

AI Virtual Painter

Submitted in partial fulfillment of the requirements of the degree of

BACHELOR OF COMPUTER ENGINEERING

by

Viraj Mehta - 21202011

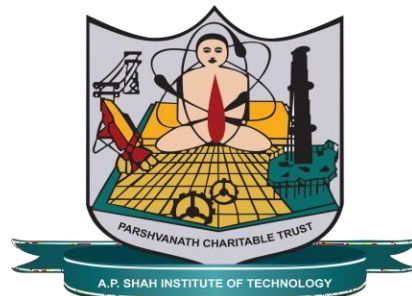
Soumya Madargi - 20102058

Tejas Pathak - 20102042

Hrugved Parab - 20102045

Guide:

Prof. Suchita Dange



Department of Computer Engineering

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

(2022-2023)



A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

CERTIFICATE

This is to certify that the Mini Project 2B entitled “**AI VIRTUAL PAINTER**” is a bonafide work of “**VIRAJ MEHTA (21202011), SOUMYA MADARGI (20102058), TEJAS PATHAK (20102042), HRUGVED PARAB (20102045)**” submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering in Computer Engineering**.

Guide
Prof. Suchita Dange

Project Coordinator
Prof. D.S. Khachane

Head of Department
Prof. S. H. Malave



A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

Project Report Approval for Mini Project-2B

This Mini project report entitled “**AI Virtual Painter**” *by VIRAJ MEHTA, SOUMYA MADARGI, TEJAS PATHAK, HRUGVED PARAB* is approved for the partial fulfilled of the degree of *Bachelor of Engineering in Computer Engineering, 2022-23*.

Examiner Name

Signature

1. _____

2. _____

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Viraj Mehta - 21202011

Soumya Madargi- 20102058

Tejas Pathak - 20102042

Hrugved Parab - 20102045

Date:

ABSTRACT

“AI Virtual Painter” is a fun application with an interactive-looking interface designed to draw figures, and shapes, designs with just a drawing in Air. It's been really tough to teach students on an online platform and make the lesson interesting during the COVID-19 pandemic. As a result, there was a need for a dust-free classroom for children.

Using MediaPipe and OpenCV, this article presents a unique paint application that identifies hand movements and tracks hand joints.

This program uses hand gestures to give users with an intuitive method of Human-Computer Interaction (HCI). HCI's major goal is to improve human-computer interaction.

In modern technologies, video tracking and processing the feed has been very essential. This processed data can be used for many research purposes or to express a particular output on a particular system.

There are various methods for processing and manipulating of data to get the required output. This paint application is created using the OpenCV module and Python programming language which is an apex machine learning tool to create an application like this.

Given the real-time webcam data, this paint-like Python application uses the OpenCV library to track an object of interest (a bottle cap in this case) and allows the user to draw by moving the object, which makes it both awesome and challenging to draw simple things.

Key Words: Hand Gesture Recognition, Human-Computer Interaction, Computer Vision, Paint, MediaPipe,

Machine learning, OpenCV, Morphing Techniques, Human-Computer Interactions, Air Writing.

CONTENTS

Sr. No.	Chapter Name	Page No.
1	Introduction	8
2	Literature Survey	10
3	Problem Statement, Objective & Scope	12
4	Proposed System	15
5	Project Plan	22
6	Experimental Setup	23
7	Implementation Details	24
8	Results	28
9	Conclusion	40
10	References	41

LIST OF FIGURES

Sr. No.	Figure Name	Page No.
1	Gantt Chart	22
2	Architecture Diagram	13
3	Block Diagram	14
3	Use Case Diagram	15
4	Data Flow Diagram	16
5	Activity Diagram	18
6	Flow Chart	20

Chapter 1

Introduction

The AI Virtual Paint Web Application using OpenCV and MediaPipe is a novel system that uses computer vision and machine learning to provide a personalized approach to painting. The application utilizes OpenCV and MediaPipe libraries to detect and track the movements of a user's hand, allowing them to paint virtually in real-time. The system uses machine learning algorithms to analyze the data and provide users with real-time feedback on their painting technique.

The web application provides a user-friendly interface for individuals to choose from a variety of painting tools and colors. The system tracks the user's movements and provides feedback on their painting technique, including suggestions for improvement based on machine learning algorithms. Additionally, the application allows users to save their artwork for future reference.

The data will also give technicians important insights into the general condition of the lab equipment, enabling them to spot recurring problems and take preventative action to stop them from occurring again. In addition, the information will allow lab administrators to generate reports on the functionality of the lab apparatus, simplifying resource allocation and future update planning.

The AI Virtual Paint Web Application has the potential to revolutionize the way we approach painting, providing individuals with a new, interactive way to express their creativity. By utilizing computer vision and machine learning algorithms. Overall, this technology has the potential to make painting more accessible and enjoyable for individuals of all skill levels, while also providing a new platform for artists to showcase work.

OpenCV (Open Source Computer Vision) - is a programming language library consisting of different types of functions mainly for computer vision. To explain in a simple language or in general way it is a library used for Image Processing. It is used mainly to do all the operations which are related to Images. They can also be used by artists to generate ideas.

Chapter 2

Literature Survey

[1] "PyTorch implementation of 'A Neural Algorithm of Artistic Style'" by L. Gatys (2016).

This article presents a PyTorch implementation of the neural algorithm of artistic style, which is a technique for generating images that combine the content of one image with the style of another. The author used a convolutional neural network to extract features from the content and style images and then optimized the image to match both sets of features.

[2] "Creating Art with Artificial Intelligence: A Case Study on a Deep Learning Approach for Painterly Rendering" by B. Nguyen et al. (2019).

This paper presents a deep learning approach for generating painterly images. The researchers trained a neural network using a dataset of real paintings and used it to generate new paintings with various styles and brushstrokes. They also developed a user interface for users to control the style and parameters of the generated artwork.

[3] "Neural Painterly Harmonization with PaintsTransfer" by Y. Li et al. (2021).

This paper presents a method for neural painterly harmonization, which is the process of transferring the style and color of one painting onto another. The researchers used a convolutional neural network to learn the style and color of a reference painting and then applied it to a target painting. They also developed a user interface for users to interactively adjust the transfer parameters.

[4] "Artistic Style Transfer for Videos Using Python and TensorFlow" by D. Mayorga et al. (2021).

This article presents a Python-based implementation of artistic style transfer for videos. The authors used TensorFlow and the VGG-19 neural network to extract style and content features from a reference image and apply them to a video sequence. They also developed a user interface for users to choose the style and adjust the parameters of the transfer.

Overall, these papers and articles demonstrate the potential of Python and AI algorithms for creating virtual paintings with various styles and techniques. They also highlight the importance of developing user interfaces that allow users to interactively control the parameters and customize the generated artwork.

Research Paper	ANALYSIS
1. “Artificial.Intelligence application of virtual reality technology in digital media art creation by Xiaoyan Wang” (2019).	This article presents a PyTorch implementation of the neural algorithm of artistic style, which is a technique for generating images that combine the content of one image with the style of another. The author used a convolutional neural network to extract features from the content and style images and then optimized the image to match both sets of features.
2. "Virtual Reality Application for Fostering Interest in Art” by laura raya” (2019).	This paper presents a deep learning approach for generating painterly images. The researchers trained a neural network using a dataset of real paintings and used it to generate new paintings with various styles and brushstrokes. They also developed a user interface for users to control the style and parameters of the generated artwork.
3. " Real Time Hand Gesture Based User Friendly Human Computer Interaction System by Koushik roy. (2020).	This paper presents a method for neural painterly harmonization, which is the process of transferring the style and color of one painting onto another. The researchers used a convolutional neural network to learn the style and color of a reference painting and then applied it to a target painting. They also developed a user interface for users to interactively adjust the transfer parameters.
4. “Paint / Writing Application through.WebCam.using MediaPipe and OpenCVby D. Mayorga et al” by shaurya gulati (2021).	This article presents a Python-based implementation of artistic style transfer for videos. The authors used TensorFlow and the VGG-19 neural network to extract style and content features from a reference image and apply them to a video sequence. They also developed a user interface for users to choose the style and adjust the parameters of the transfer.

Chapter 3

Problem Statement, Objectives, and Scope

3.1 Problem Statement

The problem statement for an AI virtual painter would be to develop an artificial intelligence system that can generate digital paintings or images in a way that resembles human artists' styles and techniques using hand movements.

The AI virtual painter should be able to take a user's input, such as a rough sketch or a set of colors, and generate a complete painting using its own creative algorithm

The AI virtual painter should be able to learn from a large dataset of existing paintings and styles to produce unique, high-quality artwork. It should also be able to adapt to different styles and genres, such as impressionism, realism, abstract, or surrealism, based on the user's preferences

Furthermore, the AI virtual painter should have the ability to continuously learn and improve its output based on feedback from users, making the system more personalized and adaptable to the user's preferences.

However, developing such an application would require solving several technical challenges. These challenges include creating an accurate gesture recognition algorithm, designing a responsive and intuitive user interface, and developing an efficient image processing pipeline to render digital paintings in real time. Overall, the development of an AI virtual paint web application has the potential to democratize digital art creation by providing an accessible and intuitive tool for people of all skill levels to create digital paintings.

3.2 Objective

To create a virtual canvas to sketch. To detect the human finger as a color marker. To do the morphological operations. To create an interface between the user and the system. Generating realistic and visually appealing artwork: One of the primary objectives of an AI virtual painter would be to generate high-quality and aesthetically pleasing artwork that is indistinguishable from art created by human artists. The AI should be able to generate a variety of styles, genres, and mediums of art, and produce images that are visually coherent and attractive. Learning from existing art: The virtual painter could be trained on a large corpus of existing art to learn and replicate the styles and techniques of various artists. This could include classical art, modern art, and contemporary art, among others.

