

**18CSC206J- Software Engineering and Project Management**

**TOPIC-PEDESTRIAN AND VEHICLE DETECTOR**

**MADE BY-**

**MAYANK SINHA(RA1911003010386)**

**SHIVAM PANDEY(RA1911003010383)**

**ABHIJEET PADHI(RA1911003010393)**

**FACULTY INCHARGE- MS. B. JOTHI**

**INDEX**

* OVERVIEW
* WORKING
* WHAT IS HAAR CASCADE?
* REQUIREMENTS TEMPLATE
* BUSINESS TEMPLATE
* USER CASE STORY TEMPLATE
* STAKE HOLDER TEMPLATE
* PROJECT DIAGRAMS
  1. USE CASE DIAGRAM
  2. CLASS DIAGRAM
  3. ACTIVITY DIAGRAM
  4. STATE DIAGRAM
  5. SEQUENCE DIAGRAM
  6. COLLABARATION DIAGRAM
  7. ER DIAGRAM
  8. DFD DIAGRAM
  9. DEPLOYMENT DIAGRAM
* ARCHITECTURE DIAGRAM
* CODE
* TEST CASES
* SITE LINK

**PEDESTRIAN AND VEHICLE DETECTOR**

OVERVIEW:

AS THE NAME SAYS **PEDESTRIAN AND VEHICLE DETECTOR** DETECTS PEOPLE AND OTHER VEHICLES BEFOREHAND AND GIVES YOU THE INDICATIONS**.**

THIS SOFTWARE WORKS AS A DRIVER ASSISTANCE SYSTEM AIMING TO ALERT DRIVERS ABOUT DRIVING ENVIRONMENTS AND POSSIBLE COLLISION WITH OTHER VEHICLE OR PEOPLE. THIS SOFTWARE ACTUALY REPLACES A **MAGNETOMETER** WHICH WORKS BY USING PASSIVE SENSING TECHNOLOGY TO DETECT LARGE FERROUNS OBJECTS BY MEASURING MAGNETIC FIELD. BUT IT WAS NOT VERY ACCURATE SO THIS METHOD BECOMES A BIT DANGEROUS AT HIGH SPEED.

AS THE AUTOMOBILE INDUSTRY IS ENTERING INTO SELF DRIVING CARS, THIS SOFTWARE IS AN ESSENTIAL FEATURE OF THESE CARS. AFTER TESLA AND OTHER BIG BRANDS COMING INTO THE INDIAN MARKET WITH SUCH SIMILAR SOFTWARE IN THEIR VEHICLE, THE INDIAN CAR MAKING COMPANIES DEFINITELY NEED THIS SOFTWARE TO COMPETE WITH THEM.

THIS SOFTWARE WORKS EXTREMELY WELL WITH AUTOMATIC EMERGENCY BRAKING SYSTEM WHICH IS DESIGNED TO AUTOMATICALLY ENGAGE THE BRAKES TO REDUCE THE IMPACT OF AN UNAVOIDABLE FRONTAL COLLISION. **AS WE TURN ON THE VEHICLE IT COMES INTO POWER AND IT SCANS THE ENVIROMENT, MARKING THE PEDESTRIANS IN GREEN AND OTHER VECHILES IN RED, AND SHOWS IT ON THE SCREEN.**

THIS SOFTWARE IS BETTER THAN THE OLD TECHNOLOGY AS IT SHOW THE EXACT PICTURES OF WHAT IS IN THE SURROUNDING OF THE VEHILCE WITHOUT MISSING ANY.

**WORKING:**

**THIS SOFTWARE WORKS IN FOLLOWING STEPS:**

* FIRST IT CHECKS FOR THE CAMERA TO RUN UNTIL WE COMMAND IT TO STOP.
* IT CAPTURES THE LIVE COLOURED RUNNING IMAGES AND TURN IT INTO BLACK AND WHITE FOR EASY DETECTION.
* THIS SOFTWARE USES THE **PRETRAINED HAAR CASCADE (**it is a machine learning based approach where a cascade function is trained from a lot of positive and negative images.it is then used to detect objects in other images**)** MODEL FOR THE DETECTION.
* WE GET ALL THE COORDINATES, HIGHT AND WIDTH OF THE REGION OF THE VEHICLES AND THE PEDESTRIANS.
* THEN IT SHOW THE VIDEO WITH RECTANGLE BOXES AROUND THE DETECTED VEHICLES AND PEDESTRIANS SIGNNALING DANGER IF ANY.
* THE SOFTWARE KEEPS THE VIDEO FEED ON UNTIL WE COMMAND IT TO SHUT DOWN.
* IT IS MUCH BETTER THAN THE LAST TECH.BECAUSE IT DOES NOT DEPEND ON THE MAGNETIC FIELDS AND RATHER WORKS DIRECTLY ON THE IMAGE PROCESSED.

**WHAT IS HAAR CASCADE?**

Object Detection using Haar feature-based cascade classifiers is an effective method proposed by Paul Viola and Michael Jones in the 2001 paper, "Rapid Object Detection using a Boosted Cascade of Simple Features". It is a machine learning based approach in which a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.



Now all possible sizes and locations of each kernel are used to calculate plenty of features. For each feature calculation, we need to find the sum of the pixels under the white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of the sum of the pixels, how large may be the number of pixels, to an operation involving just four pixels.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by Ad boost.

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone cannot classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow. Wow. Isn't it a little inefficient and time consuming? Yes, it is. Authors have a good solution for that.

In an image, most of the image region is non-face region. So, it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Do not process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very a smaller number of features). If a window fails the first stage, discard it. We do not consider remaining features on it. If it passes, apply the second stage of features, and continue the process. The window which passes all stages is a face region. How is the plan !!!Authors' detector had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in first five stages. (Two features in the above image is obtained as the best two features from Ad boost). According to authors, on an average, 10 features out of 6000+ are evaluated per sub-window.



