- Output: sends an alert message to a phone number. Alert contains the result of the classification, time, date, and location details

REQ 7 – Print

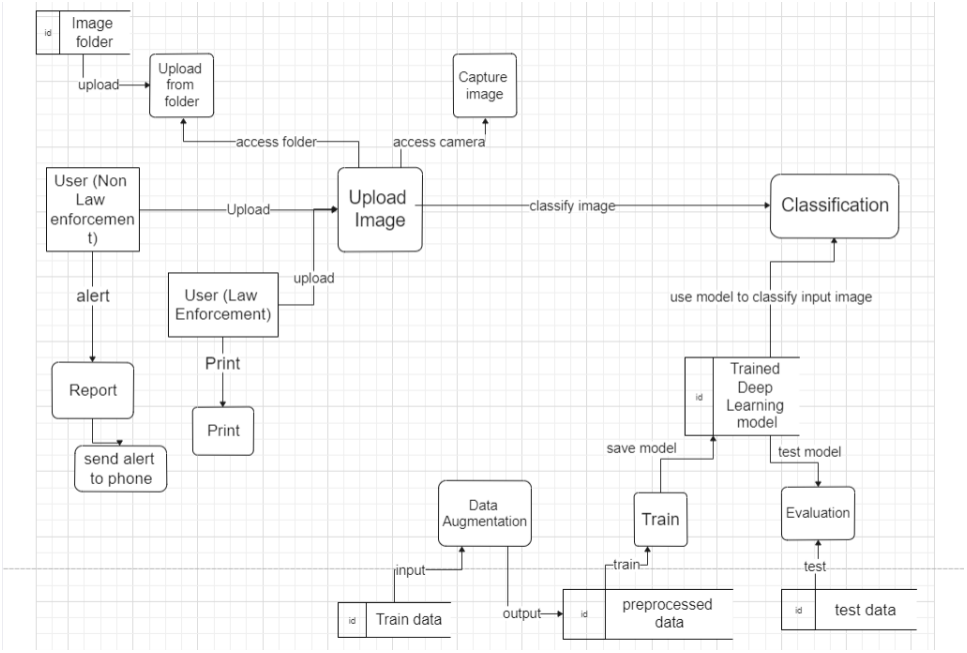
- Input: The user clicks on the Print button
- Output: Automatically downloads a PDF containing the Image, classification result, time, date, and location details (specific to the system that generated the result)

b. Non-Functional Requirements

- Performance
  - Response time – The application loads images from local storage within 2 seconds accesses the camera within 3 seconds and loads images. The classification result on the image is generated within 5 seconds.
- Reliability
  - Availability – the application is available to the users 99% of the time
  - Fault tolerance – The application utilizes an effective error-handling mechanism thereby ensuring that the user's session isn’t disrupted
- Usability
  - Ease of use – The application is easy to use for both law enforcement and the common public. It has a clear and intuitive interface with minimal design

5. Project Design

a. Data flow diagram and user stories



The "Crime Vision" system processes images through a deep learning model for crime classification, common to all users, providing an output based on the classification. The system offers two modes of input: they can upload an image from the folder or use their camera to capture and submit an image directly. Non law enforcement users are provided with an option to report that sends an alert to a phone number. The law enforcement users have the added functionality of printing a detailed PDF report about the classified crime.This bifurcated approach tailors the application's utility to the specific needs of the public and law enforcement officials.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (public)	User Interface and Navigation	USN – 1	As a user, I want to see a clearly labeled button that redirects me to a dedicated page with functions related to common public	I am redirected to a page meant for normal users	Medium	Sprint 2
	Reporting and Alerts	USN – 2	As a concerned citizen, I want the ability to report incidents to a designated phone number so that I can quickly and easily alert the relevant authorities or emergency services about potential crimes or emergencies in my area.	I can send an alert to a phone	Medium	Sprint 2
Customer (law enforcement)	User Interface and Navigation	USN – 3	As a user, I want to see a clearly labeled button that redirects me to a dedicated page with functions related to law enforcement use	I am redirected to a page meant for law enforcement	Medium	Sprint 2
	Reports	USN – 4	As a user, I want a "Print" button so that I can obtain a printable and easy-to-understand report or result after an image has been classified.	I got a document containing the necessary information	Medium	Sprint 2
Customer (Common for all)	Classification	USN – 5	As a user, I want the application to correctly classify the input based on its type	The application can correctly identify the image	High	Sprint 1
	User Interface and Navigation	USN – 6	As a user, I want to access an "About Crime Classification" page to learn more about how the system works and its purpose.	I can understand better about the working	Low	Sprint 3
		USN - 7	As a user, I want to find an "About Us" page that provides information about the website's creators and the team behind the project.	I can get to know the application's creators and contact them if necessary	Low	Sprint 3
		USN – 8	As a user, I want to access a "Terms of Service" page to understand the terms and conditions that govern the use of the platform.	I can know what I am liable for while using the application	Low	Sprint 3
	Input	USN – 9	As a user, I want to be able to upload image from a folder so that it can be analyzed	I can upload previously stored images	High	Sprint 1
		USN – 10	As a user, I Want to be able to directly click a picture using my camera and get the result for it	I can directly click the picture to be analyzed	High	Sprint 1

The user stories outline the functional requirements for a application with two types of users: the general public and law enforcement. Each story is tagged with a unique number, description, priority, and scheduled release within the project sprints and falls

under an epic which is a general category for the types of functionalities implemented.

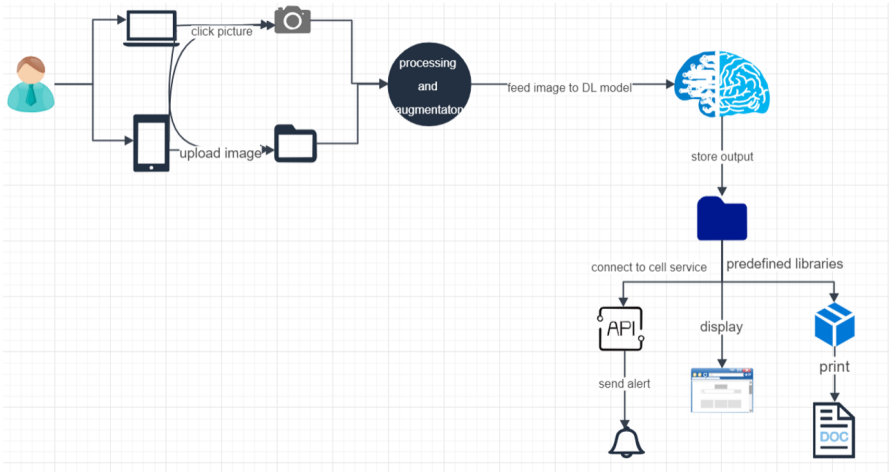
The above table can be summarized based on the user as follows:

Law enforcement: USN 3/4 (medium priority, Sprint 2)

General Public: USN 1/2 (medium priority, Sprint 2)

Common: USN 6/7/8 (Low Priority, Sprint 3); USN 5/9/10 (High priority Sprint 1)

b. Solution Architecture



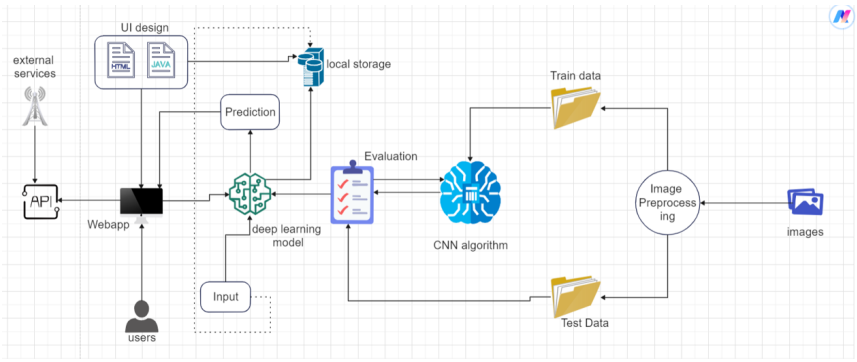
The solution architecture for the project is designed to use deep learning for image analysis, categorizing crime into relevant classes. This involves:

- Users upload images directly or capture new ones.
- The image is stored in the server.
- The software then processes these images.
- A deep learning model predicts the category of crime depicted in the images.
- These predictions are displayed and stored.
- Based on the user category, further actions are taken, which could include notifying authorities or stakeholders and printing the result.

