A MINI PROJECT REPORT ON

"VIRTUAL PAINTER"

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

B.Anitha	G.Mohini	N.Jagruthi
19911A12C4	19911A12D1	19911A12F5

Under the Esteemed Guidance of

Mrs. G. INDIRA PRIYADARSHINI

Assoc.Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

(Accredited by NBA, Approved by AICTE, Affiliated to JNTU Hyderabad)

Aziz Nagar Gate, C.B.Post, Chilkur Road, Hyderabad – 500075

2022 -2023.

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

(Accredited by NBA, Approved by AICTE, Affiliated to JNTU Hyderabad)

Aziz Nagar Gate, C.B.Post, Chilkur Road, Hyderabad - 500075

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the Project Report on **"VIRTUAL PAINTER"** is a bonafide work by **B.Anitha(19911A12C4), G.Mohini (19911A12D1) and N.Jagruthi (19911A12F5)** in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **"INFORMATION TECHNOLOGY"** JNTU Hyderabad during the year **2022 – 2023.**

Project Guide	Head of the department
Mrs. G.Indira Priyadarshini	Mr. B. Srinivasulu ,
M.Tech,	M.E,
Associate Professor,	Professor,

External Examiner

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

(Accredited by NBA, Approved by AICTE, Affiliated to JNTU Hyderabad)

Aziz Nagar Gate, C.B.Post, Chilkur Road, Hyderabad - 500075

DEPARTMENT OF INFORMATION TECHNOLOGY

DECLARATION

We, **B.Anitha (19911A12C4), G.Mohini (19911A12D1) and N.Jagruthi (19911A12F5)** hereby declare that Project Report entitled "VIRTUAL PAINTER", is submitted in the partial fulfilment of the requirement for the award of **Bachelor of Technology** in "**Information Technology"** to **Vidya Jyothi Institute of Technology**, affiliated to JNTU - Hyderabad, is an authentic work and has not been submitted to any other university or institute for the degree.

B.Anitha (19911A12C4)

G.Mohini (19911A12D1)

N.Jagruthi (19911A12F5)

ACKNOWLEDGEMENT

It is a great pleasure to express our deepest sense of gratitude and indebtedness to our internal guid **Mrs. G.Indira Priyadarshini**, Associate Professor, Department of IT, VJIT, for having been a source of constant inspiration, precious guidance and generous assistance during the project work. We deem it as a privilege to have worked under her guidance. Without her close monitoring and valuable suggestions this work would not have taken this shape. We feel that this help is un-substitutable and unforgettable.

We wish to express our sincere thanks **to Dr. E. Sai Baba**, Director VJIT, for providing the college facilities for the completion of the project. We are profoundly thankful to **Mr. B.Srinivasulu**, Professor and Head of Department of IT, for his cooperation and encouragement. Finally, we thank all the faculty members, supporting staff of IT Department and friends for their kind co-operation and valuable help for completing the project.

TEAM MEMBERS

B.Anitha(19911A12C4)

G.Mohini (19911A12D1)

N.Jagruthi (19911A12F5)

TABLE OF CONTENTS

ABSTRACT

Virtual Painter

Wouldn't it be cool if you could just wave a pen in the air to draw something virtually and it draws it on the screen? It could be even more interesting if we did not use any special hardware to achieve this, just plain simple computer vision would do, in fact, we wouldn't even need to use deep learning to achieve this. In this project, we are going to create a virtual painter using Open CV. We will first track our hand and get its landmarks and then use the points to draw on the screen. We will use two fingers for selection and one finger for drawing. And the best part is that all of this will be done in real-time. This can be used for quick explanation during office meetings or during online education classes.

Keywords: python, media pipe, computer vision, numpy, hand gesture recognition.

1.INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT:

The old approach for using the existing system for drawing or writing is done by using the expensive hardware such as stylus pens and it requires more battery power and particular system compatibility

1.2 OBJECTIVE:

The purpose it to create an interesting platform for writing, drawing etc. We can also minimize the usage of hardware components; it also saves our time and it is a kind of fun activity to kids. In the present circumstances, digital art and traditional art are inclusive of the symbiotic state, so we need to systematically understand the basic knowledge of the form between digital art and traditional art. The traditional way includes pen and paper, chalk and board method of writing. The essential aim of digital art is of building hand gesture recognition system to write digitally. Here we aim to create an external hardware free virtual painter that is easy to use. This also helps dumb people to do the things so easily. Here we just need to use our hand to draw or write we need to use our two fingers to select a color one finger to draw. we will have multiple colors we can select as we required also there will be an eraser incase if you go wrong.

1.3 SCOPE OF PROJECT:

The scope of the project is the Digital art includes many ways of writing like by using keyboard, touch-screen surface, digital pen, stylus, using electronic hand gloves, etc. But in this system, we are using hand gesture recognition with the use of machine learning algorithm by using python programming, which creates natural interaction between man and machine. With the advancement in technology, the need of development of natural 'human – computer interaction (HCI)' [10] systems to replace traditional systems is increasing rapidly.

1.4 MODULES:

1.Creating frames

Start reading the frames and convert the captured frames to HSV colour space (Easy for colour detection). Prepare the canvas frame and put the respective ink buttons on it. Adjust the values of the mediapipe intilization to detect the hand and get its landmarks through the RGB frame to the mediapipe and then it finds the forefinger coordinates and keep storing them in the array for successive frames. Finally draw the points stored in array on the frames and canvas.

2.Hand detection and fingertip:

When we show our hand to the webcam it starts detecting hand and trace all the landmarks. It detects total 21 points on the hand. Out of 21 points it uses only 6 points for the process. Later it detects the fingertip for drawing or writing purpose.

3.Drawing and clear:

We will use two fingers such as middle finger and the index finger for selecting option from the selection box. Here we use index finger for drawing or writing and for selecting the color and deleting the work we use both fingers.

Algorithm:

Step1: To implement Virtual Painter we are using OpenCV with ML algorithm using mediapipe.

Step2: Later, start creating the frames and convert the captured frames to HSV color.

Step3: Create the canvas frame and then we will put the respective ink buttons on it.

Step4: Later we will run the file and it turn on the webcam.

Step5: When we show our hand to the webcam it starts detecting hand and trace all the landmarks.

Step6: By using fingers we can select color for drawing or writing and clearing.

Step7: Index finger and middle finger are used to select the options from the seletion box and index finger is used to draw or write.

2. SYSTEM ANALYSIS

CHAPTER 2

2.1. HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements:

Python-3.x (We used 3.8.8 for this project)

OpenCV-0.8.11

Mediapipe-0.8.13

Numpy-1.20.1

HARDWARE REQUIREMENTS:

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hadware requirements include configuration characteristics.

System	: Pentium/V2.4GHz.
Hard Disk	: 100 GB.
Monitor	: 15 VGA Colour.
Mouse	: Logitech.

RAM : 1 GB.

2.2 SOFTWARE REQUIREMENT SPECIFICATIONS:

Hand Detection is a python software which is designed to detect hand movements and patterns. This can help to increase or decrease volume and brightness. Open cv in our project is used to install the packages Here we also use NumPy for better quality of our project.

2.3 EXISTING SYSTEM:

We use touch screen pens (stylus) for drawing, which are more expensive and also pairing required for full functionality. This system requires a good battery power and it requires a particular system compatibility.

Drawbacks of Existing system:

1. This system requires a good battery power.
2. It requires a particular system compatibility.
3. It uses hardware components.

2.4 PROPOSED SYSTEM:

With the use of virtual painter, we don't need any special hardware requirements for functioning just your fingers are enough, no charging required and compatible with any system. Virtual Painter system looks like an user interface actually consists of a work space called as whiteboard that comes along with tools such as RGB color box and clear and no need of other additional devices. This is the simple and easy way to write and draws with figures gesture. This is an application that is developed in python with the help of opencv.

Advantages of Proposed system:

1. No charging is required.
2. Compatible with any system.
3. No external Hardware is required.
4. Anyone can easily use such as dumb, deaf people etc.

3.TECHNOLOGIES USED

CHAPTER-3

3.1 PYTHON:

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

Python concepts

If you not interested in the how and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it is a great one to start programming with.

