A PROJECT REPORT ON

POTHOLE DETECTION AND IT'S AVOIDANCE

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

Of

SAVITRIBAI PHULE PUNE UNIVERSITY

By

Deveyash Bharat	**B150234230**
Charudatta Bangal	**B150234224**
Aryan Vaid	**B150234217**
Siddharth Changed	**B150234235**
Aditya Raj	**B150234203**

Under the guidance of

PROF. S.N. BHOSALE

DEPARTMENT OF COMPUTER ENGINEERING
SINHGAD COLLEGE OF ENGINEERING, PUNE-41
Accredited by NAAC

2019-2020

Sinhgad Technical Education Society,
Sinhgad College of Engineering, Pune-41
Department of Computer Engineering

Sinhgad Institutes

Date: 23/09/2020

CERTIFICATE

This is to certify that the project report entitled

"POTHOLE DETECTION AND IT'S AVAOIDANCE"

Submitted by

Deveyash Bhurat	**B150234230**
Charudatta Bangal	**B150234224**
Aryan Vaid	**B150234217**
Siddharth Changede	**B150234235**
Aditya Raj	**B150234203**

Is a Bonafide work carried out under the supervision of Prof. S.N. BHOSALE and it is approved for the partial fulfilment of the requirements of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering) during the year 2019-2020.

Prof.S.N.Bhosale
Internal Guide
Department of Computer Engineering

Prof.M.P.Wankhade
Head,
Department of Computer Engineering

Dr. S. D. Lokhande
Principal
SCOE, Pune

Acknowledgement

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose cease less cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to our project guide Prof. S. N. Bhosale Ma'am for the guidance, inspiration and constructive suggestions that helped us in the preparation of this project. We are also thankful to all our mentors for the right guidance and advice at the crucial junctures and for showing us the right way. We owe our deep gratitude to Dr. S.D. Lokhande, Principal, Sinhgad College of Engineering and Prof. M.P. Wankhade, HOD Department of Computer Engineering, who took keen interest on our project work and for providing all facilities and every help for smooth progress of project.

Abstract

Pothole detection is one of the most important tasks for road maintenance. Computer vision approaches are generally based on either 2D road image analysis or 3D road surface modelling. However, these two categories are always used independently. Furthermore, the pothole detection accuracy is still far from satisfactory. The proposed pothole detection and avoid any system focuses not only on the detection of potholes but also assist the driver to avoid potholes; by giving directions towards left or right to avoid the pothole. Moreover, it will also have an automatic control over the speed of vehicle which will reduce the speed of the vehicle as the car approaches towards the pothole. All the data will be collected in the local database/cloud. The data will be further used for various types of analysis such as the road that has a greater number of potholes, tracking the lifeline of the road and prioritize the road reconstruction.

Table of Figures

List of Tables

Abbreviation

- SRS: - System Requirements
- UML: - Unified Modelling Language
- RL: - Reinforcement Learning
- AI: - Artificial Intelligence
- ML: - Machine Learning
- NLTK: - Natural Language Toolkit
- JSON: - JavaScript Object Notation
- CNN: - Convolutional Neural Networks

ACM Keyword

- IOT
- CNN
- Grey Scale
- Pothole Detection

TABLE OF CONTENTS

Chapter 1
INTRODUCTION

1.1 Background and Basics

Traffic congestion has been increasing world-wide as a result of increased motorization, urbanization, population growth and changes in population density. Congestion reduces utilization of the transportation infrastructure and increases travel time, air pollution, fuel consumption and most significantly, traffic accidents. While different factors contribute to the leading cause of around 95% percent of all accidents. The driver behaviour can be improved by either alerting him about the probable collision or controlling the vehicle itself.

At present, the commercially available traffic detecting equipment's include loop detectors, pressure sensors, infrared, radar, ultrasound-based sensors and video cameras. are cheap to manufacture, their installation and repair are very expensive because they involve digging and re-surfacing of the road, which is labour intensive, time consuming and causes disruption to the traffic. The pressure-based traffic sensors have the same problem. Infrared, radar and ultrasound sensors, on the other hand, are more expensive to make. The use of these active devices in urban areas may have safety and other regulatory implications. The effectiveness of This type of sensors can also be affected by bad weather. Also, the images have to be processed further to obtain useful data from them. Road conditions is a key part for safe and comfortable driving and also for maintaining traffic efficiency in one's day to day life. It is desirable to have a mechanism by which people can know about the road conditions on the routes on which they wish to travel, in real time.

Also understanding conditions of road surface is also very important for road maintenance and asset management.

The proposed system will provide, vehicle assistance to driver, in case if potholes are detected. Assistance includes giving directions to the driver, to move left or right in order to avoid potholes. System will consist of camera and an IR sensor. Camera will be used to detect speed-breaker.

1.2 Literature Survey

NAME OF THE PAPER	PUBLISHING YEAR	AUTHOR & PUBLISHING INSTITUTE	Proposed System
Pothole Detection Based on Disparity Transformation and Road Surface Modelling	2019	Rui Fan,Umar Ozunlap,BrettHosking,Mingliu,loannispitas	Pothole detection system by using 3-D surface modelling, disparity maps
Detecting a pothole Using Deep CNN Models for An Adaptive Shock Observing in A Vehicle Driving	2018	KwangEunAn,Sung Won Lee, Seung-Ki Ryu, DongmahnSeo.	To investigate the performance of deep Convolutional neural network in detecting a pothole
Machine Learning Approach for Predicting Bumps on Road.	2017	ManjushaGhadge, Dheerajpandey , Dhananjay kalbande	This paper proposes a smart phone-based method which uses Accelerometer and GPS sensors to analyze the road conditions
Pothole detection system using machine learning on android	2017	AniketKulkarni ,Nitishmhalgi,Sagargurnani,Dr. NupurGiri	This paper investigates an application of mobile sensing detection of pothole on roads. This system uses the accelerometer sensor of android smart phone for detection of pothole and GOS of mobile for plotting the location of pothole on google map.

Large-Scale Image Processing Research Cloud	2016	YuzhongYan,Lei Huang	This paper gives a brief idea about how images, videos which are being produced on large scale are processed with the help of image processing algorithm and image processing cloud by supporting the image process computing and big data analysis tech.
Machie Learnig based approach for pothole detection	2018	Karmel. A,Adhithiyan. M,Senthil Kumar. P	This paper explains the idea of automatic detection of pothole, damaged roads and cracks. In this paper the pothole and depression are framed. And data is converted into HSV colour space, sample patching for image mask and applying contour detection, convex hull calculation and final extraction of image

Table (1.2.1) Literature Survey

1.3 Project Undertaken

The collected data will be used by the IOT to and the relative condition of the road and the abnormal road condition will be marked. Real time classifications of road surface conditions is very important for the control of a vehicle properly within its handling limits.

1.3.1 Problem Definition

It consists of an IR based sensing module and a python-based user interface, it is to be designed and implemented for vehicles. Road conditions are a key part for the driving safety, comfort and efficiency of traffic in people’s day-to-day lives.

1.3.2 Scope Statement

It is to be designed and implemented for vehicles. Road conditions are a key part for the driving safety comfort and efficiency of traffic in people’s day-to-day lives. It is desirable to have a mechanism by which people can know about the road conditions in the routes on which they wish to travel, in real time. Also understanding conditions of road surface is also very important for road maintenance and asset management.

1.4 Organization of Project Report

The overall report consists of nine chapters. First chapter deals with introduction, in that we have included background and basics, literature survey and Project Undertaken.

- System Requirements Specification (SRS), Project Process Modelling, Cost and Effort Estimates and finally Project scheduling.
- Third chapter deals with Analysis and Design. In this we include Idea Matrix and UML Diagrams.
- Fourth chapter deals with Implementation and coding where we have provided all the database details, operational details, coding details, what each class deals with and screenshots of major functionalities.
- Fifth chapter deals with the testing modules implemented on the project. In this we include Unit Testing, Integration Testing, Acceptance Testing and GUI Testing.
- Sixth chapter deals with Results and discussion where main GUI screenshots result tables and discussion are being made.
- Seventh chapter is the conclusion where we conclude the whole project.
- Eighth and last chapter is future work where we discuss about the enhancement we can provide to the project in future. In this way we complete the report in the given sequence of chapters.

Chapter 2
PROJECT PLANNING AND MANAGEMENT

2.1 Introduction

This chapter covers the project planning and management details. It also covers System Requirement specifications. SRS is considered as the base for the effort estimations and project scheduling.

2.2 System Requirement Specification (SRS)

Detail System Requirement Specification (SRS)

2.2.1 System Overview

This purpose system provides an IR based sensing module and an IR based user interface. The low-cost modules are placed on the vehicles so its efficient for users as they don't have to have an external sensing function when they get on or off the vehicle. vehicle position is stored on a local/cloud database with information including latitude and longitude data.

2.2.2 Functional Requirements

- Detection of Pothole:
  Detecting most of the potholes on the road leaving some anomalies aside.
- Avoiding Potholes:

  Providing assistance
  Reducing speed of the vehicles as they approach near the pothole.

2.2.3 Non-Functional Requirements

Performance Requirements

- The performance of the functions and every module must be well. The overall performance of the software and its simplicity will enable the users to work efficiently.

Safety Requirements

- The application is designed in modules where errors can be detected and fixed easily. This makes it easier to install and update new functionality if required. The module used for reducing the speed will not reduce the speed suddenly rather will decrease it gradually avoiding the possibility of accidents.

Security Requirements

- All data will be encrypted using AES (Advanced Encryption Standard) encryption algorithm. The data will mainly include location of potholes from various areas. Location will be in the form of co-ordinates (Latitude, Longitude). The server where all the locations will be stored will have a pass key for security purposes.

Software Quality Attributes

1. **Adaptability:** This software is adaptable by all users.
2. **Availability:** This software is freely available to all users who have installed the system in their vehicles. The availability of the software is easy for everyone.
3. **Maintainability:** After the deployment of the project if any error occurs then it can be easily maintained by the software developer.
4. **Reliability:** The performance of the software is consistent and robust which will increase the reliability of the Software.
5. **User Friendliness:** The ease with which this system can be handled contributes to the user friendliness of the system.
6. **Integrity:** Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled. This system is well integrated
7. **Security:** The database in which location of the potholes is stored will have an admin module which will require authentication.
8. **Testability:** The software will be tested considering all the aspects.

2.2.4 Deployment Environment

Hardware Requirements

- Raspberry Pi

• System	:	Intel i3 Processor and above.
• Hard Disk	:	20GB
• Ram	:	4GB
• Sensors	:	IR(Infrared)
• Camera	:	Camera Module

Software Requirements

• Operating System	:	Windows7.
• Coding Language	:	Python3
• IDE	:	PyCharm
• Database	:	SQLite/Cloud

2.2.5 External Interface Requirements

User Interfaces

- User of the system will be provided with Graphical User Interface; there is no command line interface for the functions of the product. Communication Interfaces

- The system can use the HTTP protocol for communication over the Internet and for the intranet communication will be through TCP/IP protocol suite.

2.2.6 Other Requirements (Software Quality Attributes)

- Since the system must run over the Wi-Fi, all the hardware shall require to connect Wi-Fi will be hardware interface for the system. As for e.g. GPS, Cloud, IR sensor, Motor,

2.3 Project Process Modelling

The process model which will be followed while implementing the project is waterfall model. The waterfall model is the sequential approach, where each fundamental activity of a process represented as a separate phase, arranged in linear order. In the waterfall model, schedule of all the activities must be planned before starting the work. The phases of waterfall model are requirements, design, implementation, testing and maintenance. Here in the generative adversarial networks first the generator and discriminator models need to be implemented first separately. They are trained separately. Thus, the waterfall model is suitable for the project as it implements models step by step separately and then integrates the whole system.