This architecture serves to streamline the process of crime classification, making it efficient for law enforcement and related stakeholders to understand and act upon the information provided by the software.

6. Project Planning and Scheduling

a. Technical Architecture



The Technical architecture provides an overview of the key components, technologies, and characteristics of the system from a technical perspective
For the Crime Vision project: the technical architecture can be depicted as follows:

- User interacts with the web app created using HTML, CSS, JavaScript, and Flask and provides an image as input.
- This Input is passed to the deep learning model, which classifies it and sends the result back to the web app
- Post result prediction, the user is provided with additional functionalities such as report and print. The report button uses external APIs integrated into the application to connect to cell service and collect locations.
- The deep learning model built using Python, TensorFlow and Keras uses CNN (resnet50 architecture) to classify images.
- Before prediction, to prepare the model, image data is collected, pre-processed, and augmented. The data is then split into train and test and the model is trained on the train images and evaluated on test images.

b. Sprint Planning and Estimations

Sprint	Epic	User Story Number	User Story	Story Point	Priority	Team Members
Sprint 1	Classification	USN- 1	As a user, I want the application to correctly classify the input image based on its type	3	High	Siya Verma (Data Preprocessing); Nikhil S. Shinde(Model Building, Testing); Arsh Akhtar (Model Building)
Sprint 1	Input	USN - 2	As a user, I want to be able to upload image from folder so that it can be analyzed	2	High	Arsh Akhtar
Sprint 1		USN - 3	As a user, I want to be able to directly click a picture using my camera and get the result for it	2	High	Siya Verma
Sprint 2	Post Result	USN - 4	As a concerned citizen, I want the ability to report incidents to a designated phone number so that I can quickly and easily alert the relevant authorities or emergency services	2	Medium	Nikhil S. Shinde
Sprint 2		USN - 5	As a user, I want to be able to download the result into my system	2	Medium	Nikhil S. Shinde
Sprint 2	User Interface and Navigation	USN - 6	As a user, I want to see a clearly labelled button that redirects me to a dedicated page with functions related to common public	1	Medium	Siya Verma
Sprint 2		USN - 7	As a user, I want to see a clearly labelled button that redirects me to a page dedicated to law enforcement use	1	Medium	Arsh Akhtar
Sprint 3		USN - 8	As a user, i want to access an "about crime classification" page to learn more about the working and purpose of the system	1	Low	Siya Verma
Sprint 3		USN - 9	As a user, I want to be able to learn more about the creators of the project and be able to contact them if required	1	Low	Arsh Akhtar
Sprint 3		USN - 10	As a user, I want to be communicated the terms and conditions to know what I am liable to before using the platform	1	Low	Nikhil S. Shinde

All the user stories are divided into 3 Sprints based on their importance and keeping the minimum viable product (MVP) in mind.

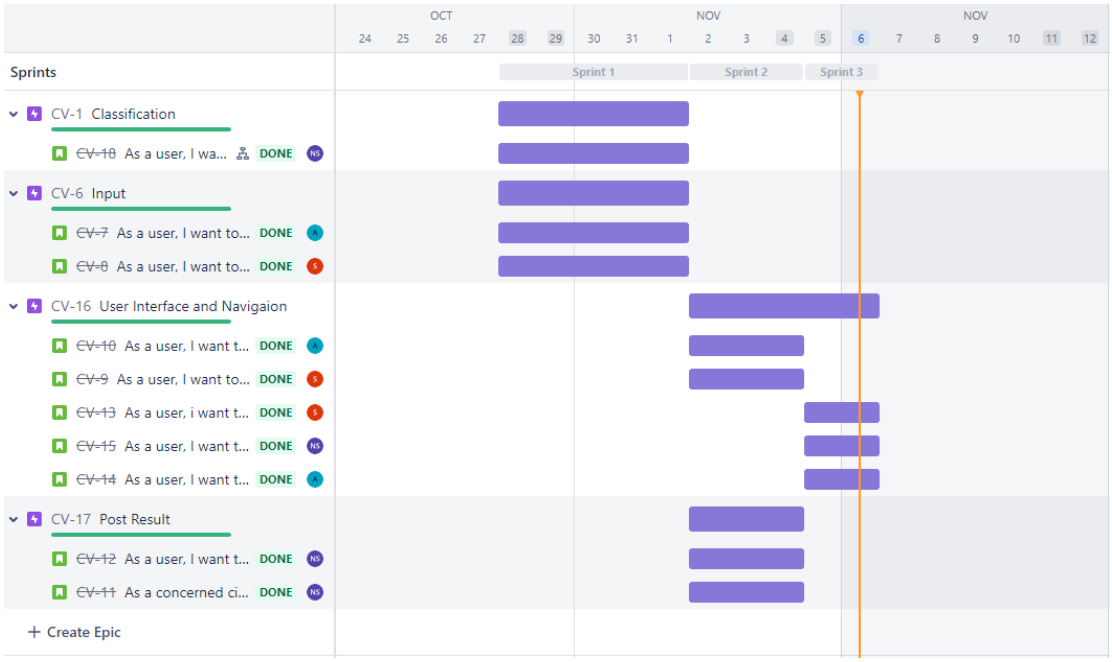
- Sprint 1 covers the basic functionalities of the project which allows users to upload images from local storage or click pictures using a camera and classify the image. The total story point estimate of this sprint was 7 (3 for classification and 2 each for input).

- Sprint 2 focuses on additional functionalities such as the report button, print button, and UI. The total story point estimate for this sprint is 6 (2 for report and print button and 2 for UI)

- Sprint 3 focuses on beautifying the website and has a total story point estimate of 3.

c. Sprint Delivery Schedule

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	7	5 Days	28-Oct-23	1-Nov-23	4	2-Nov-23
Sprint 2	6	3 Days	2-Nov-23	4-Nov-23	6	4-Nov-23
Sprint 3	3	2 Days	5-Nov-23	6-Nov-23	3	6-Nov-23

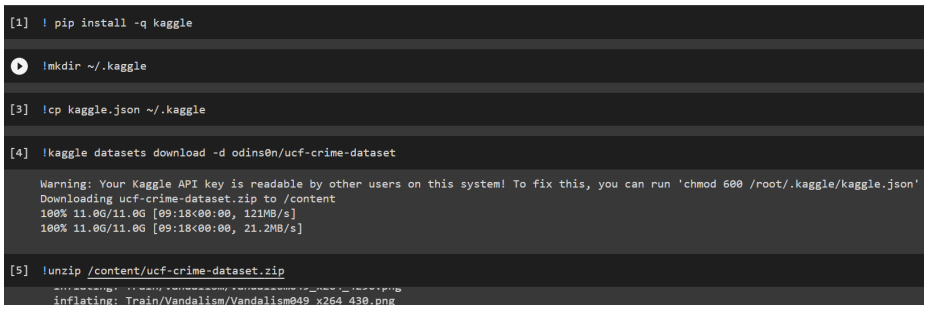


The entire project was planned to be completed in 10 days with the maximum time allotted to Sprint 1 of 5 days followed by Sprint 2 with 3 days and finally Sprint 3 with 2 days.