- Open-source general-purpose language.
- Object Oriented, Procedural, Functional.
- Easy to interface with C/Object/Java/Fortran.
- Easy-ish to interface with C++ (via SWIG).
- Great interactive environment.

Python is a high-level, interpreted, interactive and object-oriented scripting language.

Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginnerlevel programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python Features

Python's features include −

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

3.2 NUMPY

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank.

- Offers Matlab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra etc.

POWERFUL N-DIMENSIONAL ARRAYS:

- Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS:

- NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

INTEROPERABLE:

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT:

• The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE:

• NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

OPEN SOURCE:

• Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

3.3 MEDIAPIPE

MediaPipe powers revolutionary products and services we use daily. Unlike power-hungry machine learning Frameworks, Media Pipe requires minimal resources. It is so tiny and efficient that even embedded IoT devices can run it. In 2019, MediaPipe opened a whole new world of opportunity for researchers and developers following its public release. If you have just started with MediaPipe and this is one of the first articles you are going through, congratulations, you are in the right 10 place. This article will cover the basics of Media Pipe, the difference between Solutions, and the Framework. The official documentation states that inferencing is real-time, and it takes just a few lines of code to create a perception pipeline. Is it that simple? Does it provide real-time output for any kind of data we throw at it? Follow along the article to know more. MediaPipe Toolkit comprises the Framework and the Solutions. The following diagram shows the components of the MediaPipe Toolkit.

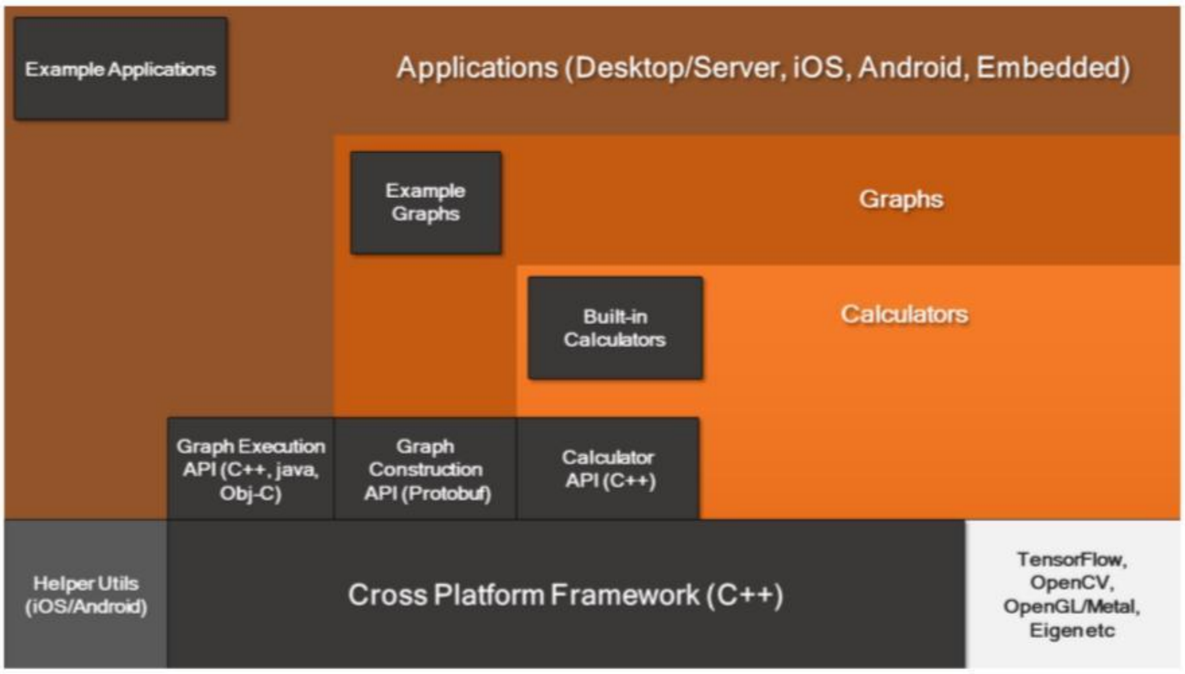

Example Applications
Applications (Desktop/Server, iOS, Android, Embedded)
Example Graphs
Graphs
Built-in Calculators
Calculators
Graph Execution API (C++, java, Obj-C)
Graph Construction API (Protobuf)
Calculator API (C++)
Helper Utils (iOS/Android)
Cross Platform Framework (C++)
TensorFlow, OpenCV, OpenGL/Metal, Eigen etc

FIGURE: 3.3.1. MEDIAPIPE TOOL KIT

MediaPipe offers ready-to-use yet customizable Python solutions as a prebuilt Python package. MediaPipe Python package is available on PyPI for Linux, macOS and Windows.

3.4 COMPUTER VISION

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do. Computer vision tasks include methods for acquiring, processing, analysing, and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that make sense to thought processes and can elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory. The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multidimensional data from a 3D scanner, or medical scanning device. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems. Sub-domains of computer vision include scene reconstruction, object detection, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual serving, 3D scene modelling, and image restoration.

4. SYSTEM ANALYSIS

&

UML DESIGN

CHAPTER - 4

SYSTEM DESIGN & UML DIAGRAMS

4.1 System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements.

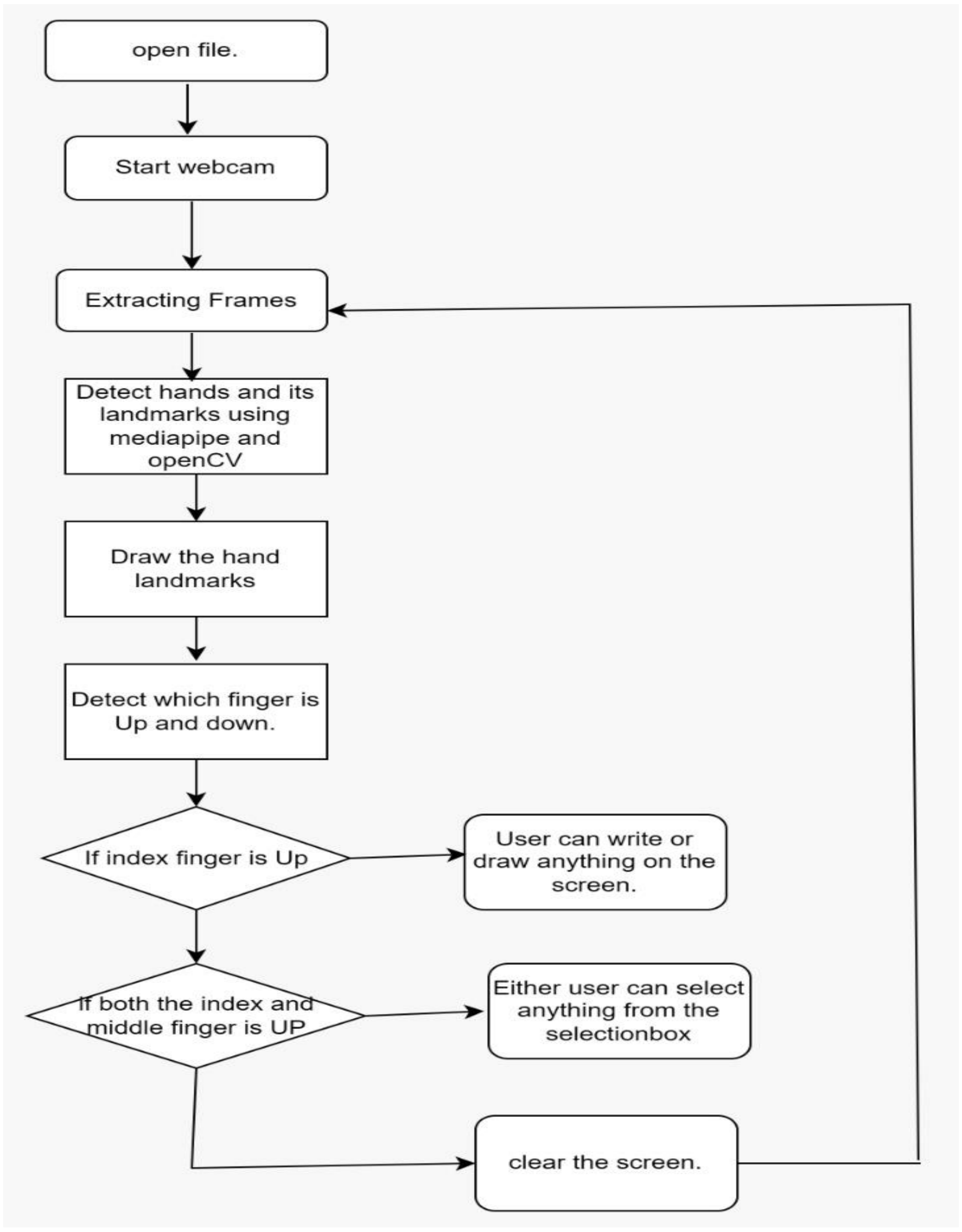
The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