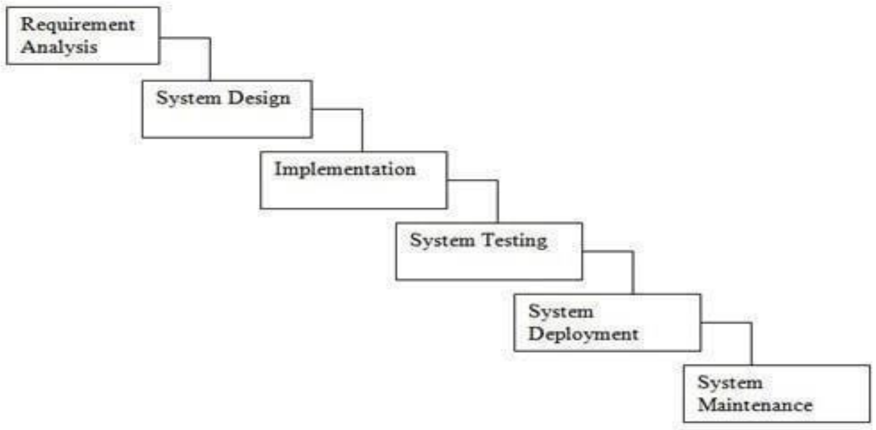


Figure 2.3.1: Waterfall Model

1. Requirement Gathering and Analysis: In this step of waterfall, we identify what are various requirements are need for our project such are software and hardware required, database, and interfaces.
2. System Design: In this system design phase, we design the system which is easily understood for end user i.e. user friendly. We design some UML diagrams and data flow diagram to understand the system flow and system module and sequence of execution.
3. Implementation: In implementation phase of our project, we have implemented various module required of successfully getting expected outcome at the different module levels. With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
4. Testing: The different test cases are performed to test whether the project modules are giving expected outcome in assumed time. All the units

developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

5. Deployment of System: Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
6. Maintenance: There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment. All these phases are cascaded to each other in which progress is seen as flowing steadily downwards like a waterfall through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and itis signed off, so the name Waterfall Model. In this model phases do not overlap.

2.4 Cost & Efforts Estimates

Cost Estimate:

The project cost can be found using any one of the models.

COCOMO-1 Model

COCOMO-2 Model

Model -1: The basic COCOMO model computes software development efforts as a function of program size expressed in estimated lines of code.

Model-2: The intermediate COCOMO model computes software development efforts as a function of program size and a set of cost drivers that include subjective assessment of the product, hardware, personnel, project attributes

Model-3 The advanced COCOMO model incorporates all characteristics of the intermediate version with an assessment of the cost drivers impact on each step of the software engineering process. Following is the basic COCOMO -2model.

Software Project	A(b)	B(b)	C(b)	D(b)
Organic	2.4	1.05	2.5	0.38

Semi-detached	3.0	1.22	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table (2.4.1) COCOMO Model

The basic COCOMO -2 model equations take form:

E=A(b)KLO

$C_{B(b)}$

$D=C(b)E^{D(b)}$

Where E is the effort applied in person months. D is development time in chronological month. KLOC is estimated number of delivered lines of code for the project. This project can be classified as Semidetached software project. The rough estimate of number of lines of this project is 9. 072k.Applying the above formula

$E=3.0*(9.072)^{1.22}$

= 44.20 person- months

$D=2.5* 44^{.35}$

$^{=}$ 9.40 months

Hence according COCOMO -2 model the time required for completion of the project is 9 (~9.40) months.

✞ Cost of Project:

Equation for calculation of cost of project using COCOMO - 2 model

is: $C = D * C_p$

Where,

C = Cost of project

D = Duration in month

C_p= Cost incurred per person-month, C_p=Rs.5000/- (per person-month) (approx.)

C = 9 * 5000= 45000/-

Hence according COCOMO - 2 model the cost of project is 45000/-(approx.)

2.5 Project Scheduling

Time Line Chart

Project timeline tracks the chronological order of events. These timelines give an under- standing of a project at just a glance, keeping informed and aligned at every stage of the project. The timeline is composed of a series of tasks, each of which has a due date and duration.

SR NO.	APPROVAL	DATE OF APPROVAL
1	Project Title Approval	4 September 2019
2	Problem Statement	11 September 2019
3	Objectives	24 September 2019
4	Research Gap	18 September 2019
5	Project Point of View	1 October 2019
6	Cost Estimation	1 October 2019
7	Requirements	3 October 2019
8	Risks Identification	3 October 2019
9	Risks Prioritization	4 October 2019

10	Review 1	5 October 2019
11	Review 2	7 October 2019

Table. (2.5) Timeline

Chapter 3
ANALYSIS AND DESIGN

3.1 Introduction

This chapter covers the analysis and design of the considered system.

3.2 IDEA Matrix

An IDEA matrix is a concept that evaluate various effects that the idea has. This tells us almost everything about the project.

I	D	E	A
INCREASE: Awareness in public accident.	**DRIVE:** Records to local database.	**EDUCATE:** User needs knowledge about vehicle and about the proposed system.	**ACCELERATE:** In this system we extend Road detection management system and we save the location of the pothole on the database which will help us in future analysis.
IMPROVE: Road journey experience as our journey will be hassle free.	**DELIVER:** Detects potholes and Speed breakers.	**EVALUATE:** Area wise detection of road anomalies.	**ASSOCIATE:** Vehicle assistance and optimum automatic speed reduction.
IGNORE: Traffic condition of the road.	**DECREASE:** The possibility of accidents and threat to life of passenger and driver.	**ELIMINATE:** The possibility of vehicle stopping on road due to road anomalies.	**AVOID:** Road anomalies and traffic congestion.

Table 3.2.1 IDEA Matrix

The letters in the word idea are the initials of the components analysed regarding the project. It is represented in a tabular form. Along with the component name,

its description is written beside it in another column.

3.3 Mathematical Model

Let S be the Whole system which consists:

S= {IP, Pro, OP}.

Where,

A. IP is the input of the system.

B. Pro is the procedure applied to the system to process the given input.

C. OP is the output of the system.

A. Input:

IP = {u, F,}.

Where,

1. You be the user.
2. F be set of files used for sending

B. Process

1. In this project Detection of bad road conditions such as potholes, bump, steep shoulders and objects on the road.
2. Real time road conditions in the routes on which they wish to travel.

C. Output:

Vehicles position on GPS with information including latitude and longitude data.

3.4 Feasibility Analysis

The proposed model comes under the non-deterministic polynomial time hardness in computational complexity theory because the proposed model is having constraints. There must be an available dataset for potholes in particular district. If any random pothole other than the dataset is given, then it cannot identify its authenticity in polynomial time. It cannot suggest whether it is suitable or not in polynomial time. Therefore, it is NP-Hard.

3.5 Architecture Diagram

Overview diagram of the system covering all the modules of the system.

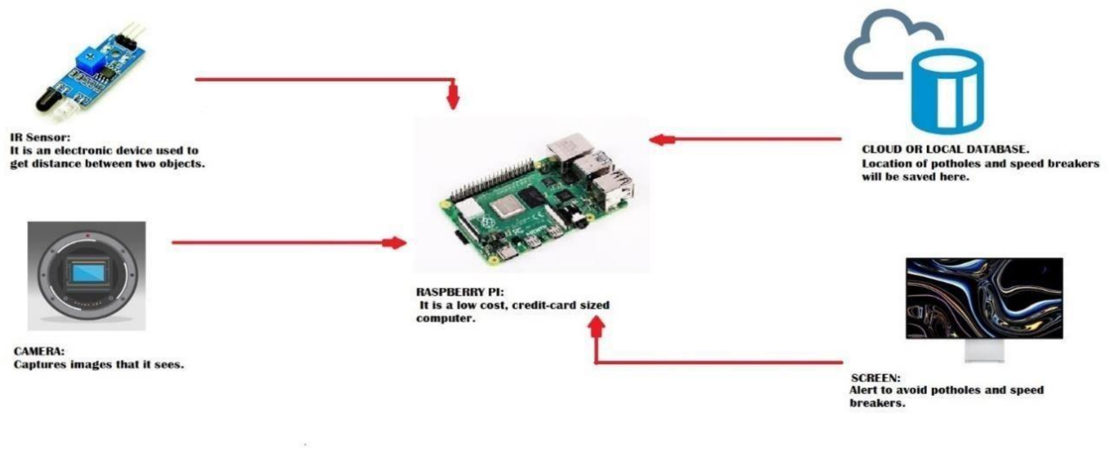

IR Sensor:
It is an electronic device used to get distance between two objects.
CLOUD OR LOCAL DATABASE.
Location of potholes and speed breakers will be saved here.
CAMERA:
Captures images that it sees.
RASPBERRY PI:
It is a low cost, credit-card sized computer.
SCREEN:
Alert to avoid potholes and speed breakers.

Figure 3.5.1: System Architecture

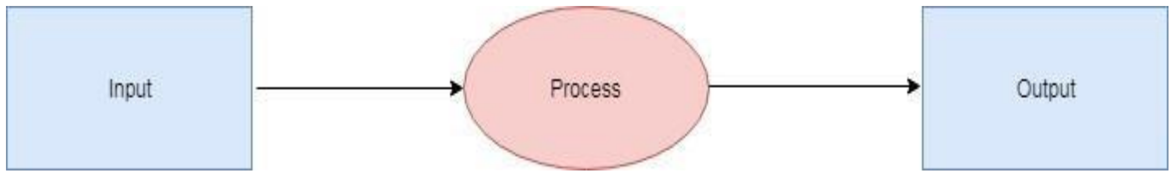


Figure 3.5.2: Dataflow Diagram (level 0)

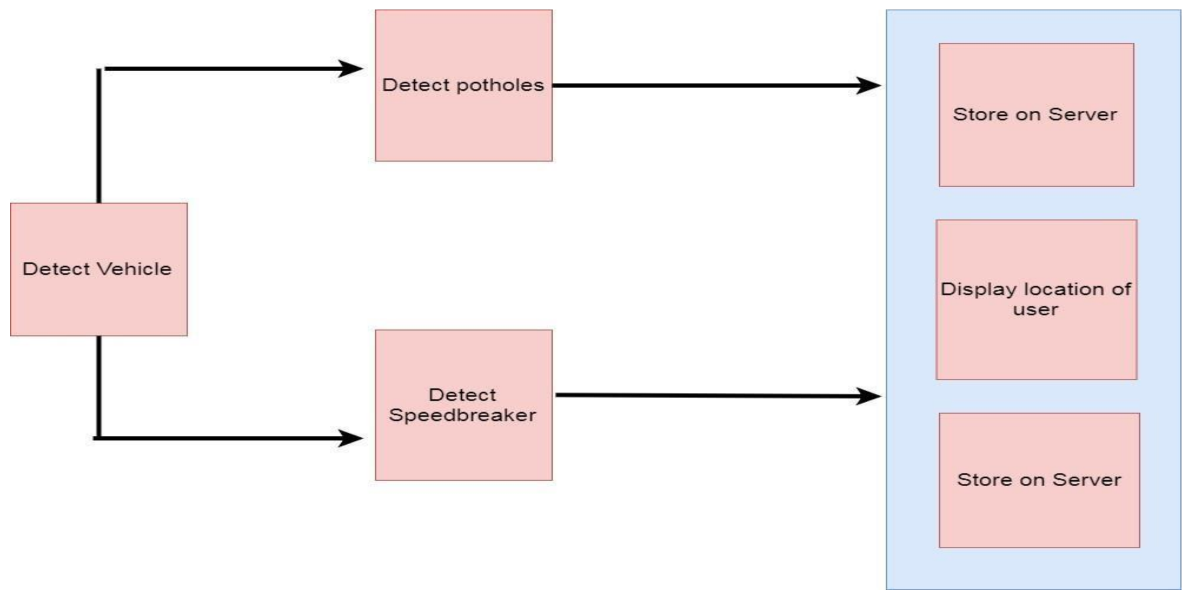


Figure 3.5.3: Dataflow Diagram (level 1)

3.6 UML Diagrams

3.6.1 Use-Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

This diagram is the use-case diagram for the overall application of the customer platform.

