

## CHAPTER-1

# INTRODUCTION

## 1.1 INTRODUCTION

Object detection is one of the most important research directions for computer vision. Object detection is a technique that detects the objects of a particular class in digital images and videos. One of its real-time applications is self-driving cars or even an application for visually impaired that detects and notify the disabled person that some object is in front of them. Object detection algorithms can be divided into the traditional methods which used the technique of sliding window where the window of specific size moves through the entire image and the deep learning methods that includes YOLO algorithm. In this, our aim is to detect multiple objects from an image. The most common object to detect in this application are the bus, bottle, and mobile. For locating the objects in the image, we use concepts of object localization to locate more than one object in real-time systems. There are various techniques for object detection, they can be divided into two categories, first one is the algorithms based on Classifications. CNN and RNN come under this category. In this category, we have to select the interested regions from the image and then have to classify them using Convolutional Neural Network. This method is much slow as we have to run a prediction for every selected region. The next category is the algorithms based on Regressions. The research focuses on YOLOv3, even though there are other various object detection techniques (You Only Look Once). A real-time object detection system called You Only Look Once, Version 3 (YOLOv3) can identify particular objects in videos, live streams, and still images. YOLO uses features that a deep convolutional neural network has learned to identify items. YOLO is implemented using the Keras or Open CV deep learning packages. As is common for object detectors, the classifier uses the knowledge gained by the convolutional layers to predict the detection.

The size of the prediction map and the size of the feature map are same since YOLO's prediction makes use of 11 convolutions. An object recognition system that operates in real-time is the YOLO. The advantage of YOLO is that it is substantially faster than other networks while still keeping accuracy. At test time, it allows the model to see at the complete image, enabling it to make predictions based on the image's total context.

Contextual Common Objects (COCO) in YOLO (COCO) The COCO database aims to support future research in the fields of object detection, instance segmentation, picture captioning, and localisation of human keypoints. A large-scale dataset for object detection, segmentation, and labelling is called COCO.

## **1.2 PROBLEM STATEMENT**

The project “Object Detection using YOLOV3” detects objects efficiently based on YOLO algorithm and apply the algorithm on image data and video data to detect objects.

## **1.3 OBJECTIVES**

1. The main purpose of object detection is to identify and locate one or more effective targets from still image or video data.
2. It comprehensively includes a variety of important techniques, such as image processing, pattern recognition, artificial intelligence and machine learning.

## **1.4 LIMITATIONS**

1. Speed for real-time detection
2. poor image quality.

## CHAPTER-2

### SYSTEM ANALYSIS

#### 2.1 EXISTING SYSTEM

You Only Look Once: Unified, Real-Time object detection. The project is on detecting objects using a regression algorithm. To get higher accuracy and good predictions they have proposed YOLO algorithm.

1. Understanding of Object Detection Based on YOLO. In this project, generally explained about the object detection families like CNN, R-CNN and compared their efficiency and introduced YOLO algorithm to increase the efficiency
2. Learning to Localize Objects with Structured Output Regression. This project is about Object Localization. In this I have used the Bounding box method for localization of the objects to overcome the drawbacks of the sliding window method.

#### 2.2 PROPOSED SYSTEM

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its accuracy and speed. It has been used in various applications to detect traffic signals, people and animals. This project introduces readers to the YOLO algorithm for object detection and explains how it works. It also shows some of its real-life applications.

#### 2.3 FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main aim of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. The feasibility study is a management-oriented activity. The aim of a feasibility study is to find out if an information system project can be done and to suggest possible alternative solutions.

There are aspects in the feasibility study portion of the preliminary investigation:

1. Technical Feasibility
2. Operational Feasibility
3. Economic Feasibility

### **2.3.1 TECHNICAL FEASIBILITY**

It refers to whether the software that is available in the market fully supports the present application. It studies the pros and cons of using a particular software for the development and its feasibility. It also studies the additional training needed to be given to the people to make the application work. The technical requirement are then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirement. The analyst must find out whether current technical resources can be updated or added to in a manner that fulfills the request under consideration.

### **2.3.2 OPERATIONAL FEASIBILITY**

It refers to the feasibility of the product to be operational. Some products may work good at design and implementation but may fail in the real time environment. It includes the study of additional human resource required and their Technical expertise. It dependent on human resources available for the project and involves projecting whether the system will be used if it is developed and implemented. It measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the analysis phase of system development. It reviews the wish of the organization to support the proposed system. Intending to determine this feasibility, it is important to understand the management commitment to the proposed project.

### **2.3.3 Economic Feasibility:**

It refers to the benefits or outcomes we are deriving from the product as compared to the total cost we are spending for developing the product. If the more or less same as the older system, then it is not feasible to develop the product. Economic analysis could also be referred to as price/benefits analysis. It is frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with price. If benefits outweigh price, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the price versus benefits before taking an action.

## **CHAPTER-3:**

# **SYSTEM ENVIRONMENT**

### **3.1 HARDWARE REQUIREMENT:**

1. Processor : Pentium Core I5
2. RAM : 4GB
3. Hard Disk : 500GB
4. Monitor : 15 inch Color Monitor
5. Mouse : Optical Mouse

### **3.2 SOFTWARE REQUIREMENT:**

1. Operating System : Windows 10
2. Front End : PYTHON
3. Back End : SQL SERVER 2008

## CHAPTER-4:

### MODULE DESCRIPTION:

#### 4.1 READ VIDEO:

A video is really an arrangement of pictures which gives the presence of movement. Recordings can be viewed as an assortment of pictures (outlines).The capacity cv2.VideoCapture and make a class case. As a contention we can indicate the info video document name. Then again, to get to a video transfer, we will put the camera boundaries all things being equal.

#### **YOLO module perform four functionalities-**

1. **OBJECT REGISTRATION:** This module allows the object to be registered with the model.
2. **OBJECT DETECTION:** This module allows the product to detect the object based on the training received.
3. **FRAME ANALYSIS:** This module allows the model to analyze the capture frame.
4. **GENERATE PREDICTION:** This module allows the module to generate a text prediction of what the object is System with YOLO will generate the output file by using the COCO dataset which is defined inside.

## **CHAPTER-5**

# **SYSTEM DESIGN**

### **INTRODUCTION:**

The purpose of the design phase is to plan a solution of the problem specified by the requirements documents. This phase is the first step in moving from the problem state to the solution state. In other words, starting with what is needed; design takes us towards how to specify the needs.

### **UML DIAGRAMS:**

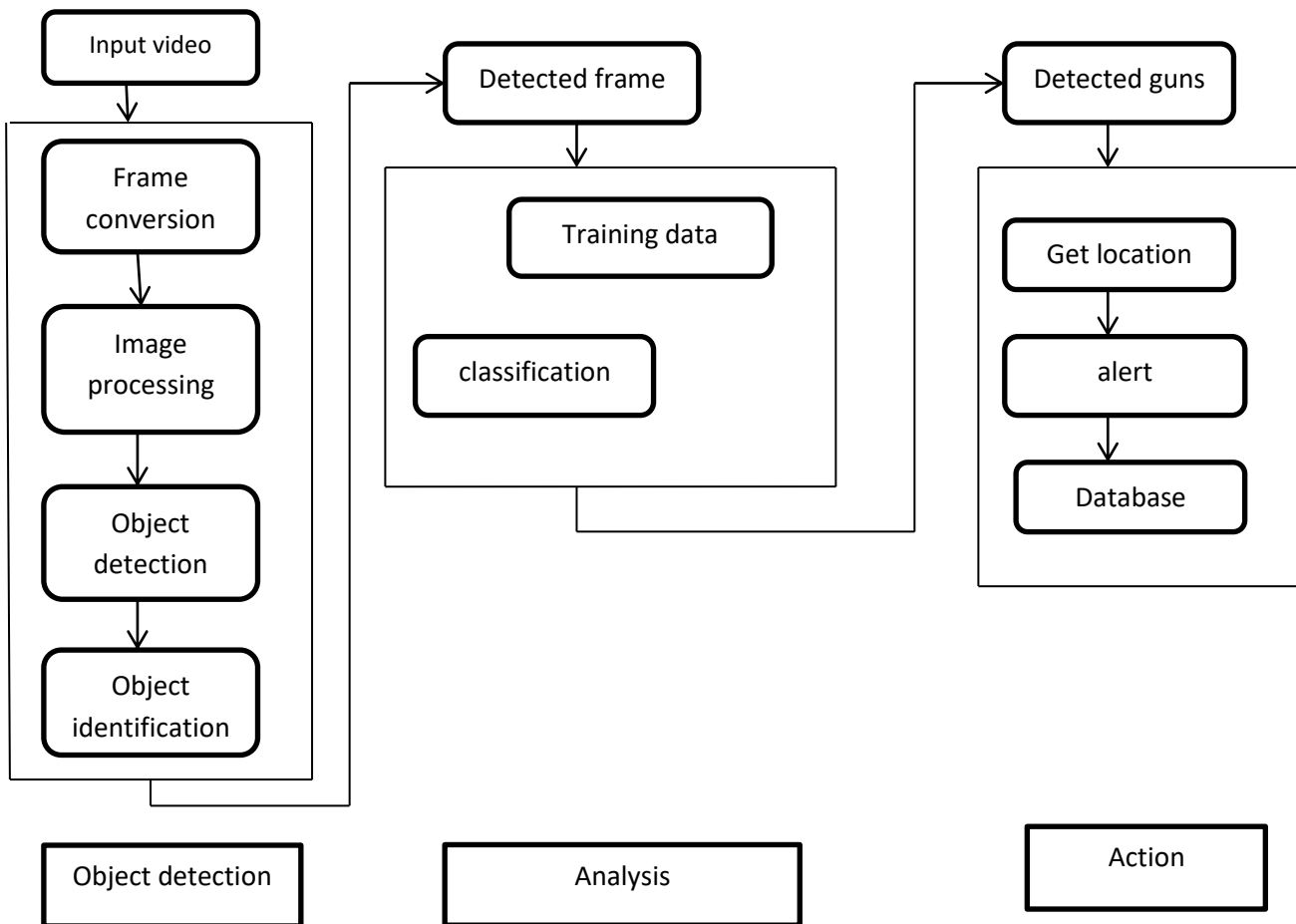
The unified modelling language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