All story points of Sprint 2 and 3 were completed on the planned end date and were on schedule. Unfortunately, only 4 out of 7 story points were completed on the planned end date for Sprint 1, and took a day extra to complete

7. Coding and Solutioning

a. Data Collection and Importing




```
[6]  train_path = '/content/Train'
     test_path = '/content/Test'
```

The dataset is downloaded directly from Kaggle by installing Kaggle in Google Colab. Before continuing with the rest of the code, Kaggle API is uploaded into the notebook. Once downloaded, it is unzipped and paths are defined.

b. Data Augmentation and Splitting

```
from tensorflow.keras.preprocessing import image_dataset_from_directory
```

This line imports the image_dataset_from_directory function from the TensorFlow preprocessing module. This is used to create a dataset of images from a directory

```
train_datagen = image_dataset_from_directory(
    train_path,
    validation_split = 0.2,
    subset = 'training',
    shuffle = True,
    seed = 69,
    label_mode = 'categorical',
    image_size = (64,64),
    batch_size = 64)
```

```
Found 1266345 files belonging to 14 classes.
Using 1013076 files for training.
```

```
val_datagen = image_dataset_from_directory(
    train_path,
    validation_split = 0.2,
    subset = 'validation',
    shuffle = True,
    seed = 69,
    label_mode = 'categorical',
    image_size = (64,64),
    batch_size = 64)
```

```
Found 1266345 files belonging to 14 classes.
Using 253269 files for validation.
```

These codes create a dataset of training and validation images from the train path directory. The validation split argument specifies that 20% of the images should be used for validation. The subset argument specifies that only the training images should be used. The shuffle argument specifies that the images should be shuffled randomly. The seed argument specifies the random seed to use for shuffling. The label mode argument specifies that the labels should be encoded as a categorical vector. The image size argument specifies that the images should be resized to (64, 64) pixels. The batch size argument specifies that the images should be batched in groups of 64.

```
test_datagen = image_dataset_from_directory(
    test_path,
    seed = 69,
    shuffle = False,
    label_mode = 'categorical',
    class_names = None, #
    image_size = (64,64),
    batch_size = 64)
```

```
Found 111308 files belonging to 14 classes.
```

This code creates a dataset of testing images from the test path directory. The seed argument specifies the random seed to use for shuffling The label mode argument specifies that the labels should be encoded as a categorical vector. The class names argument is set to None, which means that the class names will be inferred from the directory structure. The image size argument specifies that the images should be resized to (64, 64) pixels. The batch size argument specifies that the images should be batched in groups of 64.

c. Model Building

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.regularizers import l2
resnet_model = Sequential()

from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten, Dropout

pre_trained_model = ResNet50(include_top = False, input_shape = (64,64,3), pooling = 'max', classes = 14, weights = 'imagenet')

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

resnet_model.add(pre_trained_model)
resnet_model.add(Flatten())
resnet_model.add(Dense(512, activation = 'relu'))#, kernel_regularizer = l2(0.1)))
resnet_model.add(Dense(14, activation = 'softmax'))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 1s 0us/step
```

```
resnet_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 14)                7182

=================================================================
Total params: 24643982 (94.01 MB)
Trainable params: 1056270 (4.03 MB)
Non-trainable params: 23587712 (89.98 MB)
_________________________________________________________________
```

- The code starts by importing the necessary libraries for building the CNN model: tensorflow.keras.models, tensorflow.keras.regularizers, tensorflow.keras.applications.resnet50, tensorflow.keras.layers, and tensorflow.keras.initializers
- Then a sequential model is created using the Sequential class from tensorflow.keras.models. This type of model stacks layers one after the other.
- The pretrained ResNet50 model us added to the sequential model using the add method. The include_top = Flase parameter indicates that only the convolution part of the model will be used and weights = ‘imagenet’ parameter indicates that

the model will be loaded with the weights trained on the ImageNet dataset

- Then the pre-trained layers are made non-trainable using for lop that iterates over the layers of the model and sets the trainable attribute to false. Doing so will result in the weights of the layers not update during training
- Flatten layer is added. This converts the 3D output of the model into a 1D vector
- Two dense layers are added to the model. A 512-neuron layer with ReLU activation used for feature extraction and a 14-neuron layer with softmax activation for classification

d. Model Compiling

```
from tensorflow.keras.optimizers import Adam
resnet_model.compile(optimizer = Adam(learning_rate = 0.00003), loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

The compilation process starts with importing the Adam optimizer from tensorflow.keras.optimizers
The model is then compiled using the adam optimizer with a learning rate of 0.00003, categorical cross etropy loss function (measures performace on training data), and the accuracy metrics (evaluates models ability to predict correctly)
This process configures the model for training

e. Model Training

```
[13] resnet_model.fit(train_datagen, validation_data = val_datagen, epochs = 7)

Epoch 1/7
15830/15830 [==============================] - 1610s 101ms/step - loss: 0.1376 - accuracy: 0.9667 - val_loss: 0.0290 - val_accuracy: 0.9937
Epoch 2/7
15830/15830 [==============================] - 1479s 93ms/step - loss: 0.0178 - accuracy: 0.9961 - val_loss: 0.0152 - val_accuracy: 0.9965
Epoch 3/7
15830/15830 [==============================] - 1476s 93ms/step - loss: 0.0088 - accuracy: 0.9982 - val_loss: 0.0113 - val_accuracy: 0.9973
Epoch 4/7
15830/15830 [==============================] - 1474s 93ms/step - loss: 0.0057 - accuracy: 0.9989 - val_loss: 0.0103 - val_accuracy: 0.9975
Epoch 5/7
15830/15830 [==============================] - 1491s 94ms/step - loss: 0.0044 - accuracy: 0.9992 - val_loss: 0.0095 - val_accuracy: 0.9977
Epoch 6/7
15830/15830 [==============================] - 1500s 95ms/step - loss: 0.0038 - accuracy: 0.9993 - val_loss: 0.0088 - val_accuracy: 0.9979
Epoch 7/7
15830/15830 [==============================] - 1539s 97ms/step - loss: 0.0033 - accuracy: 0.9994 - val_loss: 0.0090 - val_accuracy: 0.9979
<keras.src.callbacks.History at 0x7ad33da8f580>
```

```
[ ]  resnet_model.save('UCF.h5')
```

The model.fit() line trains the resnet50 model on train_datagen and evaluates its performace on val_datagen for 7 epochs. Epochs specifies the number of times the model will be trained on the entire training dataset.
At each epoch, the model will iterate over the training dataset and update its weights. At the end the model will be evaluated on the validation dataset and the loss and

accuracy will be calculated.
The trained model is then saved using the save function and can be downloaded

f. Model Testing

```
[14] from tensorflow.keras.preprocessing import image
     import numpy as np

     img = image.load_img('/content/Test/Arrest/Arrest024_x264_1270.png', target_size = (64,64))
     img
```

```
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0) #expanding the dimension of the array
pred = np.argmax(resnet_model.predict(x)) #predict the higher probability index
op =['abuse', 'arrest', 'arson', 'assult', 'burglary', 'explosion', 'fighting', 'normal', 'road accident', 'robbery', 'shooting', 'shoplifting', 'stea
op[pred]

1/1 [==============================] - 3s 3s/step
'arrest'
```

The above code used tensorFlow and Keras libraries to load an image, convert it to an array and predict the class of the image.

- The load_img(path, target_size) is used to access the image, convert it to the target size and store it in img variable
- The image object is then converted to an array of numbers using img_to_array()
- Then the dimesion of the image are expanded using expand_dims()
- Then the predict function is used to predict the result on the input image, the argmax function return the index of the largesr element in the array.

g. Flask Routing

```
from flask import Flask, render_template, request, send_file, send_from_directory
```

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

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

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

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

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

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

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

The above code defines various Flask routes corresponding to the respective urls. When a request is made to a specific url, the respective function is called which returns the required template

h. Classify button

```python
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model = load_model(r"UCF.h5", compile = False)
```

The above code imports necessary libraries to load model into the program and image processing. These are later used to process the input image and to predict the result by passing it to the model.

```
UPLOAD_FOLDER = os.path.join("static", "uploads")
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