Creativity and innovation: While replicating existing styles is important, an AI virtual painter could also be designed to generate new and innovative artwork. This could involve incorporating randomness or other algorithms that allow the AI to create unique pieces of art that are not based on any particular style.

User interactivity: An AI virtual painter could be designed to interact with users in real-time. Users could provide feedback on the artwork generated by the AI and make suggestions for modifications, leading to a collaborative creation process.

Commercial applications: An AI virtual painter could have commercial applications in the design, marketing, and advertising industries. For example, the AI could be used to generate artwork for branding and marketing campaigns or to create custom designs for clients.

3.3 Scope

- To ensure that the interface is very simple and easily understandable by the user.
- The user should be able to draw what he wishes to draw without any interruptions.
- In the future, this is useful for making kids learn drawing in schools in an interactive way.
- It helps people with hearing impairments to communicate well.
- Various purposes, such as sending messages, e-mails, etc., as the generated text can also be used for that.

Chapter 4

Proposed System Architecture

4.1 Diagrams:

4.1.1 Architecture Diagram:

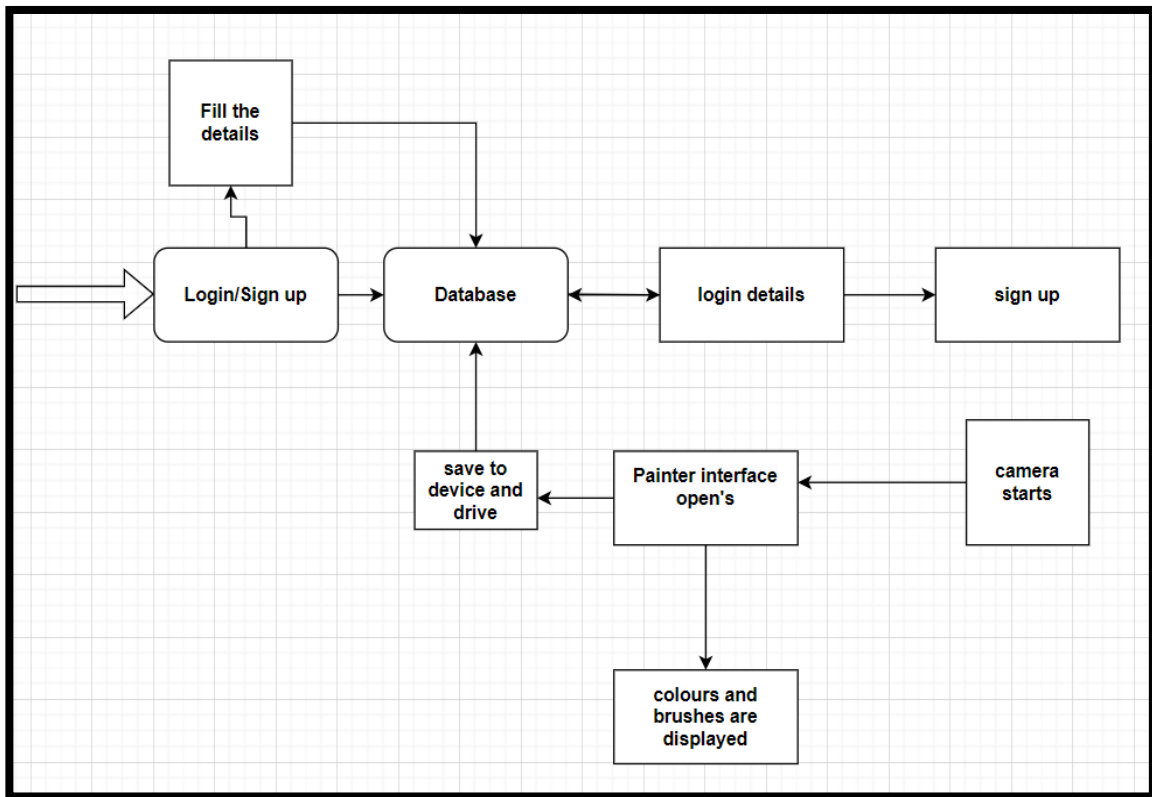


Fig no. 4.1.1 - Architecture Diagram

This is a proposed design that represents the process of AI Virtual Paint application. The device has to scan the hand movement, then verify his fingers and then draw figures and save them accordingly.