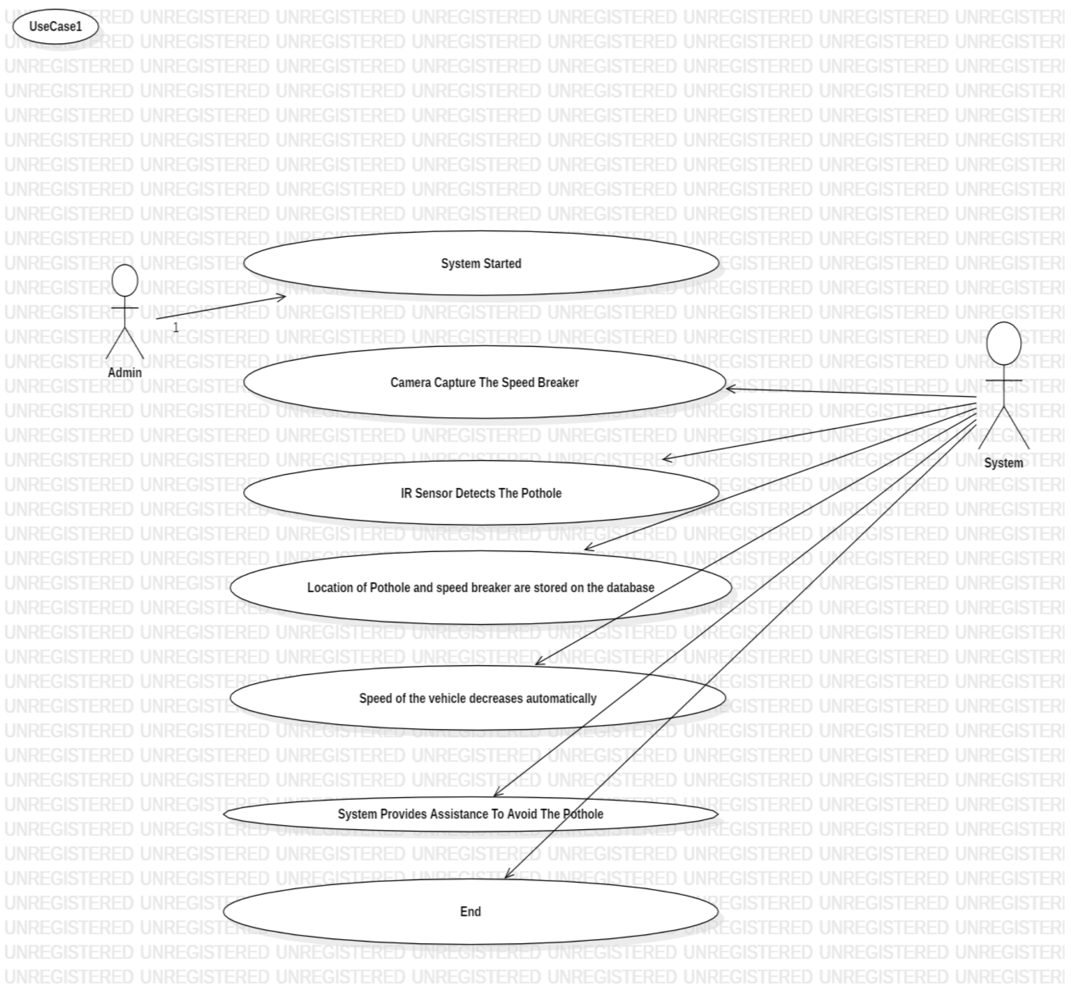

UseCase1
System Started
Admin
1
Camera Capture The Speed Breaker
System
IR Sensor Detects The Pothole
Location of Pothole and speed breaker are stored on the database
Speed of the vehicle decreases automatically
System Provides Assistance To Avoid The Pothole
End
UNREGISTERED

Fig. (3.6.1) Use-Case Diagram

3.6.2 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. This diagram represents various activities in the system along with the different actions that take place within each activity. It also depicts the flow of the actions within the activities and relation between different activities.

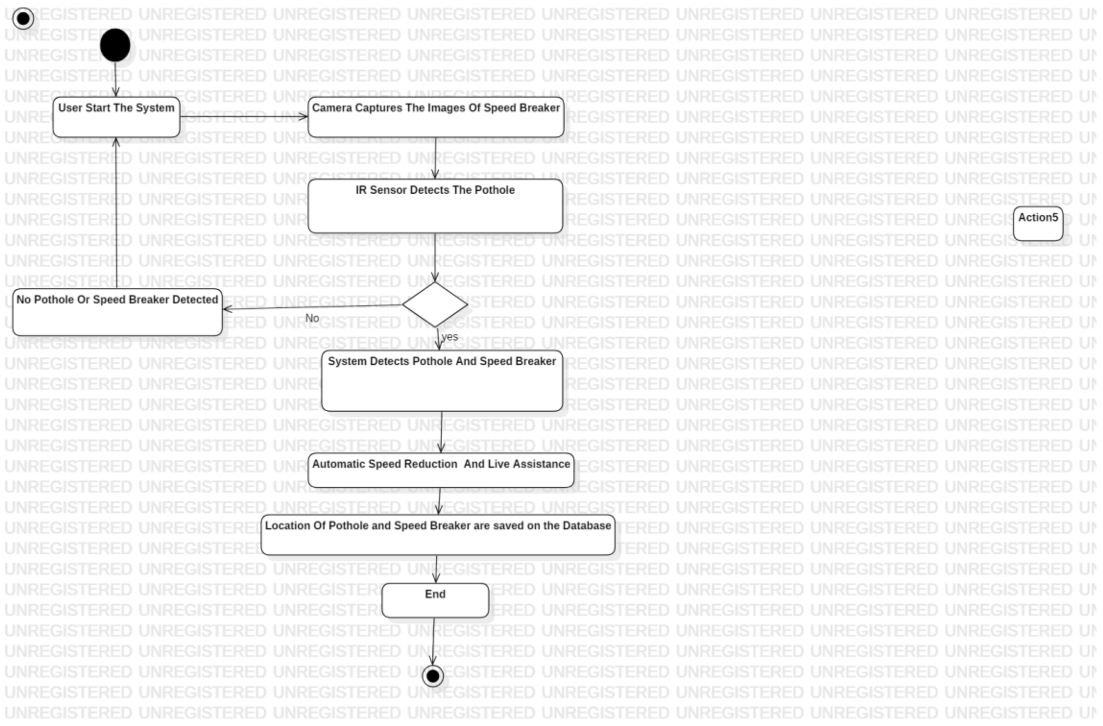

User Start The System
Camera Captures The Images Of Speed Breaker
IR Sensor Detects The Pothole
Action5
No Pothole Or Speed Breaker Detected
No
yes
System Detects Pothole And Speed Breaker
Automatic Speed Reduction And Live Assistance
Location Of Pothole and Speed Breaker are saved on the Database
End

Fig. (3.6.2) Activity Diagram

3.6.3 Class Diagrams

A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The interactions of various classes and their members is shown.

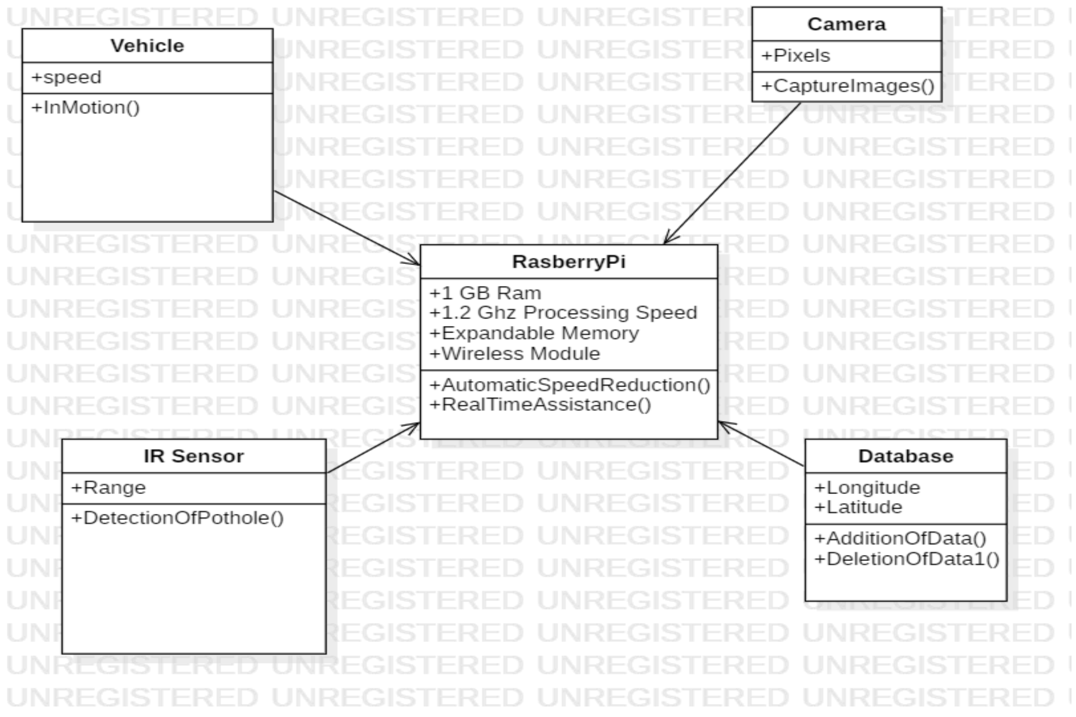


Fig. (3.6.3) Class Diagram

3.6.4 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms

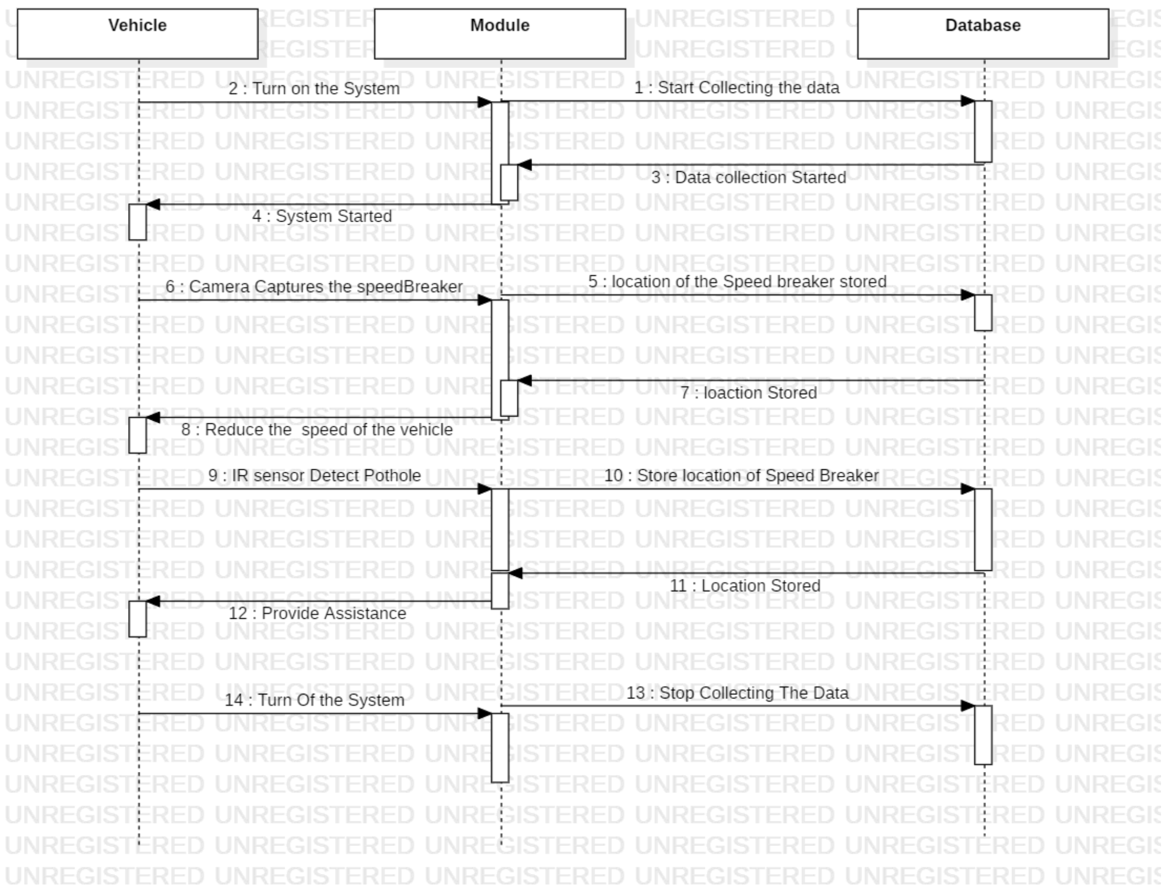

Vehicle
Module
Database
2 : Turn on the System
1 : Start Collecting the data
3 : Data collection Started
4 : System Started
6 : Camera Captures the speedBreaker
5 : location of the Speed breaker stored
7 : loaction Stored
8 : Reduce the speed of the vehicle
9 : IR sensor Detect Pothole
10 : Store location of Speed Breaker
11 : Location Stored
12 : Provide Assistance
14 : Turn Of the System
13 : Stop Collecting The Data

Fig. (3.6.4) Sequence Diagram

vent diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function.