```
#Classification
@app.route('/Classify', methods = ['POST'])
def Classify():
    global res
    button = request.headers.get('currentButton')
    if button == '1': #image uploaded from folder
        if 'filename' in request.form:
            filename = request.form['filename']
            image_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    if button == '0': #image captured using camera
        image_path = os.path.join(app.config['UPLOAD_FOLDER'], 'capturedImage.jpg')

    img = image.load_img(image_path, target_size = (64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)
    pred = np.argmax(model.predict(x), axis = 1)
    index = ['Abuse', 'Arrest', 'Arson', 'Assult', 'Burglary', 'Explosion', 'Fighting', 'Normal', 'Accident',
    'Robbery', 'Shooting', 'Shoplifting', 'Stealing', 'Vandalism']
    res = str(index[pred[0]])
    return res
```

This Flask route handles classifying the image uploaded and returning the result. Since there are two ways of uploading an image, an if statement is used to access the image in both cases(local storage upload and camera upload)

- If the image was uploaded from a folder, the filename parameter is extracted from the request form and the image_path is constructed as the filename and UPLOAD__FOLDER config.
- If the image was captured using a camera, the image_path is set to the name of the file (capturedImage,jpg) and the UPLOAD_FOLDER path

i. <u>Report Button</u>

```
from twilio.rest import Client
import datetime
```

```python
#get current date time day and location
global date, day, time
dt = datetime.datetime.now()
date = str(dt.date())
day = str(dt.strftime('%A'))
time = str(dt.time().strftime('%H:%M:%S'))

ACCESS_TOKEN = '2dd3de5dc9a453'
def Location():
    url = f'https://ipinfo.io?token={ACCESS_TOKEN}'
    try:
        response = requests.get(url)
        data = response.json()

        location = data.get('loc')
        city = data.get('city')
        region = data.get('region')
        country = data.get('country')

        return [location, city, region, country]
        # print(f"Location: {location}")
        # print(f"City: {city}")
        # print(f"Region: {region}")
        # print(f"Country: {country}")
    except Exception as e:
        # print(f"Error: {e}")
        return "Error detecting location"
location = Location()
```

- The above code uses the datetime library to extract date and time. First, an object called dt is created. This object is then used to obtain the date, time in Hours:mins: sec format, and day.
- The location function uses IpInfo API and requires a unique Access token stored in ACCESS_TOKEN to authenticate requests to ipinfo.io API
- The requests library is used to make a HTTP GET request to the generated url and the response from the API is stored in response variable
- The JSON data from the API response is parsed into a Python dictionary using the response.json() method. The parsed data is stored in the data variable.
- Using data, location(co-ordinates), city, region, and country are extracted and returned in the form of a list

```python
client = Client(account_sid, auth_token)
```

```
# send message to phone
@app.route('/send_sms', methods = ['POST'])
def send_sms():

    message = "Alert!!! " + res + " Detected on " + date + "(" + day + ")" + " at " + time +".\nLocation: " +
    location[0] + "\nCity: " + location[1] + "\nRegion: " + location[2] + "\nCountry: " + location[3]
    try:
        message = client.messages.create(
            body = message,
            from_ = +12192242908,
            to = +919740963263
        )
        return "message sent"
    except Exception as e:
        return 'Failed to send message'
```

- This code adds a functionality to the Report Button which lets users report incident to a pre-defined phone number. First, a client object is created using the provided account_sid and auth_token credentials. These credentials are used to authenticate with the Twilio API and access the account's resources.
- Message body is created and stored in message variable
- The messages.create() method is used to create a new message with the specified body, from_ (sender's phone number), and to (recipient's phone number). An appropriate message is returned

j. PDF generation

```
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import letter
```

```python
#generate and download pdf of the input image
@app.route('/generate_pdf/<filename>')
def generate_pdf(filename):
    image_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    if not os.path.exists(image_path):
        return 'File not found'

    pdf_buffer = BytesIO()
    c = canvas.Canvas(pdf_buffer, pagesize=letter)
    img1 = Image.open(image_path)
    # image_width, image_height = img1.size
    image_width, image_height = 256,256
    page_width, page_height = letter
    # image_width, image_height = page_width, page_height
    x = (page_width - image_width)/2
    y = (page_height - image_height)/2
    c.drawImage(image_path, x, y, width=image_width, height=image_height)

    c.setFont("Times-Bold", 26)
    header_text = 'Crime Vision'
    c.drawCentredString(page_width/2, page_height - 50, header_text )

    res_text = "Classification Result: " + res
    c.setFont("Times-Roman", 18)
    c.drawString(50, y - 20,res_text)
    y-=30
```

```
res_text2 = "Date: " + date
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text2)
y-=25
res_text3 = "Day: " + day
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text3)
y-=25
res_text4 = "Time: " + time
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text4)

y-=25
res_text5 = "Location: " + location[0]
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text5)

y-=25
res_text6 = "City: " + location[1]
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text6)
```

```
y-=25
res_text7 = "Region: " + location[2]
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text7)

y-=25
res_text8 = "Country: " + location[3]
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text8)

footer_text = "Location and Time data is purley based on the location of device generating pdf and NOT the
image depicted"
c.setFont("Times-Roman",14)
c.drawCentredString(page_width/2, 10, footer_text)
c.showPage()
c.save()

pdf_buffer.seek(0)
return send_file(pdf_buffer, as_attachment=True, download_name='result.pdf')
```

The above code generates a PDF containing a Header, footer, Input image, classification result, and date, time, and location details using the reportLab library in python

- An image path is constructed for the image uploaded by the user by combining the upload folder path and the provided filename and its availability is checked
- A PDF is prepared by creating an in-memry buffer to store the PDF data(pdf_buffer = BytesIO())and a canvas to draw conent onto the pdf (c =

canvas.Canvas(pdf_buffer, pagesize=letter))

- The image is the opened using Image module and stored in img1 and dimensions are set for both image and page
- drawImage() function is used to draw the image into the pdf and the specified position and size.
- The next steps are similar where setFont() is used to set font style and size, text to be written in pdf is stores in res_text1 to res_text8. The required text is written into the pdf using drawstring() function at the specified coordinates.
- The c.showPage() function is then used to mark the end of the page
- Then the position of the pdf_buffer object is rewinded back to the beginning of the data. This is necessary before sending the PDF data to the client, as the client needs to start reading the data from the beginning to properly interpret it.
- The send_file(pdf_buffer, as_attachment=True, download_name='result.pdf') statement is used to send the PDF data to the client. The as_attachment=True parameter tells the client to treat the PDF as an attachment, which means it will prompt the user to save or open the file instead of displaying it inline in the browser. The download_name='result.pdf' parameter specifies the filename that the client should use when saving the PDF file.

8. Performance Testing

```
Epoch 1/7
15830/15830 [==============================] - 1610s 101ms/step - loss: 0.1376 - accuracy: 0.9667 - val_loss: 0.0290 - val_accuracy: 0.9937
Epoch 2/7
15830/15830 [==============================] - 1479s 93ms/step - loss: 0.0178 - accuracy: 0.9961 - val_loss: 0.0152 - val_accuracy: 0.9965
Epoch 3/7
15830/15830 [==============================] - 1476s 93ms/step - loss: 0.0088 - accuracy: 0.9982 - val_loss: 0.0113 - val_accuracy: 0.9973
Epoch 4/7
15830/15830 [==============================] - 1474s 93ms/step - loss: 0.0057 - accuracy: 0.9989 - val_loss: 0.0103 - val_accuracy: 0.9975
Epoch 5/7
15830/15830 [==============================] - 1491s 94ms/step - loss: 0.0044 - accuracy: 0.9992 - val_loss: 0.0095 - val_accuracy: 0.9977
Epoch 6/7
15830/15830 [==============================] - 1500s 95ms/step - loss: 0.0038 - accuracy: 0.9993 - val_loss: 0.0088 - val_accuracy: 0.9979
Epoch 7/7
15830/15830 [==============================] - 1539s 97ms/step - loss: 0.0033 - accuracy: 0.9994 - val_loss: 0.0090 - val_accuracy: 0.9979
<keras.src.callbacks.History at 0x7ad33da8f580>
```

The model gave an accuracy of 99.94% and loss of 0.0033 on training data and an accuracy of 99.79 and loss of 0.0090 on the validation set