4.1.2 Block Diagram

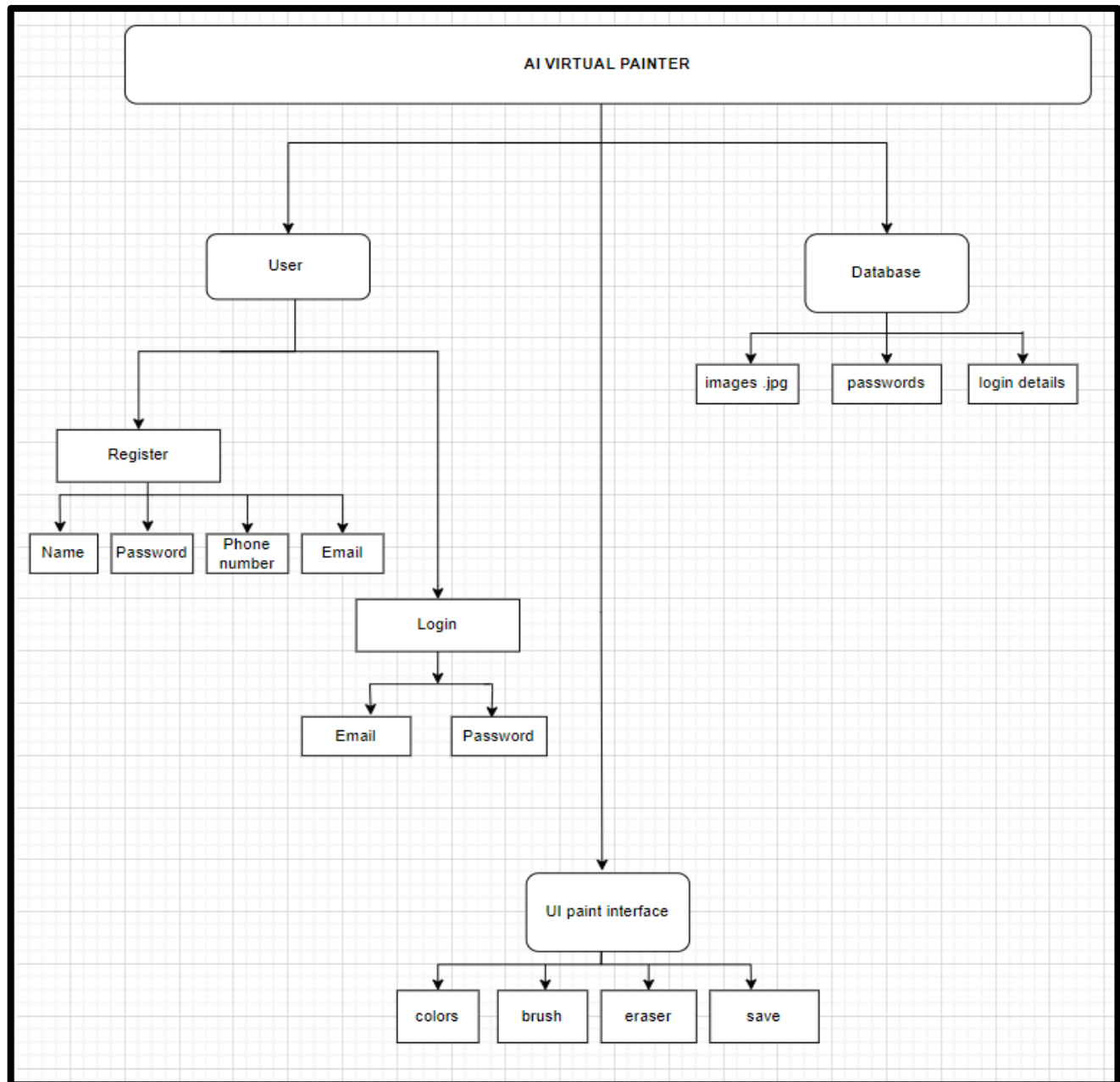


Fig no. 4.1.2 - Block Diagram

4.1.3 Use Case Diagram

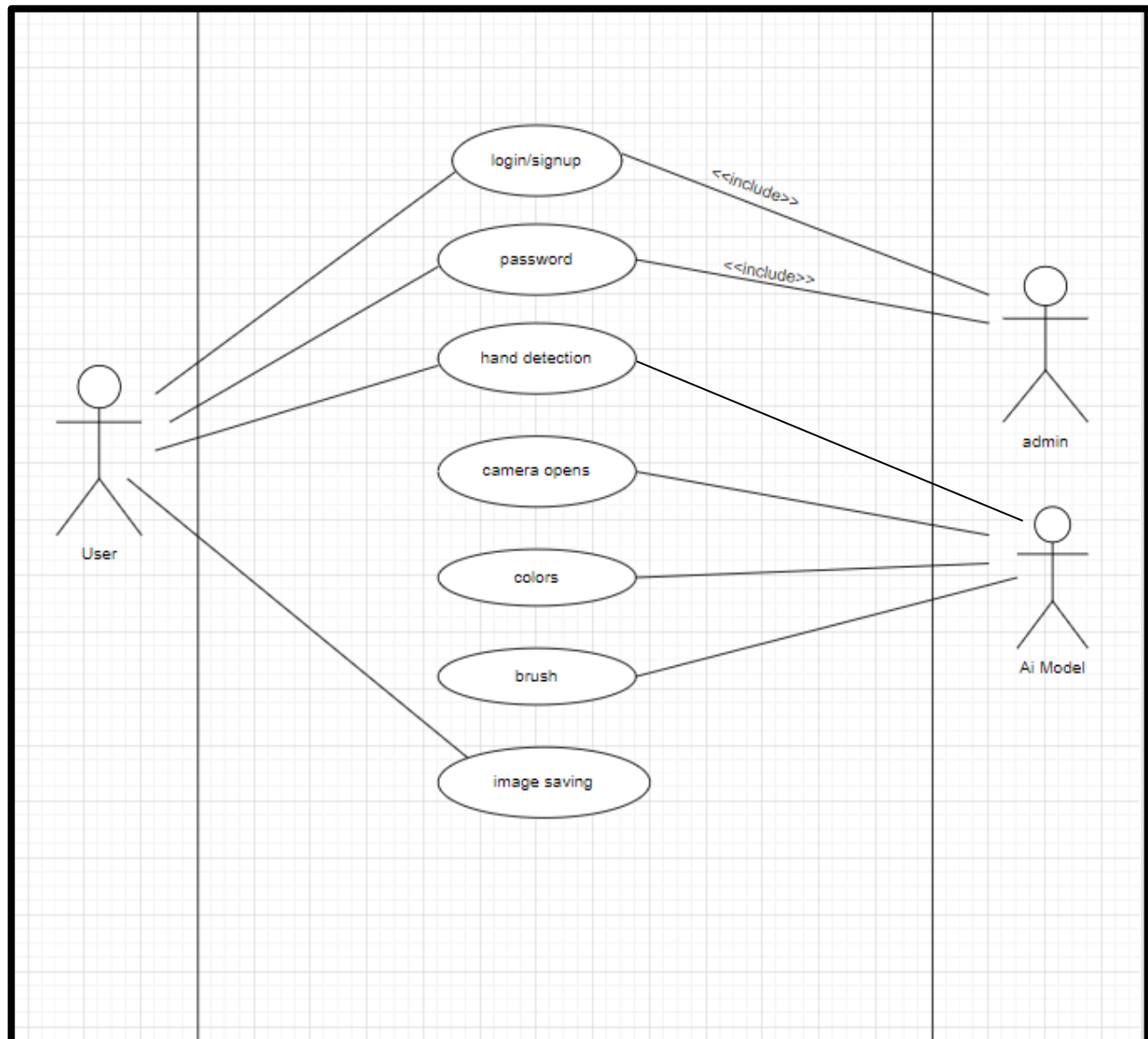
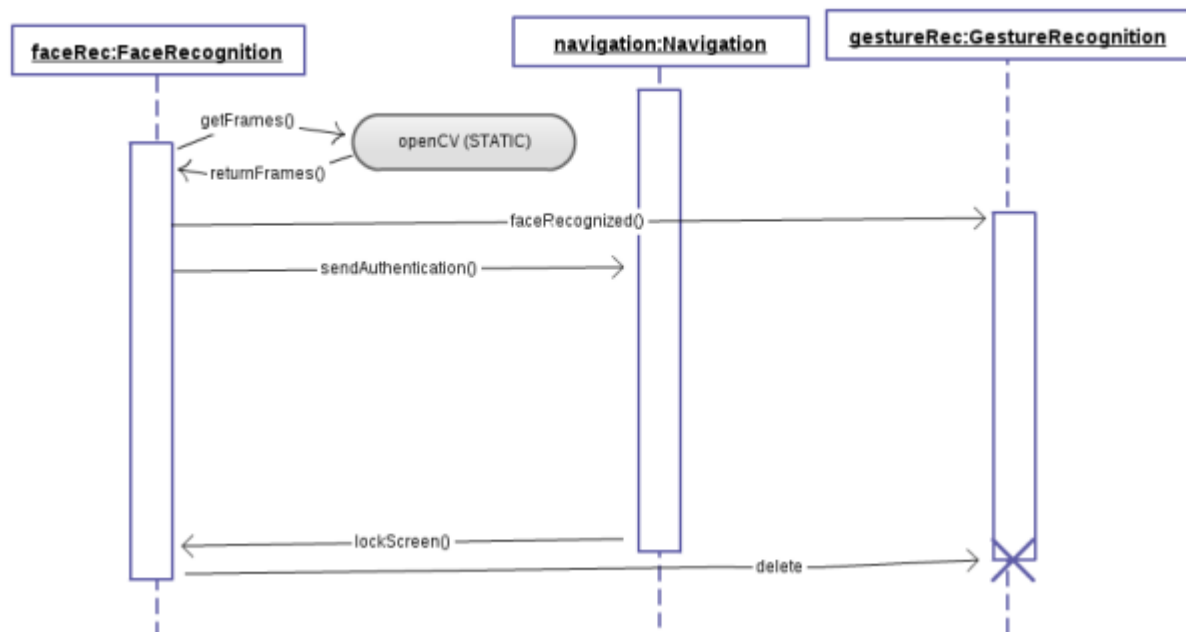