### **DATA FLOW DIAGRAM (DFD):**

A Data Flow Diagram (DFD) is a diagrammatic representation of the information flows within a system, showing: how information enters and leaves the system, Where information is placed. Data flow diagrams can be used to provide a clear representation of any business function.

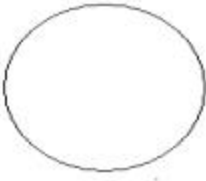



### **ADVANTAGES OF DFD:**

These are simple notations, which are easily understood by users and those involved in the system. Users can be involved the study of DFD for more accuracy. User can examine charts and starts avoiding mistakes early may prevent system's failure. A DFD is shown as a "bubble Chart" has the purpose of clarifying system requirement and identifying major transformation that will become programs in system design. So it is the starting point to design to the lowest level of detail. A Data flow diagram consists of a series of bubbles joined by data flows in the system.

**5.1 DATA FLOW DIAGRAM (DFD):****Fig 5.1.1 Data Flow Diagram**

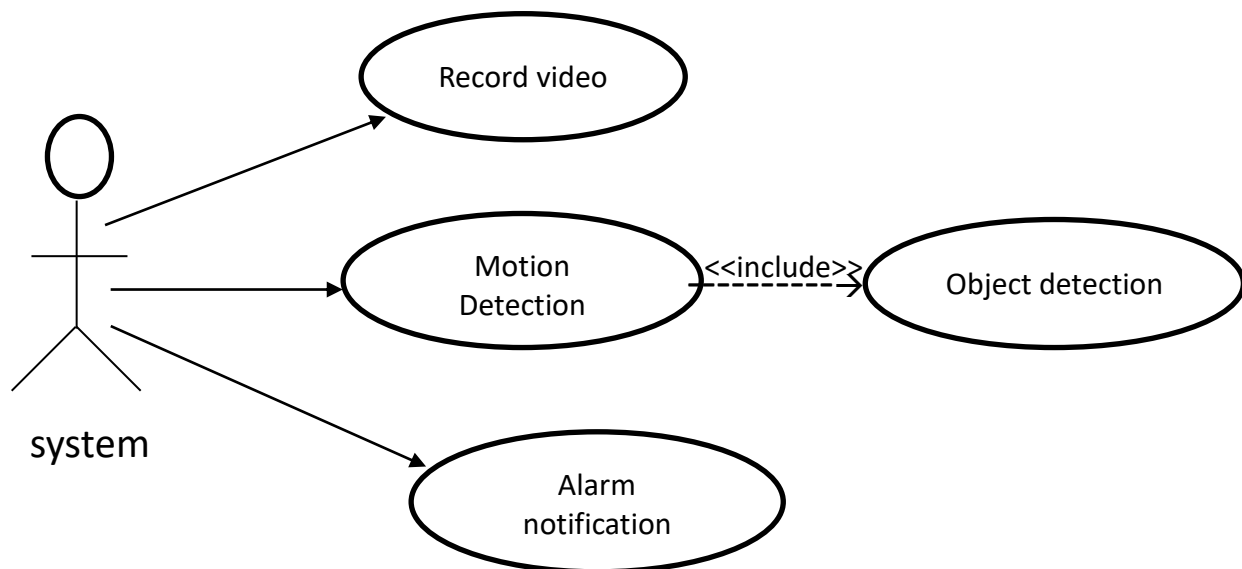


**THE NOTATIONS:**

Name	Symbol	Meaning
process		Transforms of incoming data flows(s) to outgoing data flows(s).
Data Store		A repository of data that is to be store for use by one or more processes.
Data Flow		Movement of the data in the system.
External Entity		Sources and Destination outside the specified system boundary.