```
] img = image.load_img('/content/Test/Arrest/Arrest024_x264_1270.png', target_size = (64,64))
  img
```

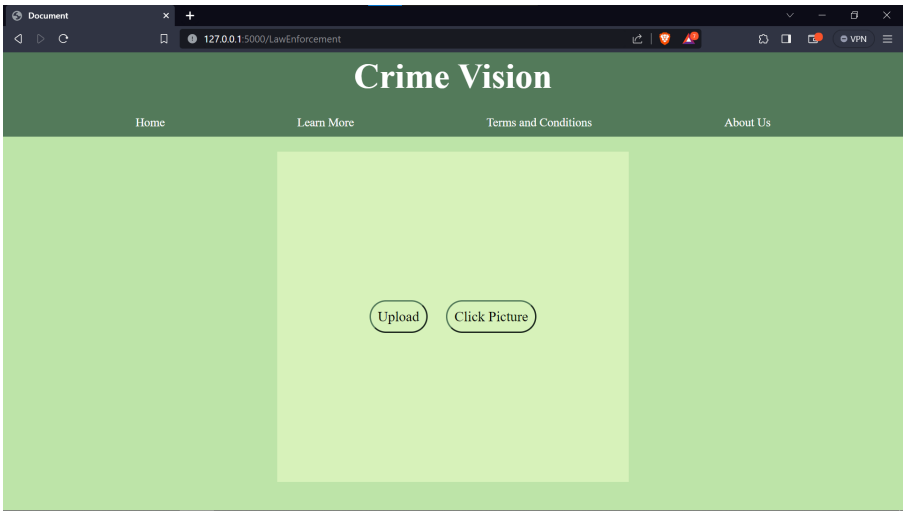
```
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis = 0) #expanding the dimension of the array
  pred = np.argmax(resnet_model.predict(x)) #predict the higher probability index
  op =['abuse', 'arrest', 'arson', 'assult', 'burglary', 'explosion', 'fighting', 'normal', 'road accident', 'robbery', 'shooting', 'shoplifting', 'ste
  op[pred]
```