***OPENCV***

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general-purpose programming language started by Guido van Rossum that became extremely popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-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.

**REQUIREMENT TEMPLATE:**

# Executive Summary

VECHILE AND PEDESTRIAN DETECTOR.

# Project Scope

THIS SOFTWARE WORKS AS A DRIVER ASSISTANCE SYSTEMS AIMING TO ALERT DRIVERS ABOUT DRIVING ENVIRONMENTS AND POSSIBLE COLLISION WITH OTHER VEHICLE OR PEOPLE.

|  |  |  |
| --- | --- | --- |
| **S. No** | **Activities in Scope** | **Activities Out of Scope** |
| 1 | Auto detection of vehicles and  pedestrians | Poor recognition software |
| 2 | Shows danger around if any | Does not work at night |
| 3 | Can be used in automatic cars. | Might not work properly on high speed |
| 4 | Can be used to keep check on traffic | Excessive skew angle |

## In Scope

## THIS SOFTWARE WORKS EXTREMELY WELL WITH AUTOMATIC EMERGENCY BRAKING SYSTEM WHICH IS DESIGNED TO AUTOMATICALLY ENGAGE THE BRAKES TO REDUCE THE IMPACT OF AN UNAVOIDABLE FRONTAL COLLISION. AS WE TURN ON THE VEHICLE IT COMES INTO POWER AND IT SCANS THE ENVIROMENT, MARKING THE PEDESTRIANS IN GREEN AND OTHER VECHILES IN RED, AND SHOWS IT ON THE SCREEN

## Out of Scope

## THE SOFTWARE DOES NOT PROVIDE PROPER ASSISTANCE FOR VERY FAST-MOVING OBJECTS AND IT ALSO DOES NOT WORK PROPERLY IN PLACES HAVING VERY LOW LIGHT.

## THE SOFTWARE MIGHT NOT WORK PROPERLY IN CROWDED PLACES AS THIS IS JUST THE BASIC MODEL OF ITS KIND.

# Epics [Major Functions]

* FIRST IT CHECKS FOR THE CAMERA TO RUN UNTIL WE COMMAND IT TO STOP.
* IT CAPTURES THE LIVE COLOURED RUNNING IMAGES AND TURN IT INTO BLACK AND WHITE FOR EASY DETECTION.
* THIS SOFTWARE USES THE **PRETRAINED HAAR CASCADE (**it is a machine learning based approach where a cascade function is trained from a lot of positive and negative images.it is then used to detect objects in other images**)** MODEL FOR THE DETECTION.
* WE GET ALL THE COORDINATES, HIGHT AND WIDTH OF THE REGION OF THE VEHICLES AND THE PEDESTRIANS.
* THEN IT SHOW THE VIDEO WITH RECTANGLE BOXES AROUND THE DETECTED VEHICLES AND PEDESTRIANS SIGNNALING DANGER IF ANY.
* THE SOFTWARE KEEPS THE VIDEO FEED ON UNTIL WE COMMAND IT TO SHUT DOWN.

# Requirements

 Functional Requirements

THIS SOFTWARE WORKS MORE OR LESS ON THE PRETRAINED HAARCASCADE MODEL FOR THE DETECTION. THIS IS BASICALLY A MACHINE LEARNING APPROACH WHERE A CASCADE FUNCTION IS TRAINED FROM A LOT OF POSITIVE AND NEGATIVE IMAGES. IT IS THENUSED TO DETECT OBJECTS IN THOSE IMAGES.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement (#)** | **Requirement Specification** | **Department** | **Name of Business User** | **Status** |
| E1FR1 | Real time video capture. |  |  | low |
| E1FR2 | Frame extraction. |  |  | high |
| E1FR3 | Vehicle and pedestrian detection |  |  | high |
| E1FR4 | Image acquisition |  |  | high |
| E1FR5 | Vehicle extraction |  |  | low |
| E1FR6 | Pedestrian extraction |  |  | low |
| E1FR7 | Danger recognition |  |  | low |

## Non-Functional Requirements

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Requirement (#)** |  | **Requirement Specification** | **Department** | **Name of Business User** | **Status** |
| NFR1 |  | Security |  |  | High |
| NFR2 |  | Portability |  |  | Moderate |
|  |  | Durability |  |  | Low |
|  |  | Accuracy |  |  | Low |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| E1NFR2 |  | Execution Speed |  |  | Medium |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| E1NFR1 |  | High resolution image capture |  |  | High |
|  |  | Moderate speed. |  |  | Low |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Infrastructure Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement (#)** | **Requirement Specification** | **Department** | **Name of Business User / Project Team Member** | **Status** |
| IR1 | Database for positive and negative images |  |  | Ready |
| IR2 | Amazon web service |  |  | Ready |
| IR3 | Haar cascade classifier trainer |  |  | Ready |
| IR4 | Image capturing system |  |  | Ready |
|  |  |  |  |  |

Requirement definition in Agile

* **Who are we building it for, who the user is? —** AS A CUSTOMER THIS SOFTWARE IS VERY MUCH REQUIRED IN TODAYS VECHILES AS SELF DRIVING CARS IS THE NEW GAME.APART FROM THAT EVEN IN MANAUL VECHILES WE HAVE SEEN A LOT OF ACCIDENTS HAPPENING BECAUSE OF SOME MINOR ISSUES.HENCE THIS WILL ALSO COVER UP FOR THOSE ACCIDENTS.
* **What are we building, what is the intention? —** I WANT THIS SOFTWARE TO BE USED BY THE AUTOMOBILE INDUSTRY TO USE IT FOR THE BETTERMENT OF THE PEOPLE.THE INDIAN AUTOMOBILE INDUSTRY DOES HAVE SUCH FEATURE BUT THEY PRODUCE MINOR ISSUES AND HENCE WE NEED TO RECTIFY THEM. THESE CAN ALSO BE USED BY THE GOVERNMENT IN THE ARMY AND BY THE TRAFFIC POLICE.
* **Why are we building it, what value it brings for the user? —** SO THAT THI SSOFTWARE HELP AND MAKE EASE FOR THE DRIVERS. WITH A SIMPLE VIEW AT THE SCREEN THEY CAN MAKE OUT WHOS NEAR THEIR VECHILE OR CAN BE.