3.6.5 Deployment Diagram

A deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modelling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

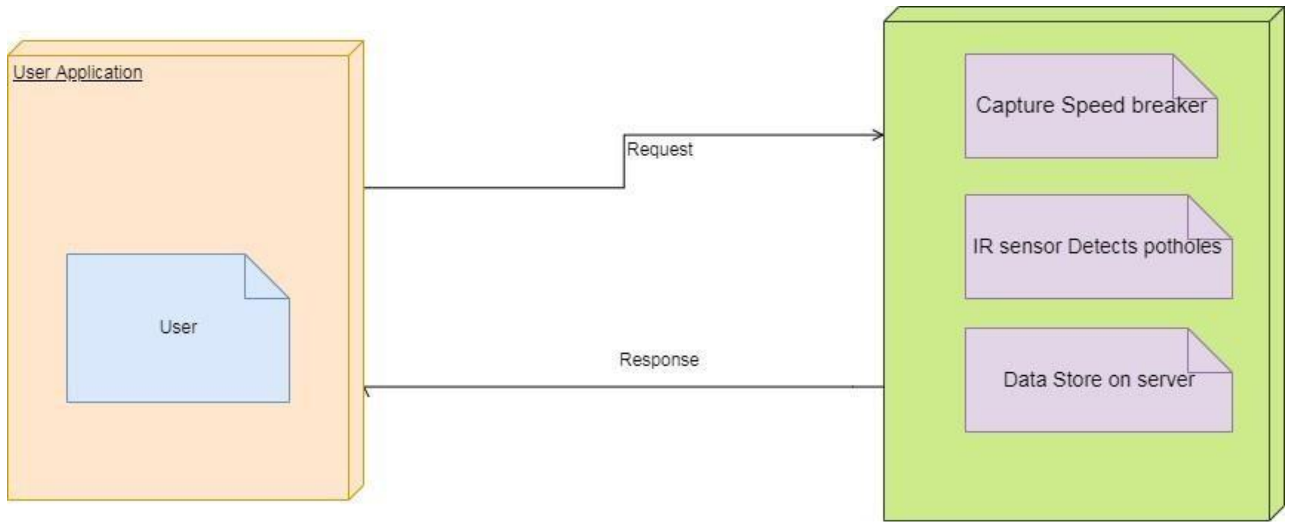


Fig. (3.6.5) Deployment Diagram

Chapter 4
IMPLEMENTATION & CODING

4.1 Introduction

This chapter covers the role of various subsystems/modules/classes along with implementation details listing of the code for the major functionalities.

In this coordination there is an IR based sensing component and IR based user interface. The low costcomponentsareplacedontheautomobilessoitsefficientforusersastheydon’thavetoought to an external sensing function when they get on or off the automobiles. Vehicles position on GPS with statistics including latitude and longitude data.

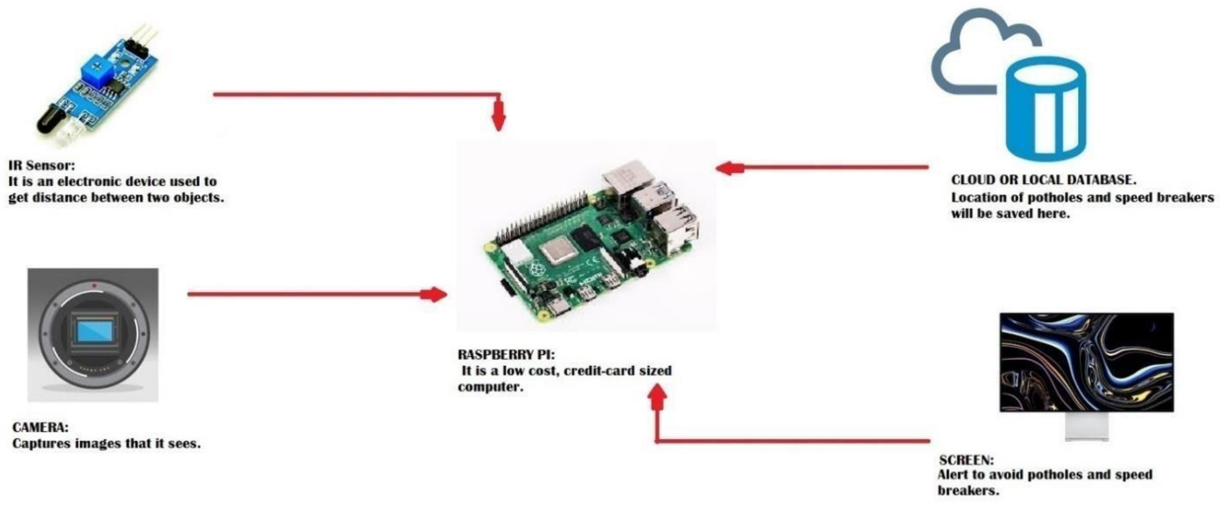
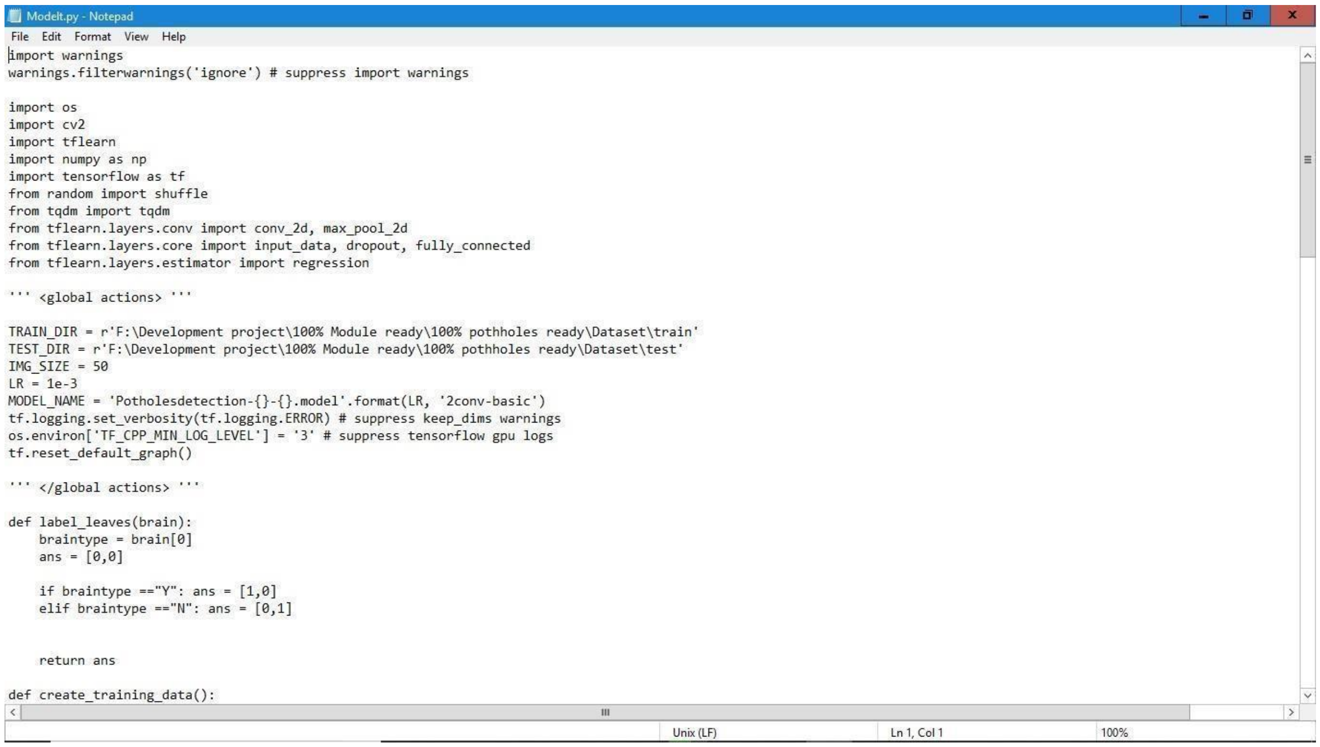

IR Sensor:
It is an electronic device used to get distance between two objects.
CAMERA:
Captures images that it sees.
RASPBERRY PI:
It is a low cost, credit-card sized computer.
CLOUD OR LOCAL DATABASE.
Location of potholes and speed breakers will be saved here.
SCREEN:
Alert to avoid potholes and speed breakers.