## 5.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

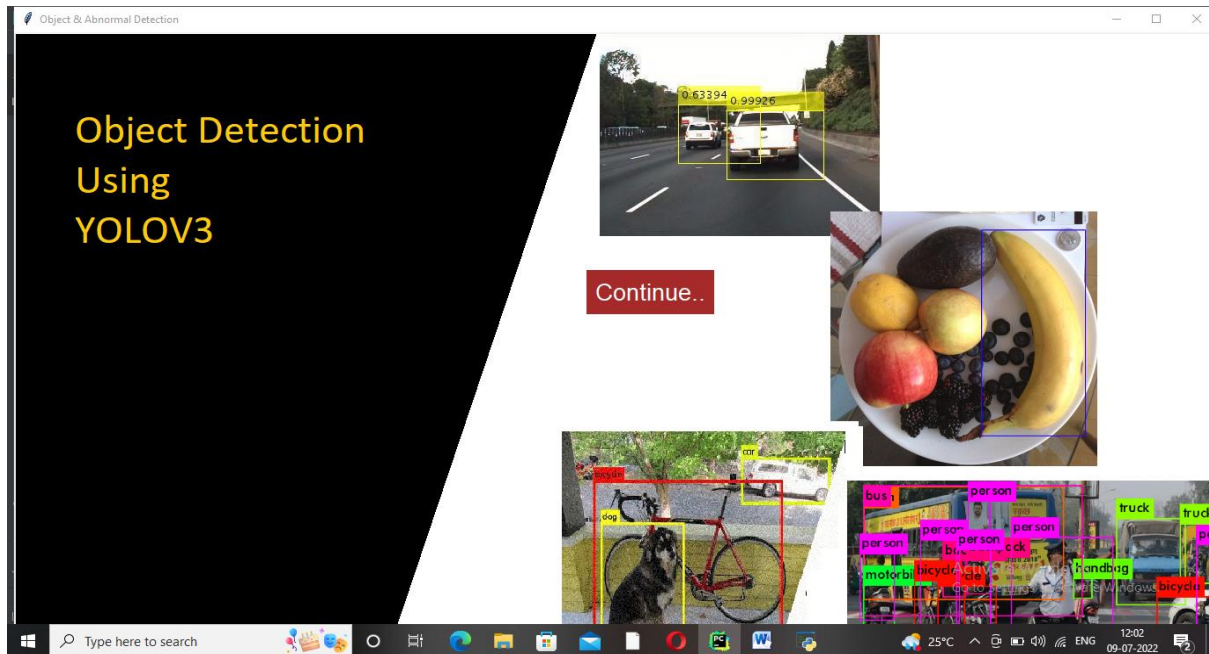


**Fig: 5.2.2 Use Case Diagram**

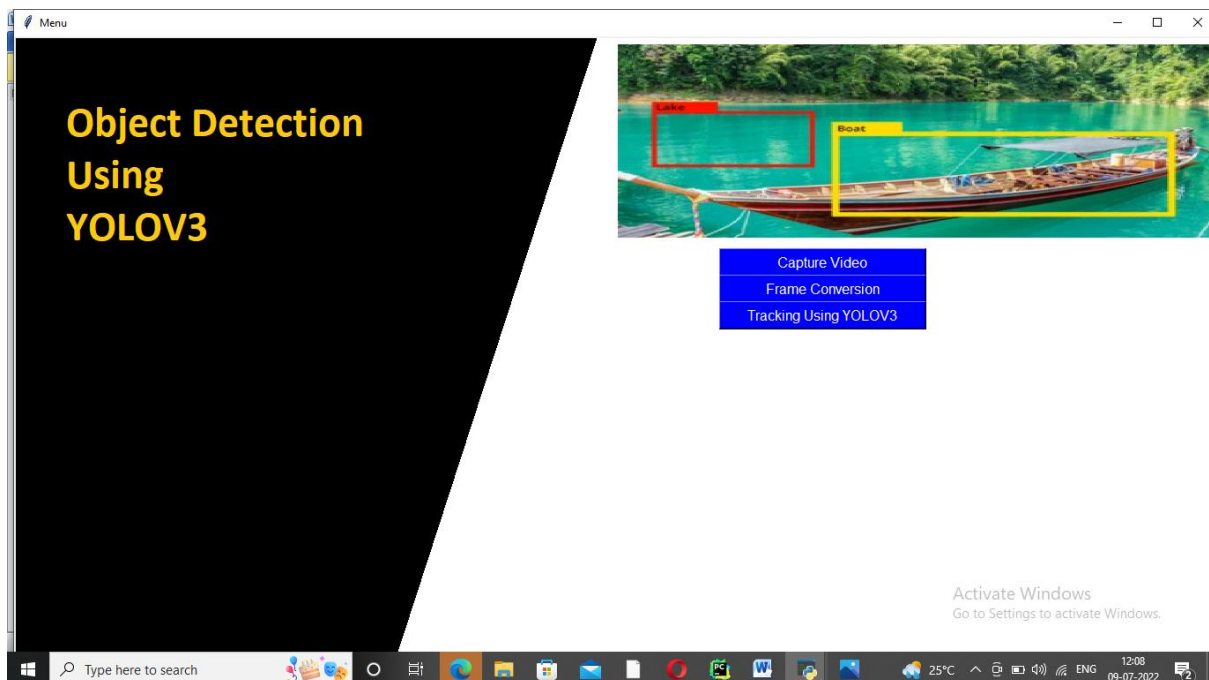
## CHAPTER-6

## FORM DESIGN

## Main



## Menu



## **CHAPTER-7**

### **SYSTEM IMPLEMENTATION:**

#### **7.1 QUALITY REQUIREMENTS:**

##### **LOGICAL DESIGN:**

A system's inputs, outputs, and data flows are all represented abstractly in the system's logical architecture. A model of the actual system that is overly abstract (and occasionally graphical) is frequently used for this. Included in the context of system design. Entity Relationship Diagrams, or ER Diagrams, are a part of logical design.

##### **PHYSICAL DESIGN:**

The system's real input and output operations are related to the physical design. This is specified in terms of how information is entered into a system, validated or authenticated, processed, and outputted. The following system requirements are chosen during physical design. The following five needs must be met:

1. Input requirements
2. Output requirements
3. Storage requirements
4. Processing requirements
5. System control and backup or recovery

To put it another way, there are often three sub-tasks that make up the physical part of systems design:

1. Designing user interfaces
2. Data Planning
3. Process Planning

##### **PROCESS DESIGN:**

The design of the user interface considers both how users input information into the system and how the system displays information to them. The representation and storage of the data within the system are both aspects of data design. Last but not least, process design is concerned with how data flows into, through, and out of the system as well as how and where it is validated, secured, and/or changed. The three sub-tasks are documented at the conclusion of the systems design phase and made available for use in the following phase.

## **PHYSICAL DESIGN:**

In this sense, the term "physical design" does not refer to the actual physical layout of an information system. Using a personal computer as an example, input is made possible by a keyboard, processing is done by the CPU, and output is made possible by a display, printer, etc. The physical arrangement of the hardware, such as the monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB ports, etc., would not be an issue. It entails a carefully thought-out user, product database structure, and control processor architecture. For the suggested system, the H/S personal specification is established.

## **INPUT AND OUTPUT REPRESENTATION:**

The information system and the user are connected through the input design. It entails creating specifications and procedures for data preparation, which are necessary to convert transaction data into a form that can be processed. This can be done either by having people key the data directly into the system or by having the computer read the data from a written or printed document. The input process is designed with an eye toward minimising the quantity of input necessary, minimising errors, minimising delays, minimising extra stages, and maintaining a straightforward workflow. The input is made in such a way that it offers security, usability, and privacy preservation. These factors were taken into account by Input Design:

What information should be provided as input?

1. How should the data be organised or coded?
2. The conversation to direct the operating staff's input-giving.
3. Methods for creating input validations and procedures to take in the event of an error.

## **OBJECTIVES:**

The process of transforming a user-centered description of the input into a computer-based system is known as input design. This design is crucial to preventing mistakes in the data entry process and providing management with clear instructions for receiving the right information from the computerised system.

It is accomplished by designing displays that are easy for users to utilise when entering bigamounts of data. The purpose of input design is to make data entering simpler and error-free. The data entering panel is created such that any data manipulations are possible. Additionally, it offers record viewing capabilities.

The veracity of the entered data will be verified. Screens can be used to help enter data. When necessary, appropriate messages are sent so that the person is not immediately surrounded by corn. Thus, the goal of input design is to produce an intuitive input arrangement.

## **OUTPUT DESIGN:**

A quality output is one that shows the information clearly and complies with the end user's needs. Any system's outputs are how processing results are transmitted to users and other systems. It is decided during output design how information will be displaced for immediate demand as well as the hard copy output. It is the user's most crucial and direct source of information. The interaction between the system and aiding user decision-making is improved by efficient and intelligent output design.

1. Designing computer output should be done in an organised, well-thought-out manner; the correct output must be generated while making sure that each output element is built so that users may use the system expeditiously. When analysing computer-generated output, one should pinpoint the precise output that is required to satisfy the specifications.
2. Pick information presentation strategies.
3. Produce reports, documents, and other forms containing data generated by the system.

## 7.2 ALGORITHMIC COMPLEXITY:

In this system using YOLO (You Only Look Once) algorithm

YOLO is an algorithm that uses neural networks to provide real-time object detection.

This algorithm is popular because of its speed and accuracy.

It has been used in various applications to detect traffic signals, people, parking meters, and animals

YOLO algorithm works using the following three techniques:

1. **Residual blocks** : First, the image is divided into various grids. Each grid has a dimension of  $S \times S$
2. **Bounding box regression:**

A bounding box is an outline that highlights an object in an image.

Every bounding box in the image consists of the following attributes:

Width (b w)

Height (b h)

Class (for example, person, car, traffic light, etc.)- This is represented by the letter c

Bounding box center (bx ,by)

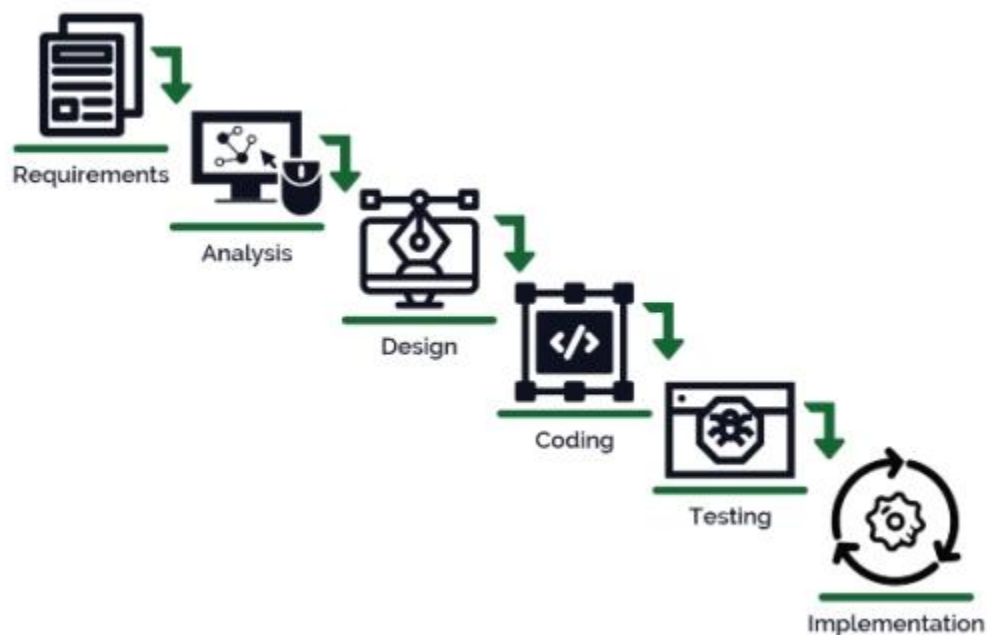
3. **Intersection Over Union (IOU):** Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap.