Elements of a System

1. **Architecture -** This is the conceptual model that defines the structure, behaviour, and more views of a system. We can use flowcharts to represent and illustrate the architecture.
2. **Modules** - This are components that handle one specific tasks in a system. A combination of the modules makes up the system.
3. **Components -** This provides a particular function or group of related functions. They are made up of modules.
4. **Interfaces -** This is the shared boundary across which the components of the system exchange information and relate.
5. **Data -** This the management of the information and data flow.

4.2 System Flowchart:

This is the conceptual model that defines the structure, behavior and more views of a system. We can use flowcharts to represent and illustrate the architecture


open file.
Start webcam
Extracting Frames
Detect hands and its landmarks using mediapipe and openCV
Draw the hand landmarks
Detect which finger is Up and down.
If index finger is Up
User can write or draw anything on the screen.
If both the index and middle finger is UP
Either user can select anything from the selectionbox
clear the screen.

4.2.1 Flow chart of system

4.3 SOFTWARE DESIGN:

The design concepts provide the software designer with a foundation from which more sophisticated methods can be applied. A set of fundamental design concepts has evolved.

They are as follows:

1. Abstraction - Abstraction is the process or result of generalization by reducing the information content of a concept or an observable phenomenon, typically in order to retain only information which is relevant for a particular purpose. It is an act of Representing essential features without including the background details or explanations.

2**. Refinement -** It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement of function in a step-wise fashion until programming language statements are reached. In each step, one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and Refinement are complementary concepts.

3. Modularity - Software architecture is divided into components called modules

4. Software Architecture - It refers to the overall structure of the software and the ways in which that structure provides conceptual integrity for a system. Good software architecture will yield a good return on 14 investments with respect to the desired outcome of the project, e.g., in terms of performance, quality, schedule and cost.

5. Control Hierarchy - A program structure that represents the organization of a program component and implies a hierarchy of control.

6. Structural Partitioning - The program structure can be divided into both horizontally and vertically. Horizontal partitions define separate branches of modular hierarchy for each major program function. Vertical partitioning suggests that control and work should be distributed top down in the program structure.

7. Data Structure - It is a representation of the logical relationship among individual elements of data.

8. Software Procedure - It focuses on the processing of each module individually.

9. Information Hiding - Modules should be specified and designed so that information contained within a module is inaccessible to other modules that have no need for such information.

AI painter user for who the application looks like an user interface actually consists of a work space called as whiteboard that comes along with different tools such as different pen tools and eraser and no need of other additional devices This is the simple and easy way to write and draws with figures gesture. This is an application that is developed in python with the help of OpenCV and hence all its features apply here as well such as platform independence.

FIGURE: 4.4 UML DIAGRAM

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

Class diagram:

Identification of analysis classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items.

There are 4 approaches for identifying classes:

a. Noun phrase approach.
b. Common class pattern approach.
c. Use case Driven Sequence or Collaboration approach.
d. Classes, Responsibilities and collaborators Approach

a. Noun Phrase Approach:

The guidelines for identifying the classes:

- Look for nouns and noun phrases in the usecases.
- Some classes are implicit or taken from general knowledge.
- All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.
- Carefully choose and define the class names After identifying the classes we have to eliminate the following types of classes:
- Adjective classes.

b. Common class pattern approach:

The following are the patterns for finding the candidate classes:

- Concept class.

- Events class.
- Organization class
- Peoples class
- Places class
- Tangible things and devices class.

c. Use case driven approach:

We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform those functionalities.

d. CRC approach:

The process consists of the following steps:

- Identify classes' responsibilities (and identify the classes)
- Assign the responsibilities
- Identify the collaborators.

Identification of responsibilities of each class:

The questions that should be answered to identify the attributes and methods of a class respectively are:

a. What information about an object should we keep track of?
b. What services must a class provide?

Identification of relationships among the classes:

Three types of relationships among the objects are:

Association: How objects are associated?

Super-sub structure: How are objects organized into super classes and sub classes? **Aggregation:** What is the composition of the complex classes?

Association:

The questions that will help us to identify the associations are:

a. Is the class capable of fulfilling the required task by itself?
b. If not, what does it need?
c. From what other classes can it acquire what it needs?

Guidelines for identifying the tentative associations:

- ● A dependency between two or more classes may be an association. Associationoften corresponds to a verb or prepositional phrase.
- ● A reference from one class to another is an association. Some associations are implicit or taken from general knowledge.

Some common association patterns are:

Location association like part of, next to, contained in….

Communication association like talk to, order to ……

We have to eliminate the unnecessary association like implementation associations, ternary or n- ary associations and derived associations.

Super-sub class relationships:

Super-sub class hierarchy is a relationship between classes where one class is the parent class of another class (derived class). This is based on inheritance.

Guidelines for identifying the super-sub relationship, a generalization are:

1. Top-down: Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub classes have significant behaviour.

2. Bottom-up: Look for classes with similar attributes or methods. Group them by moving the common attributes and methods to an abstract class. You may have to alter the definitions a bit.

3. Reusability: Move the attributes and methods as high as possible in the hierarchy.

4. Multiple inheritances: Avoid excessive use of multiple inheritances. One way of getting benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class as an attribute.

Aggregation or a-part-of relationship:

It represents the situation where a class consists of several component classes. A class that is composed of other classes doesn't behave like its parts. It behaves very difficultly. The major properties of this relationship are transitivity and anti-symmetry.

The questions whose answers will determine the distinction between the part and whole relationships are:

- Does the part class belong to the problem domain?
- Is the part class within the system's responsibilities?
- Does the part class capture more than a single value? (If not then simply include it as an attribute of the whole class)
- Does it provide a useful abstraction in dealing with the problem domain?

There are three types of aggregation relationships.

They are:

Assembly:

It is constructed from its parts and an assembly-part situation physically exists.

Container:

A physical whole encompasses but is not constructed from physical parts.

Collection member:

A conceptual whole encompasses parts that may be physical or conceptual. The container and collection are represented by hollow diamonds but composition is represented by solid diamond.

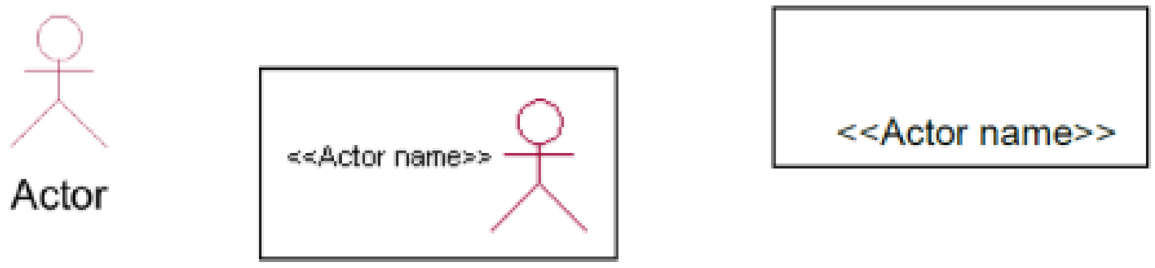
Global Use Case Diagrams:

Identification of actors:

Actor:

Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

Questions to identify actors:

1. Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
2. Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
3. Which external hardware or systems (if any) use the system to perform tasks?
4. What problems does this application solve (that is, for whom)?
5. And, finally, how do users use the system (use case)? What are they doing with the system?

