CHAPTER–I

INTRODUCTION

1.1 OBJECTIVE

The goal of "object detection" is to find the location of an object of an object in a given picture Accurately and mark the object with the appropriate category. To be precise the problem that object detection seeks to solve involves determining where the object is, and what it is. However, solving this problem is not easy. Unlike the human eye, a computer process Image's in two dimensions. Furthermore, the size of the object, its orientation in the space, its attitude, and its location in the image can all vary greatly.

1.2 INTRODUCTION

Object detection is technologically challenging and practically useful problem in the field of computer vision. Object detection deals with identifying the presence of various individual objects in an image. Great success has been achieved in controlled environment for object detection recognition problem but the problem remains unsolved in uncontrolled places, in particular, when objects are placed in arbitrary poses in cluttered and occluded environment. AS an example, it might be easy to train a domestic help robot to recognize the presence of Coffee machine with nothing else in the image.

On the other hand, imagine the difficulty of such robot in detecting the machine on a kitchen slab that is cluttered by other utensils, gadgets, tools, etc. The searching or recognition process in such scenario is very difficult. So far, no effective solution has been found for this problem. A lot of research is being done in the area of object recognition and detection during the last two decades. The research on object detection is multi-disciplinary and often involves the fields of image processing, machine learning, linear algebra, topology, statistics probability, optimization,

etc. The research innovations in this field have become so diverse that getting a primary first-hand summary of most state-of-the-art approaches is quite difficult and time consuming.

The approach used incorporates four computer vision and machine learning concepts: sliding Window's to extract sub-images from the image, feature extraction to get meaningful data from the sub-images, Support Vector Machines (SVMs) to classify the objects in sub image, and Principal Component Analysis (PCA) to improve efficiency. As a model problem for the Motivating application, we focused on the problem of recognizing objects in images, in particular, soccer balls and sunflowers. For this algorithm to be useful as a real-time aid to the visually-impaired, it would have to be enhanced to distinguish between "close" and "far" objects, as well as provide information about relative distance between the user and the object, etc. We do not consider these complications in this project; we focus on the core machine learning issues of object recognition. The training and testing of the proposed algorithm was done using data sets.

CHAPTER- II

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Existing system has the problems when object changes its appearances or object is moving out of camera view and again coming in front of camera. Tracking fails due to scaling, rotation, Changes and does not perform in case of full out of plane rotation. In existing system, we can detect Only one object at a time

2.2 **PROPOSED SYSTEM**

The proposed system overcomes the above-mentioned drawback and system can track and learn the real-time object automatically using principal component analysis-N learning, Template matching and eigen object detection. Latency has been optimized. Supports random Colors on bounding box. The proposed methods detect various multiple object detection using Image processing.

2.3 ADVANTAGES

- Using only a single camera for the depth finding
- Having uncomplicated calculations
- Requiring no auxiliary devices.
- This method can be used for both stationary and moving targets

CHAPTER III

DEVELOPMENT ENVIRONMENT

The Development Environment comprises of hardware requirements and software requirements. The Hardware requirement consists of Processor, Hard Disk, Mouse, RAM and Keyboard. The Software requirement consists of Operating System, Front end tool, Back-end tool and coding language.

3.1 HARDWARE REQUIREMENTS

Processor and RAM play a vital role in hardware process. For the development of this application, the following hardware requirements have been considered.

Processor	Ryzen 3 3250u, Intel 3 3rd gen
RAM	4 GB RAM minimum; 8 GB RAM recommended
Storage	HDD-250GB+ [Recommended-SSD]
Camera	5mp+
Network Connection	Yes

3.2 SOFTWARE REQUIREMENTS

Operating System is the major part of software requirements. The Front-End Tool and Back End Tool are used for storing and retrieving the information. The Coding Language is most important in developing the application. For the development of this application, the following software requirements have been considered.

Operating System	Linux, Windows 7/8/10 (64 bit)
Front End	React JS, CSS.
Coding Language	JavaScript
IDE	Visual Studio, WebStorm
Frame Work	ReactJs,TensorflowJS

3.3 SOFTWARE DESCRIPTION

3.3.1 FRONT END

REACT JS

React is a JavaScript library that aims to simplify the development of visual interfaces Developed at Face book and released to the world in 2013, it drives some of the most widely used apps, powering Face book and Instagram among countless other applications. Its primary goal is to make it easy to reason about an interface and its state at any point in time. It does this by dividing the UI into a collection of components.

React makes many things easier, and its ecosystem is filled with great libraries and tools. React in itself has a very small API, and you basically need to understand 4 concepts to get started:

Components

JSX

State

Props

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content. media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium.

3.3.2 BACK END

WASM

Web Assembly (abbreviated WASM) is a binary instruction format for a stack-based virtual machine. WASM is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications. The WASM stack machine is designed to be encoded in a size- and load-time-efficient binary format. Web Assembly aims to execute at native speed by taking advantage of common hardware capabilities available on a wide range of platforms. WASM supports on major browser engines such as Google Chrome, Mozilla Firefox, Microsoft Edge.

CHAPTER – IV

SYSTEM DESIGN

4.1 DATA MODEL

Data Model is a set of concepts to describe the structure of the database and certain constraints that the database should obey. The main aim of data model is to support the development of information system by providing the definition and format of data. A data model can be a diagram or flowchart that illustrates the relationships between data. Usually, data models are specified in a data modeling language. Although capturing all the possible relationships in a data model can be very time intensive, it's an important step and shouldn't be rushed. Well documented models allow stake holders to identify errors and make changes before any programming code has been written. Data modelers often use multiple models to view the same data and ensure that all processes, entities, relationships, and data flows have been identified. Data Model can be classified into various evolutions. Some of the evolutions are Hierarchical Model, Network Model, Relational Model, Entity Relationship Model and Object-Oriented Model.

The structural part of a data model theory refers to the collection of data structures which make up a data when it is being created. These data structures represent entities and objects in a database model. The manipulation part of a data model refers to the collection of operators which be applied to the data structures.

ROLE OF DATA MODEL

The main aim of data model is to support the development information system by the definition and format of data. If this is done consistently across systems then compatibility of data can be achieved. If the same data structures are used to store and access data then different applications can share data. Data model is based on data, data relationship, data semantic and data constraint.

CATEGORIES OF DATA MODEL

i) Conceptual Data Model
ii) Physical Data Model
iii) Implementation Data Model

Conceptual Data Model: This data model provides the concept that is close to the way many users perceive data.

Physical Data Model: This data model provides the concept that describes the details of how data is stored in the computer.

Implementation Data Model: This data model provides the concept that fall between the above two, balancing user views with some computer storage details.

4.2 DESIGN NOTATION

4.2.1 PROCESS

A procedure or process does operations and give the output on the supplied arguments The pure Functions are considered as low-level process that do not have side effects. A process data flow component is represented as an ellipse.

4.2.2 DATA FLOWS

The nexus between one process to another or one sub identity to mother is represented by the with the intermediate value or the label on it.

4.2.3 EXTERNAL ENTITY

Any external entity which can access the flow in DFD like a librarian, is called as External Entity component. It is represented as a rectangle.

4.2.4 OUTPUT SYMBOL

While the user interaction with the system the DFD depicts it in the form of a below polygon.