## 7.3 METHODOLOGIES:

### Steps for object Detection using YOLO v3:

1. A number of shape-related photos are the inputs (m, 416, 416, 3).
2. This image is sent to a convolutional neural network by YOLO v3
3. The output volume obtained by flattening the final two dimensions of the aforementioned output is (19, 19, 425):
4. In this case, a grid of  $19 \times 19$  cells yields 425 results.
5. 5 is the number of anchor boxes in a grid, so 425 is equal to  $5 * 85$ .
6.  $85 = 5 + 80$ , where 5 is (pc, bx, by, bh, and bw) and 80 is the total number of classes we're looking to find.
7. The result includes a list of bounding boxes and any classes that were identified. Six integers are used to symbolise each bounding box (pc, bx, by, bh, bw, c). Each bounding box is represented by 85 numbers if "c" is stretched into an 80-dimensional vector.
8. To prevent picking overlapping boxes, IoU (Intersection over Union) and Non-Max Suppression are used.

A traditional model used in the system development life cycle to design a system with a linear and sequential approach is the waterfall model. Because the model progresses methodically from one phase to the next in a downward direction, it is known as a waterfall model. The waterfall methodology does not outline how to address requirement changes by going back to a prior phase. The waterfall method was the first one to be used to development.





The initial Process Model introduced was the Waterfall Model. The term "linear-sequential life cycle model" is also used to describe it. It is incredibly easy to use and comprehend. There is no overlap between phases in a waterfall model; each step must be finished before the subsequent phase can start. The first SDLC methodology for software development was the waterfall model. The software development process is depicted using the waterfall model, which follows a linear sequential flow. This implies that a phase of development can only start if the one before it is finished. The phases in this waterfall model do not cross over.

The Waterfall technique was the first SDLC model that was widely utilised in software engineering to guarantee the project's success. The entire software development process is split into distinct phases using "The Waterfall" technique. Typically, the results of one step in this waterfall model serve as the input for the subsequent phases in turn.

The Waterfall model's successive phases are:

1. Requirement Analysis and Gathering – During this stage, all prospective system requirements are gathered and compiled into a requirement specification document.
2. System Design – In this phase, the requirement specifications from the first phase are examined, and a system design is created. This system design aids in determining the overall system architecture as well as the hardware and system requirements.
3. Implementation – The system is first created as a collection of short programmes known as units, which are then combined in the following phase, using input from the system designs. Unit testing is the process of developing and evaluating each unit for functionality.
4. Integration and Testing – Following the testing of each unit created during the implementation phase, the entire system is integrated. The entire system is tested for errors and failures after integration.
5. System deployment – After functional and non-functional testing, the product is either provided to customers or deployed in their environments.
6. Maintenance – The client environment occasionally experiences problems. Patches are published to address certain problems. Additionally, improved versions of the product are issued. To bring about these changes in the surroundings of the consumer, maintenance is performed.
7. The progression is viewed as flowing smoothly downward (like a waterfall) through the phases as all of these phases are connected to one another. The "Waterfall Model" gets its name because the following phase doesn't begin until the prior phase's established set of goals have been met and it has been approved. Phases do not cross over in this model.

## Modeling a waterfall: Application

• Because each piece of software generated is unique, a proper SDLC strategy must be used based on both internal and external considerations. The following are some scenarios in which using the waterfall paradigm is most appropriate:

- Requirements are very well documented, distinct, and fixed.

1. The product definition is consistent.
2. Technology is stable and well-understood.
3. The prerequisites are clear-cut.

The project is brief, there are many resources with the necessary skills, and the product is supported.

### Waterfall Model: Benefits

1. Simple, clear, and simple to use
2. Due to the model's rigidity, it is simple to manage. Specific deliverables and a review process are included at each phase.
3. Each phase is processed and finished separately.
4. Effective for smaller projects with clearly defined needs.
5. Stages with a clear definition.
6. Milestones that are clear to all.
7. Simple task organisation.

Results and procedures are clearly documented.

## 7.4 MEASURING LANGUAGE USAGE:

The following metrics can help you understand how successful software development projects impact the bottom line:

1. Retention rate/churn
2. Number of active users (daily, weekly, monthly)
3. Customer acquisition rate
4. Conversions
5. Customer lifetime value (CLV)
6. Monthly recurring revenue (MRR)

Business performance, or revenue, metrics are the most obvious method of measuring project success. Note that they don't help you understand why it was successful or which factors had the greatest impact on its overall success.

## **CUSTOMER CONTENTMENT:**

Metrics on customer satisfaction assist you in understanding how well your team can fulfil end-user expectations and how it connects to brand perception.

To gauge satisfaction, you could monitor the metrics listed below:

1. Feature emotion
2. NPS, CSAT ratings
3. Support ticket themes

Finally, you should monitor satisfaction over time to make sure you keep becoming better.

A new McKinsey analysis claims that executives are becoming more and more aware that CX surveys fall short of meeting their genuine CX requirements. Only 15% of respondents claimed they were entirely satisfied with their strategy, despite the fact that 93 percent said they use survey measures like NPS or CSAT as their main tool for monitoring performance.

Important information is indeed captured by survey analytics. But focusing solely on these measures ties into the previous point: brands are thinking too little and too obviously to make any significant advancements.

Instead, be sure to examine a variety of data sets. Customer-related insights are readily available in a variety of formats, including brand mentions, chatbot dialogues, behavioural data, support tickets, and reviews.

Henry Martinez of 3Pillar adds that it's a good idea to monitor "quality metrics and timetable adherence—from the perspective of the client. Reality will dictate perception. It must be accepted before it can be considered "true."

In essence, you're attempting to ascertain whether your capacity to consistently provide high-quality items on time (or faster than anticipated) and to release products has any measurable impact on customer satisfaction.

## **PRODUCT PARTICIPATION:**

"A project can meet all objectives in terms of cost and deliverables, but if it doesn't make an influence in the market, it's a failure," claims Eddy Vidal Nunez Garcia of 3Pillar. If it resonates with clients, that is the best metric.

Naturally, "impact on the market" also includes measures for customer satisfaction and sales, but KPIs for product engagement are one of the best methods to assess the most crucial aspect of success: if you were able to create something your consumers really value.

You can learn more about user usage of your product using these metrics. Use them to find out whether clients use certain features more frequently than others.

How frequently do they log in?

Are they finishing the onboarding procedures? Where do they leave the process if not?

Examples include:

1. User adoption
2. New user onboarding completion percentage
3. App usage and time spent there
4. Number of actions/sessions.

By highlighting warning signs like incomplete onboarding, underutilised features, or abrupt drops in usage, these KPIs assist you in identifying clients who may be at risk of leaving your business. These analytics can reveal which users need direct assistance from a sales or support representative on an individual basis, and over the long term, they can assist your development team in locating and eliminating friction points for a more seamless experience going forward.

## **AGILE MEASURES**

Agile metrics assess the effectiveness of team planning and decision-making.

They are helpful for enhancing the development process, which enhances the quality of their output, even though they cannot tell you whether the software itself had any impact on your strategic goal.

1. Lead time
2. Velocity
3. cycle time
4. the impact of code changes on software development projects
5. open/close rates
6. Mean time between failures are a few examples.

Before taking any action based on any presumptions you may have about the issue, it's critical to consult with the team if any of these KPIs deviate from the intended range or show that the team is headed in the wrong incorrect direction.

Learn the complete story, then collaborate to find a solution—if one is even required].

Agile metrics can help you highlight areas that need attention and launch a discussion about the best course of action, but they can't help you find the fundamental cause of an problem.

## **DELIVERABLES:**

The finest software development indicators, according to Abel Gonzalez Garcia of 3Pillar, "verify whether the project's goals have been realised. The work we put into our initial estimate will then be compared.

You should concentrate on deliverables if your objective is to gauge how effectively development teams are performing their work.

The optimal use of these indicators is to hasten the release of (high-quality) software solutions. Use them if you want to find out more about the reasons behind delays, locate potential bots, or if your team is committing errors that necessitate rework.