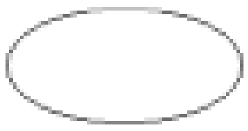
The actors identified in this system are:

a. System Administrator
b. Customer
c. Customer Care

Identification of use cases:

1. Use case: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

● Pattern of behaviour the system exhibits

● A sequence of related transactions performed by an actor and the system

● Delivering something of value to the actor.

Use cases provide a means to:

● capture system requirements

● communicate with the end users and domain experts

● test the system Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

● For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal.

● Name the use cases.

● Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous Questions to identify use cases:

● What are the tasks of each actor?

● Will any actor create, store, change, remove or read information in the system?

● What use case will store, change, remove or read this information?

- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What use cases will support and maintains the system?

Flow of Events :

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

Construction of Use case diagrams:

Use-case diagrams graphically depict system behavior (use cases).

These diagrams present a high-level view of how the system is used as viewed from an outsider's (actor's) perspective.

A use- case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

- actors ("things" outside the system)
- use cases (system boundaries identifying what the system should do)
- Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Relationships in use cases:

1. Communication: The communication relationship of an actor in a use case is shown by connecting the actor symbol to the use case symbol with a solid path. The actor is said to communicate with the use case.
2. Uses: A Uses relationship between the use cases is shown by generalization arrow from the use case.
3. Extends: The extend relationship is used when we have one use case that is similar to another use case but does a bit more. In essence it is like subclass.

4.4.1 USECASE DIAGRAMS

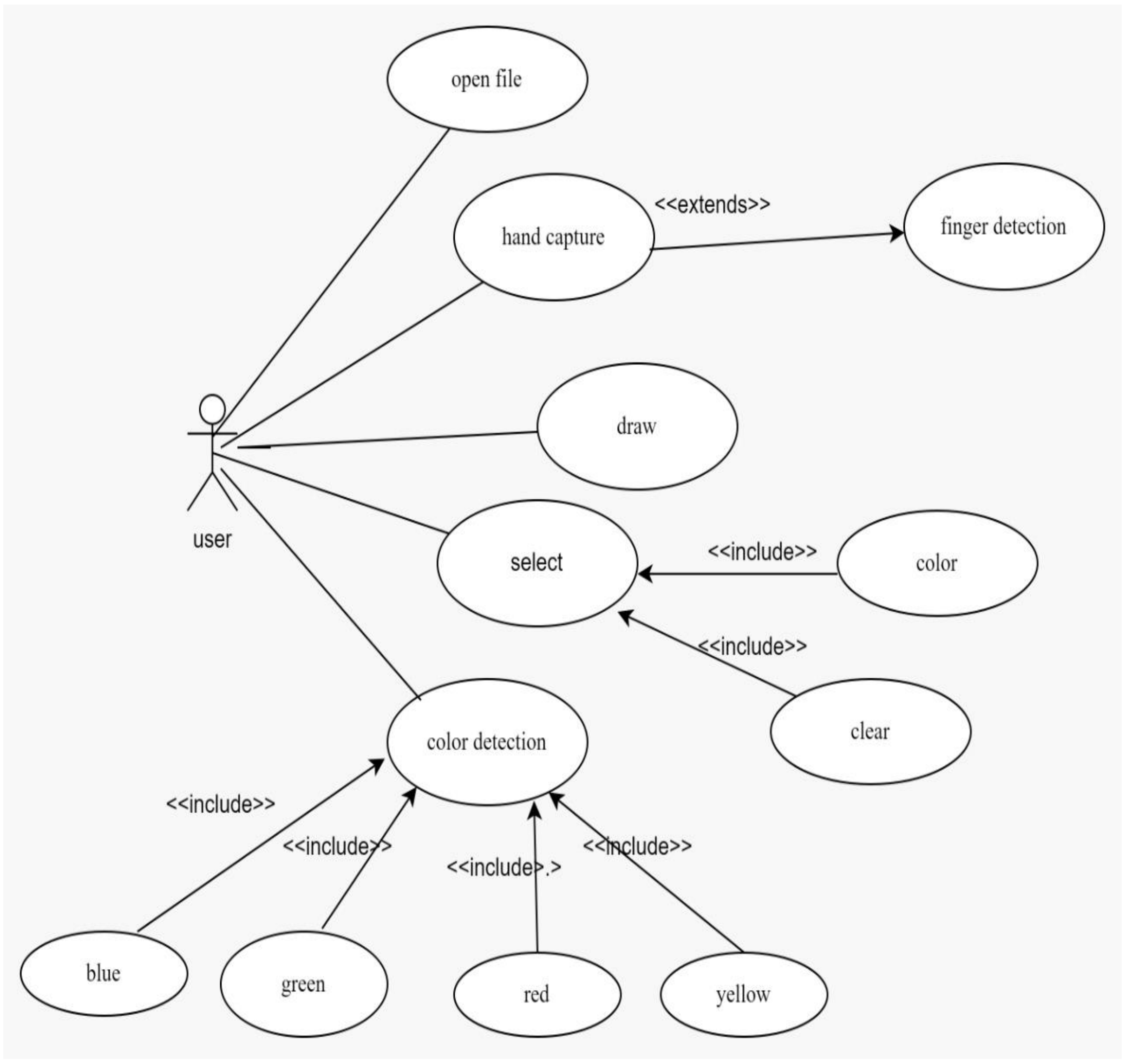
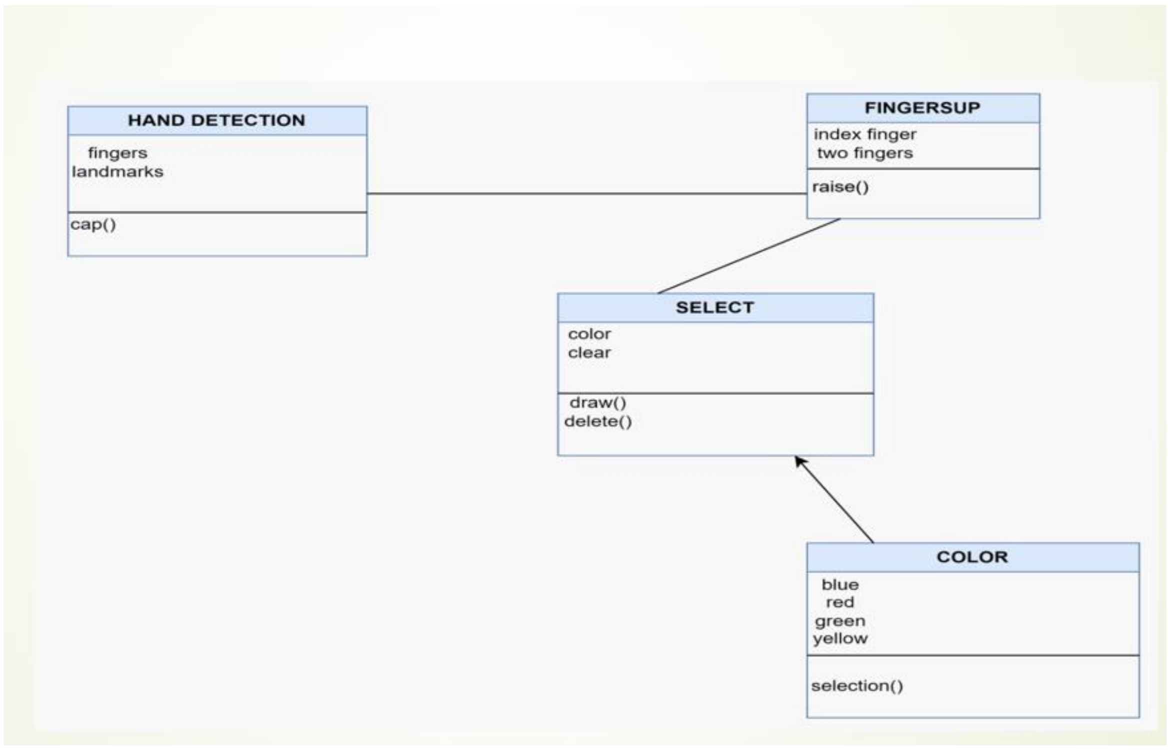

open file
hand capture
<<extends>>
finger detection
user
draw
select
<<include>>
color
<<include>>
clear
color detection
<<include>>
<<include>>
<<include>.>
<<include>>
blue
green
red
yellow