4.3 DATA FLOW DIAGRAM

4.3.1 ZERO LEVEL DFD

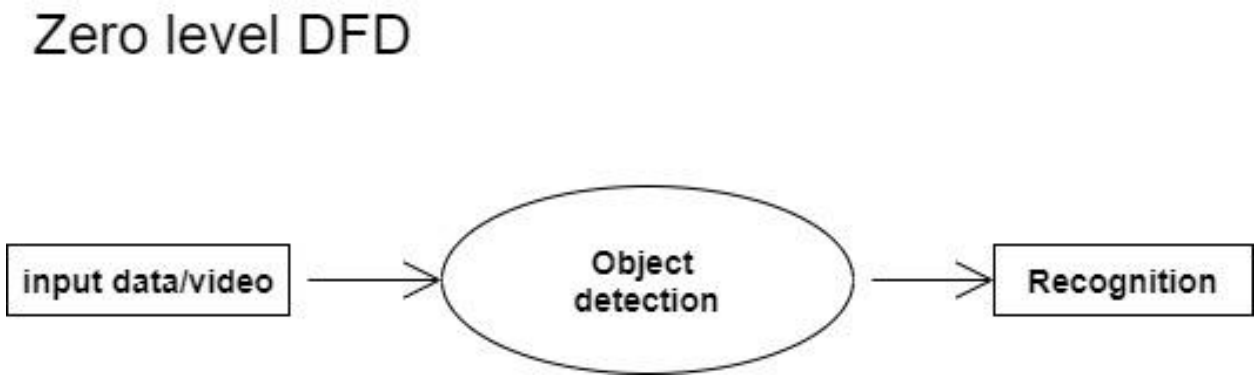
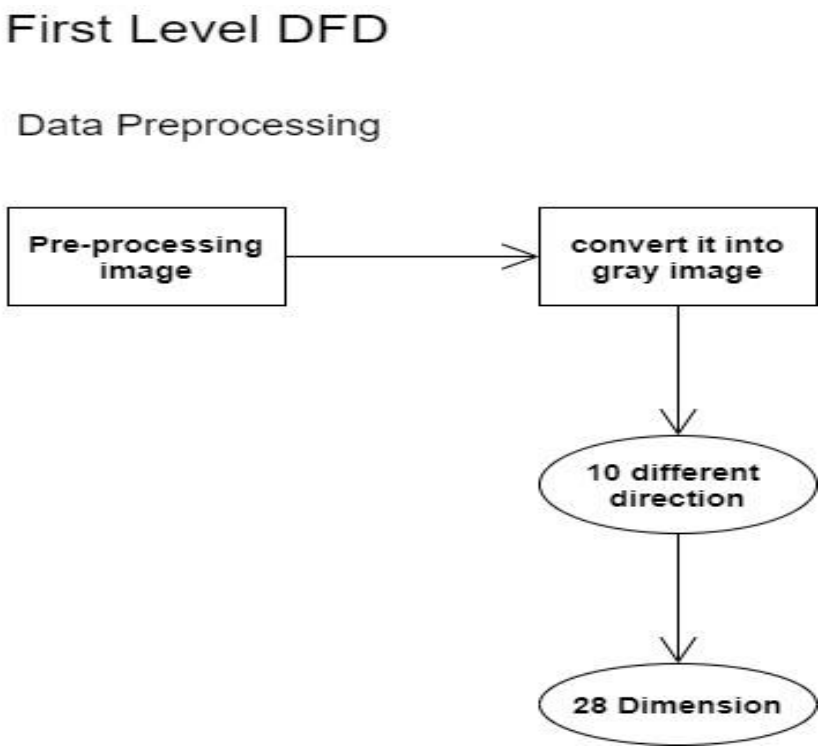


Fig 4.3.1

4.3.2 FIRST LEVEL DFD-DATA PREPROCESSING



4.3.3 FIRST LEVEL DFD-PROCESSING

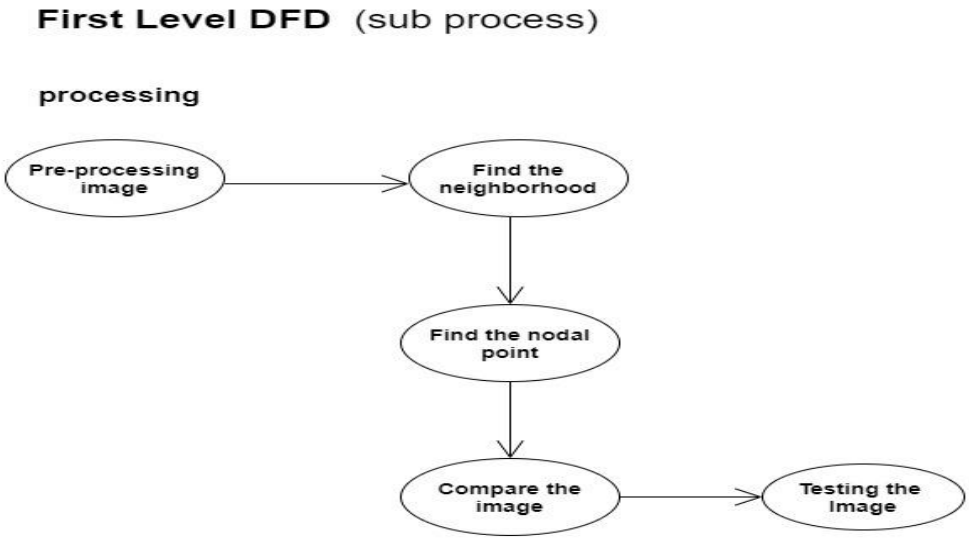
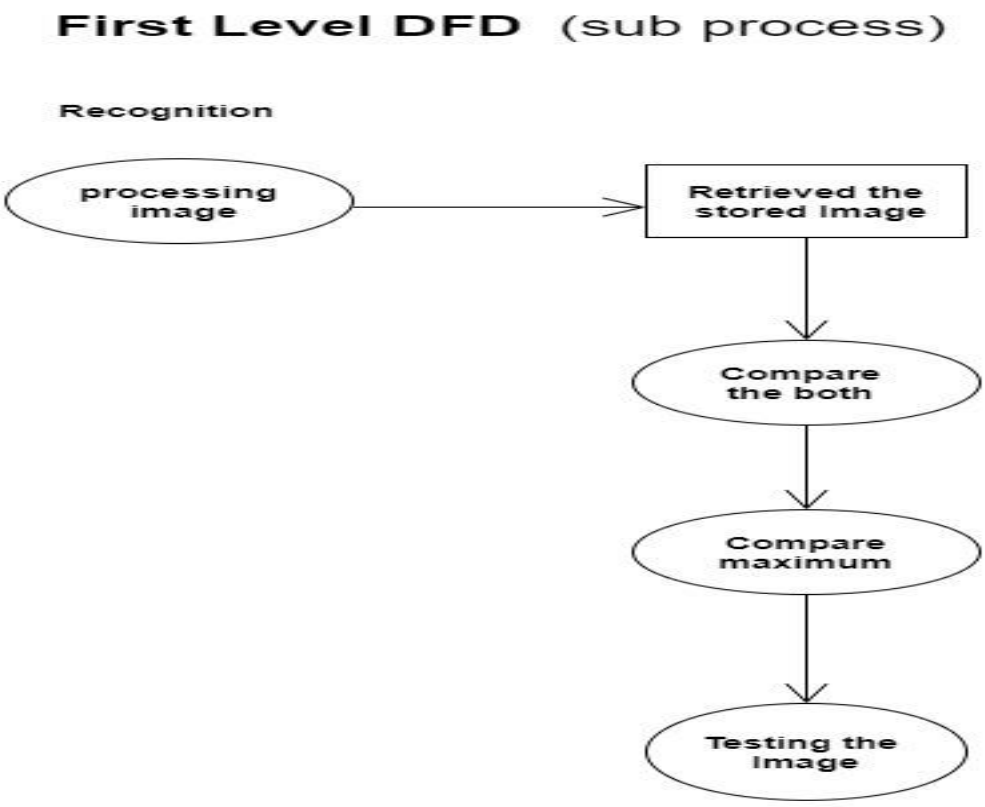