Fig (4.1.1) System Architecture-1

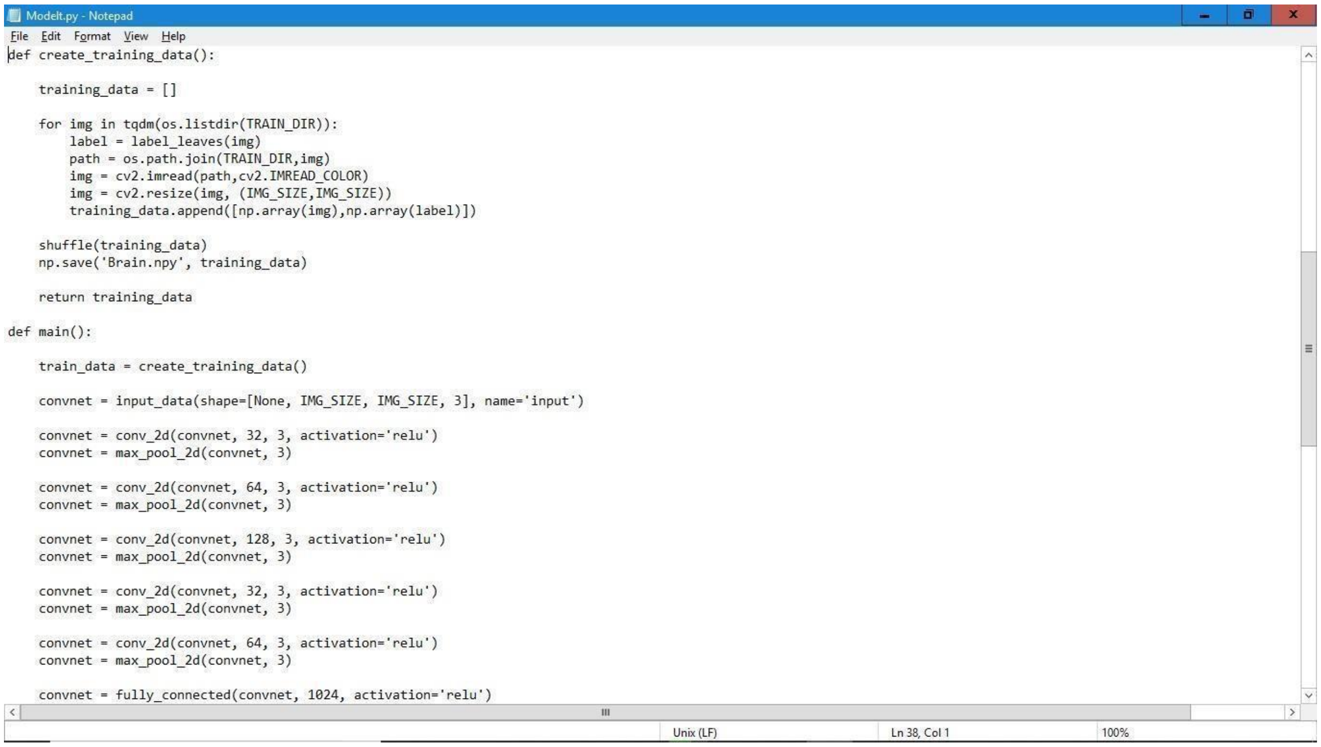
Fig (4.1.2) System Architecture-2

4.2 Database



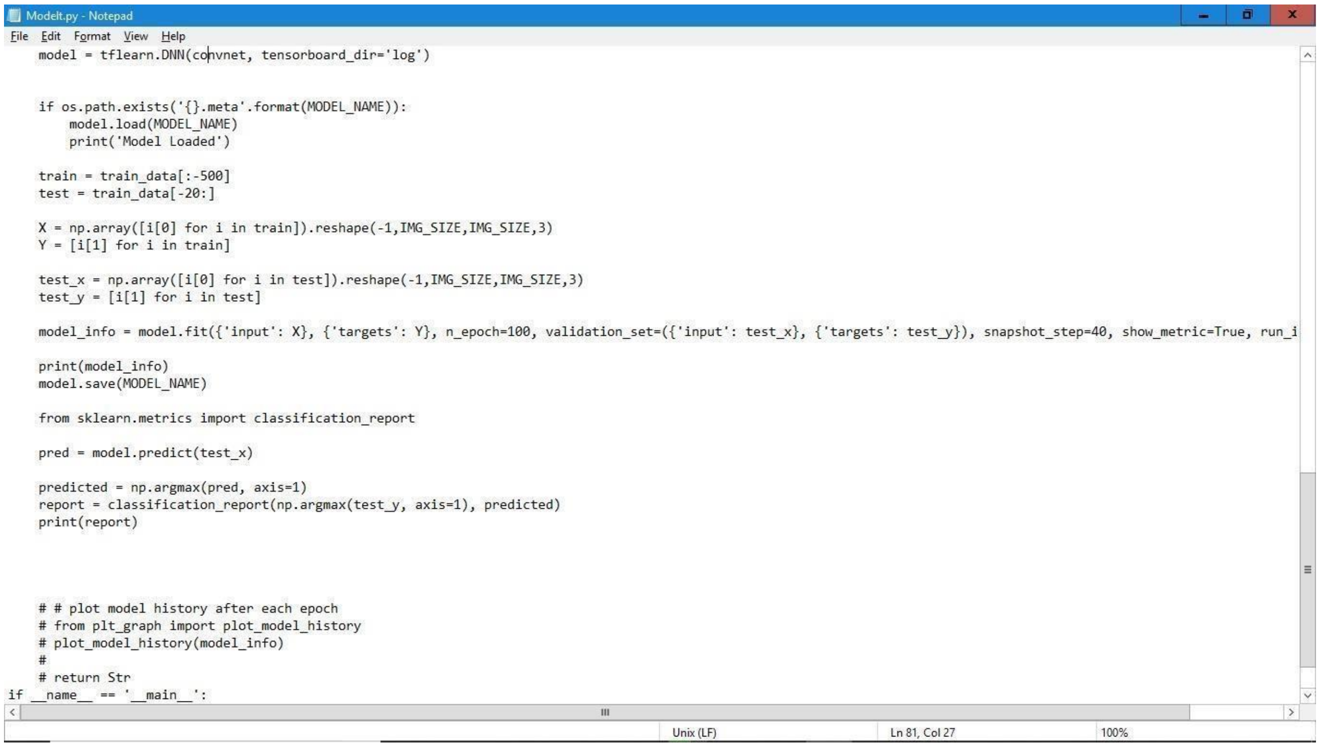
```
import warnings
warnings.filterwarnings('ignore') # suppress import warnings

import os
import cv2
import tflearn
import numpy as np
import tensorflow as tf
from random import shuffle
from tqdm import tqdm
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

''' <global actions> '''

TRAIN_DIR = r'F:\Development project\100% Module ready\100% pothholes ready\Dataset\train'
TEST_DIR = r'F:\Development project\100% Module ready\100% pothholes ready\Dataset\test'
IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = 'Potholesdetection-{}-{}.model'.format(LR, '2conv-basic')
tf.logging.set_verbosity(tf.logging.ERROR) # suppress keep_dims warnings
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # suppress tensorflow gpu logs
tf.reset_default_graph()

''' </global actions> '''

def label_leaves(brain):
    braintype = brain[0]
    ans = [0,0]

    if braintype =="Y": ans = [1,0]
    elif braintype =="N": ans = [0,1]


    return ans

def create_training_data():
```

Fig (4.2.1) Model Training-1



```
def create_training_data():

    training_data = []

    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_leaves(img)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])

    shuffle(training_data)
    np.save('Brain.npy', training_data)

    return training_data

def main():

    train_data = create_training_data()

    convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 128, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 32, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = conv_2d(convnet, 64, 3, activation='relu')
    convnet = max_pool_2d(convnet, 3)

    convnet = fully_connected(convnet, 1024, activation='relu')
```

Fig (4.2.2) Model Training-2



```
    model = tflearn.DNN(convnet, tensorboard_dir='log')


    if os.path.exists('{}.meta'.format(MODEL_NAME)):
        model.load(MODEL_NAME)
        print('Model Loaded')

    train = train_data[:-500]
    test = train_data[-20:]

    X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
    Y = [i[1] for i in train]

    test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,3)
    test_y = [i[1] for i in test]

    model_info = model.fit({'input': X}, {'targets': Y}, n_epoch=100, validation_set=({'input': test_x}, {'targets': test_y}), snapshot_step=40, show_metric=True, run_i

    print(model_info)
    model.save(MODEL_NAME)

    from sklearn.metrics import classification_report

    pred = model.predict(test_x)

    predicted = np.argmax(pred, axis=1)
    report = classification_report(np.argmax(test_y, axis=1), predicted)
    print(report)



    # # plot model history after each epoch
    # from plt_graph import plot_model_history
    # plot_model_history(model_info)
    #
    # return Str
if __name__ == '__main__':
```

Fig (4.2.3) Model Training-3

Operational Details

This module covers functionalities.

Tools Used Are:

•OpenCV python OpenCV-

Python is a library of Python bindings designed to solve computer vision problems. OpenCV (Open-Source Computer Vision Library) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language. Open CV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

•TensorFlow

TensorFlow is an open-source software library for machine learning across a range of tasks. It is a symbolic math library, and also used as a system for building and training neural networks to detect and decipher patterns and correlations, analogous to human learning and reasoning. Tensor Flow provides a Python API as well as C++, Haskell, Java, Go and Rust APIs.

- PyCharm
- Anaconda

Algorithm

- Convolutional neural network is one of the main categories to do image recognition, Image classification, object detection widely used.

- CNN image classification takes the input image, process it and classify' it. Computer sees an input image as array of pixels depends on the image resolution. (h*w*h)

  - Convolutional layer
  - Non-Linearity (ReLU)Layer
  - Pooling Layer
  - Fully connected Layer

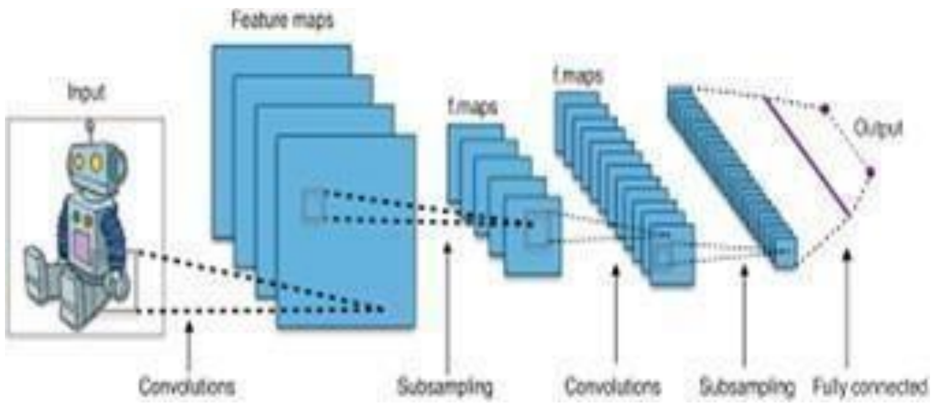
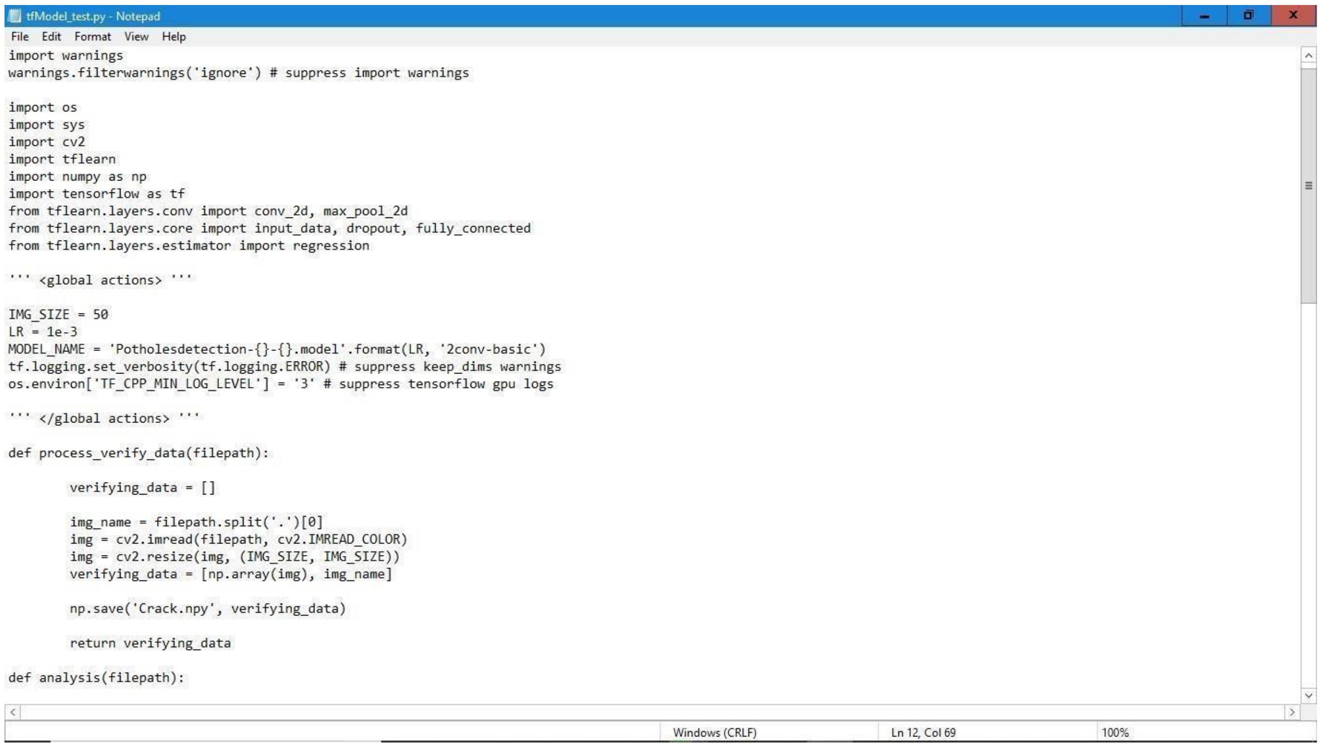