FIG.4.4.2.CLASS DIAGRAM



5.INPUT / OUTOUT DESIGN

CHAPTER – 5

INPUT/OUTPUT DESIGN

5.1 INPUT DESIGN:

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So, inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations. Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

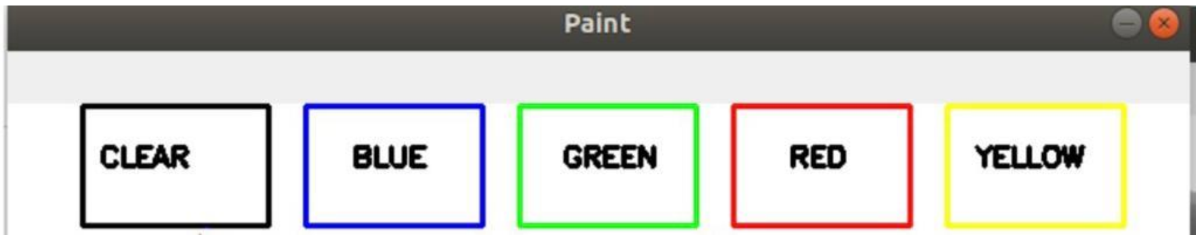


FIGURE 5.1.1 SELECTION BOX

5.2 OUTPUT DESIGN:

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity, and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself.

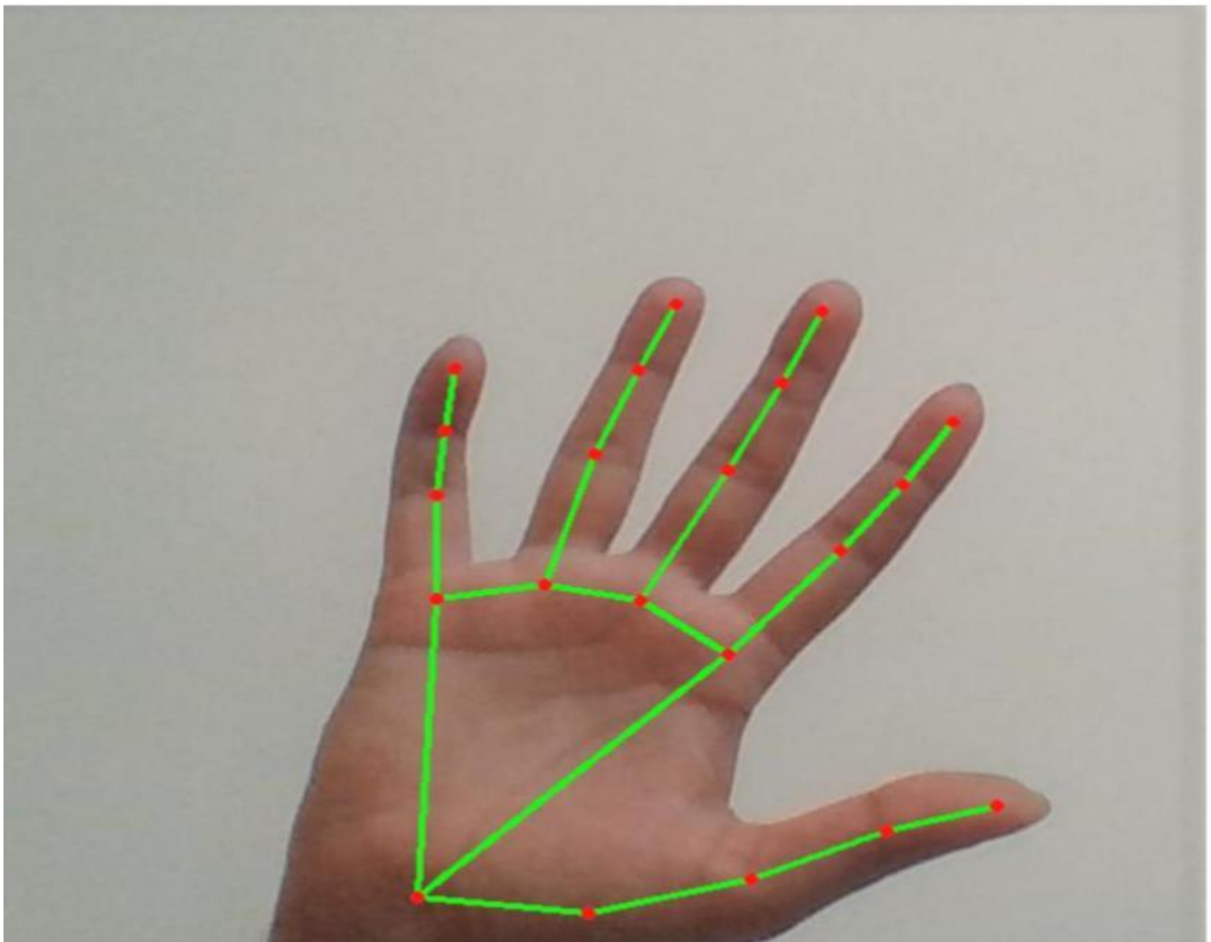


FIGURE:5.2.1 HAND DETECTION

6.IMPLEMENTATION AND TEST CASES

CHAPTER – 6

IMPLEMENTATION

6.1 IMPLEMENTATION:

The most exciting part of this system is the workflow phase. Writing involves a lot of functionalities. So, the number of gestures used for controlling this system is equal to these number of actions involved. The basic functionalities that are included in this system are:

1. Writing Mode - In this state, the system will trace the fingertip coordinates and stores them.

2. Colour Mode – The user can change the color of the text among the various available colors.

3. Backspace - Say if the user goes wrong, we need a gesture to add quick backspace.

```
    # All the imports go here
import cv2
import numpy as np
import mediapipe as mp
from collections import deque




# Giving different arrays to handle colour points of different colour
bpoints = [deque(maxlen=1024)]
gpoints = [deque(maxlen=1024)]
```