1. Total output
2. On-time delivery
3. Number of features implemented
4. Tasks finished
5. Bugs addressed
6. Code churn
7. Active days
8. Effects of code modifications
9. Mean time to recover
10. Mean time between failures are all important metrics.

It's crucial to establish what successful software developments look like from a productivity perspective in collaboration with team leads and project managers. It's crucial that you communicate these choices to the entire team along with clear guidelines for success in the context of each person's unique duties and responsibilities.

Productivity KPIs aren't the most helpful software development indicators on their own. Business leaders have the genuine risk of concentrating too much on output rather than outcome. Having said that, these indicators can assist firms in enhancing their management approach, highlighting individual accomplishments, and identifying precise areas for development—leading to the quicker delivery of better items to market.

## **FINAL REFLECTIONS**

In the end, evaluating how each component affects the bigger picture is how to gauge a project's success.

It's crucial to realise that this can't be accomplished (at least not effectively) in a single report; rather, you'll need to concentrate on one target at a time to gauge its true impact and pinpoint concrete prospects for development the following time.

## 7.5 DEBUGGING:

### What is Debugging?



Any software that is being developed undergoes rigorous testing, troubleshooting, and maintenance in order to provide products that are bug-free. Nothing can be done perfectly on the first try.

Therefore, it should come as no surprise to anybody that when software is written, it has a lot of faults. This is because nobody is flawless, and while having errors in the code is not a problem, avoiding them or failing to prevent them is!

We can infer that debugging is nothing more than a process of eliminating or correcting the faults included in a software programme because all those bugs and errors are routinely eliminated.

Debugging is a step-by-step process that begins with error detection, analysis, and error removal. When a piece of software is unable to produce the desired outcome, we need a software tester to examine the problem and find a solution.

Despite how effective the outcome was, since faults are fixed at every stage of debugging in software testing, we can infer that the work is time-consuming and hard.

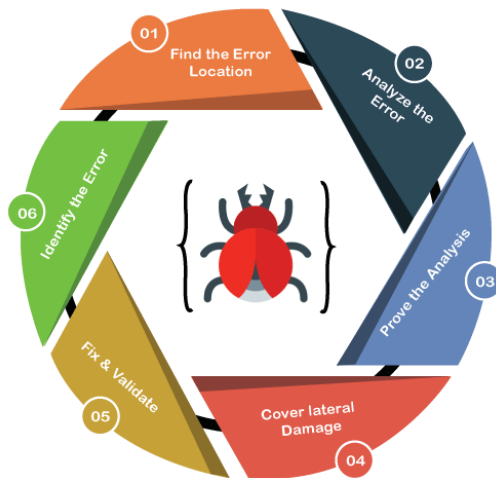
### Why is Debugging necessary?

When we begin developing the software program's code, debugging begins. Because the code is combined with various other programming units to create a software product, it gradually continues in the subsequent stages to deliver it.

### These are the advantages of Debugging:

1. Debugging has the ability to report an erroneous condition as soon as it arises. By identifying flaws earlier, it prevents results from being hampered, making software development stress-free and easy.
2. It provides pertinent data structure-related information that makes interpretation easier.
3. Debugging helps the developer cut out useless and disruptive information.
4. The developer may quickly avoid complex, one-use testing code with debugging, which will save time and effort while creating software.

## Steps involved in Debugging:



1. **IDENTIFY THE ERROR:** Time might be wasted by failing to spot an error right away. It is quite clear that the production problems that consumers report are difficult to evaluate, and occasionally the data we receive is inaccurate. Therefore, it is essential to pinpoint the actual error.
2. **LOCATE THE ERROR:** Once the error has been appropriately identified, you must meticulously analyse the code several times to determine its exact location. This step generally focuses on locating the error rather than recognising it.
3. **ANALYZE THE ERROR:** The third phase entails error analysis, a bottom-up strategy that begins with the error's location and moves on to a study of the code. The faults are simpler to understand after this procedure. The two main objectives of error analysis are to reevaluate faults in order to uncover defects that already exist and to postulate the possibility of collateral damage from a remedy.
4. **SUPPORT THE ANALYSIS:** After studying the main issues, it's important to seek for any potential additional application flaws. The fourth phase is used to create automated tests for these areas by integrating the test framework.
5. **COVER LATERAL DAMAGE:** Compiling all of the unit tests for the code that has to be modified is the goal of the fifth phase. These unit tests need to pass when you run them.
6. **FIX & VALIDATE:** The final stage, fix and validate, focuses on correcting defects after which all test scripts are performed to see if they pass.

## STRATEGIES FOR DEBUGGING:

1. An in-depth study of a system is required for a better comprehension of it. It facilitates the debugger's ability to create precise drawings of the systems that need to be debugged.
2. The backward analysis examines the programme starting at the point where the failure notice first appeared in order to identify the problem region. To comprehend the cause of faults, it is vital to grasp the defect's region.
3. By using the breakpoints or print statements incurred at various points in the programme, the software monitors the issue in the forward direction during the analysis. The areas where the incorrect results are obtained are highlighted.
4. It is advised to draw on previous experience to analyse and resolve difficulties of a similar nature. The effectiveness of the debugger directly relates to the success rate of this strategy.

## DEBUGGING TOOLS

You might think of the debugging tool as a computer programme that is used to test and troubleshoot a number of other programmes. Currently, there is a wide variety of free software available on the market for debugging, including gdb and dbx. These programmes provide command-line interfaces that are console-based. Code-based tracers, profilers, interpreters, and other tools for automated debugging are some examples.

The following is a list of some of the most popular debuggers:

1. Radare20
2. WinDbg
3. Valgrind.

## 7.6 PROGRAMMING LANGUAGES:

Python is a programming language created by Guido van Rossum. In 1989, Guido van Rossum began using Python. A free and open-source programming language is Python. Python is a powerful, interactive, object-oriented, and interpreted scripting language. Python has been created to be very readable. It typically employs English terms rather than punctuation in the other languages. Compared to other languages, it has fewer syntactical structures.

**Python Interpretation:** The interpreter processes Python while it is running. Your software does not need to be compiled before running. This is comparable to PHP and PERL.

**Python is interactive:** When writing programmes, you can actually sit at a Python prompt and communicate with the interpreter immediately. Python's support for object-oriented programming, which encapsulates code within objects, makes it an object-oriented language.



## FEATURES OF PYTHON:

1. **Readable:** The language Python is quite readable.
2. **Python is Simple to Learn:** Python is a simple programming language that is expressive and high level, making it simple to grasp and simple to learn.
3. **Python is available** and may be used with a variety of operating systems, including Mac, Windows, Linux, and Unix. It is portable and cross-platform as a result.
4. **Open Source:** The programming language Python is free and open source.
5. **Large standard library:** Python includes a sizable standard library that contains some useful functions and codes that we can utilise when building Python programmes.
6. **Python can be downloaded and used without cost.** This implies that you can use it in your application and download it for free. See the Python Open Source License. Python is an example of FLOSS (Free/Libre Open Source Software), which permits unrestricted copying, reading of the source code, and modification of the software.
7. **Supports handling of exceptions:** If you're new, you might be wondering what an exception is. An exception is a circumstance that can happen during a programme exception and impede the regular operation of the programme. Because Python offers exception handling, we can test various scenarios that can lead to an exception later on and design less error-prone code.
8. **Supports generators and list comprehensions.** Advanced features. These characteristics will be discussed later.
9. **Python offers automated memory management,** which entails that memory is automatically freed and purged. There is no need to waste time erasing the recollection.

## APPLICATIONS FOR PYTHON:

1. **Web development** -Python is the foundation of web frameworks like Django and Flask. They assist you in creating server-side code that facilitates database management, backend programming logic creation, url mapping, etc.
2. **Machine learning** -Python is widely used to create machine learning applications. Writing a computer's logic in such a way that it can learn and solve a specific problem on its own is known as machine learning. For instance, a machine learning algorithm that recognises user interest is used on websites like Amazon, Flipkart, eBay, and others to recommend things. Another application of machine learning is voice and face recognition in smartphones.
3. **Data Analysis** -Python can also be used to create data analysis and data visualisation in the form of charts.
4. **Scripting** -Writing small programmes to automate straightforward operations, such as sending pre-written emails, is known as scripting. Python is a programming language that may also be used to create these kinds of apps.
5. **Game development** -Python may be used to create games.

6. Python may be used to create embedded programmes.
7. **Desktop apps** -Python allows you to create desktop apps using libraries like TKinter or QT.