Fig 4.3.3

4.3.4 FIRST LEVEL DFD-RECOGNITION



4.3.5 FIRST LEVEL DFD-SUB PROCESS

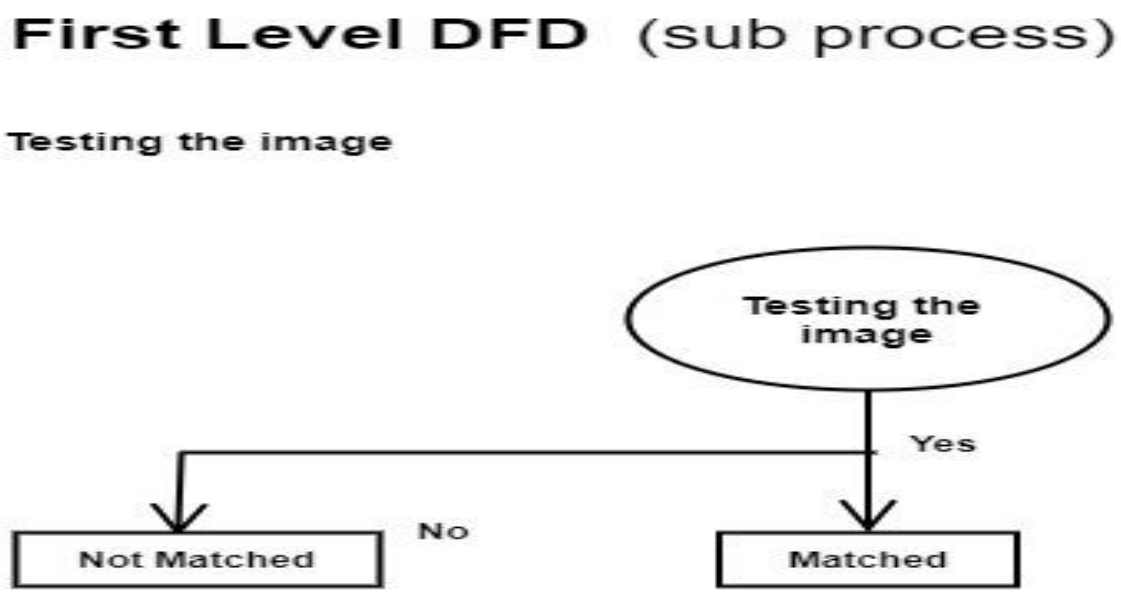


Fig 4.3.5

4.3.6 SECOND LEVEL DFD

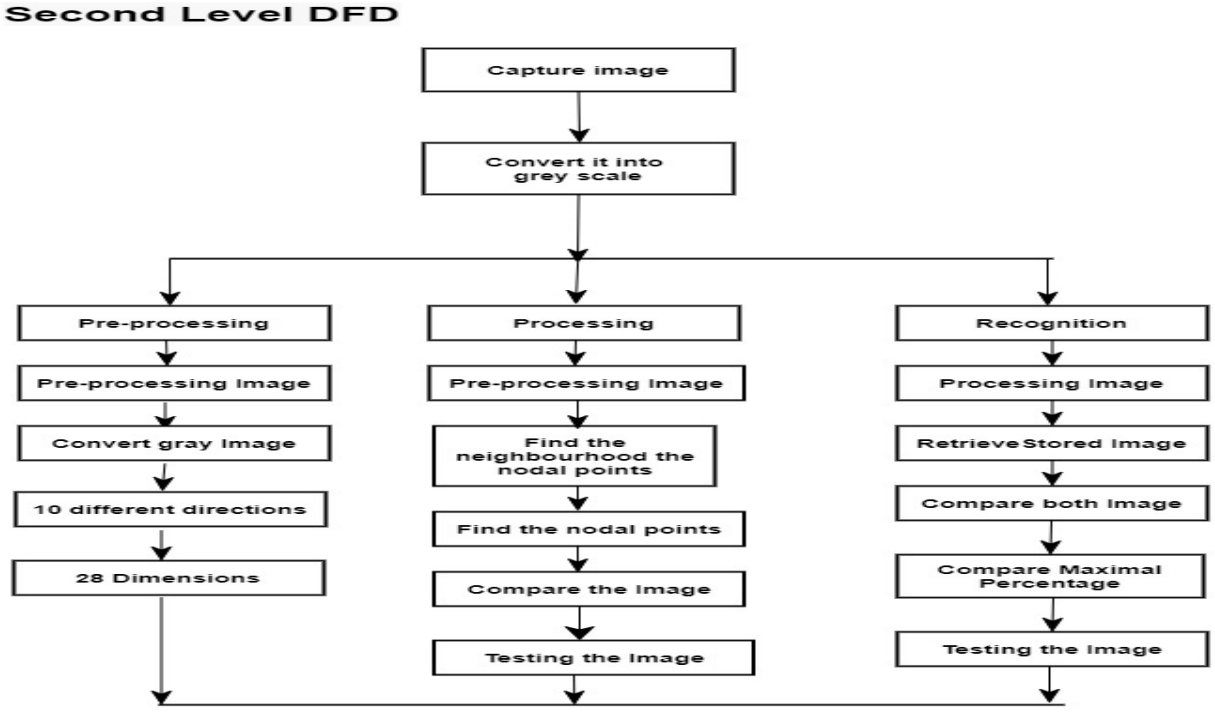


Fig 4.3.6

4.3.7 USECASE DIAGRAM

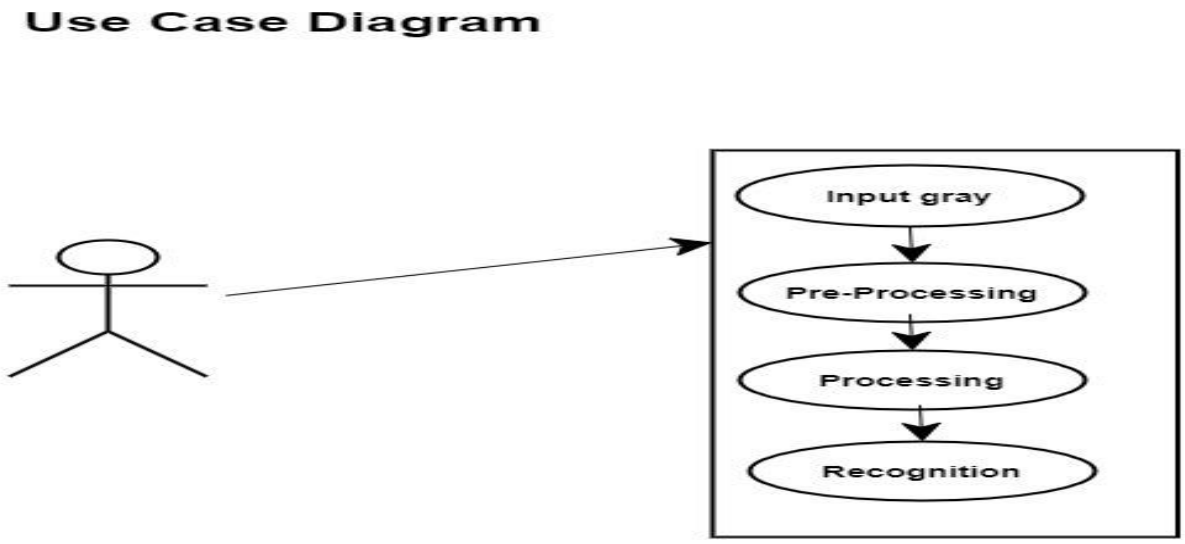


Fig 4.3.7

4.3.8 SEQUENCE DIAGRAM

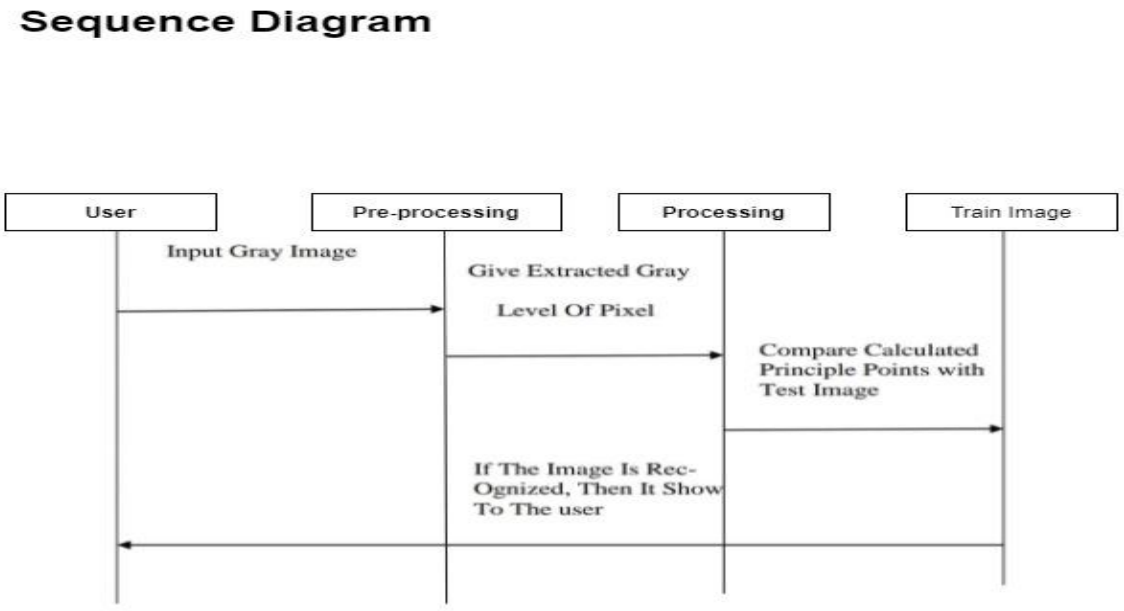


Fig 4.3.8

CHAPTER - V

SOFTWARE DEVELOPMENT

Software Development Life Cycle is a process used by software industry to design, develop and test high quality software's. The SDLC aims to produce high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

5.1 WATERFALL MODEL:

The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases,

- Requirement analysis resulting in a software requirements specification
- Software design
- Implementation
- Testing
- Integration, if there are multiple subsystems
- Deployment (or Installation)

5.2. DEVELOPMENT METHODOLOGY

Development Methodology Development methodology to be used in this research method is waterfall Software Development Life Cycle (WF-SDLC). Since this is a real time object detection with a real-time. we have seen the need to add collaborative methodology to the WF-S LC because it isn't enough for the WF- SDLC to work alone in a collaborative system.

Therefore, the system will be designed following the Collaborative Waterfall Software Development Lifecycle (C-WF- SDLC). This system is developed in a way that it will be able to detect the certain real world objects the C-WF-SDLC methodology follows six main phases, that is: Planning, Analysis, Design, Implementation and Maintenance.

5.3 SYSTEM REQUIREMENT PHASE

The purpose of this phase is to determine project's main goal and how the system will function. To gather system requirements information, these are common questions that have to be answered:

Why the system needs to be developed?

- Who are the users?
- How will they use the system?
- What are they using the system for?
- What are the input and output of the system?

This question needs to be answered thoroughly to come up with clear functionality of the system describing the functions that the system should perform. All possible requirements of the system to be developed are captured in this phase. After the requirements are understood Software Requirement Specification (SRS) document is prepared to serve as a guideline for the next phase of the model.

5.4 ANALYSIS PHASE

In this phase analysis of the user's requirement is carried out. This is to determine the scope of the users based on the SRS prepared in the requirement phase and the observations made on the current systems. Thing to be cogitated are

Scope of users

- Purpose of the system
- Information on surveillance systems
- Suitable equipment's (camera, laptop etc.)

5.5 DESIGN PHASE

This is the plan of how the system will look like and how it works. It describes the desired features and operations in detail and may include screen layouts, process diagrams, pseudocode and other documentation. A sample of the project is developed in this phase. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined and the logical system of the product is developed in this phase. It also helps in specifying hardware and system requirements and also helps in defining overall system architecture.

5.6 IMPLEMENTATION AND UNIT TESTING PHASE

This phase is considered to be the longest phase of the software development life cycle. This is so, because this is where the code is created and work is divided into small programs that are referred to as units. This phase include unit testing whereby units will be tested individually for their functionality before the whole system. Unit testing mainly verifies if the modules also known as units meet project specifications.

5.7 TEST PHASE

This is the main testing phase in the SDLC, as the project is divided in small modules in the previous phase then the modules will be integrated together to test the system as whole. This is to make sure that the modules work together as intended by the developer (as in the specifications) and required by users. It also checks for bugs, errors and ensure the system is able to work in the intended platform. After ensuring that the product had solved the problem the system is then delivered to the customers.

5.8 MAINTENANCE AND OPERATION PHASE

Not all problems can be seen directly, but they occur with time and as other problems they need to be solved. Usually these kinds of problems come in picture after the practical use of the system they are never found throughout the development life cycle. This phase of the Waterfall Model is considered to be very long, it never ends. The changes that occur after the product is handed to the users must not affect the main operation of the so a system must be developed in a way that it will adapt to change.

CHAPTER - VII

TESTING

Testing is the process or group of procedures carried out to evaluate some aspect of a piece of software. Testing plays a vital role in the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. Once program code has been developed, testing begins. The minimum aim of testing process is to identify all defects existing in software product.