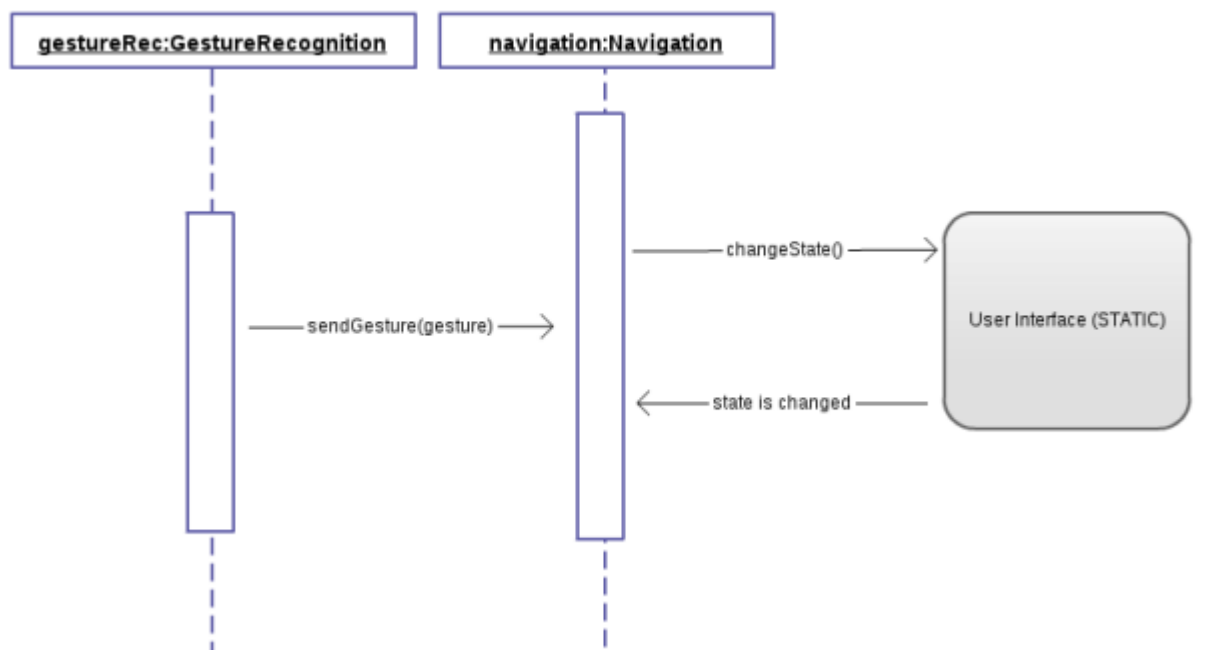
Fig no. 4.1.3 – Use Case Diagram

4.1.4 Sequence Diagram:

Step 1: Face Recognition in Virtual Painter



Step 2: Gesture Recognition and Navigation in Virtual Painter



4.1.5 Activity Diagram

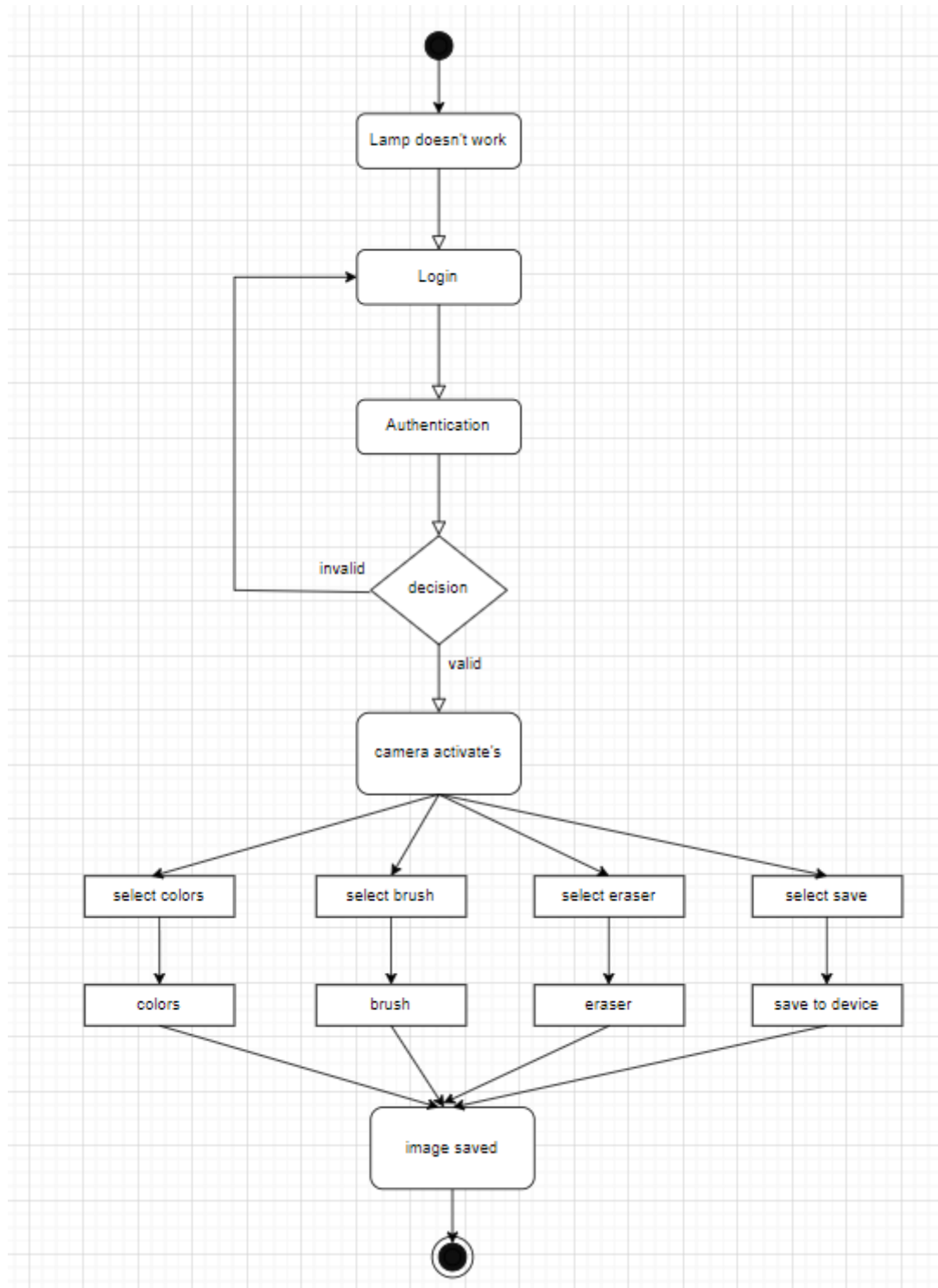


Fig no. 4.1.5 Activity Diagram of capturing an image in Virtual painter

4.1.6 Data Flow Diagram

The Data Flow Diagram (DFD) for the proposed system can be decomposed into three levels such as level 0, level 1, and level 2.

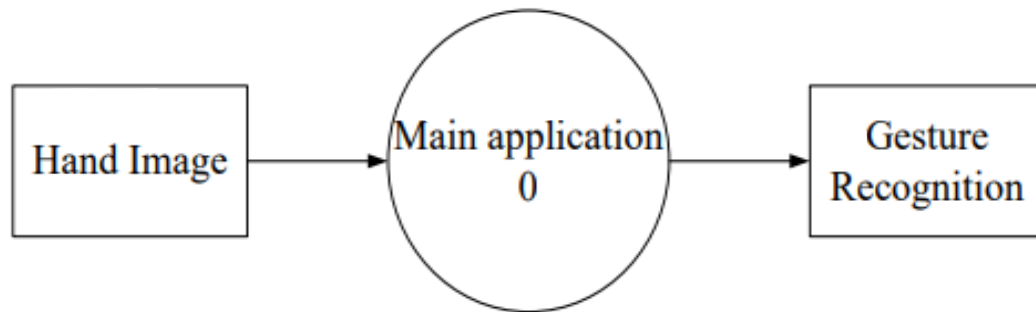


Fig no. 4.1.6a DFD level 0

The above figure represents a level zero data flow diagram where the main application {0} is shown to take input from a hand image and then using algorithms, gives the output as gesture recognition for Virtual Painter.

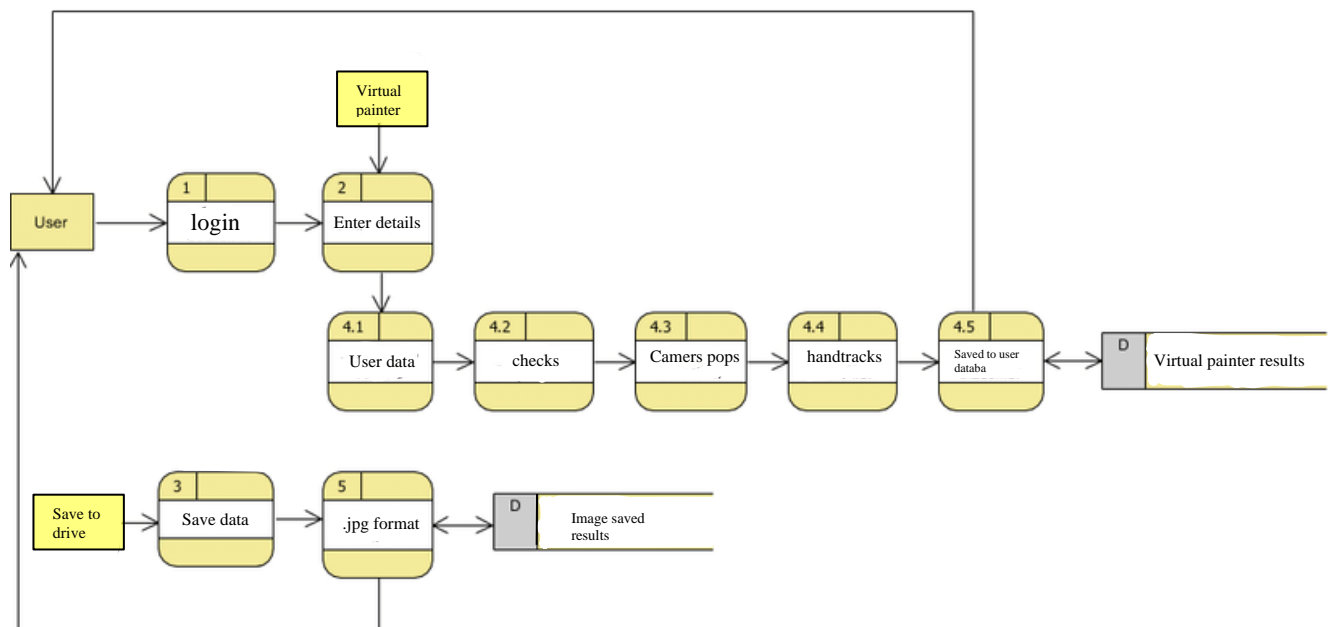


Fig no. 4.1.6b DFD level 1

4.2. Methodology

To create a CSS grid for our project, we utilized a function called `render.grid`. This function takes input from the system administrator and creates a grid layout based on the specified number of rows and columns. This grid layout serves as a visual representation of the lab layout. We used the `array.map` or `data.map` function to create the grid layout based on the number of rows and columns specified by the system administrator.

To create the layout, we used variables and the CSS set operator, which is a static property. However, in order to make the grid layout dynamic, we had to call or render data from the database. This is done by using the `document.getElementById` function to set the CSS property first, and then using `data.map` to arrange the PCs in a grid form instead of a linear form.

The `render.grid` function takes in the number of rows and columns as input parameters, and creates a two-dimensional array based on those parameters. This array represents the grid layout, where each cell in the array represents a PC in the lab. We then use the `array.map` function to loop through each element in the array and create a `div` element for each cell in the grid.

We also utilized the set operator in CSS to specify the CSS properties for the grid layout. This includes the `grid-template-rows` and `grid-template-columns` properties, which define the number of rows and columns in the grid, and the `grid-gap` property, which specifies the space between each cell in the grid.

To make the grid layout dynamic, we fetched data from the database using the `document.getElementById` function. This allowed us to retrieve the necessary data to populate the grid, including the PC names and their status. We then used the `data.map` function to loop through each element in the data array and update the corresponding `div` element in the grid with the appropriate PC name and status.

The result is a dynamic grid layout that accurately reflects the lab layout and the status of each PC in real-time. This allows system administrators and technicians to easily monitor the lab and quickly identify any issues with individual PCs. Overall, the use of the `render.grid` function and dynamic data fetching from the database have greatly improved the usability and functionality of our project.