**BUSINESS CASE TEMPLATE:**

|  |  |
| --- | --- |
| **DATE:** | **6.2.2021** |
| **SUBMITTED BY:** | **ABHIJEET PADHI(RA1911003010393), SHIVAM PANDEY(RA1911003010383), MAYANK SINHA(RA1911003010386)** |
| **TITLE:** | **PEDESTRIAN AND VEHICLE DETECTOR** |

**THE PROJECT**

● MOST IMPORTANT FEATURE OF SELF-DRIVEN CARS.

● DETECTS HUMANS AND OTHER VEHICLE.

● REDUCES WORKLOAD ON HUMANS.

● VERY SAFE COMPARED TO OTHER TECHNOLOGIES.

**THE HISTORY**

● MAGNOMETER WASN’T SUCCESFULL.

● PEOPLE FALL ASLEEP WHILE DRIVING.

● LOTS OF ACCIDENTS DUE TO POOR VISION.

**LIMITATIONS**

● IT REQUIRES A GREAT CAMERA.

● CAMERA ANGLE SHOULD BE CORRECT.

● OBJECT ON THE FRONT SHOULD BE AT MODERATE SPEED.

**APPROACH**

● CODING SKILLS.

● SOFTWARE MANAGEMENT AND DEVELOPMENT SKILLS.

● BASIC KNOWLEDGE ABOUT FACE DETECTOR.

**BENEFITS**

● SHOWS DANGER BEFOREHAND.

● MORE RELIABLE.

● SELF DRIVING FACILITY.

● EASY AND ACCURATE CALCULATIONS.

**USER STORY TEMPLATE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **USER STORY ID** | **AS A (TYPE OF USER)** | **I WANT TO (PERFORM SOME TASK)** | **SO THAT I CAN (ACHIEVE SOME GOAL)** |
| **1** | **PROJECT MANAGER** | **VIEW A STATUS REPORT FROM EACH TEAM MEMBER.** | **ENSURE THE PROJECT STAYS ON TRACK.** |
| **2** | **EMPLOYEE** | **BE REMINDED OF UPCOMING DEADLINE.** | **COMPLETE MY TASK ON TIME.** |
| **3** | **CUSTOMER** | **PUSRCHASE THE PRODUCT.** | **AND GET THE BENFITS OF THE FEATURES.** |
| **4** | **CUSTOMER** | **USED BY THE TRAFFIC POLICE.** | **USE THIS SYSTEM FOR MAKING THE WORK EASIER.** |
| **5** | **CUSTOMER** | **GIVE FEEDBACK ABOUT THE SOFTWARE.** | **IMPROVE OR SOFTWARE ACCORDING TO THE FEEDBACK RECEIVED.** |
| **6** | **DEVLOPER** | **USE THE FEEDBACK AND UPDATE THE SOFTWARE.** | **IMPROVE THE SOFTWARE TO MEET THE NEEDS OF THE CUSTOMERS.** |