Testing establishes the software which has attained a specified degree of quality with respect to selected attributes. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals, that is conducted tests to uncover errors and ensure that defined input will procedure actual results that agree with required results. Testing is related with two processes namely Validation and Verification.

6.1 VALIDATION:

Validation is a process of evaluating a software system or component during or at the end of the development cycle in order to determine whether it satisfies specified requirements. It is usually associated with traditional execution-based testing.

6.2 VERIFICATION

Verification is a process of evaluating a software system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. It is associated with activities such as inspection and reviews of the software deliverable.

TYPES OF TESTING

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

6.3 SYSTEM TESTING

System Testing begins at the requirements phase with the development of a master test plan and requirements-based tests. System Testing is more complicated task. It requires large amount of resources. The goal is to ensure that the system performs according to its requirements. It evaluates both functional behavior and quality requirements such as reliability, usability, performance and security. Testing is one of the important steps in the software development phase.

Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

TYPES OF SYSTEM TESTING:

- Functional Testing
- Performance Testing
- Stress Testing
- Configuration Testing
- Security Testing
- Recovery Testing

Functional Testing:

Functional Testing are used to ensure that the behavior of the system to the requirements specification. All functional requirements for the system must be achievable by the system. It focuses on the inputs and proper outputs for each function. Improper and Illegal inputs must also be handles by the system. All functions must be tested.

Some of the goals of functional testing are:

- All types or classes of legal inputs must be accepted by the software.
- All classes of illegal inputs must be rejected.
- All possible classes of the system output must exercise and examined.
- All functions must be exercised.

Performance Testing:

The goal of system performance testing is to see the software meets the performance requirements. Performance Testing allows testers to tune the system; that is to optimize the allocation of system resources. Resources for system testing must be allocated in the system test plan. Results of performance tests are quantifiable. Performance testing requires the test-bed requirement that includes special laboratory equipment and space that must be reserved for the tests.

Test Managers should ascertain the availability of these resources and allocate the necessary time for training in the test plan. Usage requirements for these resources need to be described as part of the test plan.

Stress Testing:

When a system is tested with a load that causes it to allocate its resources in maximum amounts, this is called stress testing. Stress testing is most important because it can reveal defects in real-time and other types of systems, as well as weak areas where poor design could cause unavailability of service. This is particularly important for real-time systems where

unpredictable events may occur resulting in input loads that exceed those described in the requirements documents.

Stress Testing often uncovers race conditions, deadlocks, depletion of resources in unusual or unplanned patterns and upsets in normal operation of the operating system. All these condition are likely to reveal defects and design flaws which may not be revealed under normal testing condition.

Configuration Testing:

Configuration testing allows developers/users to evaluate the system performance and availability when hardware exchanges and reconfigurations occurs. Software system interacts with the hardware devices such as disk drives, tape drives and printers. Many software systems interact with multiple CPUs, some of which are redundant. Software that controls real time processes or embedded software also interfaces with devices but these are very specialized hardware items such as missile launchers and nuclear power device sensors. Several types of operations should be performed during configuration testing.

Security Testing:

Security testing handles safety and security issues for commercial applications for use on the Internet. The Internet users believe that their personal Information is not secure and is available to those with intent to do harm; the future of e-commerce is in peril. Security testing evaluates system characteristics that relate to the scalability, integrity, and confidentially or system data and services. Damages can be done through various means such as viruses, Trojan horses, trap doors, illicit channels.