Chapter 5

Project Planning

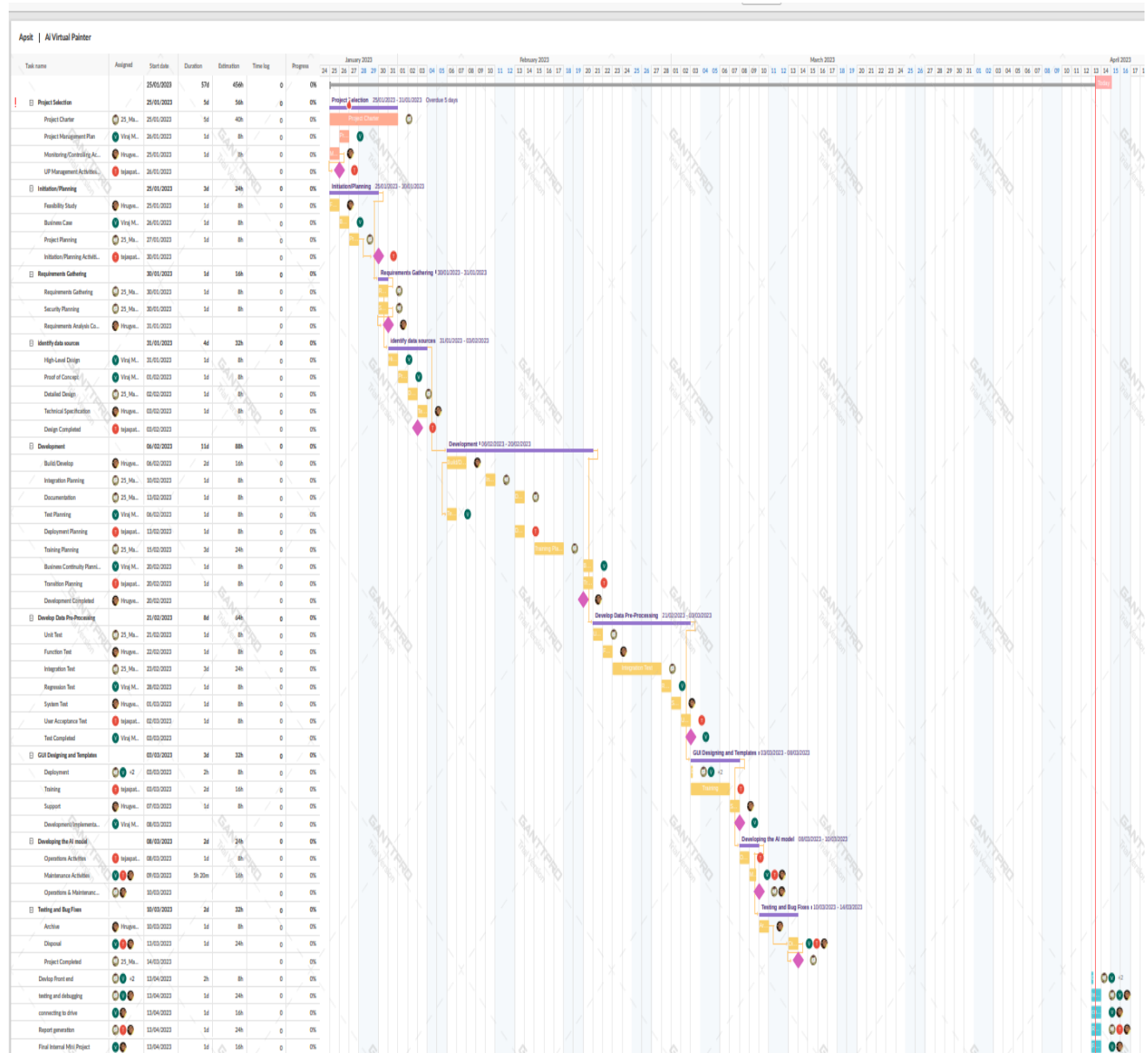


Fig no. 5 – Gantt Chart

Chapter 6

Experimental Setup

6.1 Software Requirements: -

1. Editor used for this project is Pycharm
2. Python for front end
3. Tkinter for Database

6.2 Hardware Requirements: -

Modern Operating System:

1. Windows 7 or 10
2. Mac OS X 10.11 or higher, 64-bit
3. Linux: RHEL 6/7, 64-bit (almost all libraries also work in Ubuntu) x86 64-bit CPU (Intel / AMD architecture)
4. 4 GB RAM

Chapter 7

Implementation Details

```
import sqlite3
from tkinter import *
from tkinter import ttk
import tkinter.messagebox as tkMessageBox

def database():
    global conn, cursor
    conn = sqlite3.connect("data.db")
    cursor = conn.cursor()
    q1 = "Create table if not exists user (id integer primary key AUTOINCREMENT,email varchar(150)
unique,password TEXT, name TEXT,phone_no TEXT)"
    cursor.execute(q1)

def login():
    root = Tk()
    em=StringVar(root)
    passwr=StringVar(root)
    root.title("AI virtual painter")
    root.geometry("780x450")
    root.config(bg="#343434")
    def log_in():
        database()
        try:
            e = em.get()
            p = passwr.get()
            cursor.execute(f'SELECT * FROM user WHERE email = ? AND password = ?', (e, p))
            data=cursor.fetchone()

            if data:
                name = data[3]
                f = open("cache.txt","w")
                f.write(name)
                f.close()
                try:
                    import os
                    os.mkdir(name)
                except:
                    pass
                tkMessageBox.showinfo("AI virtual painter","Logged In Successfully !!!")
                root.destroy()
                import Ai_virtual_painter
            else:
                tkMessageBox.showinfo("AI virtual painter","Failed To Login !!!")
                em.set("")
                passwr.set("")

        except Exception as e:
            tkMessageBox.showinfo("AI virtual painter",e)
```

```

h1 = Label(root, text="-- LOGIN --",fg="#DCB86F",bg="#343434",font=("",24,"bold"))
h1.place(x=300,y=15)
h2 = Label(root, text="Email: ",fg="#DCB86F",bg="#343434",font=("",20,"bold"))
h2.place(x=114,y=120)
h3 = Label(root, text="Password: ",fg="#DCB86F",bg="#343434",font=("",20,"bold"))
h3.place(x=60,y=205)
e1 = Entry(root,textvariable=em,bg="#DCB86F",fg="#343434",font=("",20),width=30)
e1.place(x=250,y=120)
e2 = Entry(root,textvariable=passwrdbg="#DCB86F",fg="#343434",font=("",20),width=30,show='*')
e2.place(x=250,y=205)
b1 = Button(root,
text="Register",bg="#DCB86F",fg="#343434",font=("",20,"bold"),bd=2,relief="solid",width=18,command=signup
)
b1.place(x=60,y=290)
b2 = Button(root,
text="LOGIN",bg="#DCB86F",fg="#343434",font=("",20,"bold"),bd=2,relief="solid",width=18,command=log_in)
b2.place(x=390,y=290)
root.mainloop()

def signup():
    global nme,passwrdb,em,phone
    root2 = Tk()
    nme=StringVar(root2)
    passwrdb=StringVar(root2)
    em = StringVar(root2)
    phone = StringVar(root2)
    branch = StringVar(root2)

    root2.title("AI virtual painter")
    root2.geometry("780x480")
    root2.config(bg="#343434")
    def reg():

        global nme,passwrdb,em,phone
        database()
        try:
            cursor.execute(f"INSERT INTO user(name,password,email,phone_no)
VALUES('{nme.get()}','{passwrdb.get()}','{em.get()}','{phone.get()}')")
            conn.commit()
            conn.close()
            tkMessageBox.showinfo("AI virtual painter","Registered Successfully !!!")
            root2.destroy()
            login()
        except Exception as e:
            tkMessageBox.showinfo("AI virtual painter",e)

    def login2():
        root2.destroy()
        login()

    h1 = Label(root2, text="-- REGISTER --",fg="#DCB86F",bg="#343434",font=("",24,"bold"))
    h1.place(x=280,y=15)

```

```

h2 = Label(root2, text="Name: ",fg="#DCB86F",bg="#343434",font=("",20,"bold"))
h2.place(x=60,y=120)
h3 = Label(root2, text="Password: ",fg="#DCB86F",bg="#343434",font=("",20,"bold"))
h3.place(x=60,y=180)
e1 = Entry(root2,textvariable=nme,bg="#DCB86F",fg="#343434",font=("",20),width=30)
e1.place(x=250,y=120)
e2 = Entry(root2,textvariable=passwrdbg="#DCB86F",fg="#343434",font=("",20),width=30)
e2.place(x=250,y=180)
h4 = Label(root2, text="Email: ",fg="#DCB86F",bg="#343434",font=("",20,"bold"))
h4.place(x=60,y=240)
h5 = Label(root2, text="Phone No.: ",fg="#DCB86F",bg="#343434",font=("",20,"bold"))
h5.place(x=60,y=300)
e4 = Entry(root2,textvariable=em,bg="#DCB86F",fg="#343434",font=("",20),width=30)
e4.place(x=250,y=240)
e5 = Entry(root2,textvariable=phone,bg="#DCB86F",fg="#343434",font=("",20),width=30)
e5.place(x=250,y=300)

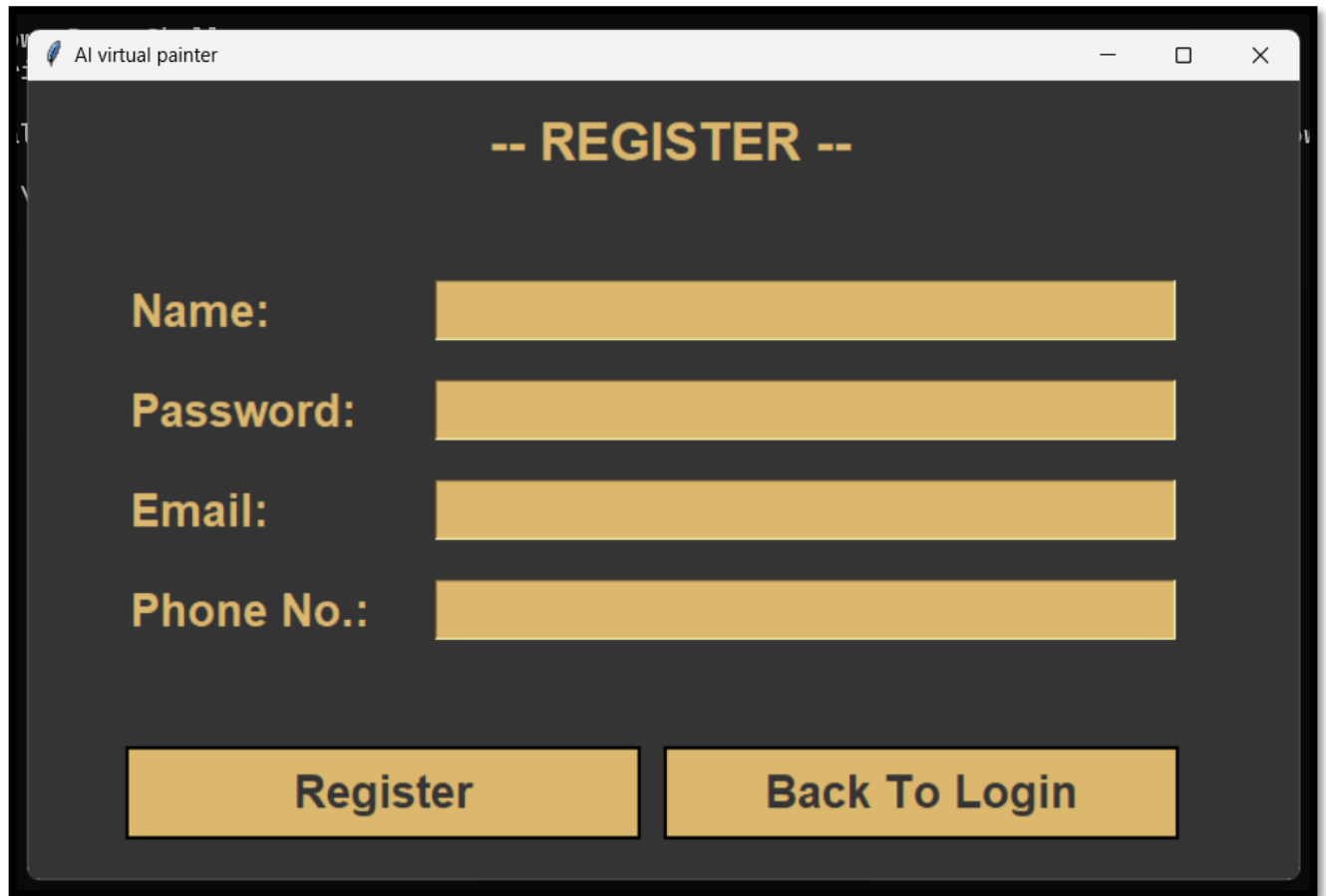
b1 = Button(root2,
text="Register",font=("",20,"bold"),bg="#DCB86F",fg="#343434",bd=2,relief="solid",width=18,command=reg)
b1.place(x=60,y=400)
b2 = Button(root2, text="Back To
Login",bg="#DCB86F",fg="#343434",font=("",20,"bold"),bd=2,relief="solid",width=18,command=login2)
b2.place(x=390,y=400)
root2.mainloop()

signup()