## **BASIC FUNDAMENTALS:**

### **TOKENS**

TOKENS are the smallest individual component of the programme. The tokens used in Python are as follows:

1. Keywords
2. Identifiers
3. Literals Operators.

## CHAPTER-8

# SYSTEM TESTING

### INTRODUCTION:

System testing is a crucial component of software quality assurance and serves as the final examination of specification, designing, and coding. Executing a programme with the goal of identifying errors is known as testing. When a programme is being tested, a collection of test cases are run alongside it, and the results are then analysed to see if the programme is functioning as it should.

Goals of the test:

1. Testing is the process of running a software with the aim of identifying errors.
2. The possibility of discovering an error that hasn't been identified yet is an indicator of a solid test case design.
3. A test that finds an error that hasn't been found yet is successful.

These aforementioned goals imply a significant shift in the view. Testing can only demonstrate the presence of software problems; it cannot demonstrate the absence of defects.

The testing procedures are as follows:

### 8.1 UNIT TESTING:

Unit testing concentrates verification efforts on the module, which is the smallest unit of software design. This test focuses on each module separately to make sure that it functions properly as a whole. Unit testing was chosen as the name because it ensures that each module is tested separately.

Sl Test Case:	UTC1
Name of Test:	Load dataset
Items be inserted:	Dataset features and labels are displayed or not
Sample Input:	Dataset csv file

Expected output:	All features and labels should be displayed
Actual output:	Total data is displayed
Remarks:	Pass.

Sl. Test Case:	UTC2
Name of Test:	Split data
Items being tested:	Data is divided in to train and test set
Sample Input:	Test and train size
Expected output:	Dataset is divided in to 2 parts
Actual output:	Based on given test size data is divided and stored in train and test sets
Remarks:	Pass

## 8.2 INTEGRATION TESTING:

It is a methodical process for integrating many programme modules into a single software framework. This test identifies and validates all errors within the module.

Sl. Test Case:	ITC1
Name of Test:	Train Model
Item being tested:	Model fit is performed
Sample Input:	Train x and train y
Expected output:	Fit is performed
Actual output:	Training is done and accuracy is displayed
Remarks:	Pass.

Sl. Test Case:	ITC2
Name of Test:	Accuracy calculation

Item being tested:	If accuracy of a algorithm is calculated
Sample Input:	Test x and test y
Expected output:	Accuracy of a algorithm
Actual output:	Accuracy of each model
Remarks:	Pass.

### 8.3 OUTPUT TESTING:

Testing the output enables us to determine whether the output is accurate or inaccurate.

### 8.4 VALIDATION TESTING:

When the software is ready for integration testing and meets the specifications, validation testing is performed. However, it needs to be verified against the specification to find any unforeseen flaws in the future and to increase its dependability.

### 8.5 SOFTWARE TESTING TECHNIQUES:

A plan for software testing serves as a guide for the software developer. Testing is a collection of tasks that can be organised in advance and carried out in a methodical manner. For this reason, a template for software testing, or a series of phases into which we can insert certain test case creation techniques, should be developed for the software engineering process.

**Any plan for software testing should include the following qualities:**

1. Testing starts with the module and moves "outward" to the integration of the full computer-based system.
2. Various testing methods are appropriate in various circumstances.
3. The software's creator does testing, together with a separate test team.
4. While debugging must be taken into account in any testing strategy, testing and debugging are two distinct processes.

## **STRATEGIES FOR SOFTWARE TESTING**

The following are the two strategic stances that testing takes:

The software developer has direction thanks to a plan for software testing. Testing is a set of procedures that can be planned ahead of time and executed methodically. In order to accommodate different test case creation approaches, the software engineering process should define a template for software testing or a group of processes. Any software testing approach should include the following characteristics:

1. Testing "outward" begins at the module level and moves toward the integration of the entire computer-based system.
2. Different testing techniques are appropriate at different times.
3. The software's designer and a different test team conduct the testing.
4. Any testing methodology must take debugging into account, even if testing and debugging are distinct responsibilities.

### **8.5.1 TESTING A BLACK BOX**

This is also known as behavioural testing and is a software experiment blueprint of the internal architecture, design, and implementation of the critical component being tested that the tester is unaware of. These tests may be useful or useless, but they are often utilitarian. Without giving much attention to internal programme learning, we merely focus on data sources and item structure surrender in this testing.

#### **Black Box Testing Types :**

Black Box Testing comes in a variety of forms, however the following are the most common:

Functional testing is a sort of black box testing that focuses on a system's functional requirements and is carried out by software testers.

Non-functional testing - This kind of black box testing focuses on non-functional requirements including performance, scalability, and usability rather than a specific capability.

Regression testing is done after code upgrades, fixes, or any other system maintenance to ensure that the new code hasn't negatively impacted the previous version of the code.

### **8.5.2 TESTING A WHITE BOX:**

Clear box experiment approaches investigate the inner structure of the used information designs, inner plan code structures, and the functionality of the product rather than only concentrating on usefulness as in a discovery experiment. Box testing is a design that generates test cases by relying heavily on the internal coding design. White box testing defines a clear report on component testing and focuses mostly on the program's internal coding.

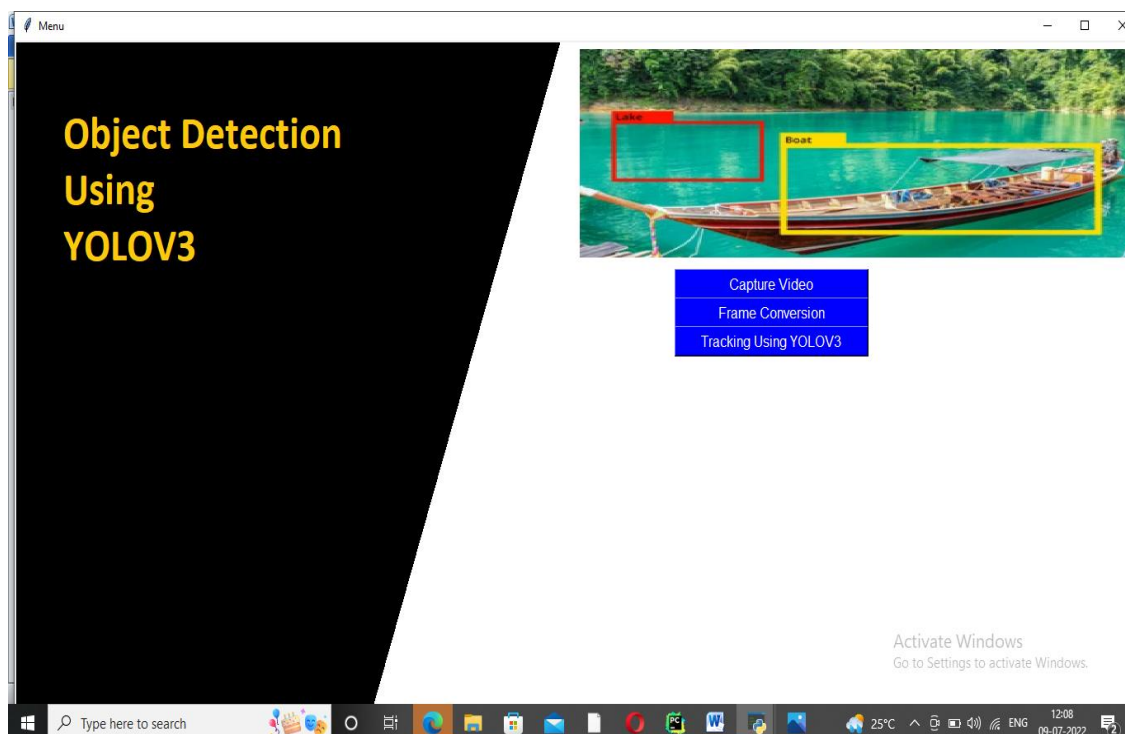
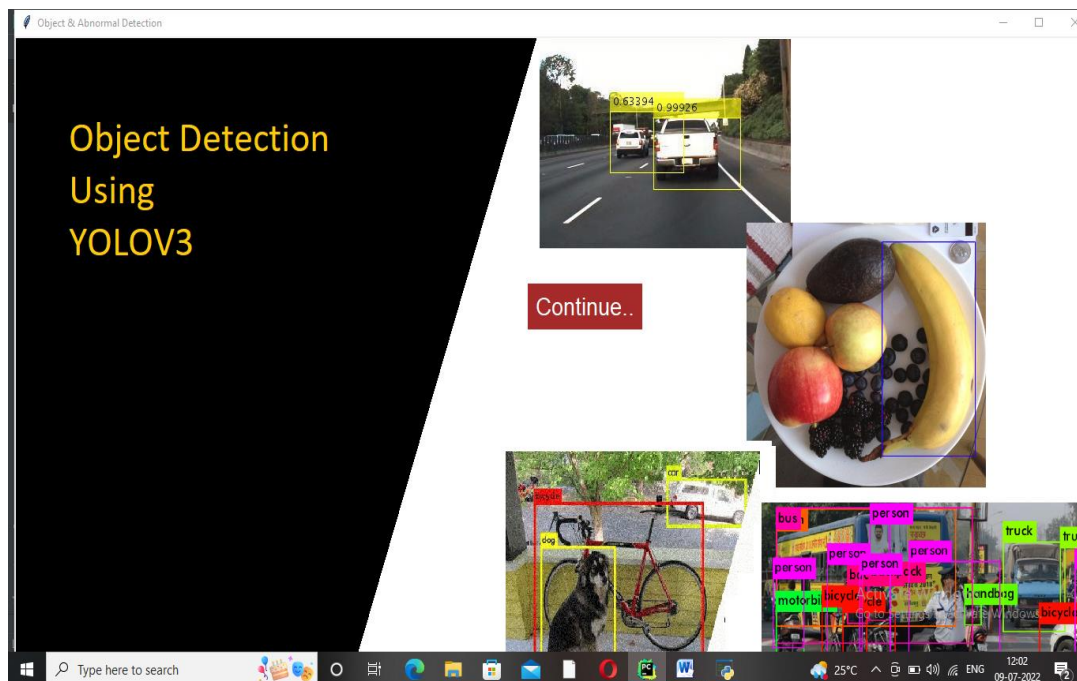
The following aspects of the software code are tested during white box testing:

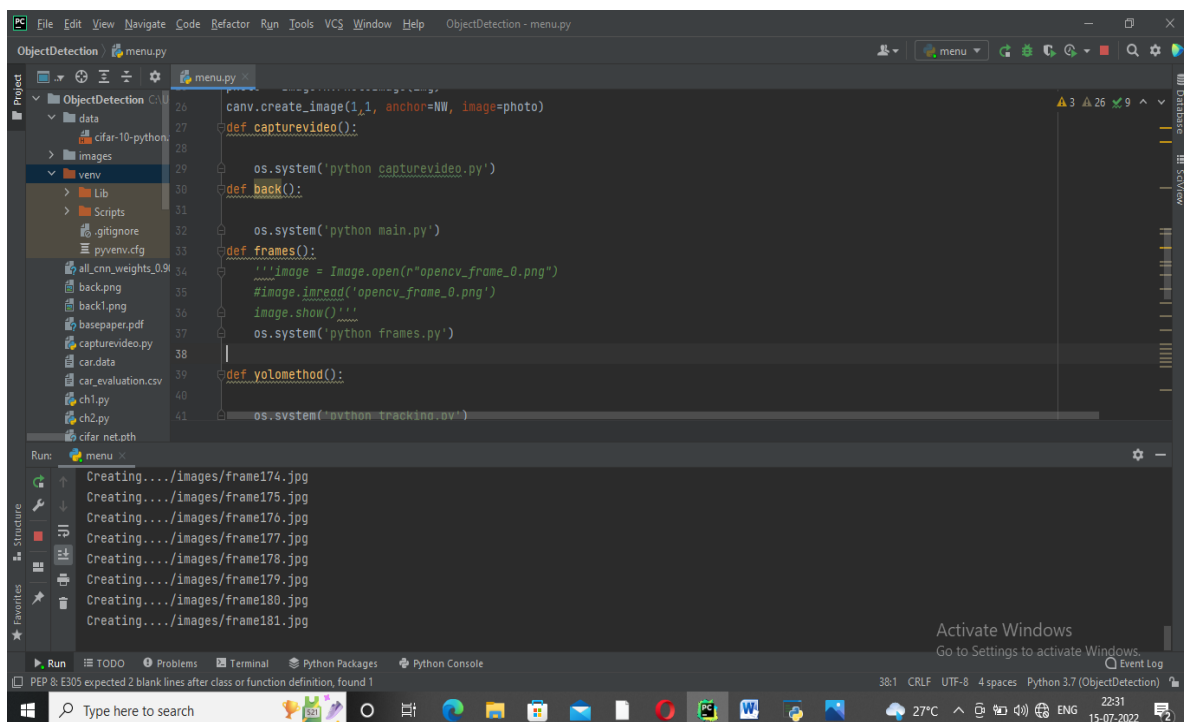
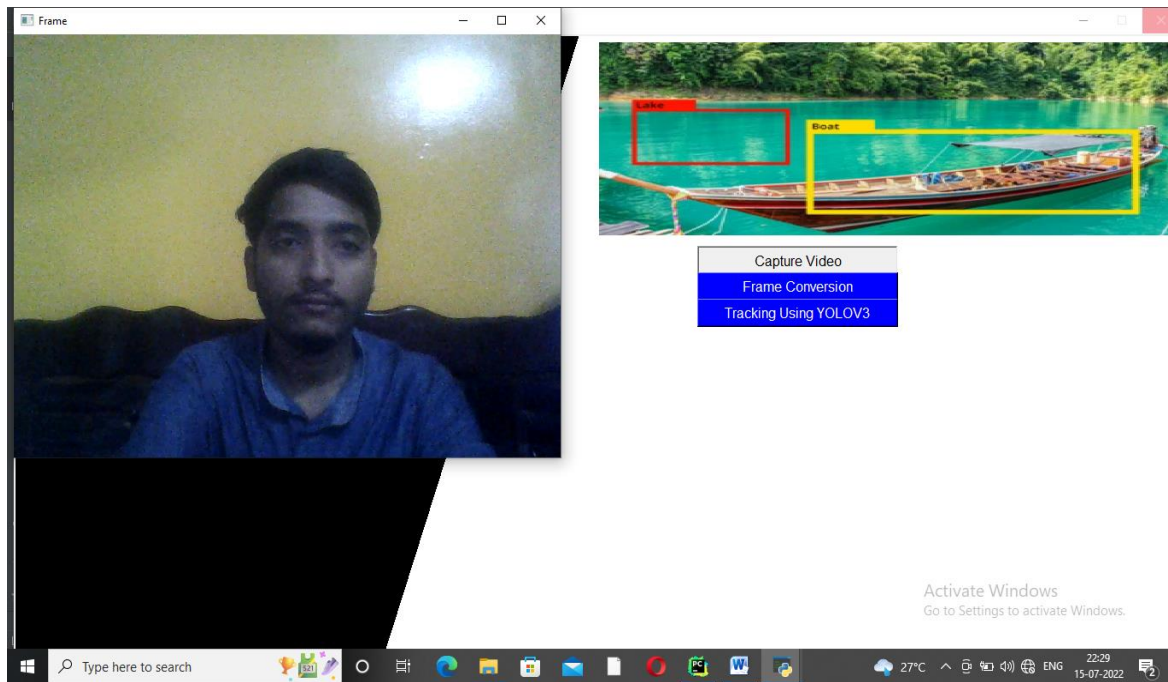
1. Gaps in the internal security.
2. Paths in the coding procedures that are broken or poorly organised.
3. How certain inputs are passed through the code.
4. Projected results.
5. The way conditional loops work.
6. Individual testing of each statement, object, and function