Some of the effects that cause Security breaches:

- Loss of information
- Corruption of information.
- Misinformation

- Privacy violations.
- Denial of services

Recovery Testing:

Recovery Testing subjects a system to losses of resources in order to determine if it can recover properly from these losses. Multiple CPUs or multiple instances of devices are used to detect the failure of devices. The recovery testers must ensure that the device monitoring system and the check point software are working properly. It focuses on the process of restart and switchover. It can be detected by loss of transaction, merging of transactions, incorrect transactions and unnecessary duplication of a transaction.

6.4 TEST DATA AND OUTPUT

Test data is data which has been specifically identified for use in tests, typically of a computer program. Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result. Other data may be used in order to challenges the ability of the program to respond to unusual, extreme, exceptional or unexpected input.

Test data may be produced in a focused or systematic way (as is typically the case in domain testing) or by using other, less-focused approaches (as is typically the case in high-volume randomized automated tests). Test data may be produced by the tester, or by a program or function that aids the tester. Test data may be recorded for re-use or used once and then forgotten.

6.5 UNIT TESTING

A unit is the smallest possible testable software component. A unit can be function or procedure implemented in a procedural programming languages. A unit may also be a small-sized COTS component purchased from an outside vendor that is undergoing evaluation by the

purchaser, or a simple module retrieved from an in-house reuse library. Unit test results are recorded for future testing process. This result document used for integration and system tests.

Some of the phases for unit test planning are:

- ❖ Describe Unit Test Approach and Risks.
- ❖ Identify Unit features to be tested.
- ❖ Add levels of detail to the test plan.

6.6 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is ―putting them together‖- interfacing. There may be the chances of data lost across on another's sub functions, when combined may not produce the

desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

6.7 TESTING TECHNIQUES / TESTING STRATERGIES

The description of behavior or functionality for the software under test may come from a formal specification, an Input/Process/Output Diagram (IPO), or a well-defined set of pre and post condition. Another source for information is a requirements specification document that usually describes the functionality of the software under test and its inputs and expected outputs.

WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing. White-box testing techniques are as follows:

- **Control-flow testing** - The purpose of the control-flow testing to set up a test case which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- **Data-flow testing** - This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

Black-box testing techniques:

- **Equivalence class** - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- **Boundary values** - The input is divided into higher and lower end values. If these values pass the test, it is assumed that all values in between may pass too.
- **Cause-effect graphing** - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- **Pair-wise Testing** - The behavior of software depends on multiple parameters. In pair wise testing, the multiple parameters are tested pair-wise for their different values.
- **State-based testing** - The system changes state on provision of input. These systems are tested based on their states and input.

6.8 VALIDATION TESTING

Validation testing takes place after integration testing. Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement.

After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specifications and are accepted.
- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus, the proposed system under consideration has been tested by using validation

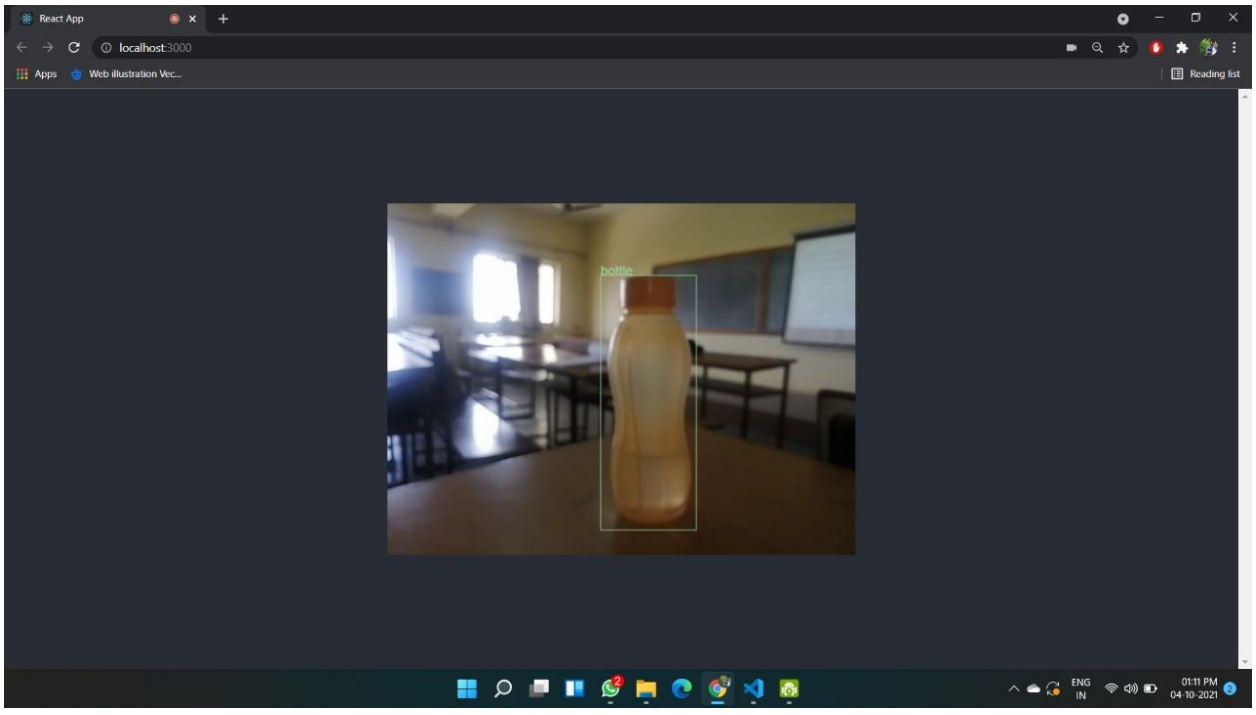
testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic.

6.9 USER ACCEPTANCE TESTING

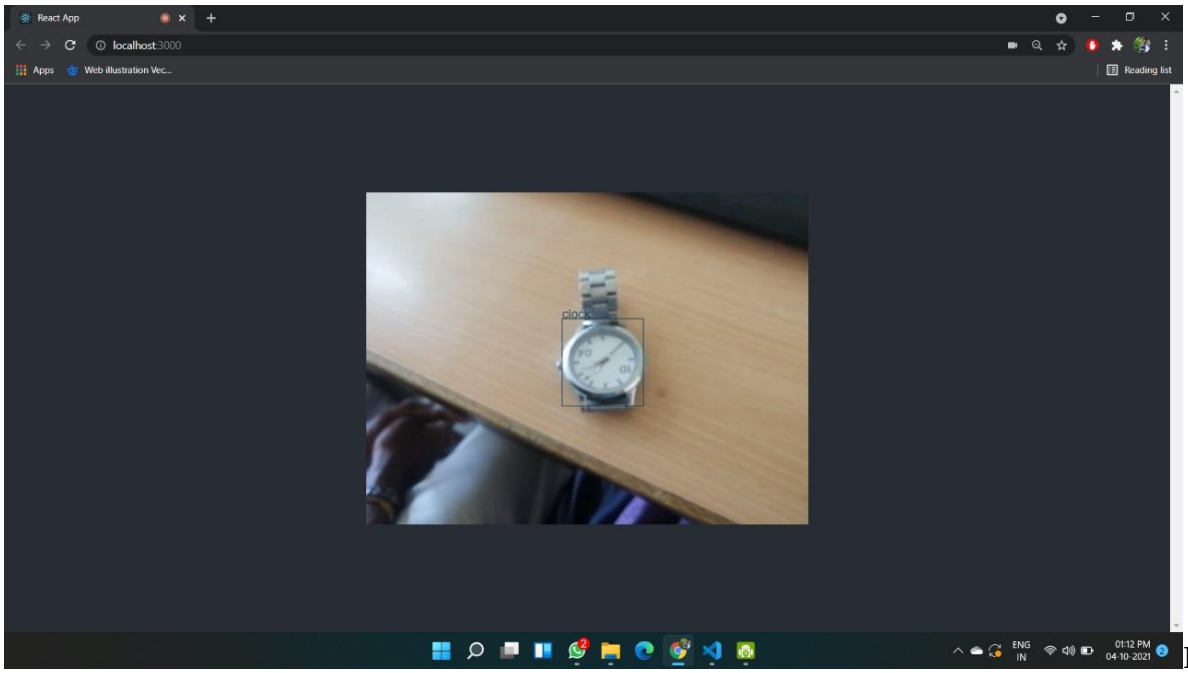
When the Web application is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the Web application matches all user requirements and if user does not like the way it appears or works, it may be rejected.