```


Chapter 8

Result



AI virtual painter

-- REGISTER --

Name:

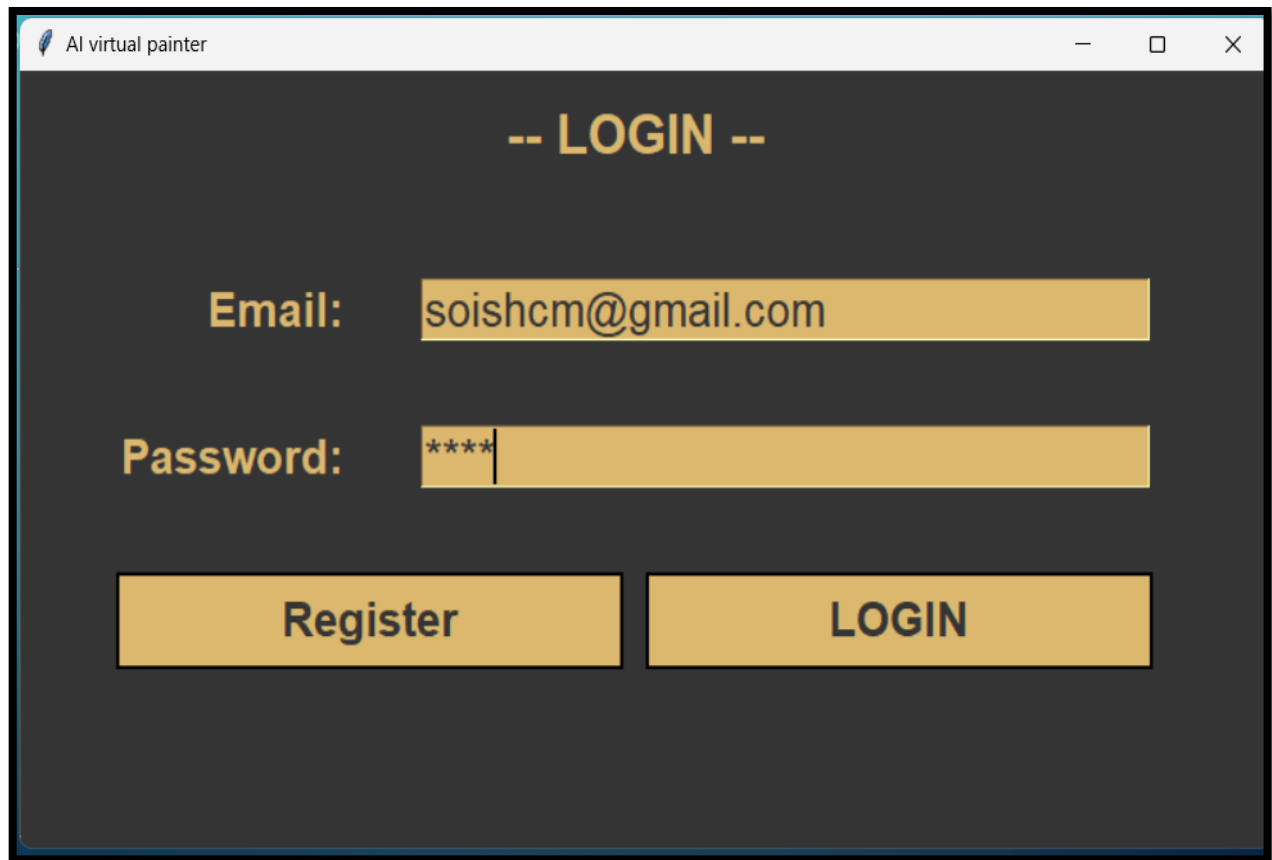
Password:

Email:

Phone No.:

Register **Back To Login**

Fig no. 8.1 - Home Page



The image shows a web browser window titled "AI virtual painter" with standard window controls. The main content area has a dark gray background and is titled "-- LOGIN --" in yellow. It contains two input fields: "Email:" with the value "soishcm@gmail.com" and "Password:" with masked characters "****". Below these fields are two yellow buttons labeled "Register" and "LOGIN".

AI virtual painter

-- LOGIN --

Email: soishcm@gmail.com

Password: ****

Register LOGIN

Fig no. 8.2 – User Login Page

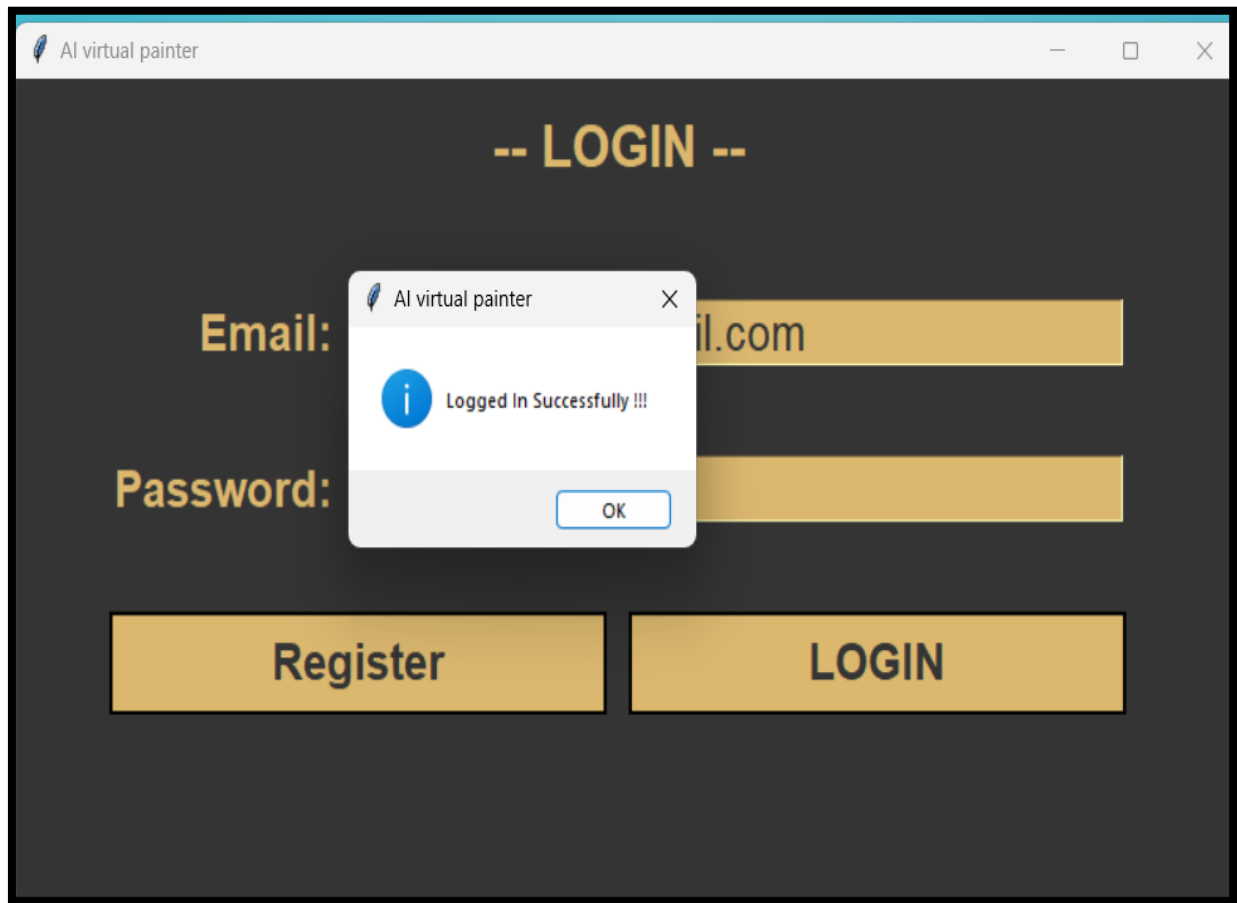


Fig no. 8.3 –Login Done

A screenshot of a Windows PowerShell terminal window. The window title is "Windows PowerShell". The text inside the terminal is as follows:
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