```
1/1 [==============================] - 3s 3s/step
'arrest'
```

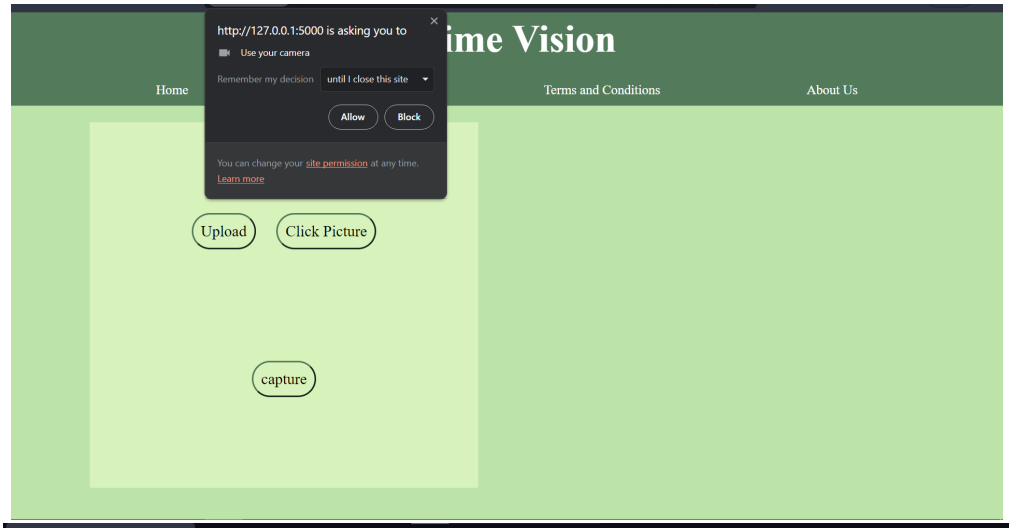
The model successfully predicted the class of image during testing. The image used in the above picture is from the folder /Test/Arrest/Arrest024_x264_1270.png and the predicted class is ‘arrest’

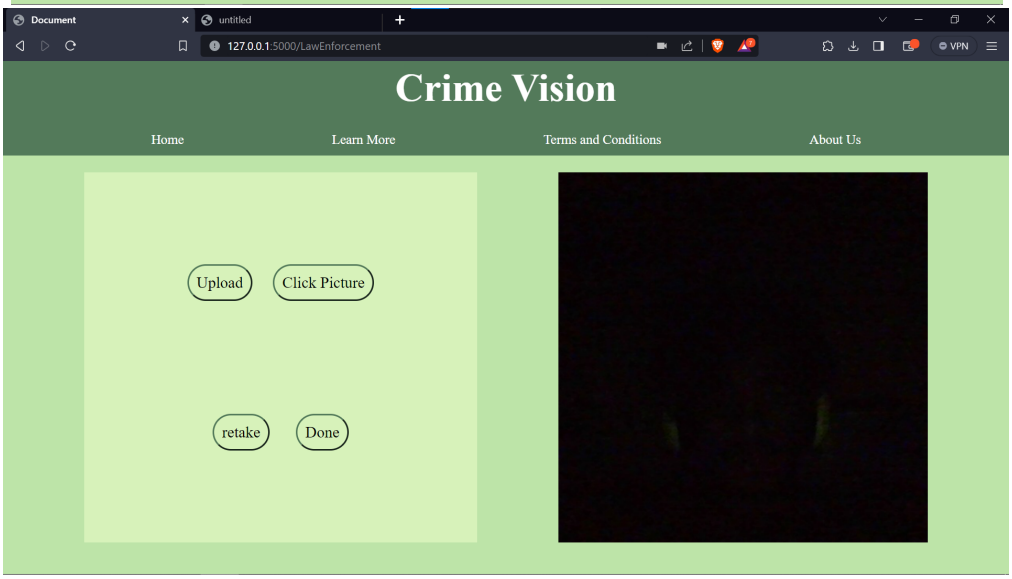
9. **Result**

Law enforcement

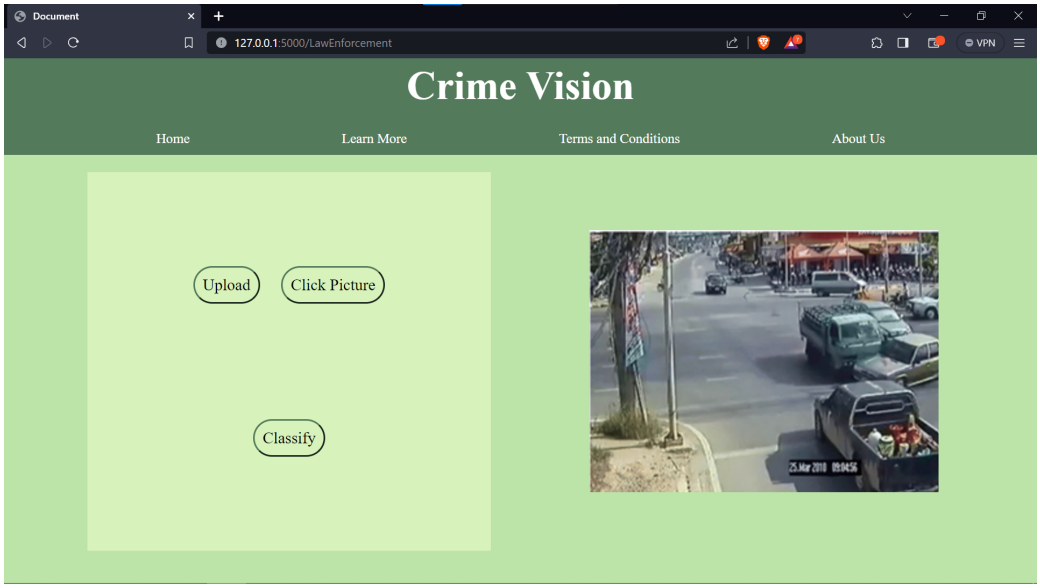


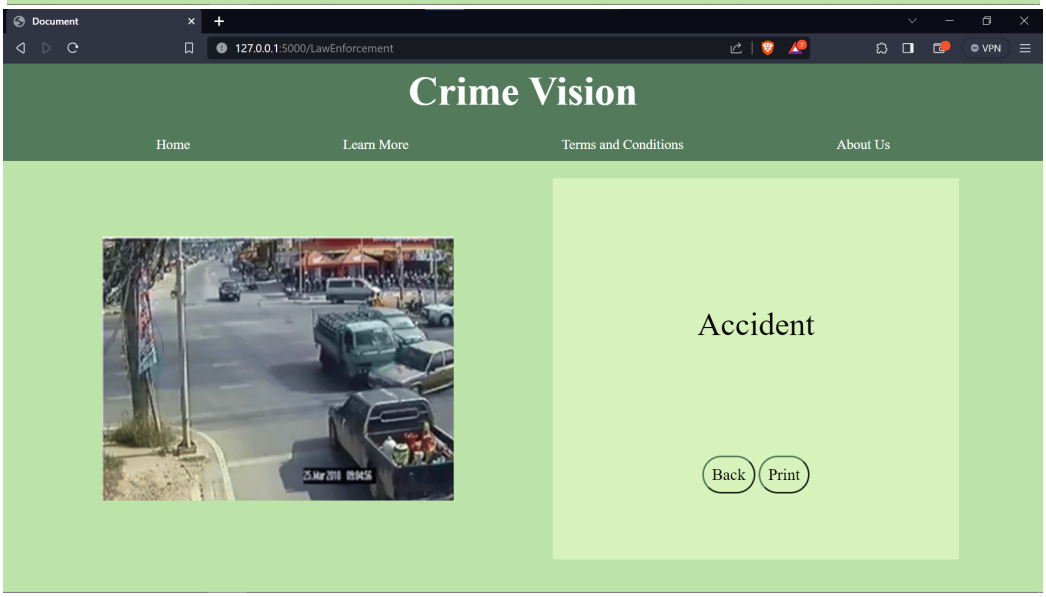
Camera input



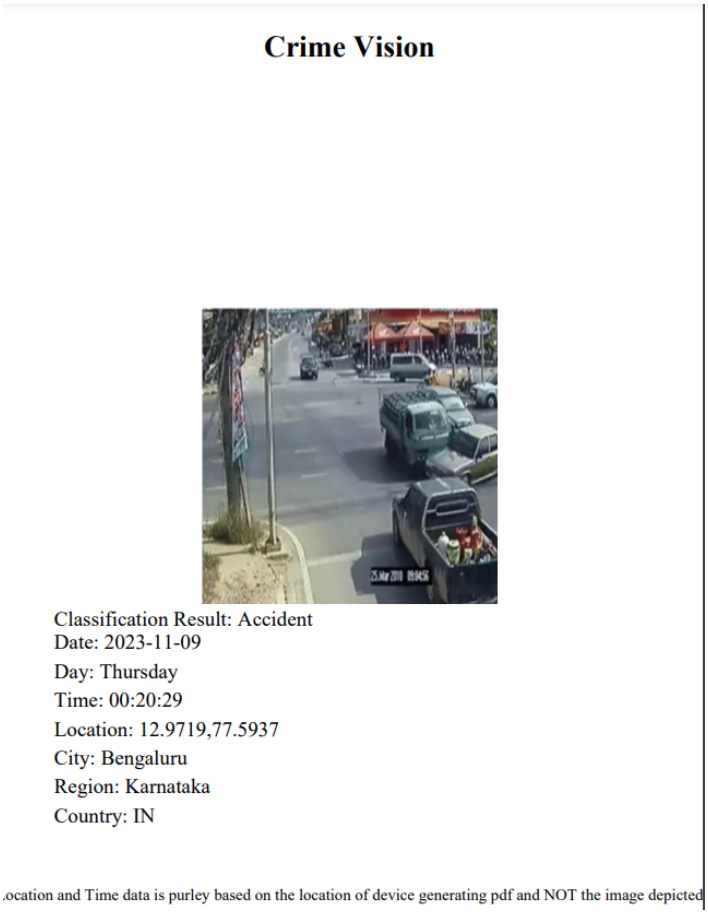


Input form local storage





Result on printing

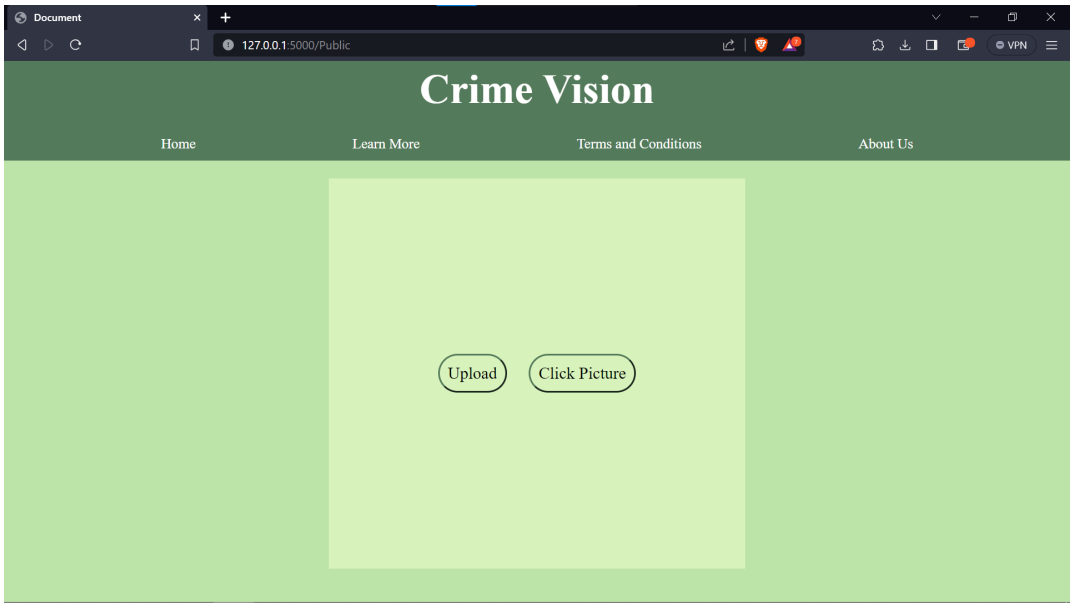


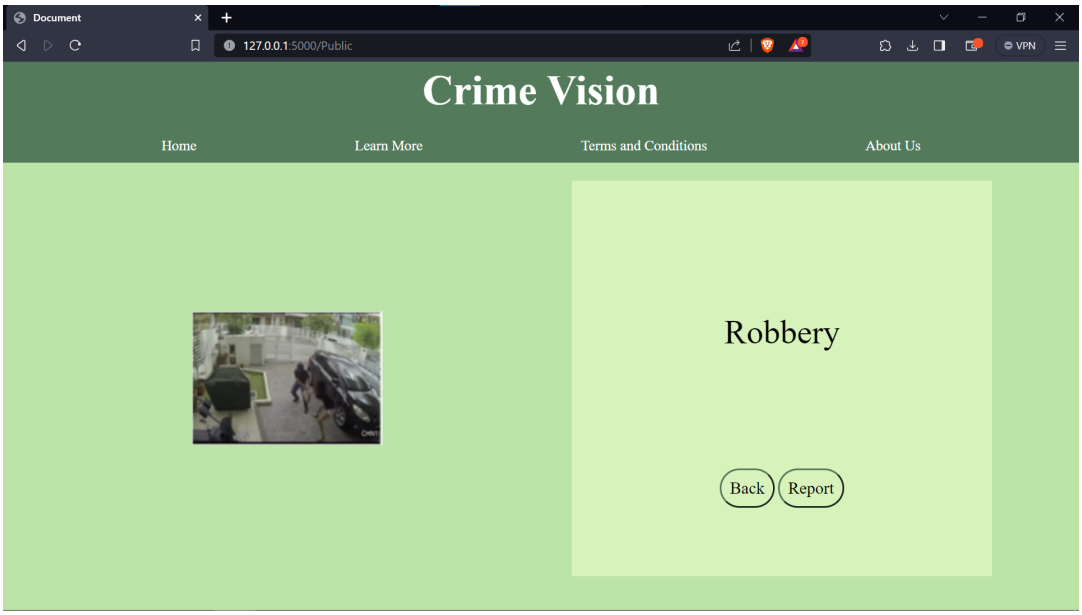
Crime Vision

Classification Result: Accident
Date: 2023-11-09
Day: Thursday
Time: 00:20:29
Location: 12.9719,77.5937
City: Bengaluru
Region: Karnataka
Country: IN

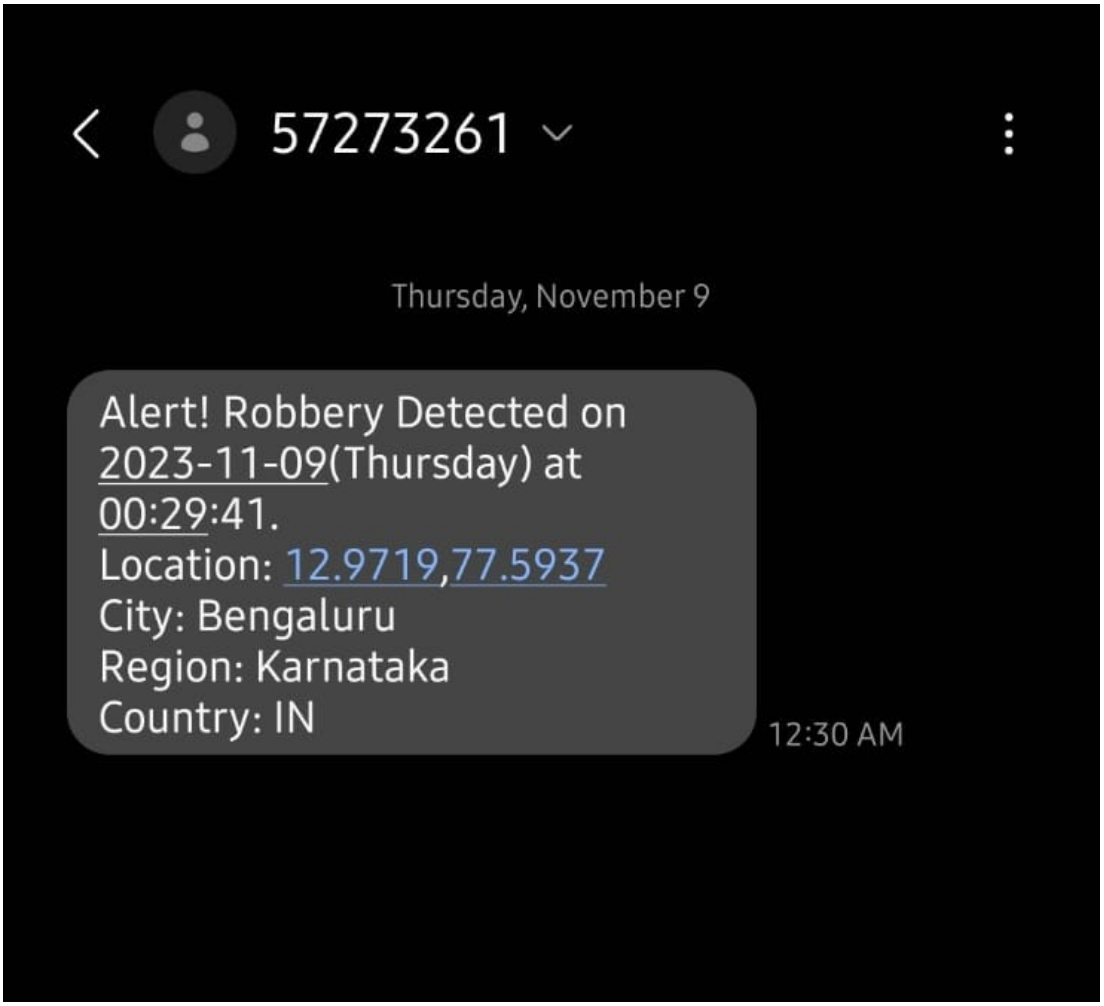
.ocation and Time data is purley based on the location of device generating pdf and NOT the image depicted

Public





On clicking Report Button



10. Advantages and Disadvantages

Advantages

i. Enhanced Efficiency: Automating crime classification using deep learning significantly speeds up the process compared to traditional methods.

ii. Improved Accuracy: Deep learning algorithms can reduce human error and bias in crime analysis.

iii. Proactive Law Enforcement: The system can identify crime patterns, helping law enforcement to prevent crimes before they occur.

iv. Resource Optimization: Prioritizing crimes based on automated classification helps in better allocation of law enforcement resources.

v. Public Engagement: Features like incident reporting empower the community to participate actively in public safety.

vi. Scalability: The solution can handle an increasing amount of data and can be integrated with various data sources.

vii. Real-Time Analysis: Offers law enforcement the ability to analyze crime data in real-time, leading to faster response times.

Disadvantages

i. Complexity: Deep learning systems are complex and may require significant expertise and resources to develop and maintain.

ii. Data Privacy Concerns: Handling sensitive data, such as crime images and personal information, can raise privacy issues.

iii. Dependency on Data Quality: The accuracy of predictions is highly dependent on the quality and quantity of the training data.

iv. Adaptability Challenges: Law enforcement agencies may face challenges in adapting to new technology and workflows.