Fig. (4.3.1.1) CNN Layer

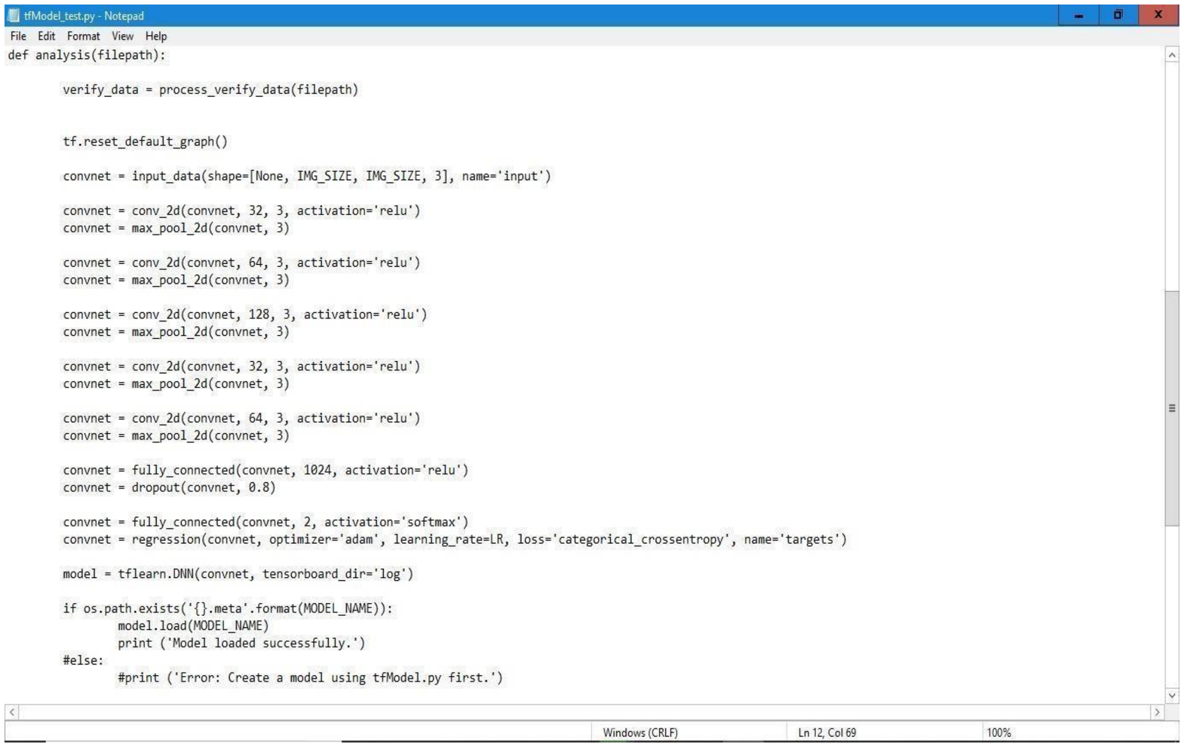
4.2.1 Major classes

Major classes from each module with their responsibilities.



```
import warnings
warnings.filterwarnings('ignore') # suppress import warnings

import os
import sys
import cv2
import tflearn
import numpy as np
import tensorflow as tf
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

''' <global actions> '''

IMG_SIZE = 50
LR = 1e-3
MODEL_NAME = 'Potholesdetection-{}-{}.model'.format(LR, '2conv-basic')
tf.logging.set_verbosity(tf.logging.ERROR) # suppress keep_dims warnings
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # suppress tensorflow gpu logs

''' </global actions> '''

def process_verify_data(filepath):

        verifying_data = []

        img_name = filepath.split('.')[0]
        img = cv2.imread(filepath, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        verifying_data = [np.array(img), img_name]

        np.save('Crack.npy', verifying_data)

        return verifying_data

def analysis(filepath):
```

Fig. (4.3.2.1) Model Testing-1

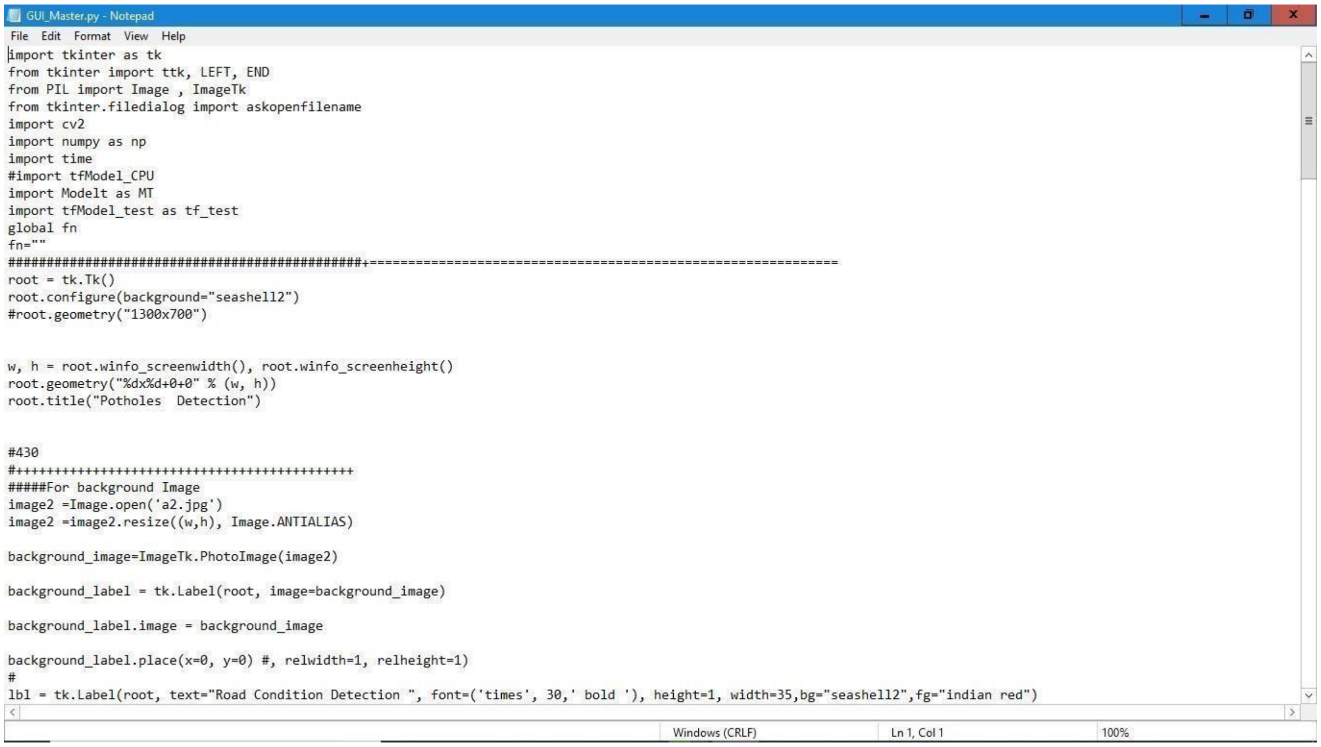


```
def analysis(filepath):

        verify_data = process_verify_data(filepath)


        tf.reset_default_graph()

        convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

        convnet = conv_2d(convnet, 32, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = conv_2d(convnet, 64, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = conv_2d(convnet, 128, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = conv_2d(convnet, 32, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = conv_2d(convnet, 64, 3, activation='relu')
        convnet = max_pool_2d(convnet, 3)

        convnet = fully_connected(convnet, 1024, activation='relu')
        convnet = dropout(convnet, 0.8)

        convnet = fully_connected(convnet, 2, activation='softmax')
        convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

        model = tflearn.DNN(convnet, tensorboard_dir='log')

        if os.path.exists('{}.meta'.format(MODEL_NAME)):
                model.load(MODEL_NAME)
                print ('Model loaded successfully.')
        #else:
                #print ('Error: Create a model using tfModel.py first.')
```

Fig. (4.3.2.2) Model Testing-2

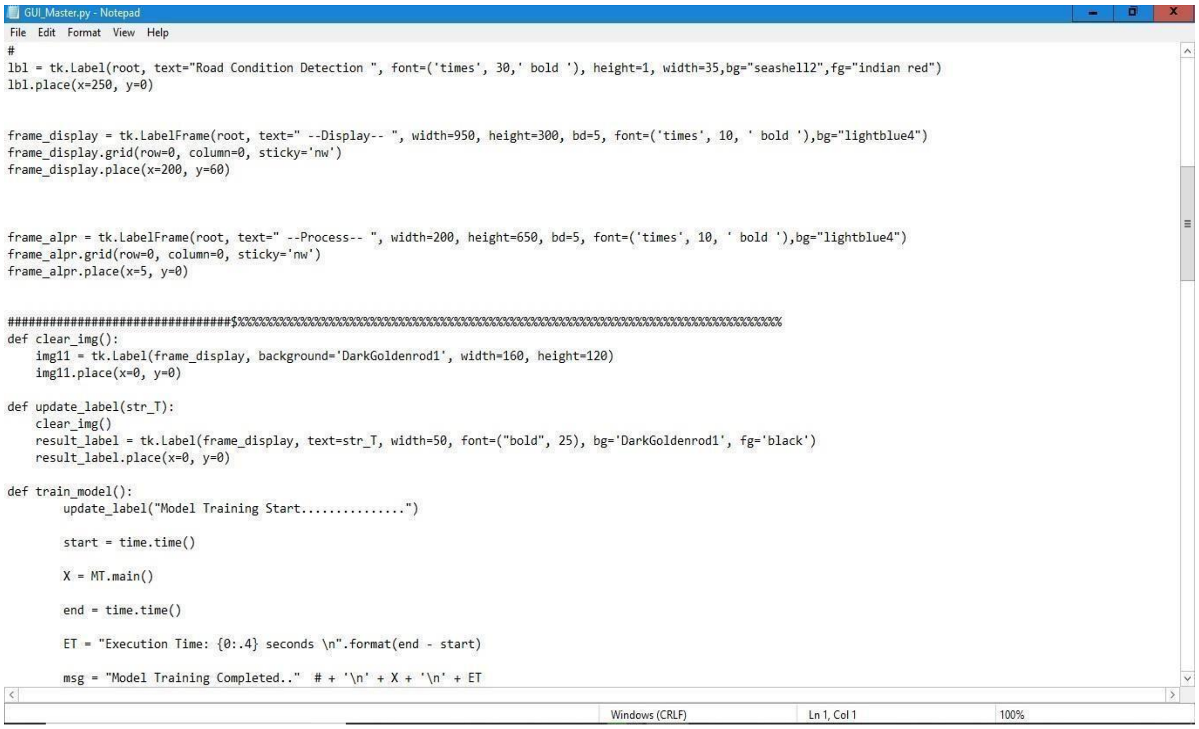
4.2.2 Code Listing

Sample code for the major functional requirements. Code is show in the images attached.



```
import tkinter as tk
from tkinter import ttk, LEFT, END
from PIL import Image , ImageTk
from tkinter.filedialog import askopenfilename
import cv2
import numpy as np
import time
#import tfModel_CPU
import Modelt as MT
import tfModel_test as tf_test
global fn
fn=""
##############################################+============================================================
root = tk.Tk()
root.configure(background="seashell2")
#root.geometry("1300x700")


w, h = root.winfo_screenwidth(), root.winfo_screenheight()
root.geometry("%dx%d+0+0" % (w, h))
root.title("Potholes  Detection")


#430
#++++++++++++++++++++++++++++++++++++++++++++
#####For background Image
image2 =Image.open('a2.jpg')
image2 =image2.resize((w,h), Image.ANTIALIAS)

background_image=ImageTk.PhotoImage(image2)

background_label = tk.Label(root, image=background_image)

background_label.image = background_image

background_label.place(x=0, y=0) #, relwidth=1, relheight=1)
#
lbl = tk.Label(root, text="Road Condition Detection ", font=('times', 30,' bold '), height=1, width=35,bg="seashell2",fg="indian red")
```