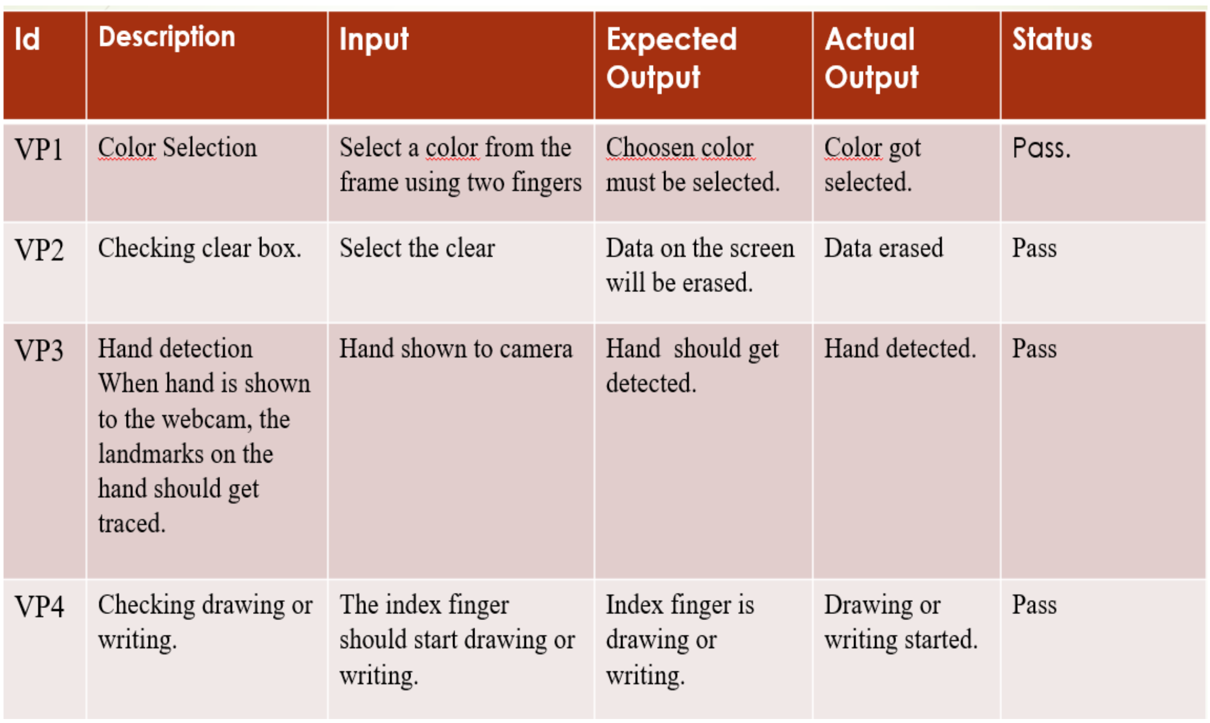
```
rpoints = [deque(maxlen=1024)]

ypoints = [deque(maxlen=1024)]




# These indexes will be used to mark the points in particular arrays of specific
colour

blue_index = 0

green_index = 0

red_index = 0

yellow_index = 0



#The kernel to be used for dilation purpose

kernel = np.ones((5,5),np.uint8



colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (0, 255, 255)]

colorIndex = 0



# Here is code for Canvas setup

paintWindow = np.zeros((471,636,3)) + 255

paintWindow = cv2.rectangle(paintWindow, (40,1), (140,65), (0,0,0), 2)

paintWindow = cv2.rectangle(paintWindow, (160,1), (255,65), (255,0,0), 2)

paintWindow = cv2.rectangle(paintWindow, (275,1), (370,65), (0,255,0), 2)
paintWindow = cv2.rectangle(paintWindow, (390,1), (485,65), (0,0,255), 2)

paintWindow = cv2.rectangle(paintWindow, (505,1), (600,65), (0,255,255), 2)



cv2.putText(paintWindow, "CLEAR", (49, 33),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA)

cv2.putText(paintWindow, "BLUE", (185, 33),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA)

cv2.putText(paintWindow, "GREEN", (298, 33),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA)

cv2.putText(paintWindow, "RED", (420, 33),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA)

cv2.putText(paintWindow, "YELLOW", (520, 33),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA)

cv2.namedWindow('Paint', cv2.WINDOW_AUTOSIZE)



# Initialize mediapipe

mpHands = mp.solutions.hands

hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)

mpDraw = mp.solutions.drawing_utils



# Initialize the webcam
```

```
cap = cv2.VideoCapture(0)
ret = True
while ret:
    # Read each frame from the webcam
    ret, frame = cap.read()


    x, y, c = frame.shape


    # Flip the frame vertically
    frame = cv2.flip(frame, 1)
    #hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)


    frame = cv2.rectangle(frame, (40,1), (140,65), (0,0,0), 2)
    frame = cv2.rectangle(frame, (160,1), (255,65), (255,0,0), 2)
    frame = cv2.rectangle(frame, (275,1), (370,65), (0,255,0), 2)
    frame = cv2.rectangle(frame, (390,1), (485,65), (0,0,255), 2)
    frame = cv2.rectangle(frame, (505,1), (600,65), (0,255,255), 2)
    cv2.putText(frame, "CLEAR", (49, 33), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, "BLUE", (185, 33), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, "GREEN", (298, 33), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 0), 2, cv2.LINE_AA)
```

```
    cv2.putText(frame, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, "YELLOW", (520, 33),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA)
    #frame = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)



    # Get hand landmark prediction
    result = hands.process(framergb)



    # post process the result
    if result.multi_hand_landmarks:
        landmarks = []
        for handslms in result.multi_hand_landmarks:
            for lm in handslms.landmark:
                # # print (id, lm)
                # print(lm.x)
                # print(lm.y)
                lmx = int(lm.x * 640)
                lmy = int(lm.y * 480)



                landmarks.append([lmx, lmy])
            # Drawing landmarks on frames
            mpDraw.draw_landmarks(frame, handslms,
mpHands.HAND_CONNECTIONS)
        fore_finger = (landmarks[8][0],landmarks[8][1])
        center = fore_finger
        thumb = (landmarks[4][0],landmarks[4][1])
        cv2.circle(frame, center, 3, (0,255,0),-1)
        print(center[1]-thumb[1])
        if (thumb[1]-center[1]<30):
            bpoints.append(deque(maxlen=512))
            blue_index += 1
            gpoints.append(deque(maxlen=512))
            green_index += 1
            rpoints.append(deque(maxlen=512))
            red_index += 1
            ypoints.append(deque(maxlen=512))
            yellow_index += 1


        elif center[1] <= 65:
            if 40 <= center[0] <= 140: # Clear Button
                bpoints = [deque(maxlen=512)]
                gpoints = [deque(maxlen=512)]
                rpoints = [deque(maxlen=512)]
                ypoints = [deque(maxlen=512)]
                blue_index = 0
                green_index = 0
                red_index = 0
                yellow_index = 0


                paintWindow[67:,:,:] = 255
            elif 160 <= center[0] <= 255:
                    colorIndex = 0 # Blue
            elif 275 <= center[0] <= 370:
                    colorIndex = 1 # Green
            elif 390 <= center[0] <= 485:
                    colorIndex = 2 # Red
            elif 505 <= center[0] <= 600:
                    colorIndex = 3 # Yellow
        else :
            if colorIndex == 0:
                bpoints[blue_index].appendleft(center)
            elif colorIndex == 1:
                gpoints[green_index].appendleft(center)
            elif colorIndex == 2:
                rpoints[red_index].appendleft(center)
            elif colorIndex == 3:
                ypoints[yellow_index].appendleft(center)
    # Append the next deques when nothing is detected to avois messing up
    else:
        bpoints.append(deque(maxlen=512))
        blue_index += 1
        gpoints.append(deque(maxlen=512))
        green_index += 1
        rpoints.append(deque(maxlen=512))
        red_index += 1
        ypoints.append(deque(maxlen=512))
        yellow_index += 1



    # Draw lines of all the colors on the canvas and frame
    points = [bpoints, gpoints, rpoints, ypoints]
    # for j in range (len(points [0])):
    #         for k in range(1, len(points[0][j])):
    #             if points[0][j] [k - 1] is None or points[0][j][k] is None:
    #                 Continue
    #             cv2.line(paintWindow, points[0][j][k - 1], points[0][j][k], colors[0], 
2)
    for i in range(len(points)):
        for j in range(len(points[i])):
            for k in range(1, len(points[i][j])):
                if points[i][j][k - 1] is None or points[i][j][k] is None:
                    continue
                cv2.line(frame, points[i][j][k - 1], points[i][j][k], colors[i], 2)
                cv2.line(paintWindow, points[i][j][k - 1], points[i][j][k], colors[i], 2)
```

```
   cv2.imshow("Output", frame)
    cv2.imshow("Paint", paintWindow)


    if cv2.waitKey(1) == ord('q'):
        break


# release the webcam and destroy all active windows
cap.release()
cv2.destroyAllWindows()
```

TEST CASES



Id	Description	Input	Expected Output	Actual Output	Status
VP1	Color Selection	Select a color from the frame using two fingers	Choosen color must be selected.	Color got selected.	Pass.
VP2	Checking clear box.	Select the clear	Data on the screen will be erased.	Data erased	Pass
VP3	Hand detection When hand is shown to the webcam, the landmarks on the hand should get traced.	Hand shown to camera	Hand should get detected.	Hand detected.	Pass
VP4	Checking drawing or writing.	The index finger should start drawing or writing.	Index finger is drawing or writing.	Drawing or writing started.	Pass