PS C:\Users\Soumya\Documents\Yoho> python main.py
|

Fig no. 8.4 – To Run Project

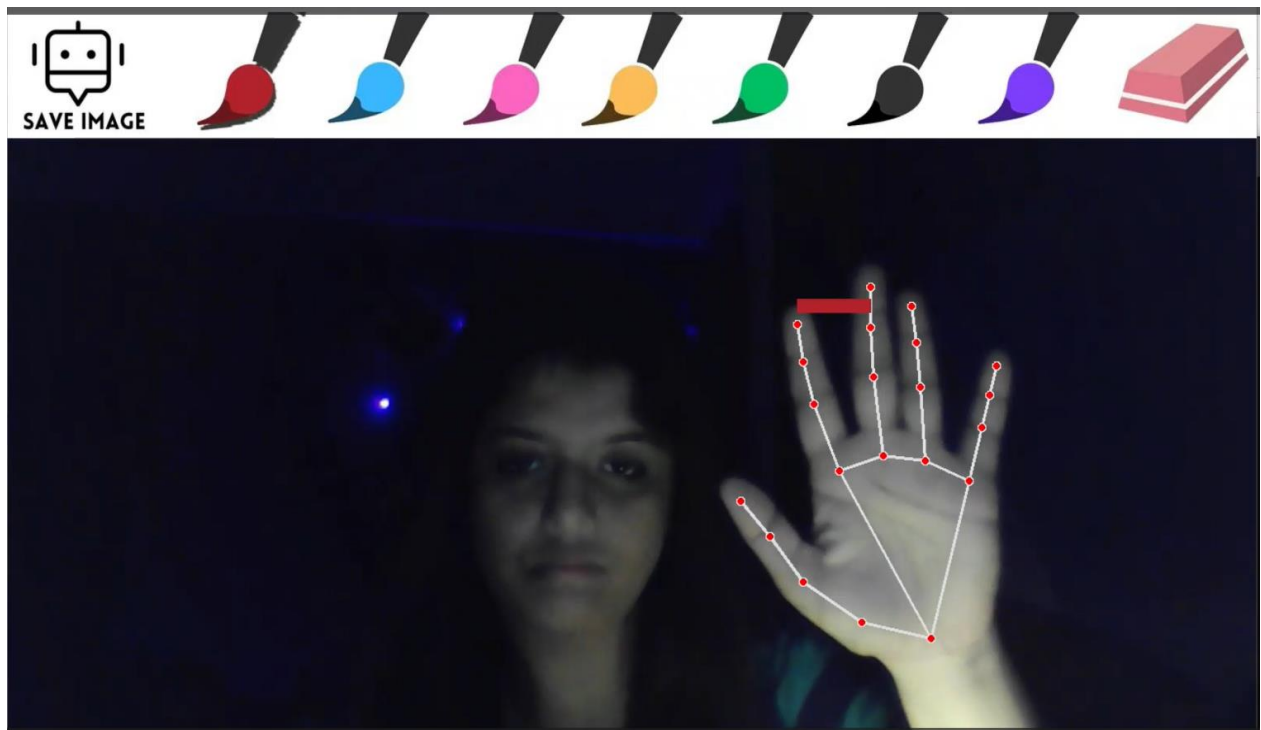


Fig no. 8.5 – UI Opened

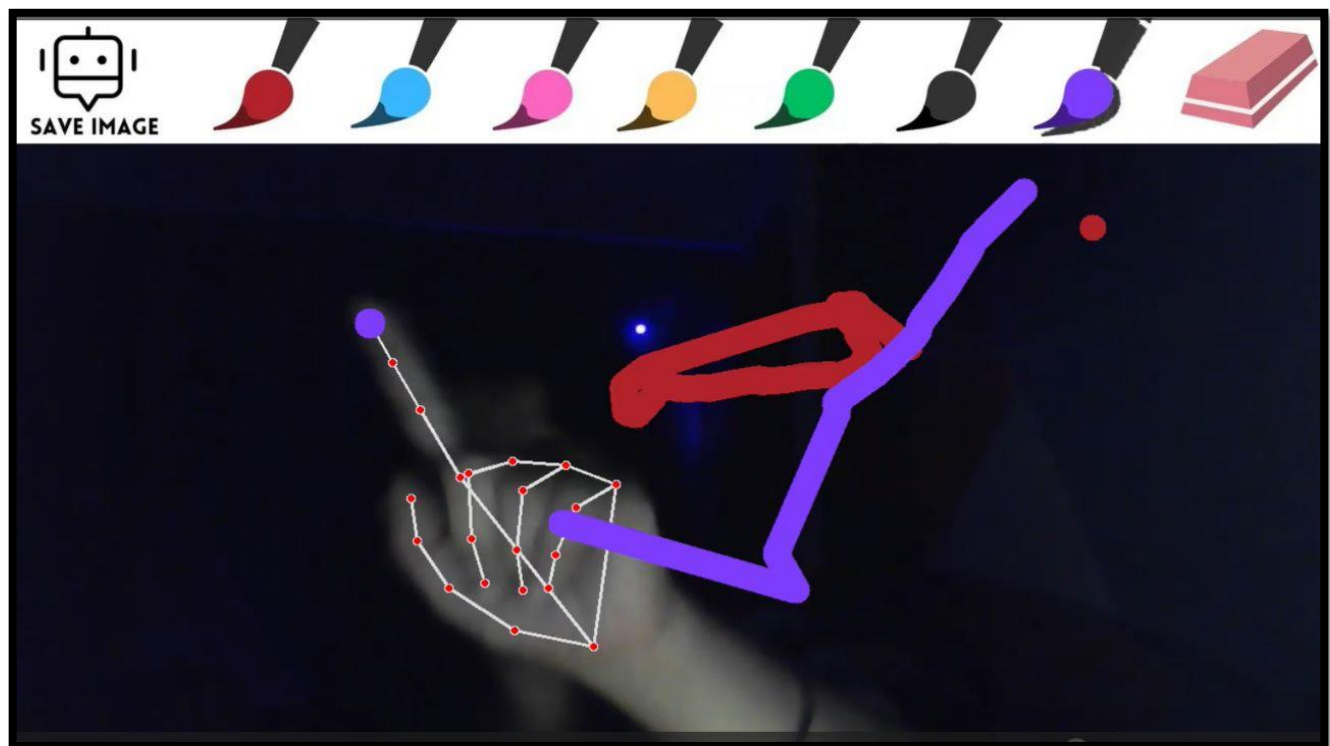


Fig no. 8.6 – Drawing in Air

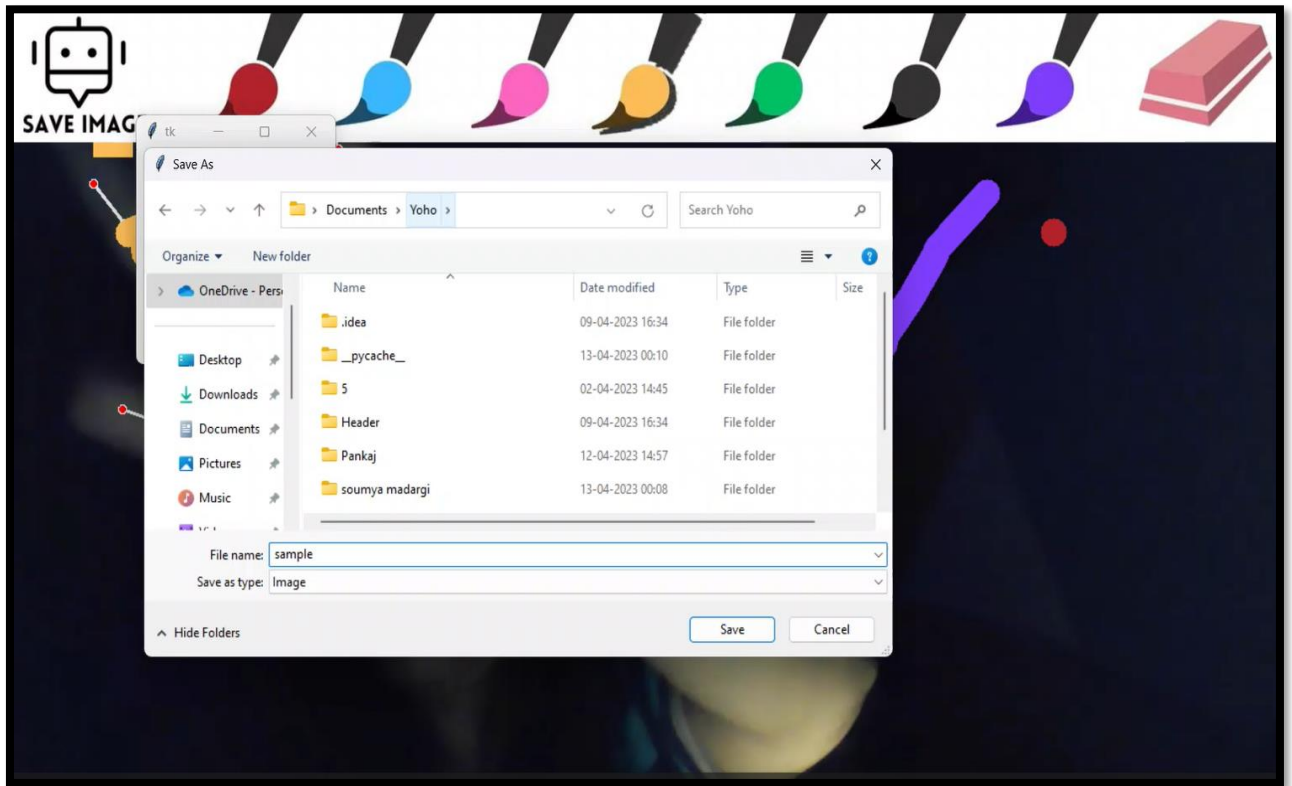


Fig no. 8.7– To Save Created Canvas on the Respected Device

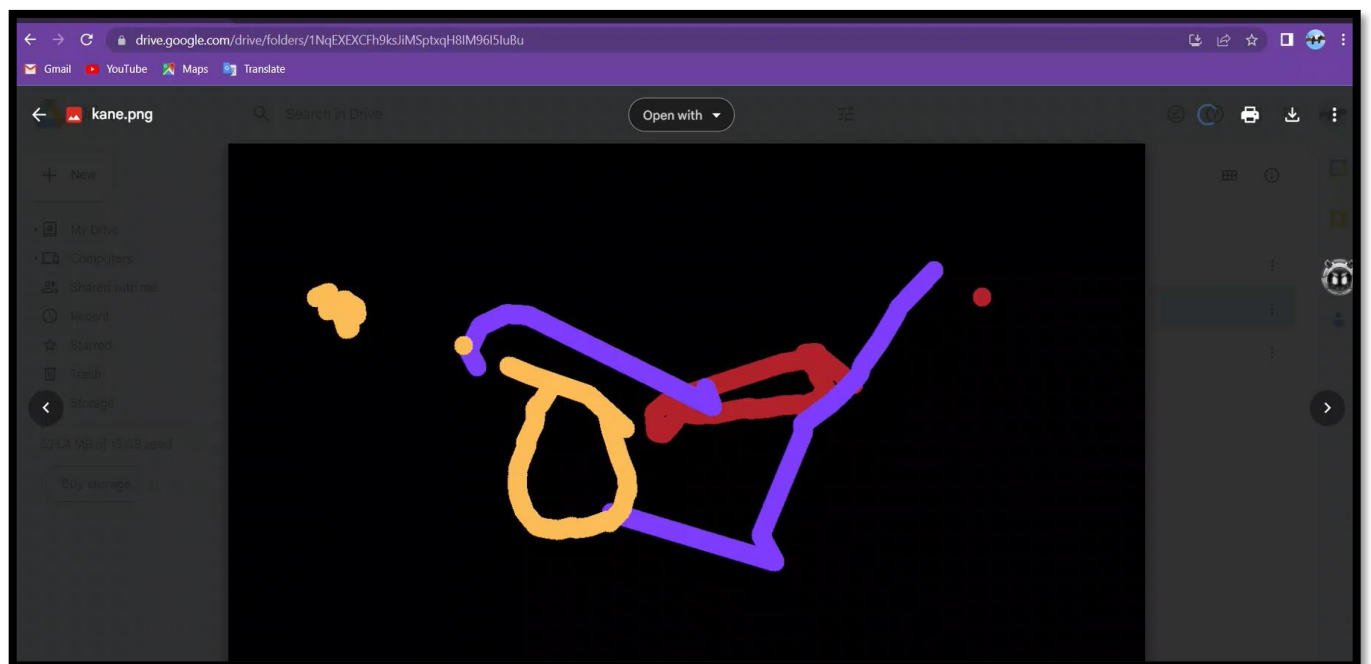


Fig no. 8.8 - Painting

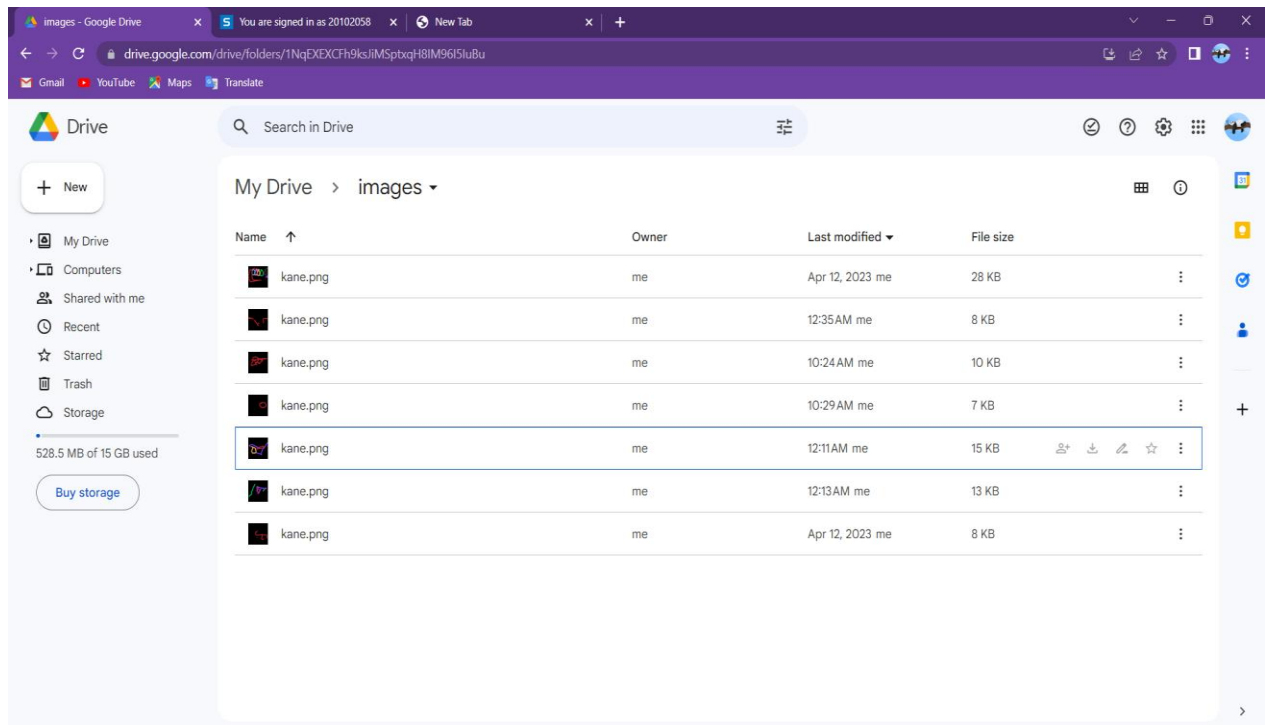


Fig no. 8.9 – Image Saved in Drive

Chapter 9

Conclusion

The virtual paint application's fundamental goal is to deliver an AI-based tool that allows users to draw anything on screen using hand movements. This system also gives the user the option of selecting any tool from the toolbar. The user can save their completed work or see their drawing process as a replay animation with this application. In conclusion, AI virtual painters are a promising new technology that has the potential to revolutionize the field of art.

By leveraging the power of machine learning and deep neural networks, these systems can generate stunning digital artworks that are virtually indistinguishable from those created by human artists. By leveraging the power of machine learning and deep neural networks, these systems can generate stunning digital artworks that are virtually indistinguishable from those created by human artists.