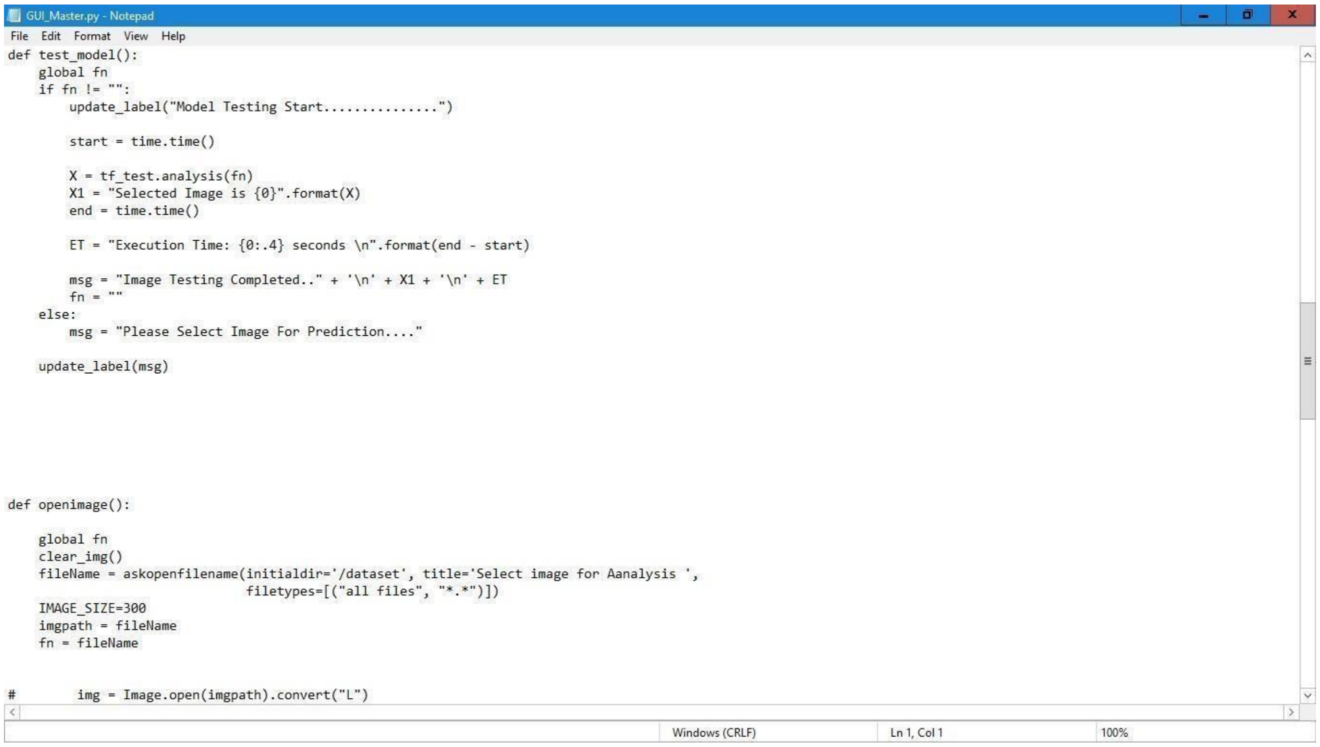
Fig. (4.3.3.1) GUI.py-1



```
#
lbl = tk.Label(root, text="Road Condition Detection ", font=('times', 30,' bold '), height=1, width=35,bg="seashell2",fg="indian red")
lbl.place(x=250, y=0)


frame_display = tk.LabelFrame(root, text=" --Display-- ", width=950, height=300, bd=5, font=('times', 10, ' bold '),bg="lightblue4")
frame_display.grid(row=0, column=0, sticky='nw')
frame_display.place(x=200, y=60)



frame_alpr = tk.LabelFrame(root, text=" --Process-- ", width=200, height=650, bd=5, font=('times', 10, ' bold '),bg="lightblue4")
frame_alpr.grid(row=0, column=0, sticky='nw')
frame_alpr.place(x=5, y=0)


################################$%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
def clear_img():
    img11 = tk.Label(frame_display, background='DarkGoldenrod1', width=160, height=120)
    img11.place(x=0, y=0)

def update_label(str_T):
    clear_img()
    result_label = tk.Label(frame_display, text=str_T, width=50, font=("bold", 25), bg='DarkGoldenrod1', fg='black')
    result_label.place(x=0, y=0)

def train_model():
        update_label("Model Training Start...............")

        start = time.time()

        X = MT.main()

        end = time.time()

        ET = "Execution Time: {0:.4} seconds \n".format(end - start)

        msg = "Model Training Completed.."  # + '\n' + X + '\n' + ET
```

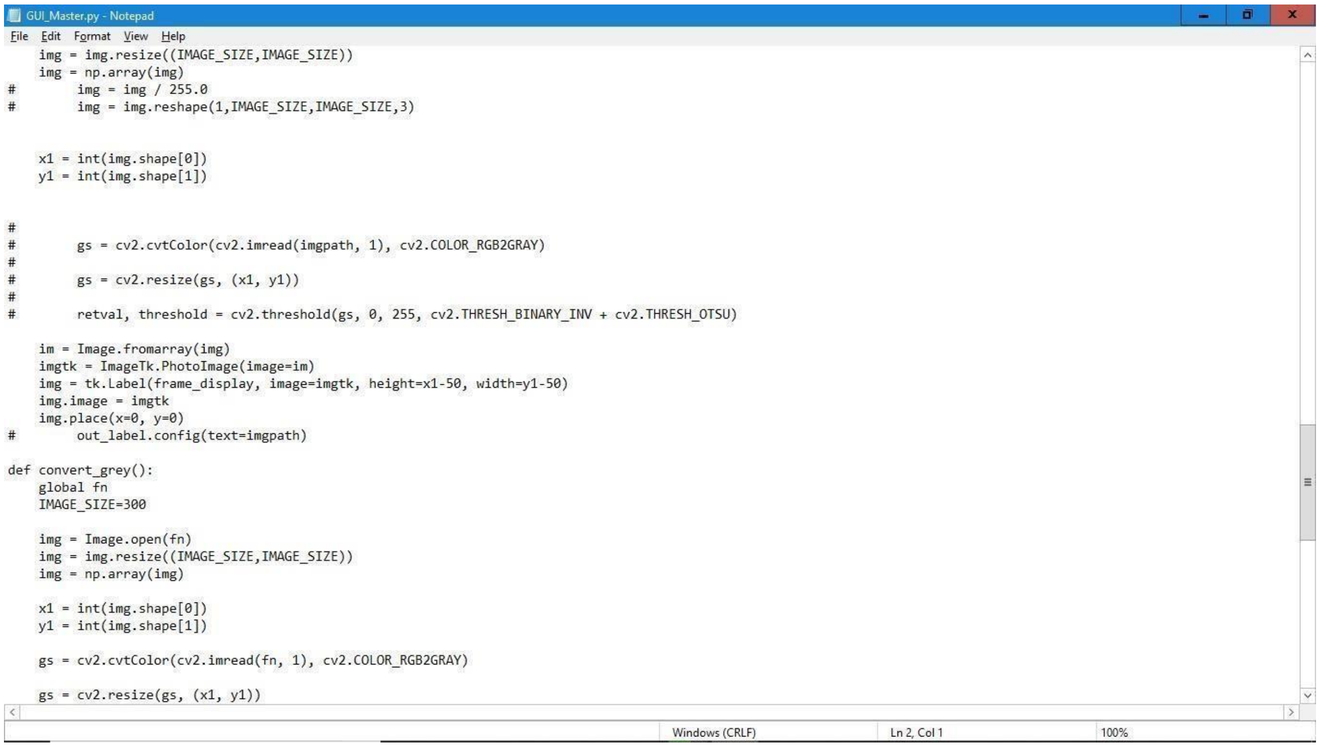
Fig. (4.3.3.2) GUI.py-2

4.3 Screenshots



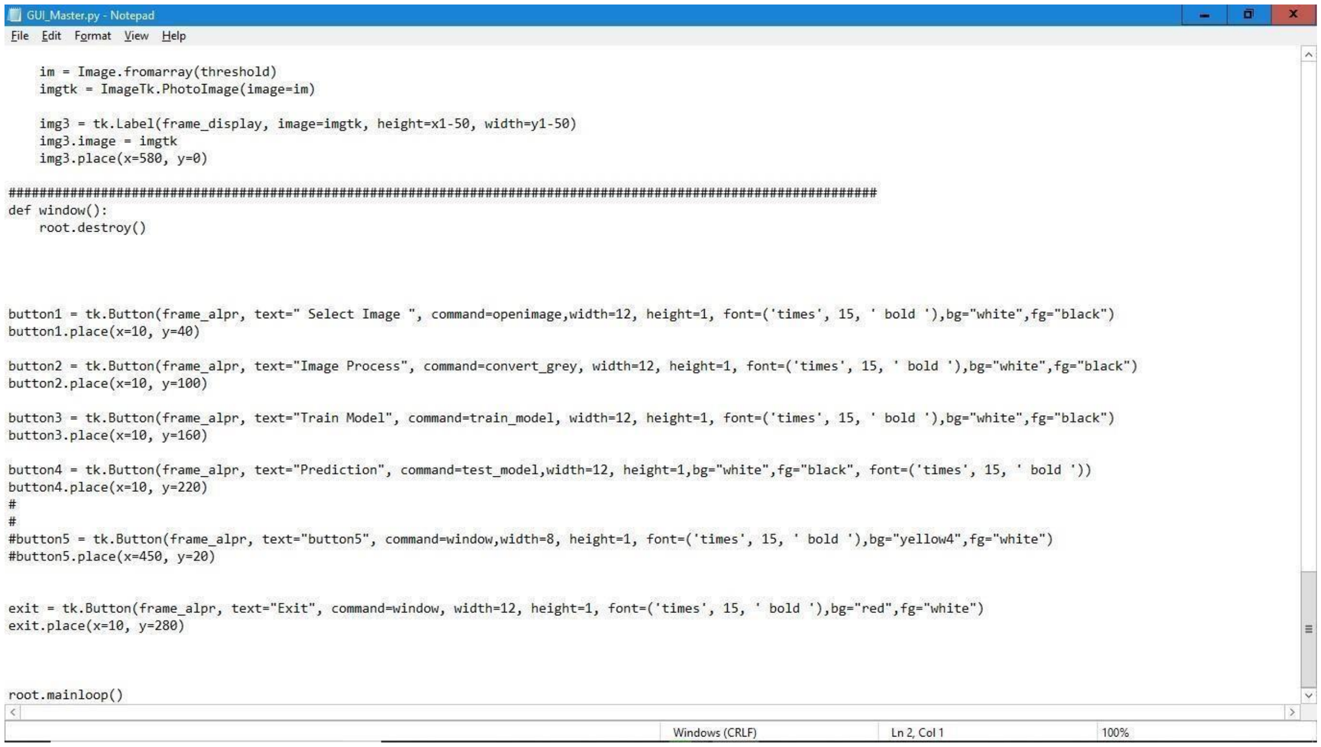
```
def test_model():
    global fn
    if fn != "":
        update_label("Model Testing Start..............")

        start = time.time()

        X = tf_test.analysis(fn)
        X1 = "Selected Image is {0}".format(X)
        end = time.time()

        ET = "Execution Time: {0:.4} seconds \n".format(end - start)

        msg = "Image Testing Completed.." + '\n' + X1 + '\n' + ET
        fn = ""
    else:
        msg = "Please Select Image For Prediction...."

    update_label(msg)




def openimage():

    global fn
    clear_img()
    fileName = askopenfilename(initialdir='/dataset', title='Select image for Aanalysis ',
                               filetypes=[("all files", "*.*")])
    IMAGE_SIZE=300
    imgpath = fileName
    fn = fileName


#        img = Image.open(imgpath).convert("L")
```