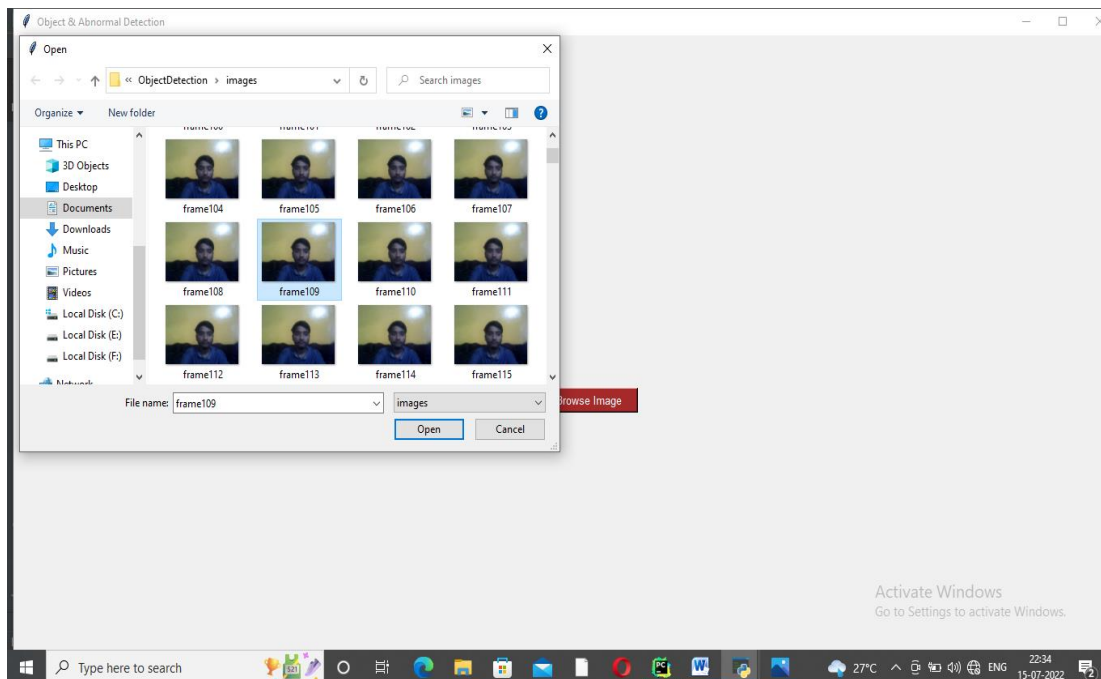
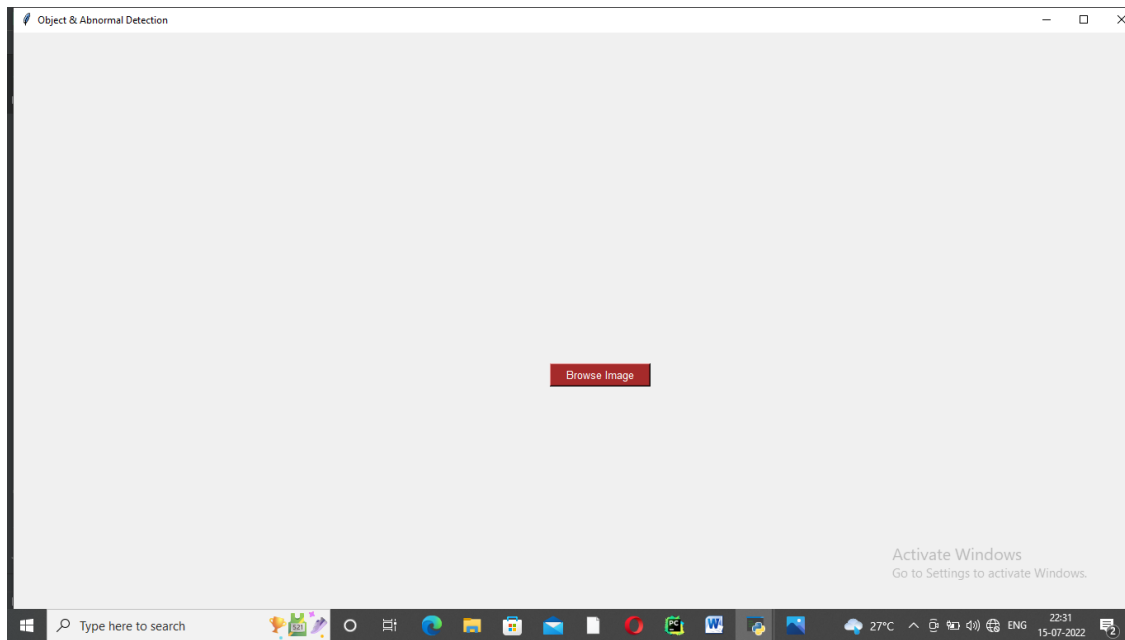


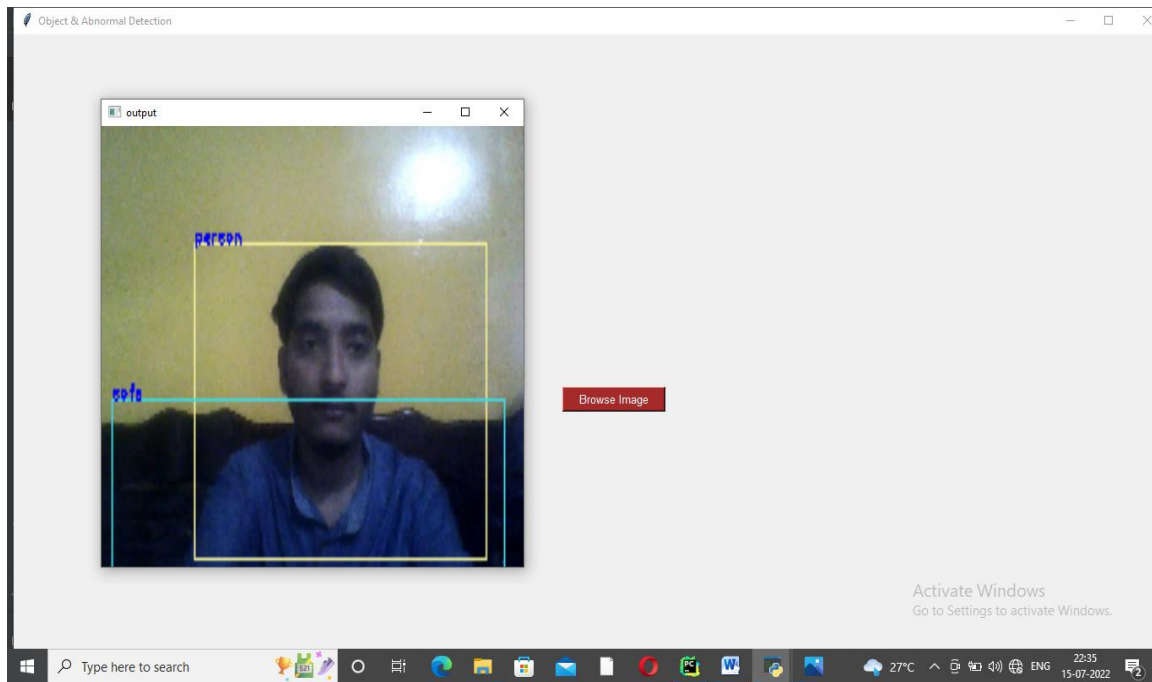
## CHAPTER-9

### RUNTIME FORMS









## CHAPTER-10

### CONCLUSION:

Because of the benefits of the YOLO method, I have utilised and suggested using it in this project for object detection. This method can be used in a variety of settings to address practical issues like security, lane monitoring, or even providing aural feedback to help the blind. This is a model that we developed to identify just three things, but it can be scaled up to identify many more. YOLOV3 is a better object detection method as a consequence, accurately tracking the object with a 91 percent accuracy and a 51 percent loss. In COCO datasets, YOLOv3 is employed to differentiate a variety of objects. Based on an analysis of the photo classes, performance metrics for YOLOv3 are tabulated. The class's precession value increases with increasing map values. YOLOv3 and Open CV are used to build multiple object tracking for traffic surveillance footage. Multiple objects are recognised and tracked in various movie frames. The main benefit of employing YOLOV3 is its incredibly fast analysis rate of 45 frames per second. The idea of generic object representation is also understood by YOLOV3. When compared to CNN approaches in terms of performance, this is one of the best object detecting algorithms.

## REFERENCES:

- [1] You Only Look Once: Unified, Real-Time Object Detection, Joseph Redmon, Santosh Divvala, and Ross Girshick, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
- [2] YOLO Juan Du1, "Understanding of Object Detection Based on CNN Family," Hisense's New Research and Development Center, Qingdao, China, 266071
- [3] "Learning to Localize Objects with Structured Output Regression," by Matthew B. Blaschko and Christoph H Lampert, appeared in Computer Vision - ECCV 2008, pp. 2–15.
- [4] The authors of "Application of Deep Learning in Object Detection," Xinyi Zhou, Wei Gong, WenLong Fu, and Fengtong Du, Communication University of China's Information Engineering School, Neuroscience and Intelligent Media Institute, and CUC
- [5] YAD2K: Yet Another Darknet 2 Keras by Allan Zelener  
(<https://pjreddie.com/darknet/yolo/>) is the official Yolo website.
- [6] YOLO explained by Andrew Ng at <https://www.youtube.com/watch?v=9sFpMpdYW8>