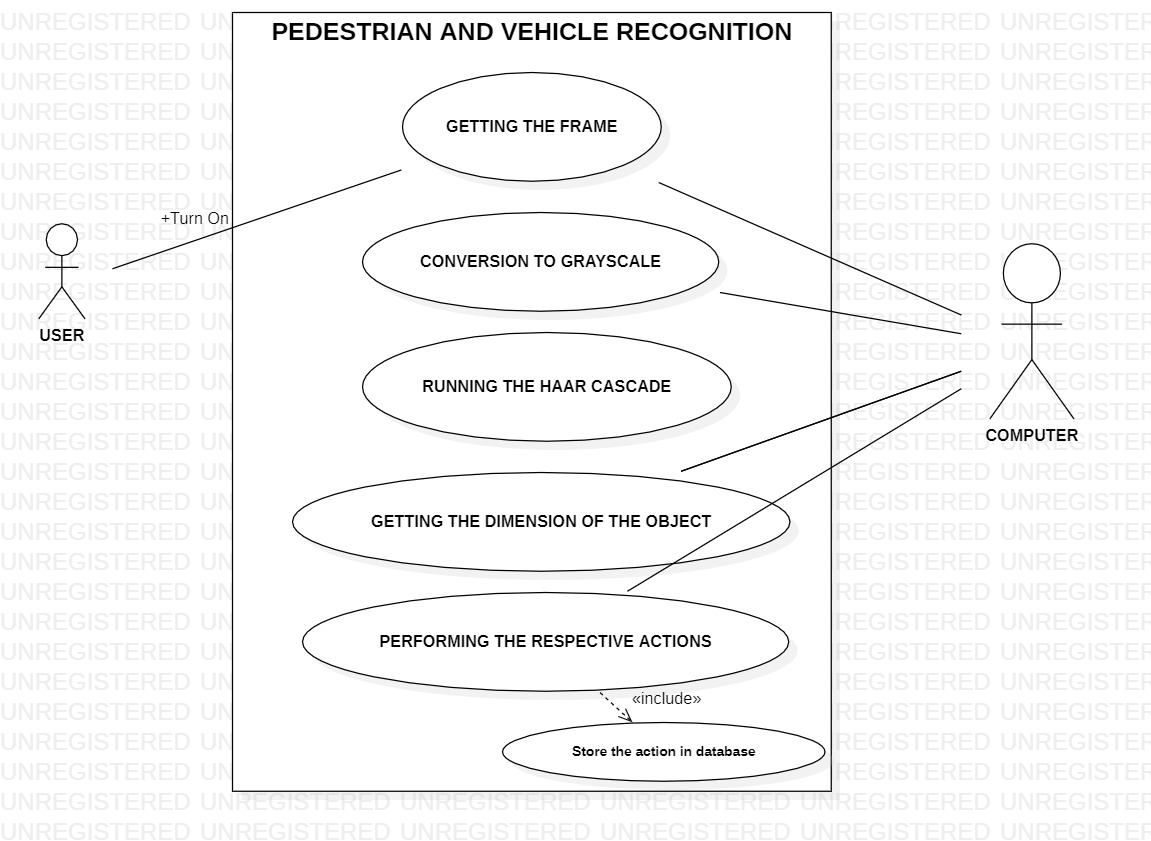
**STAKE HOLDER TEMPLATE:**

|  |  |
| --- | --- |
| PROJECT NAME: | PEDESTRIAN AND VEHICLE DETECTOR. |
| PREPARED BY: | MAYANK SINHA (RA1911003010386), SHIVAM PANDEY(RA1911003010383), ABHIJEET PADHI(RA1911003010393) |
| DATE: | 6TH FEBRUARY-2021. |

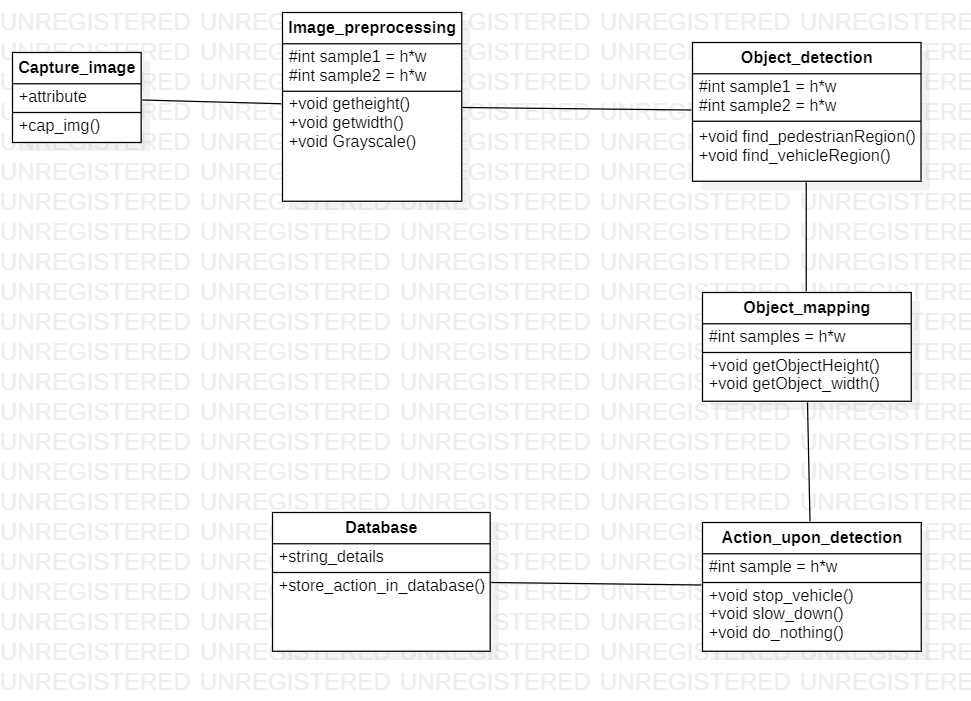
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROJECT STAKEHOLDER NAME | SPECIFIC INFORMATION NEEDS | PROJECT INTERESTS | IMPACT ON PROJECT | ROLE |
| AMIT | PROVIDES INSTRUCTIONS AND GAUIDANCE | SALARY, SHARE PORTIONS, DECISION MAKING | POSITIVE SUPPOTER | DIRECTORIAL  /MANAGEMENT |
| ALINJAR | INFORMATION ON SOFTWARE POLICIES | DIRECTLY ENGAGED WITH THE SOFTWARE | POSITIVE SUPPORTER | EMPLOYEE |
| LAVESH | INFORMATION ON POTENTIAL GROWTH AND NEW COMPETITION IN MARKET | INFULENCE THE OVERALL SALES OF THE SOFTWARE | SUPPOTER/  INFLUENCER | CUSTOMERS |
| TANMAY | NIL | ADVICE FOR THE BETTER FUNTION OF THE SOFTWARE | SUPPOTER | ADVISORS |
| ARADHAYA | REGULAR COMMUNICATIONS  SOFTWARE POLICIES AND PRACTICES | LEGISLATION AND REGULATIONS OF THE SOFTWARE. | INFLUENCER | GOVERNMENT |
| SIDDHART | NIL | PUBLIC IMAGE/ REPUTATION | INFLUENCER | RATING AGENCIES |