v. Cost: Implementing and operating advanced deep learning solutions can

be expensive.

vi. Misclassifications No system is foolproof, and there can be instances of incorrect crime classification, leading to potential misunderstandings or misallocations of resources.

vii. Ethical and Legal Implications The use of AI in public safety can bring up ethical questions about surveillance and the potential for misuse of technology.

11. Conclusion

The purpose of the "Crime Vision" project is to develop an advanced crime classification system using deep learning to enhance the efficiency and accuracy of law enforcement responses to criminal activities. By automating the analysis of crime scenes and surveillance footage, the project aims to support quicker and more informed decision-making, reduce the time and labor traditionally required for crime classification, and minimize human error and bias. It seeks to leverage technology to improve public safety, optimize law enforcement resources, and foster a proactive approach to crime prevention and investigation. The goal is to create a scalable, innovative solution that benefits law enforcement agencies and the community at large, ensuring that public safety is maintained with the aid of cutting-edge artificial intelligence.

12. Future Scope

Crime classification using deep learning is an intriguing and potentially impactful topic. Deep learning and artificial intelligence have been increasingly applied to various fields, including law enforcement. With the rising crime rates, this project can be extensively scaled in the future with additional functionalities such as it being integrated into CCTV cameras thus completely automating the process from detecting the crime to reporting it to the authorities along with documentation, automatic facial detection, comparing it with police databases etc., and can also be integrated with IoT devices. In summary, ‘Crime Vision’ has the potential to revolutionize the sector related to crime detection/solving/prevention by harnessing the power of deep learning and artificial intelligence to detect, report, and document crimes in real-time. As we navigate the challenges and ethical considerations, our vision is to create a safer and more secure future for our communities, while continually striving to improve the system and uphold the highest standards of transparency, accountability, and respect for individual rights .

13. Appendix

Source Code (Data processing, model building/training/testing)