- **Alpha testing** - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.
- **Beta testing** - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

TESTCASE SUCCESS



TESTCASE FAILURE



E

CHAPTER - VII

SYSTEM IMPLEMENTATION

7.1 INTRODUCTION

System Implementation is the process of bringing developed system into operational use. If the implementation phase is not carefully planned and controlled, it can be lead to many critical problems. Thus proper implementation is essential to provide a reliable system to meet managerial requirements. Implementation is one of the most important tasks in project. Implementation is the phase, in which one has to be cautions, because all the efforts undertaken during the project will be fruitful only if the tool is properly implemented according to the plans made. The implementation phase is less creative than system design. It is primarily concerned with user training; site preparation and file-sites, the test of the network along with the system are also included under implementation. Depending on the nature of the system extensive user training maybe required programming is itself a design works. The initial parameters of the management information system should be modified as a result of programming efforts. Programming provides a real test for the assumption made by the analyst.

Implementation is used here to mean the process of converting a new or revised system into an operation one. Here the new system is implemented to an operational use. Maintenance is far more than fixing mistakes. The maintenance can be defined using four activities that are undertaken after a program is released for use.

The second activities that contribute to a definition of maintenance occurs because of the rapid change that encountered in every aspect of computing. Adaptive maintenance – as activity that modifies software to properly interface with a changing environment – is both necessary and common place.

The third activity that may be applied to definition of maintenance occurs when software package is successful. As the software is used new recommendations for new capabilities, modifications to existing function and general enhancements are received from the user, to satisfy this request perceptive maintenance is used. The fourth maintenance activity occurs when software is changed to improve future maintainability or reliability, or to provide a better basic

for future enhancements. This is often called preventive maintenance, which is characterized by reverse engineering and re-engineering technique.

7.2 IMPLEMENTATION

The system has been tested in the location of the developer. But it is not possible to find all errors here. It may be that even after through testing the user will find errors. In such a case the user when reports the errors it is possible to correct those errors as that coding has been documents and it is possible to find out the location where the error is occurring and the reason for error can be analyzed and corrected. This developed system supports for corrective maintenance.

As this software can be run with the requirements given above and it does not involve any particular hardware as such and it can be run with the rapid development that is being encountered in the computer industry.

If there is a need to include any new modules then it has been externally and then includes to it with the exits architecture. But up to now the system holds all the possible reports generation tools, which a team needs. Later on it correspondence with this limitation may upgrade the system.

CHAPTER – VIII

PERFORMANCE AND LIMITATIONS

8.1. MERITS OF THE SYSTEM

- To identify and locate one or more effective targets from video data.
- Person detection is a variant of object detection used to detect a primary class "person" in images or video frames.
- Detecting people in video streams is an important task in modern video surveillance systems.

8.2. LIMITATIONS OF THE SYSTEM

- Viewpoint variation. One of the biggest difficulties of object detection is that an object viewed from different angles may look completely different.
- Deformation.
- Occlusion
- Illumination conditions.