6.2. OUTPUT SCREENS

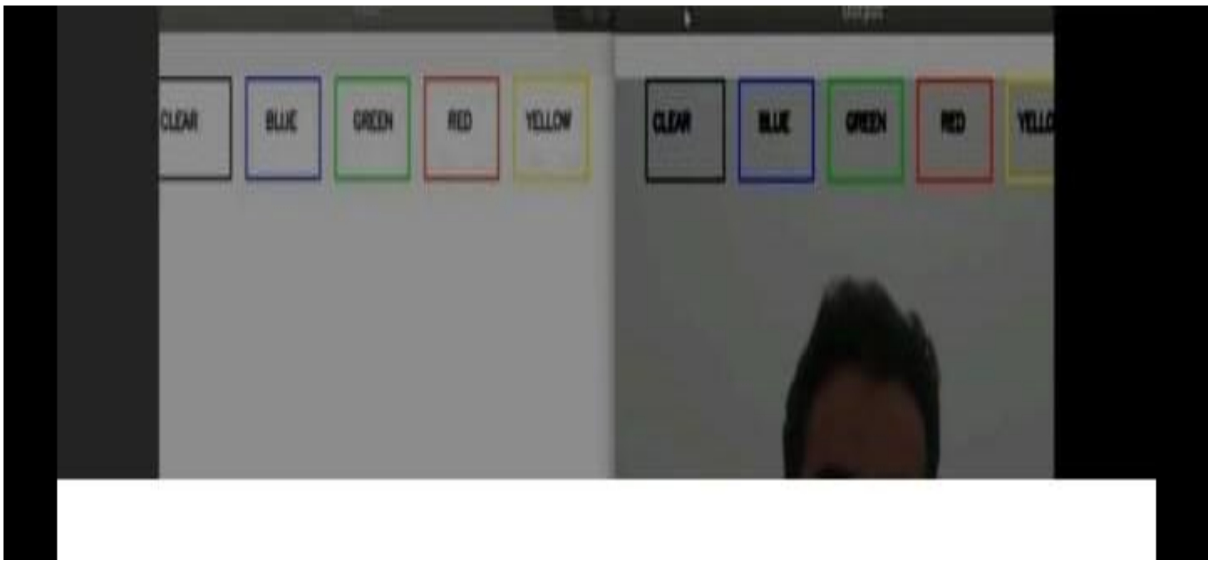


FIGURE:6.2.1 SELECTION BOX

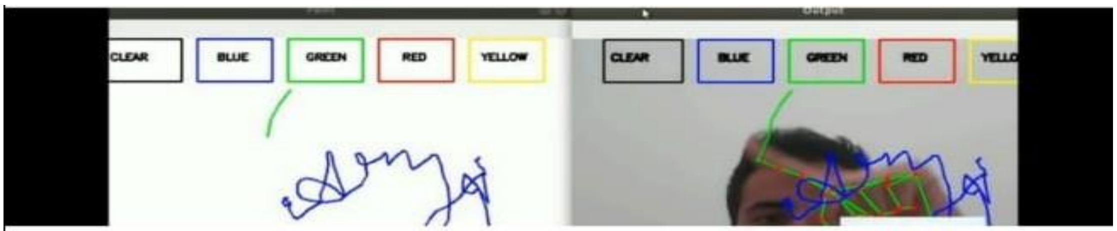


FIGURE:6.2.2

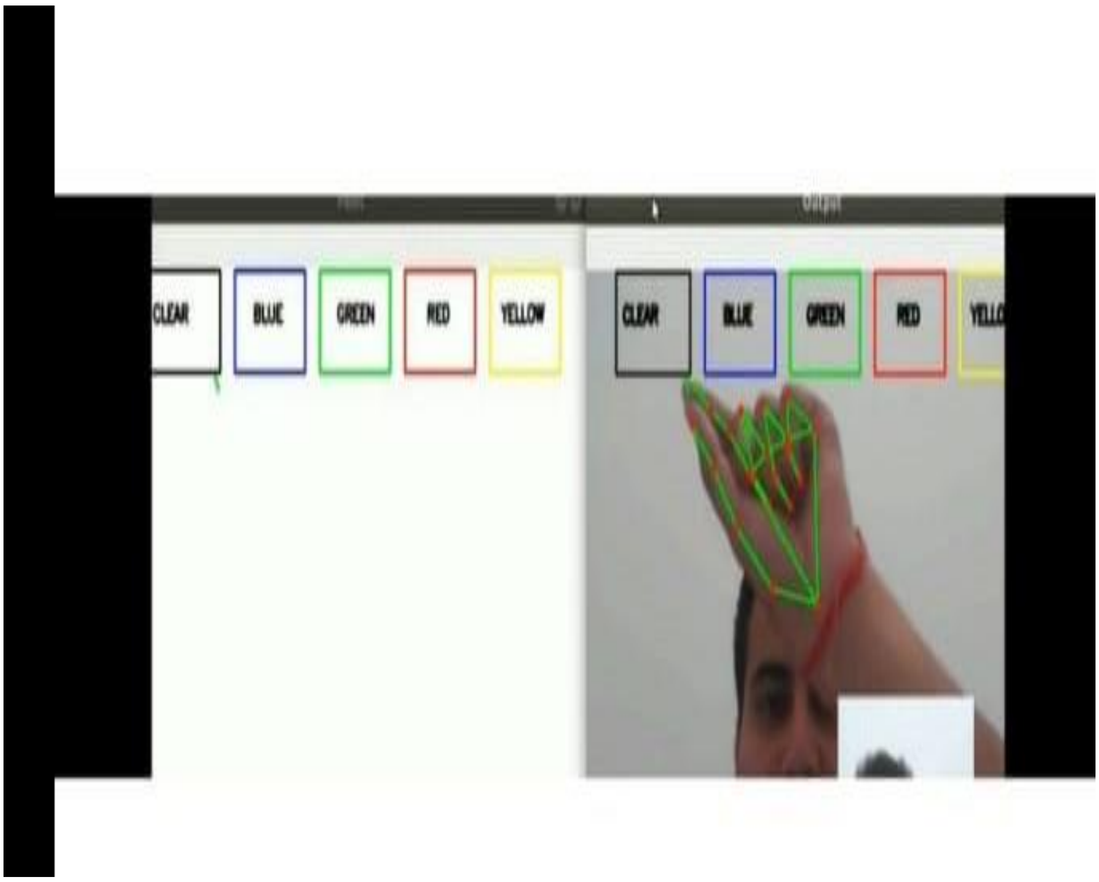


FIGURE:6.2.3

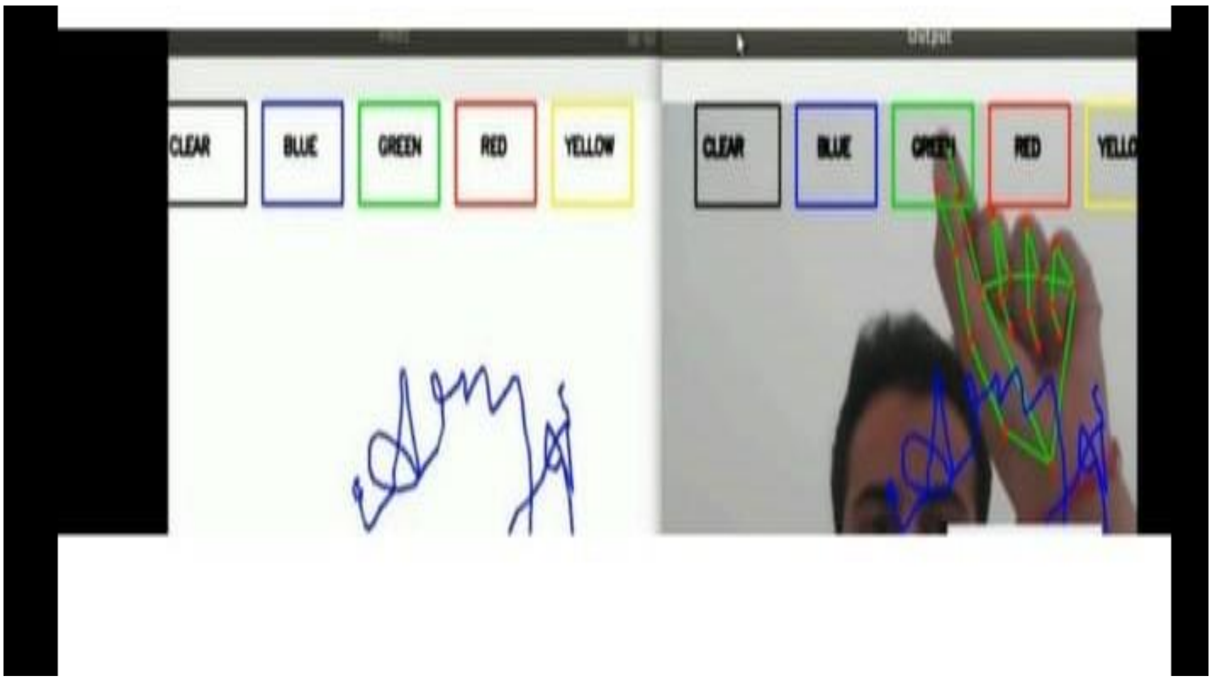


FIGURE:6.2.4

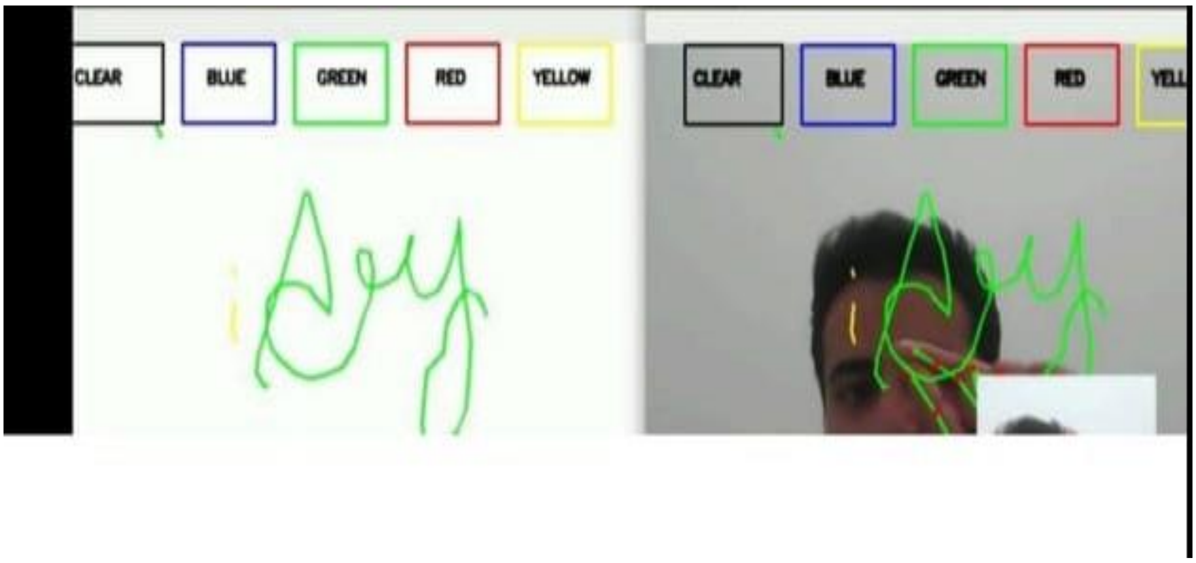


FIGURE.6.2.5

7.TESTINGS

CHAPTER – 7

TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and code generation.

7.1 TESTING OBJECTIVES:

- To ensure that during operation the system will perform as per specification.
- To make sure that system meets the user requirements during operation.
- To make sure that during the operation, incorrect input, processing and output will be detected
- To see that when correct inputs are fed to the system the outputs are correct
- To verify that the controls incorporated in the same system as intended
- Testing is a process of executing a program with the intent of finding an error
- A good test case is one that has a high probability of finding an as yet undiscovered error.

The software developed has been tested successfully using the following testing strategies and any errors that are encountered are corrected and again the part of the program or the procedure or function is put to testing until all the errors are removed. A successful test is one that uncovers a yet undiscovered error. Note that the result of the system testing will prove that the system is working correctly. It will give confidence to system designer, users of the system, prevent frustration during implementation process etc Software testing is a Virtual painter 44 critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

7.2 TESTING METHODOLOGIES:

- ✓ White box testing.
- ✓ Black box testing.
- ✓ Unit testing.
- ✓ Integration testing.
- ✓ User acceptance testing.
- ✓ Output testing.
- ✓ Validation testing.
- ✓ System testing

1)White Box Testing:

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

2) Black Box Testing:

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry

.

3) Unit Testing:

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

4) Integration Testing:

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

- ✓ **Top-Down Integration:**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module.

- ✓ **Bottom-Up Integration:**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the 29 bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated.

5) User acceptance Testing:

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system

6) Output Testing:

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

7) Validation Testing:

Validation testing is generally performed on the following fields:

✓ Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message. in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

✓ Numeric Field:

Virtual painter 47 The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform.

✓ **Preparation of Test Data:**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

✓ **Using Live Test:**

Live test data are those that are extracted from organization files. After system is partially constructed; programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

✓ **Using Artificial Test Data**:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be Virtual painter 48 prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program. The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications

7.3 USER TRAINING:

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose, the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

7.4 MAINTAINENCE:

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

7.5 TESTING STRATEGY:

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements. Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding.

7.5.1 SYSTEM TESTING:

Software once validated must be combined with other system elements (e.g., Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications, and system documentation.

7.5.2 UNIT TESTING:

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module. In Due Course, latest technology advancements will be taken into consideration. Virtual painter 50 As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this

8. CONCLUSION & FUTURE ENHANCEMENT

CHAPTER – 8
CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION:

This application can be put to use in many fields which doesn't need a lot of expertise in typing and also is efficient as it takes a brief amount of time to convey the user's thoughts. It create a space for the users to draw anything they wish. Using a single finger among the available choice of colors while using two fingers to make a choice in a very convenient and effortless manner. Even senior citizens or people who find it difficult to use keyboards will able to use system effortlessly. The principle reason of the project is to demonstrate human-computer interaction and is built using Python, Numpy and OpenCV.

Future Enhancement:

Virtual Painter using **Android Phone:** Mobile devices such as smartphones, ipads, tablet & pc etc., are equipped with cameras, they demand of image processing applications . This application need to be faster and consumes lower power because the mobile device is only powered by battery.
Robot control: controlling the robot using gestures considered as one of the interesting application to use the numbering to count the five fingers

to control a robot using hand pose signs. The order is given to the robot to perform particular task using gestures.

9.BIBLOGRAPHY

CHAPTER – 10

BIBLOGRAPY

[1]. Volume 5, Issue 1, January 2015 ISSN: 2277 128X, International Journal of Advanced Research in Computer Science and Software Engineering: Research Paper -- Gesture Controlled Computer.

[2]. https://towardsdatascience.com/Painting-withopencv

[3]. https://www.geeksforgeeks.org/live-webcamdrawing-using-opencv.

[4]. Volume 43, Issue 1, June 2012 ISSN: 2277 128X, International Journal of Advanced Research in Computer Science and Software Engineering: Research Paper -- Hand Data Glove: A Wearable Real-Time Device for Human Computer Interaction.

[5]. International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.4, J 2012, DOI: 10.5121/ijaia.2012.3412 161- HAND GESTURE RECOGNITION: A LITERATURE REVIEW.

[6]. OpenCV for Computer Vision Applications, Proceedings of National Conference on Big Data and Cloud Computing (NCBDC'15), March 20, 2015.

[7]. https://www.geeksforgeeks.org/live-webcamdrawing-using-opencv/

[8]. https://docs.opencv.org/master/d5/d54/group__obj detect.html

[9]. https://circuitdigest.com/tutorial/real-life-objectdetection-using-opencvpython-detectingobjects-in-live-video.

[10]. Real Time Object Detection and Tracking Using Deep Learning and OpenCV Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018) IEEE Xplore Compliant Part Number: CFP18N67-ART; ISBN:978-1-5386-2456-2.

[11]. Numpy.org, 2017. Online]. Available: http://www.numpy.org

[12]. (2017, January 17). Object Detection Online]. Available: http://en.m.wikipedia.org/wiki/Object_detection.

[13]. Study on Object Detection using Open CV – Python, International Journal of Computer Applications (0975 – 8887) Volume 162 – No 8, March 2017.

[14]. Nidhi, "Image Processing and Object Detection", Dept. of Computer Applications, NIT, Kurukshetra, Haryana,