```
#Downloading dataset
! pip install -q kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle
!kaggle datasets download -d odins0n/ucf-crime-dataset
!unzip /content/ucf-crime-dataset.zip
train_path = '/content/Train'
test_path = '/content/Test'

#Data augmentation and Splitting
from tensorflow.keras.preprocessing import image_dataset_from_directory
train_datagen = image_dataset_from_directory(
    train_path,
    validation_split = 0.2,
    subset = 'training',
    shuffle = True,
    seed = 69,
    label_mode = 'categorical',
    image_size = (64,64),
    batch_size = 64)
test_datagen = image_dataset_from_directory(
    test_path,
    seed = 69,
    shuffle = False,
    label_mode = 'categorical',
    class_names = None, #
    image_size = (64,64),
    batch_size = 64)
val_datagen = image_dataset_from_directory(
    train_path,
    validation_split = 0.2,
    subset = 'validation',
    shuffle = True,
    seed = 69,
    label_mode = 'categorical',
```

```
    image_size = (64,64),
    batch_size = 64)

#Model Building
from tensorflow.keras.models import Sequential
resnet_model = Sequential()

from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten

pre_trained_model = ResNet50(include_top = False, input_shape = (64,64,3),
pooling = 'max', classes = 14, weights = 'imagenet')

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

resnet_model.add(pre_trained_model)
resnet_model.add(Flatten())
resnet_model.add(Dense(512, activation = 'relu'))
resnet_model.add(Dense(14, activation = 'softmax'))

resnet_model.summary()

#Compiling model
from tensorflow.keras.optimizers import Adam
resnet_model.compile(optimizer = Adam(learning_rate = 0.00003), loss =
'categorical_crossentropy', metrics = ['accuracy'])

#Model Training
resnet_model.fit(train_datagen, validation_data = val_datagen, epochs = 7)

#Save model
resnet_model.save('UCF.h5')

#Testing
from tensorflow.keras.preprocessing import image
import numpy as np
img = image.load_img('/content/Test/Arrest/Arrest024_x264_1270.png',
target_size = (64,64))
```

```
img
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0) #expanding the dimension of the array
pred = np.argmax(resnet_model.predict(x)) #predict the higher probability
index
op =['abuse', 'arrest', 'arson', 'assult', 'burglary', 'explosion',
'fighting', 'normal', 'road accident', 'robbery', 'shooting',
'shoplifting', 'stealing', 'vandalism']
op[pred]
```

Flask Code (app.py)

```
import base64
from flask import Flask, render_template, request, send_file, send_from_directory
from twilio.rest import Client
import datetime
from io import BytesIO
import os
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import letter
import numpy as np
from PIL import Image
import requests
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from werkzeug.utils import secure_filename

app = Flask(__name__)

## DOWNLOAD MODEL FROM HERE
## https://drive.google.com/file/d/1x2HQSPTcQxXrEg3EsB1sWHIqOoeWWpoy/view?usp=sharing
model = load_model(r"UCF.h5", compile = False)

# TWILIO
account_sid = 'AC4512273a08e04745bc4313e8740bb483'
auth_token = 'e86a599b868eae767e62cc2c19dfe8d3'
client = Client(account_sid, auth_token)

#INPUT IMAGES STORED IN THIS FOLDER
```

```
UPLOAD_FOLDER = os.path.join("static", "uploads")
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

#DATE TIME LOCATION
global date, day, time
dt = datetime.datetime.now()
date = str(dt.date())
day = str(dt.strftime('%A'))
time = str(dt.time().strftime('%H:%M:%S'))

ACCESS_TOKEN = '2dd3de5dc9a453'
def Location():
    url = f'https://ipinfo.io?token={ACCESS_TOKEN}'
    try:
        response = requests.get(url)
        data = response.json()

        location = data.get('loc')
        city = data.get('city')
        region = data.get('region')
        country = data.get('country')

        return [location, city, region, country]

    except Exception as e:
        return "Error detecting location"
location = Location()

#CHECK IS FILE IS OF ALLOWED EXTENSION
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
#DELETE FILES
def delete_files_in_uploads():
    files = os.listdir(UPLOAD_FOLDER)
    for file in files:
        os.remove(os.path.join(UPLOAD_FOLDER, file))
##########################################################################################
#########################
```

```
#HTML TEMPLATE ROUTES
@app.route('/')
def index():
    return render_template('index.html')

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

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

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

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

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

@app.route('/Public')
def Public():
    return render_template('Public.html')
#########################################################################################
#######################
#SEND MESSAGE TO PHONE (TWILIO)
@app.route('/send_sms', methods = ['POST'])
def send_sms():
    message = "Alert!!! " + res + " Detected on " + date + "(" + day + ")" + " at " + time +".\nLocation: " +
location[0] + "\nCity: " + location[1] + "\nRegion: " + location[2] + "\nCountry: " + location[3]
    try:
        message = client.messages.create(
            body = message,
            from_ = +12192242908,
```

```
            to = +919740963263
        )
        return "message sent"
    except Exception as e:
        return 'Failed to send message'

#UPLOAD IMAGE TO UPLOADS FOLDER
#when image selected from local storage
@app.route('/Upload_to_folder',methods = ['POST'])
def Upload_to_folder():
    if 'file' not in request.files:
        return 'No file part'
    file = request.files['file']
    if file.filename == '':
        return 'No selected file'
    if file and allowed_file(file.filename):
        delete_files_in_uploads()
        # Save the uploaded file to the "uploads" directory
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], file.filename))
        return 'File successfully uploaded and saved.'

    return 'Invalid file type. Only JPG, JPEG, and PNG files are allowed.'
#When image clicked using camera
@app.route('/Upload_to_folder_camera', methods = ['POST'])
def Upload_to_folder_camera():
    delete_files_in_uploads()
    image_data = request.json.get('image', None)
    if image_data:
        try:
            image_bytes = base64.b64decode(image_data.split(',')[-1])
            filename = secure_filename('capturedImage.jpg')
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            with open(file_path, 'wb') as f:
                f.write(image_bytes)
            return "FIle successfully uploaded"
        except Exception as e:
            return f"Error:{str(e)}"
    return "invalid image data"
```

```
#CLASSIFICATION
@app.route('/Classify', methods = ['POST'])
def Classify():
    global res
    button = request.headers.get('currentButton')
    if button == '1': #image uploaded from folder
        if 'filename' in request.form:
            filename = request.form['filename']
            image_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    if button == '0': #image captured using camera
        image_path = os.path.join(app.config['UPLOAD_FOLDER'], 'capturedImage.jpg')

    img = image.load_img(image_path, target_size = (64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis = 0)
    pred = np.argmax(model.predict(x), axis = 1)
    index = ['Abuse', 'Arrest', 'Arson', 'Assult', 'Burglary', 'Explosion', 'Fighting', 'Normal', 'Accident', 'Robbery',
'Shooting', 'Shoplifting', 'Stealing', 'Vandalism']
    res = str(index[pred[0]])
    return res

#GENERATE AND DOWNLOAD PDF
@app.route('/generate_pdf/<filename>')
def generate_pdf(filename):
    image_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    if not os.path.exists(image_path):
        return 'File not found'

    pdf_buffer = BytesIO()
    c = canvas.Canvas(pdf_buffer, pagesize=letter)
    img1 = Image.open(image_path)
    image_width, image_height = 256,256
    page_width, page_height = letter
    x = (page_width - image_width)/2
    y = (page_height - image_height)/2
    c.drawImage(image_path, x, y, width=image_width, height=image_height)

    c.setFont("Times-Bold", 26)
```

```
header_text = 'Crime Vision'
c.drawCentredString(page_width/2, page_height - 50, header_text )

res_text = "Classification Result: " + res
c.setFont("Times-Roman", 18)
c.drawString(50, y - 20,res_text)
y-=30
res_text2 = "Date: " + date
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text2)
y-=25
res_text3 = "Day: " + day
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text3)
y-=25
res_text4 = "Time: " + time
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text4)

y-=25
res_text5 = "Location: " + location[0]
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text5)

y-=25
res_text6 = "City: " + location[1]
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text6)

y-=25
res_text7 = "Region: " + location[2]
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text7)

y-=25
res_text8 = "Country: " + location[3]
c.setFont("Times-Roman", 18)
c.drawString(50, y - 10,res_text8)
```

```
    footer_text = "Location and Time data is purley based on the location of device generating pdf and NOT the image depicted"
    c.setFont("Times-Roman",14)
    c.drawCentredString(page_width/2, 10, footer_text)
    c.showPage()
    c.save()

    pdf_buffer.seek(0)
    return send_file(pdf_buffer, as_attachment=True, download_name='result.pdf')

@app.route('/Back')
def clear():
    delete_files_in_uploads()
if __name__ == "__main__":
    app.run(debug=True)
```

GitHub Repository Link:

https://github.com/smartinternz02/SI-GuidedProject-597912-1697648513

Project Demo Link:

https://drive.google.com/file/d/1-cZvV6As3FZf6lPJDno-w80zZ_nKLLNA/view?usp=drive_link