To overcome AI virtual painters can help democratize art by making it more accessible to a wider audience. They can enable people with no formal training or artistic background to create beautiful and meaningful pieces of art. Additionally, they can assist professional artists by providing them with new tools and techniques to enhance their creativity and productivity.

However, there are also concerns about the impact of AI virtual painters on the art world. Some critics argue that these systems might lead to a devaluation of human art, as they could replace human artists altogether. Others worry about issues related to copyright and ownership of digital artworks generated by AI systems.

Overall, the potential benefits and drawbacks of AI virtual painters are complex and multifaceted. As this technology continues to evolve, it will be important to carefully consider its implications and use it in a responsible and ethical manner.

Chapter 10

References

[1] “Artificial.Intelligence application of virtual reality technology in digital media art creation by Xiaoyan Wang” (2019).

<https://ieeexplore.ieee.org/abstract/document/9742419>

[2] "Virtual Reality Application for Fostering Interest in Art” by laura raya (2019).

<https://ieeexplore.ieee.org/abstract/document/9340297>

[3] "Real Time Hand Gesture Based User-Friendly Human-Computer Interaction System" by Kaushik Roy (2020).

<https://ieeexplore.ieee.org/abstract/document/9775918>

[4] "Paint / Writing Application through.WebCam.using MediaPipe and OpenCVby D. Mayorga et al" by Shaurya Gulati (2021).

<https://ieeexplore.ieee.org/abstract/document/9753939>