**PROJECT DIAGRAMS:**

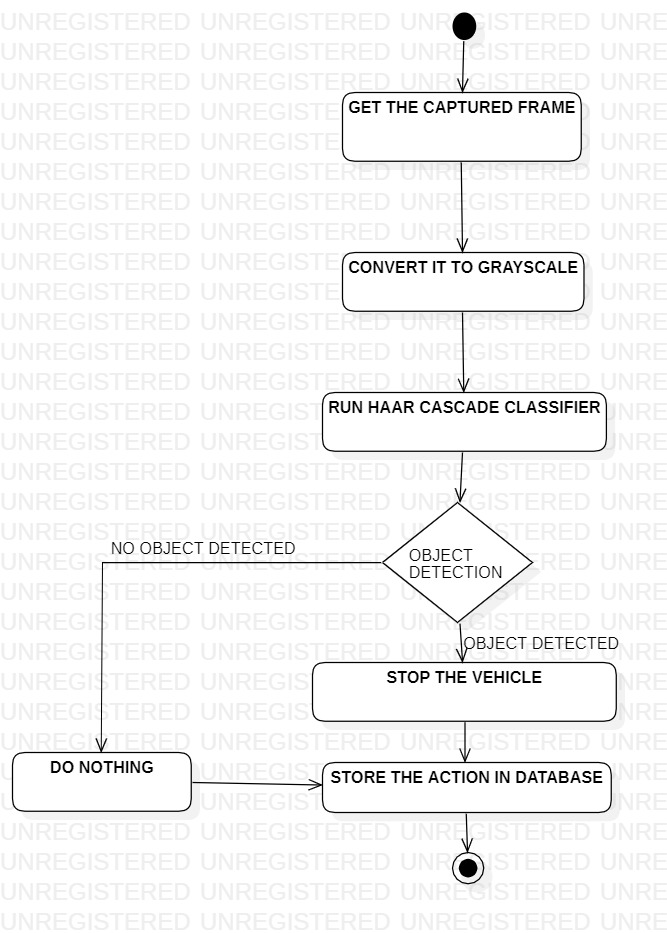
USE CASE DIAGRAM



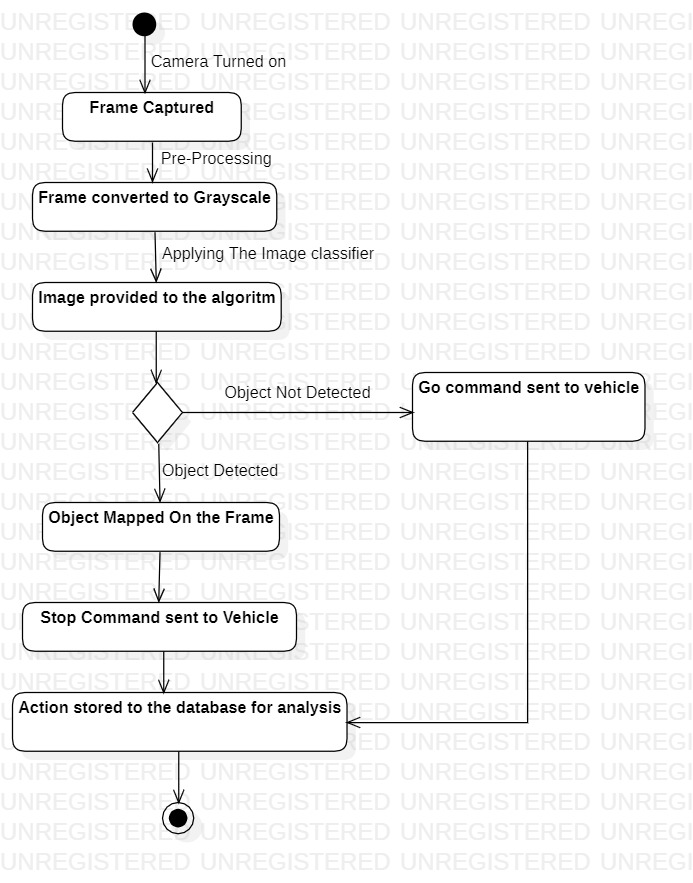
CLASS DIAGRAM



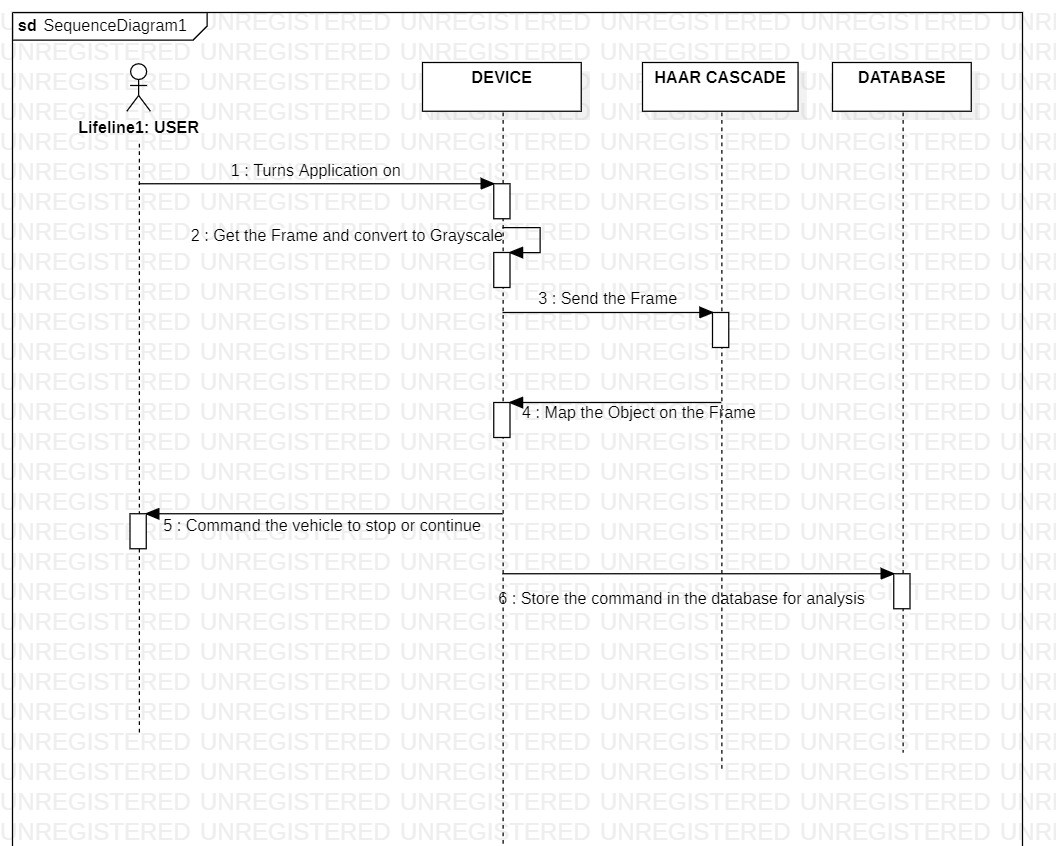