Fig. (4.4.1) GUI.py-3



```
    img = img.resize((IMAGE_SIZE,IMAGE_SIZE))
    img = np.array(img)
#        img = img / 255.0
#        img = img.reshape(1,IMAGE_SIZE,IMAGE_SIZE,3)


    x1 = int(img.shape[0])
    y1 = int(img.shape[1])


#
#        gs = cv2.cvtColor(cv2.imread(imgpath, 1), cv2.COLOR_RGB2GRAY)
#
#        gs = cv2.resize(gs, (x1, y1))
#
#        retval, threshold = cv2.threshold(gs, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

    im = Image.fromarray(img)
    imgtk = ImageTk.PhotoImage(image=im)
    img = tk.Label(frame_display, image=imgtk, height=x1-50, width=y1-50)
    img.image = imgtk
    img.place(x=0, y=0)
#        out_label.config(text=imgpath)

def convert_grey():
    global fn
    IMAGE_SIZE=300

    img = Image.open(fn)
    img = img.resize((IMAGE_SIZE,IMAGE_SIZE))
    img = np.array(img)

    x1 = int(img.shape[0])
    y1 = int(img.shape[1])

    gs = cv2.cvtColor(cv2.imread(fn, 1), cv2.COLOR_RGB2GRAY)

    gs = cv2.resize(gs, (x1, y1))
```

Fig. (4.4.2) GUI.py-4



GUI_Master.py - Notepad

```
    im = Image.fromarray(threshold)
    imgtk = ImageTk.PhotoImage(image=im)

    img3 = tk.Label(frame_display, image=imgtk, height=x1-50, width=y1-50)
    img3.image = imgtk
    img3.place(x=580, y=0)

##############################################################################################################
def window():
    root.destroy()



button1 = tk.Button(frame_alpr, text=" Select Image ", command=openimage,width=12, height=1, font=('times', 15, ' bold '),bg="white",fg="black")
button1.place(x=10, y=40)

button2 = tk.Button(frame_alpr, text="Image Process", command=convert_grey, width=12, height=1, font=('times', 15, ' bold '),bg="white",fg="black")
button2.place(x=10, y=100)

button3 = tk.Button(frame_alpr, text="Train Model", command=train_model, width=12, height=1, font=('times', 15, ' bold '),bg="white",fg="black")
button3.place(x=10, y=160)

button4 = tk.Button(frame_alpr, text="Prediction", command=test_model,width=12, height=1,bg="white",fg="black", font=('times', 15, ' bold '))
button4.place(x=10, y=220)
#
#
#button5 = tk.Button(frame_alpr, text="button5", command=window,width=8, height=1, font=('times', 15, ' bold '),bg="yellow4",fg="white")
#button5.place(x=450, y=20)


exit = tk.Button(frame_alpr, text="Exit", command=window, width=12, height=1, font=('times', 15, ' bold '),bg="red",fg="white")
exit.place(x=10, y=280)



root.mainloop()
```

Windows (CRLF) Ln 2, Col 1 100%

Fig. (4.4.3) GUI.py-5

Chapter 5
TESTING

5.1 Introduction

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product.

This chapter covers the testing approach used and the test cases.

5.1**UnitTesting**

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

5.2 Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It occurs after unit testing and before validation testing.

5.3 Acceptance Testing

In engineering and its various sub disciplines, acceptance testing is a test conducted to determine if the requirements of a specification or contract are met. It may involve chemical tests, physical tests, or performance tests.

Chapter 6
RESULTS & DISCUSSION

6.1 Main GUI snapshots

Preprocessing

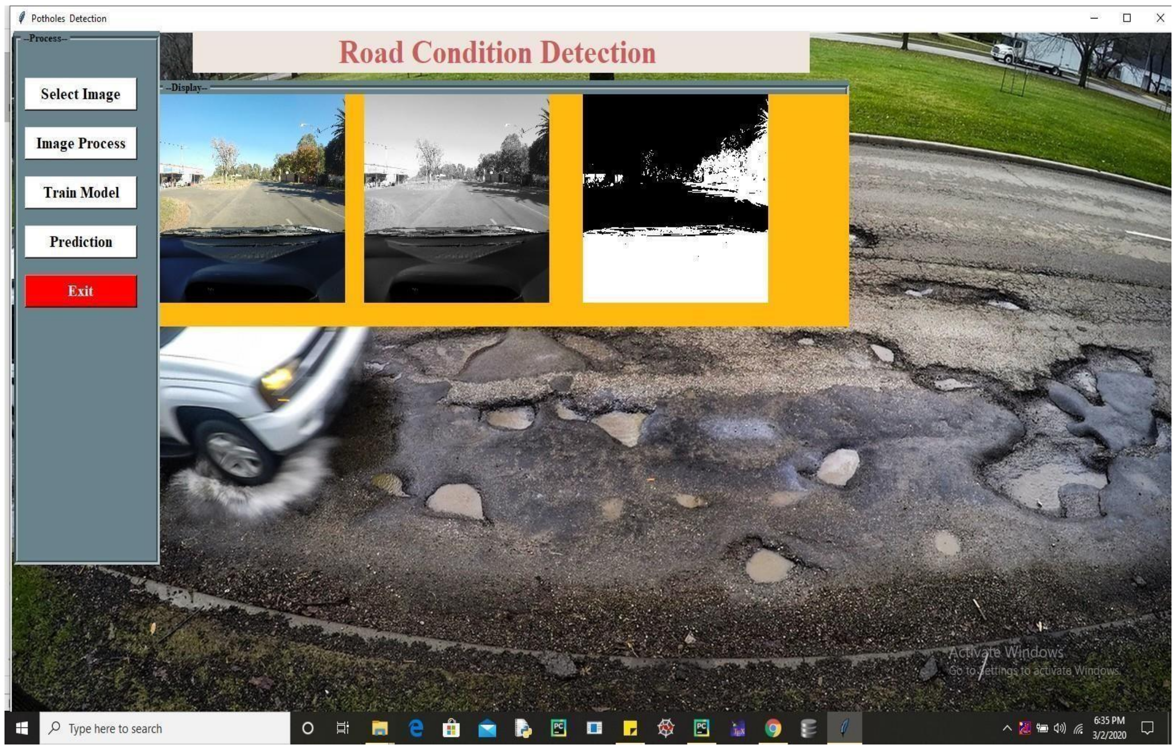
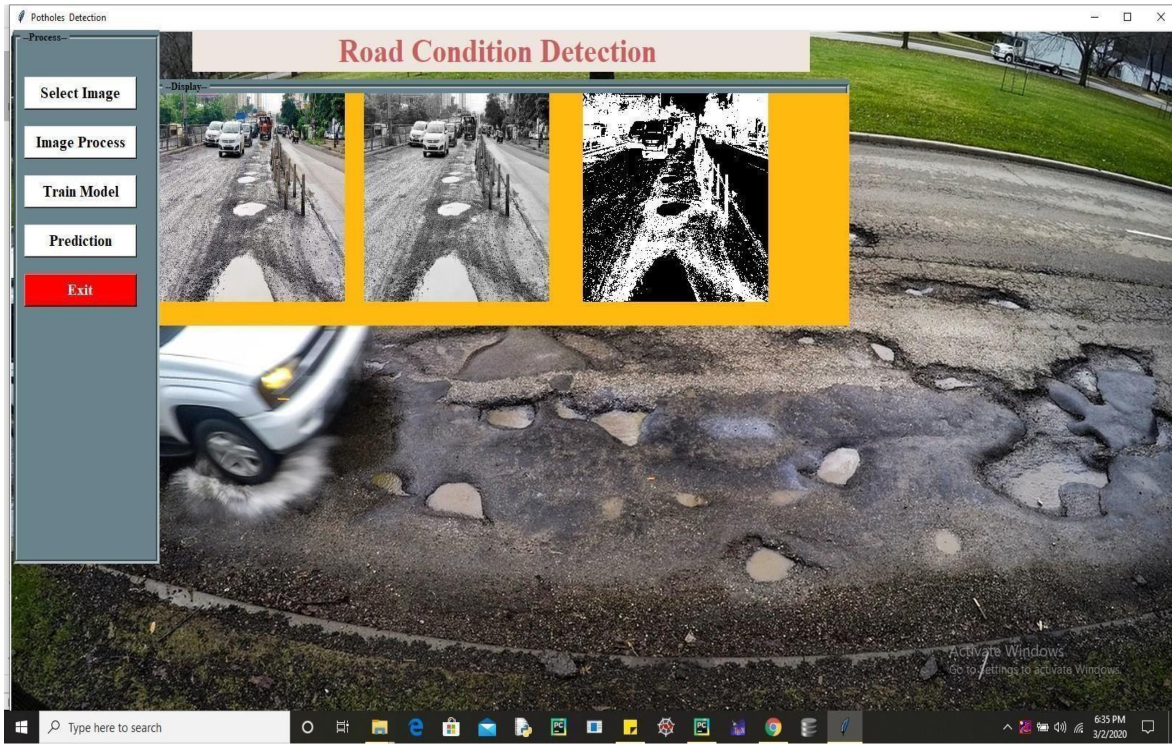


Fig (6.1.1) GUI-1



Outcome

Fig. (6.1.2) GUI-2

6.2 Results

In our system camera captures the object and verify in the system. then it will find the object and detect the object by its name.

In tracking, an object can be defined as anything that is of interest. For example, vehicles on a road, cricket match ball. Any object which is present in the nearby environment that maybe important to track in a specific domain. The appearance and shapes can be represented by object. First, we will describe the representation of object shape. Representation of objects is very important in object detection and tracking. There are various ways used to represent objects. Deep convolutional neural networks have recently achieved state-of-the-art performance on a number of image recognition benchmarks, including the Ima- geNet. The winning model on the localization sub-task was a network that predicts a single bounding box and a confidences core for each object category in the image. Such a model captures the whole-image context around the objects. In this work, we propose a saliency-inspired neural network model for detection, which predicts a set of class-agnostic bounding boxes along with a single score for each box, corresponding to its likelihood of containing any object of interest.

When we want computers to understand complex scenes. Image captioning is one such task. In these tasks, we have to train a model to predict the category of a given image is to first annotate each image in a training set with a label from a predefined set of categories

Chapter 7
CONCLUSION

According to the real time road conditions evaluation, the abnormal road condition can be detected and saved in traffic centre.

Thedrivesofthevehiclescanbeobtainnearbyroadinformationfromtheothervehiclesviaactive warning signals or goggle Maps to manage their driving behaviours for improving driving safety, comfort and efficiency.

Chapter 8
FUTURE WORK

In future if the road conditions can be improved and then there will be comfort for the people. The system can be made useful as a part of smart city campaign.

Also, applying machine learning techniques in classifying data can help the system to adapt to changing factors like nature of the road and vehicle type the users use. And the data collected can be sent to the government officials and help them in choosing the right person for the job to do a rightful job.

References

1) Rui Fan, Umar Ozgunalp, Brett Hosking, Ming Liu, Ioannis Pitas.” Pothole Detection Based on Disparity Transformation And Road Surface Modelling”.

2) Kwang Eun An, Sung Won Lee, Seung-Ki Ryu, Dongmahn Seo,” Detecting A Pothole Using Deep Convolutional Neural Network Models for An Adaptive Shock Observing in A Vehicle Driving”.

3) Manjusha Ghadge, Dheeraj Pandey, Dhananjay Kalbande,” Machine Learning Approach for Predicting Bumps
On Road”

4) Aniket Kulkarni, Nitish Mhalgi, Sagar Gurnani, Dr. Nupur Giri, “Pothole Detection System using Machine Learning on Android", V.E.S. Institute of Technology, Mumbai-74

5) Syuan-Yi Chen1, Annie Shih2 and Chun-Yi Hsiao, "Road condition Detection Device Using IR Sensing Module and python", International Conference on Consumer Electronics-Taiwan (ICCE-TW).

6) ArtisMednis, GirtsStrazdins,ReinholdsZviedris,GeorgijsKanonirs, Leo Selavo,"Real Time Pothole Detection using python”.

7) S Gayathri,MamathaRG,ManasaB,MenitaPatil,Sanjana BM,” Automatic Pothole Detection System”.

8) Adhithiyan.M,Karmel. A,Senthil Kumar. P,” Machine Learning based approach for pothole detection”.

9) Sudarshan S Rode, ShonilVijay, PrakharGoyal, Purushottam Kulkarni, Kavi Arya, “Pothole Detection And Warning System:Infrastructure Support and System Design”

10) ShambhuHegde, Harish V. Mekali, GollaVaraprasad," Pothole Detection and Inter Vehicular Communication."