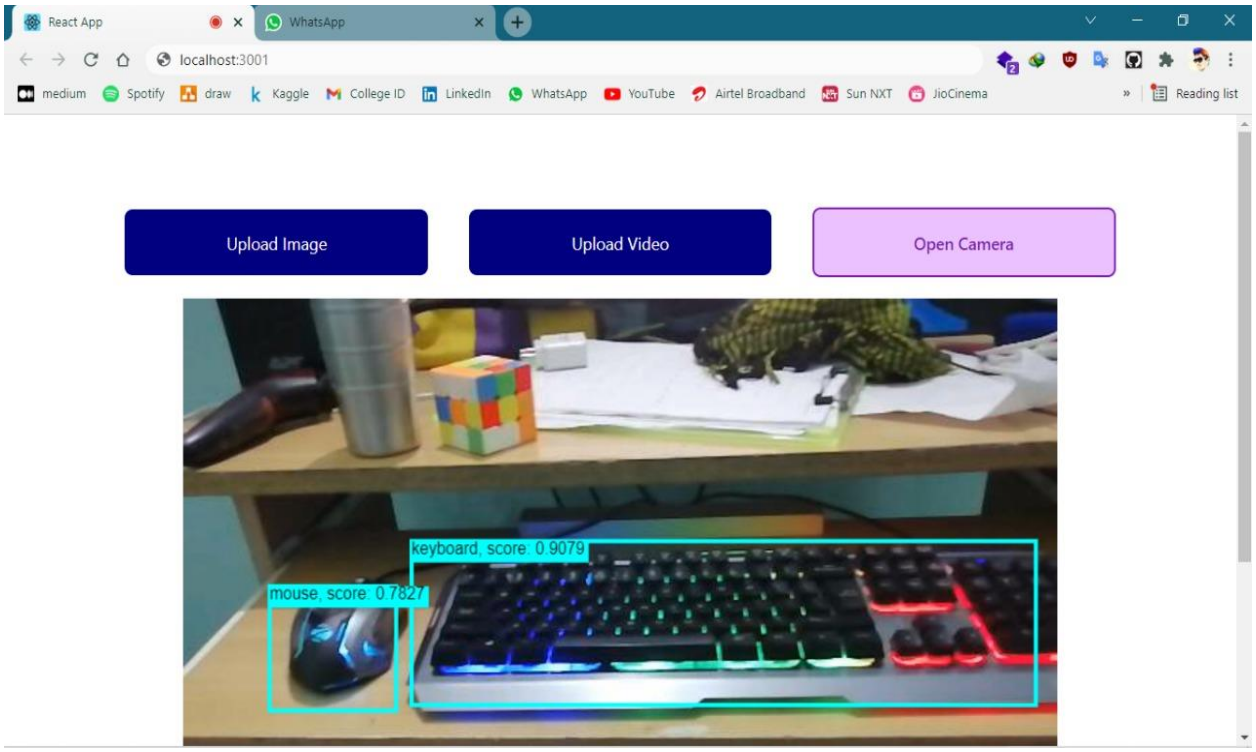
8.3. FUTURE ENHANCEMENTS

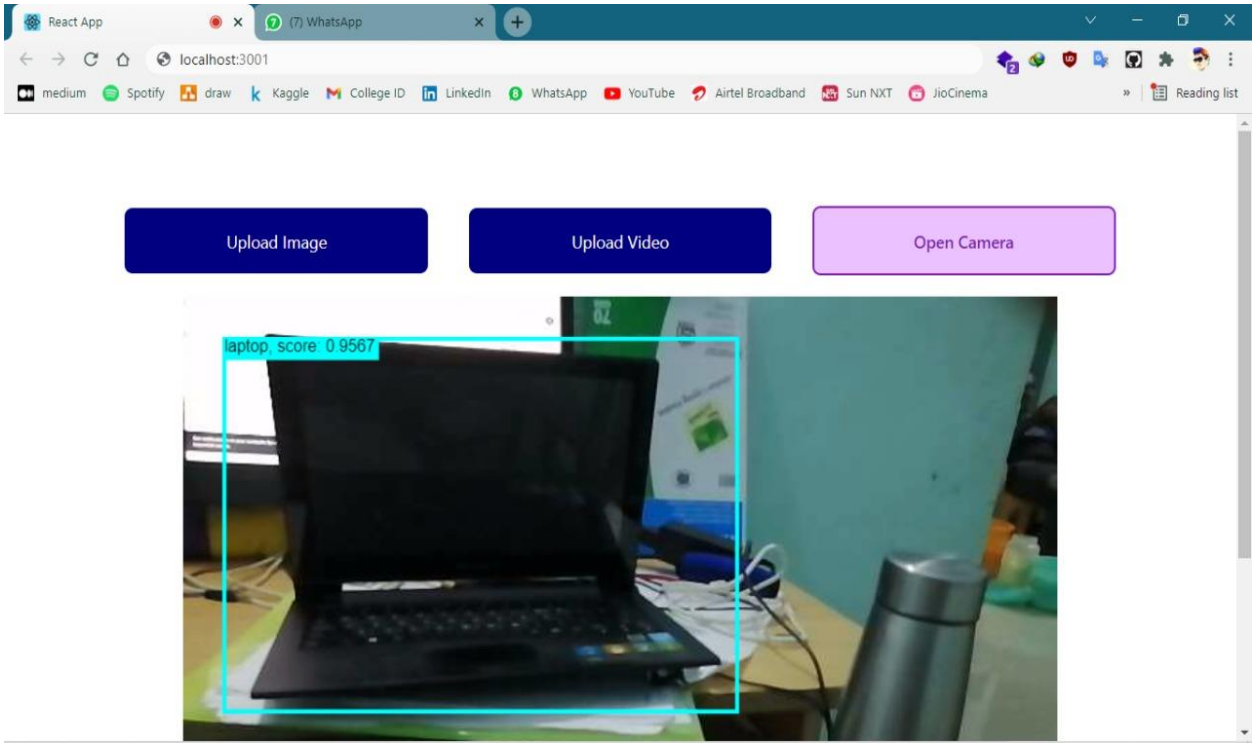
The future of object detection technology is in the process of proving itself, and much like the original Industrial Revolution, it has the potential, at the very least, to free people from tedious jobs that will be done more efficiently and effectively by machines. It will also open up new avenues of research and operations that will reap additional benefits in the future.