ACTIVITY DIAGRAM



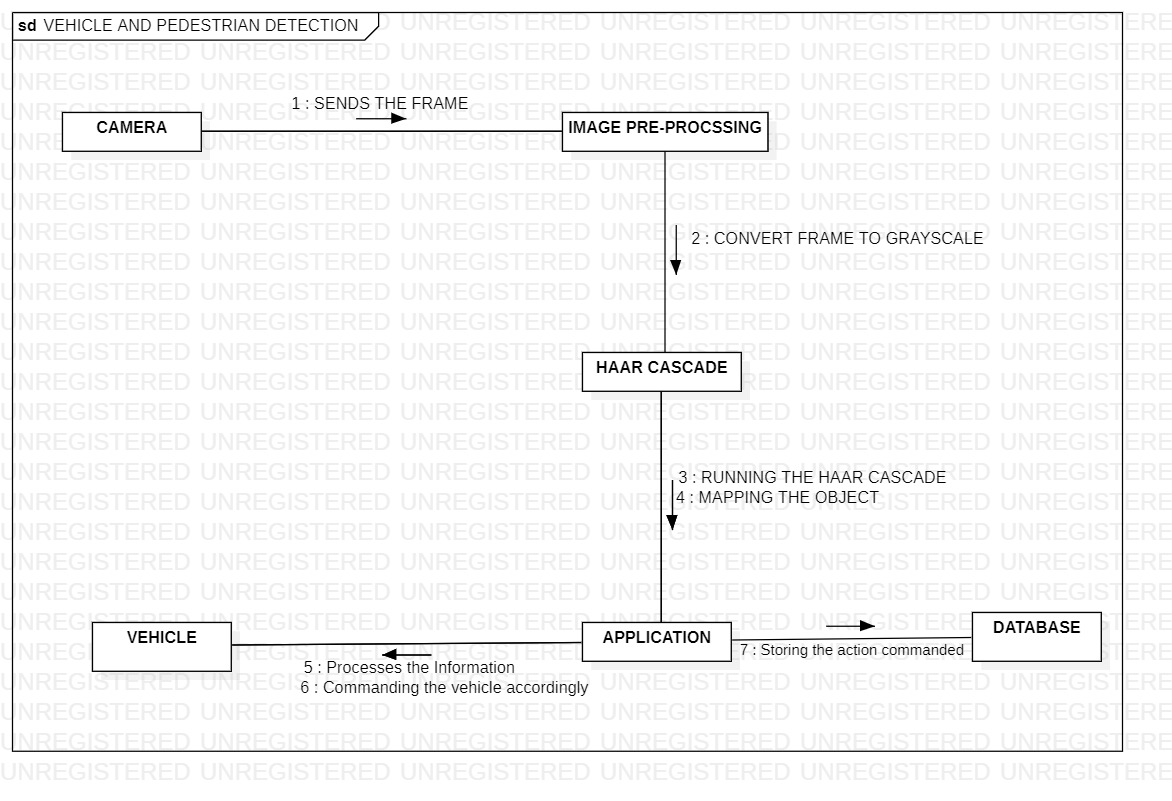
STATE DAIGRAM



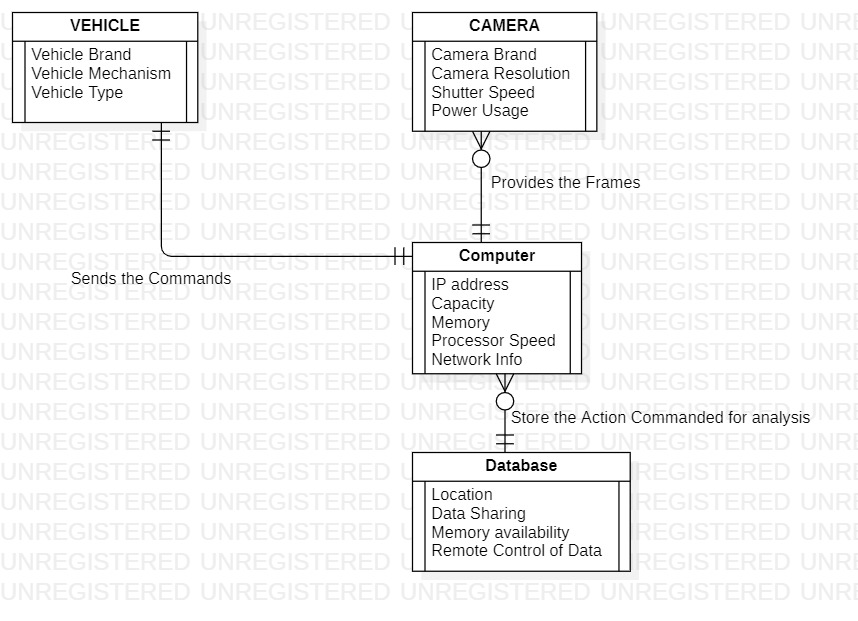
SEQUENCE DIAGRAM



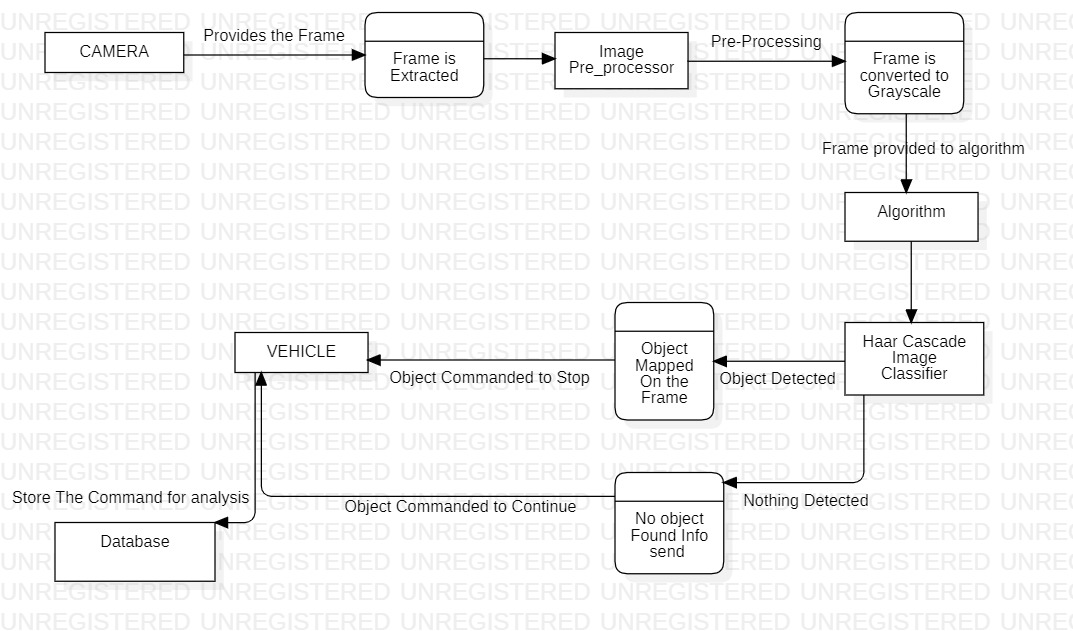