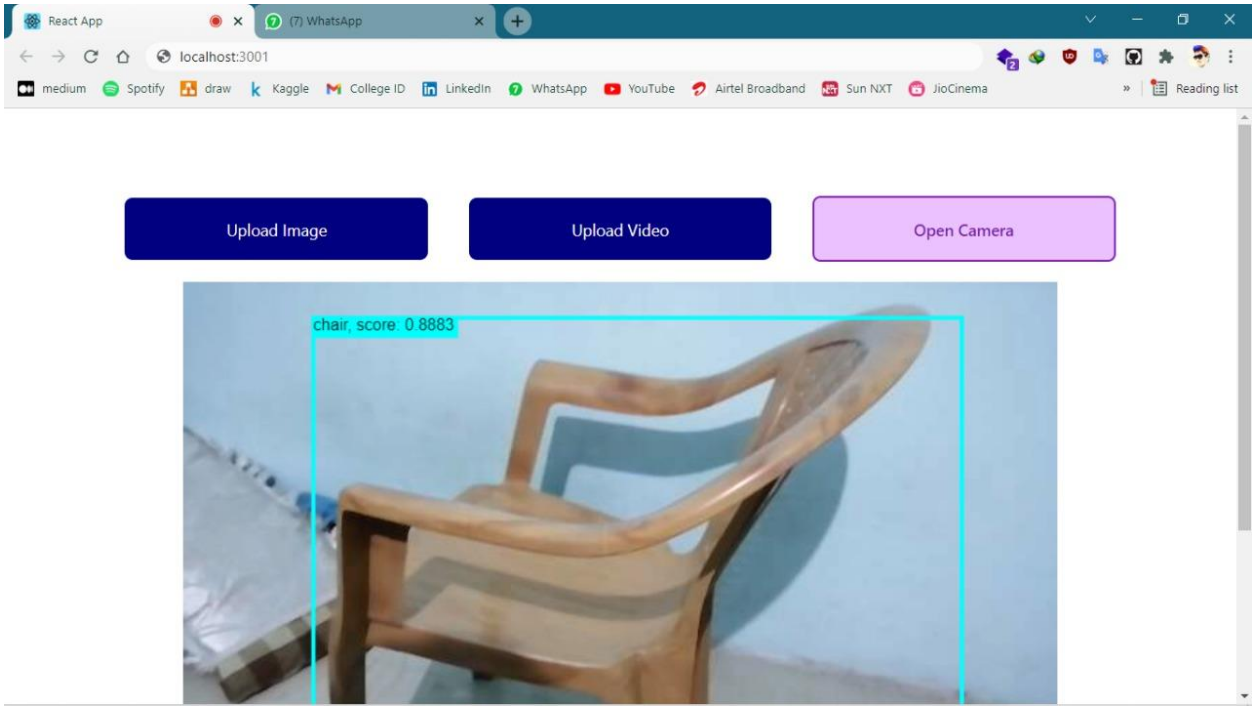
CHAPTER IX

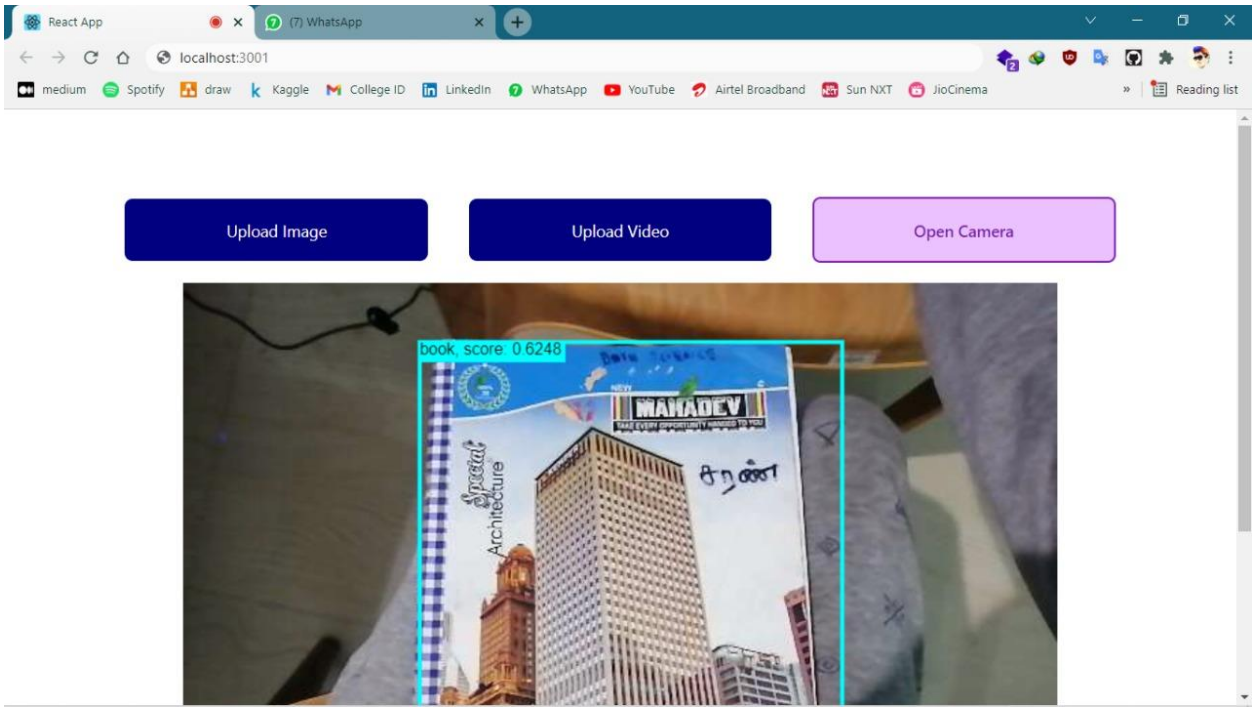
APPENDICES

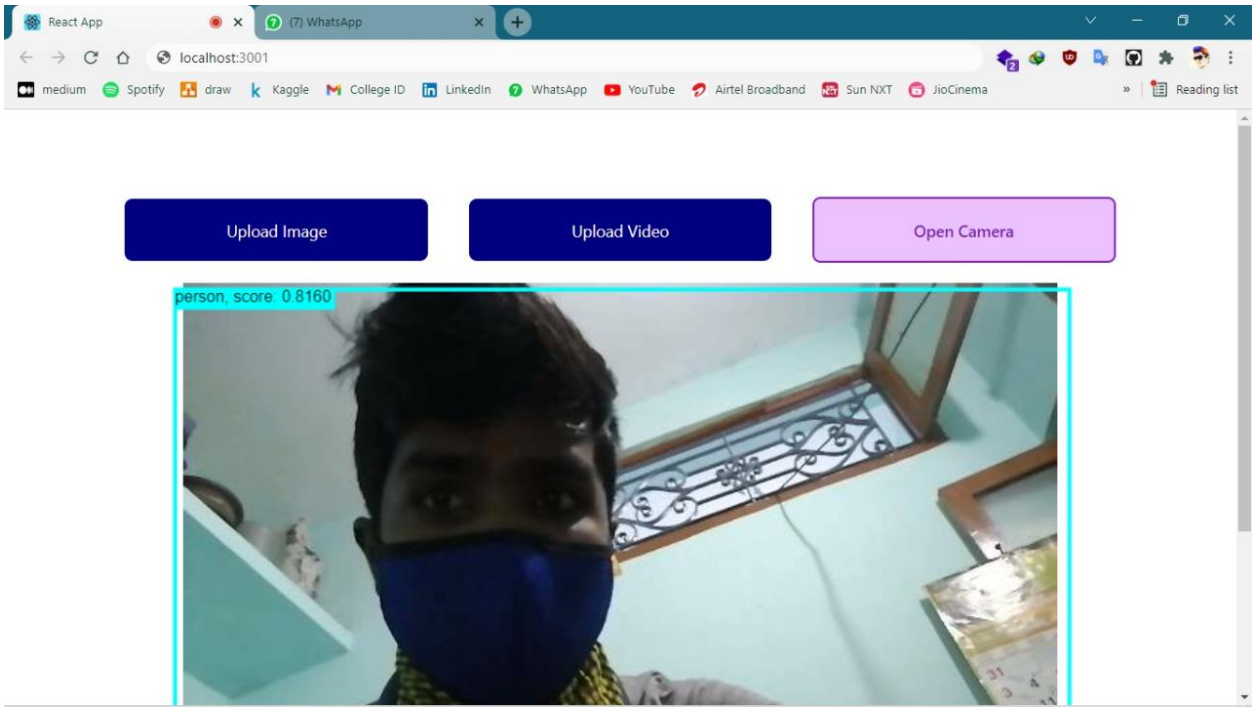
9.1 SAMPLE SCREEN SHOTS

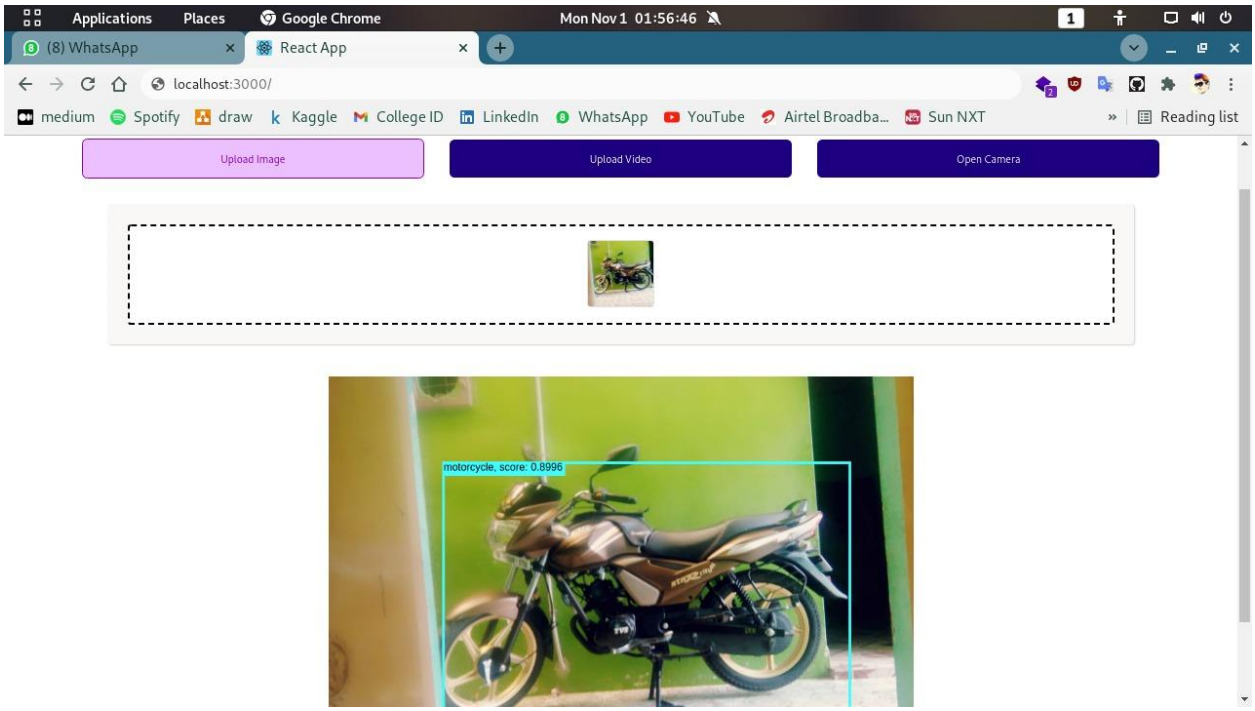



React App
(7) WhatsApp
localhost:3001
medium
Spotify
draw
Kaggle
College ID
LinkedIn
WhatsApp
YouTube
Airtel Broadband
Sun NXT
JioCinema
Reading list
Upload Image
Upload Video
Open Camera
laptop, score: 0.9567


React App
(7) WhatsApp
localhost:3001
medium
Spotify
draw
Kaggle
College ID
LinkedIn
WhatsApp
YouTube
Airtel Broadband
Sun NXT
JioCinema
Reading list
Upload Image
Upload Video
Open Camera
chair, score: 0.8883


React App
(7) WhatsApp
localhost:3001
medium
Spotify
draw
Kaggle
College ID
LinkedIn
WhatsApp
YouTube
Airtel Broadband
Sun NXT
JioCinema
Reading list
Upload Image
Upload Video
Open Camera
book, score: 0.6248
MAHADEV


React App
(7) WhatsApp
localhost:3001
medium
Spotify
draw
Kaggle
College ID
LinkedIn
WhatsApp
YouTube
Airtel Broadband
Sun NXT
JioCinema
Reading list
Upload Image
Upload Video
Open Camera
person, score: 0.8160


Applications
Places
Google Chrome
Mon Nov 1 01:56:46
(8) WhatsApp
React App
localhost:3000/
medium
Spotify
draw
Kaggle
College ID
LinkedIn
WhatsApp
YouTube
Airtel Broadba...
Sun NXT
Reading list
Upload Image
Upload Video
Open Camera
motorcycle, score: 0.8996

9.2 CONCLUSION

Object detection is a key ability for most computer and robot vision system. Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smartphones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning. It should be noted that object detection has not been used much in many areas where it could be of great help. As mobile robots, and in general autonomous machines, are starting to be more widely deployed (e.g., quad-copters, drones and soon service robots), the need of object detection systems is gaining more importance. Finally, we need to consider that we will need object detection systems for nano-robots or for robots that will explore areas that have not been seen by humans, such as depth parts of the sea or other planets, and the detection systems will have to learn to new object classes as they are encountered. In such cases, a real-time open-world learning ability will be critical.

CHAPTER X

REFERANCE

WEB REFERANCE

ReactJS:- https://reactjs.org/docs/getting-started.html

TensorflowJS:- https://www.tensorflow.org/js

Javascript:- https://developer.mozilla.org/en-US/docs/Web/JavaScript