COLLABARATION DAIGRAM



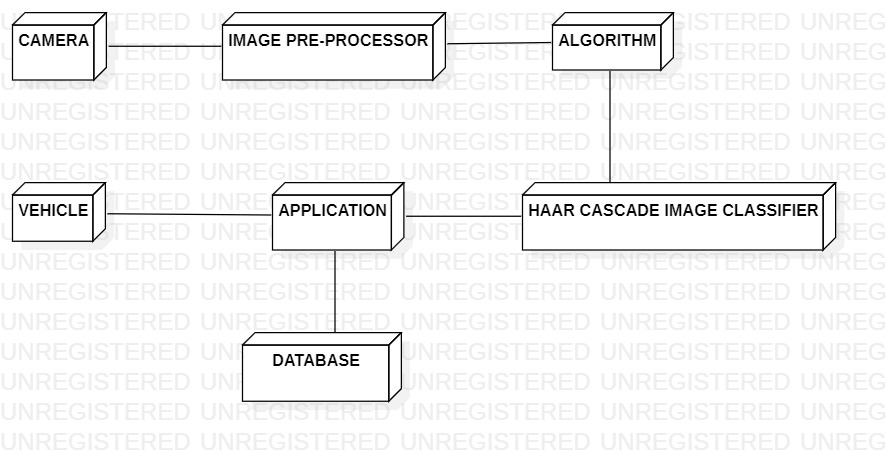
ER DIAGRAM



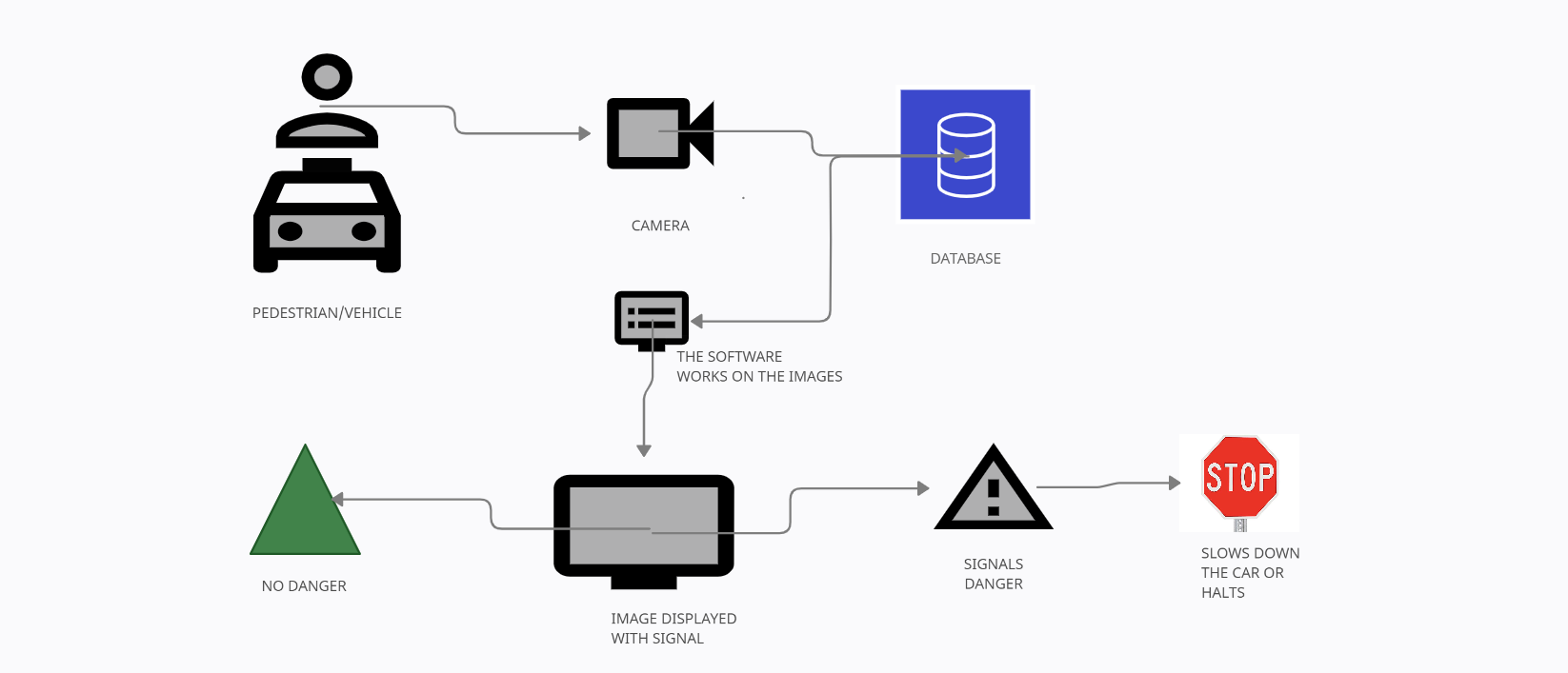
DFD DAIGRAM



DEPLOYMENT DIAGRAM



ARCHITECTURE DIAGRAM



CODE:

#WE IMPORT THE OPEN-CV LIBRARY

import cv2

from random import randrange

#PROVIDING PRE-TRAINED HAAR CASCADE IMAGE CLASSIFIER

car\_tracker=cv2.CascadeClassifier('car\_detector.xml')

#

pedestrian\_tracker=cv2.CascadeClassifier('haarcascade\_fullbody.xml')

#FOR CAPTURING THE VIDEO ON THE SYSTEM

cam=cv2.VideoCapture('Pedestrians Compilation.mp4')

#WE USE WHILE TRUE SO THAT THE CAM RUNS UNTIL WE COMMAND IT TO STOP

while True:

#WE GET THE CAM READ-> THIS HAS TWO PARTS THE ACTUAL FRAME STORED IN FRAME AND A BOOL VALUE WHICH IS ALWAYS TRUE IN SUCCESSFUL\_READ

successful\_read, frame=cam.read()

#WE CONVERT THE COLORED IMAGES TO BLACK AND WHITE TO MAKE THE DETECTION EASIER FOR HAARCASCADE

grayscaled\_image=cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

#WE GET THE COORDINATES FOR THE PLACE WHERE THE CAR&PEDESTRIANS ARE PRESENT

car\_coordinates=car\_tracker.detectMultiScale(grayscaled\_image,1.4,1)

pedestrian\_coordinates=pedestrian\_tracker.detectMultiScale(grayscaled\_image,1.2,3)

#WE GET X-COORDINATE, Y-COORDINATE OF THE TOP LEFT CORNER AND THE HEIGHT AND WIDTH OF THE REGION OF THE CARS & PEDESTRIANS

for(x,y,h,w) in car\_coordinates:

#WE PLOT A RECTANGLE AROUND THE FACE

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)

for(x,y,h,w) in pedestrian\_coordinates:

#WE PLOT A RECTANGLE AROUND THE FACE

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

#CODE IF U WANT TO USE AND ELLIPSE INSTEAD OF A RECTANGLE

#center = (x + w//2, y + h//2)

#frame = cv2.ellipse(frame, center, (w//2, h//2), 0, 0, 360, (255, 0, 255), 4)

#THIS SHOWS THE VIDEO WITH THE RECTANGLE SHAPE OVER THE DETECTED CAR&PEDESTRIANS

cv2.imshow('CAR&PEDESTRIAN DETECTOR',frame)

#THIS COMMAND KEEPS THE VIDEO FEED ON UNTIL WE COMMAND IT TO CLOSE AND 1 SIGNIFY AFTER 1ms THE FRAMES WILL CHANGE

if cv2.waitKey(1)==13:

break

#WE USE Q KEY BOTH UPPER AND LOWERCASE TO BREAK OUT OF THE LOOP AND THEN RELEASE THE FEED

#if key==81 or key==113:

cam.release()

cv2.destroyAllWindows()

**TEST CASES:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| ID | TEST CASES | TEST DATA | EXPECTED RESULT | ACTUAL RESULT | STATUS |  |  |
| 1 | Test if the vehicle is off. | Trial of the software. | Vehicle must be turned on. | System working. | Test successful | |  |
| 2 | Test if the vehicle is on. | Trial of the software | Vehicle must be turned on. | System working. | Test successful | |  |
| 3 | Check of the software detects in crowded places. | Trial of the software in crowded places | Detects the object | System failed | test failed | |  |
| 4 | Check if the software detects fast moving objects. | Trial of the software. | Should detect the object. | System pass. | Test successful | |  |
| 5 | Check if the software work properly in fog or rain | Trial if the software | Should detect the object | System failed. | Test failed | |  |
| 6 | Check if the software detects in broad sunlight | trial of the software in sunlight | detects the pedestrians clearly | detects the pedestrians clearly | test successful | |  |
| 7 | Check if the software detects in dim lighting | trial of the software in dim lighting | detects the pedestrians clearly | detects the pedestrians with some difficulty | test successful | |  |
| 8 | Check if the software detects pedestrians while the car is moving at slow speed (10-40 km/h) | trial of the software in a vehicle moving at the mentioned speed | detects the pedestrians clearly | detects the pedestrians clearly | test successful | |  |
| 9 | Check if the software detects pedestrians while the car is moving at medium speed (50-90 km/h) | trial of the software in a vehicle moving at the mentioned speed | detects the pedestrians clearly | detects the pedestrians clearly | test successful | |  |
| 10 | Check if the software detects pedestrians while the car is moving at high speed (100 km/h and above) | trial of the software in a vehicle moving at the mentioned speed | detects the pedestrians clearly | detects the pedestrians with some difficulty | test successful | |  |
| 11 | Check if the software stops the vehicle when pedestrian is detected | if the software automatically stops the vehicle | vehicle is stopped | vehicle is stopped | test successful | |  |
| 12 | Check if the payment portal is working | payment is received or declined | payment received | payment received | test successful | |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**REFRENCES:**

* **Shari Lawrence Peeger and Joanne M. Atlee Software Engineering**
* **Agile Software Development Principle, Patterns, and Practices by Robert C. Martin**
* **Rapid Development by Steve C. McConnel**
* **Applied Software Project Management by Andrew Stellman**
* **Software Project management in practice by Pankaj Jalote**

**SOFTWARE WEBPAGE LINK:**

https://shivampandeyvns.wixsite.com/vortex