**Smart Interactive Mirror System.**

**ABSTRACT**

Smart appliance design figured with multimedia intelligence to provide comfortable, convenient, and secure personal services in home becomes increasingly important in the era of information communication technology.

This investigation describes a novel design and implementation of an interactive multimedia mirror system, called “smart mirror.” The basic design of a smart mirror starts with the glass that is to be used. Two-way glass is the recommended type as it lets the graphics on the display come through clearer.

This project has been developed within the context of a time where every day we see more and more connected devices. The Internet transformed our lives by connecting us more easily to information and other people in the virtual world. Mobile phones then became smartphones and since then this concept has erupted and morphed into the Internet of Things, things which connect us to everyday objects. There are no end of objects that could be made “smarter”, some being more suited to this than others. Mirrors, for example, provide a large surface ideal for displaying information and interacting with. Most people have mirrors at home so the concept of a smart mirror that you can interact with is attractive and has been fantasized in many futuristic movies.

This final year project describes how a smart mirror was built from scratch using a hardware and custom software built on top. The goal of the project was to create a Smart Mirror device that people could interact with but also to further develop the technology so that it would let you install and develop your own applications for it.

The Smart Mirror was developed in four months, starting with the software and finally integrating it with the hardware. On the whole results were good because a higher level of interactivity has been achieved by being able to use voice commands, gestures and smartphones. A few problems arose in the construction and software side of the project, such as the glass not being reflective enough and the gesture recognition being unreliable but these drawbacks can be addressed by doing more tests and trials to further develop the Smart Mirror.

**OBJECTIVE**

The objective wherein is to develop a mirror that does smart activities like showing weather, time, date and news etc. thus keeping the user away from time consuming activities.

**PROBLEM STATEMENT**

The major problem of any existing mirror is displaying just the object in front of it or just the human face without having to interact with them. This project is developed with the intention that people spend quality time in front of the mirror.

**INTRODUCTION**

Smart mirrors arise from the transformation of a conventional mirror into an interactive information display element with special interaction capabilities. The mirror surface is an appropriate example of a natural interface because it takes part in our everyday life. Therefore, visual feedback interaction can be obtained non-intrusively through this object. In this sense, maintaining its initial functionality, the surface of the mirror is converted into a natural interface used for the visualization of information.

Everyone knows what a mirror is. It is an object found in most people’s homes. In mirrors we see our reflections. But what happens when you combine the idea of a mirror with technology? What possibilities are there and how smart could a mirror be? These are some of the questions that inspired my choice of final year project, a project which aimed to develop a smart mirror and a small operating system to power it. The device was to go beyond an ordinary mirror, to have a screen inside that you would be able to interact with by using voice commands, hand gestures and smartphones or other devices.

The smart mirror is a popular project among DIY enthusiasts and it usually consists of a one­way mirror with a screen attached to it that displays a static software. However what I wanted to achieve was something you could interact with. Once the software was almost finished I started designing the frame and finally I built the smart mirror and attached all the components Developing this project has been a great experience. I have learned a diverse range of skills in different fields, such as DIY, Python, electronics and web scrapping. To obtain the final result I’ve had to work with many different technologies. I used Photoshop and Illustrator for the UI designs, web tools for the software and electronics for the hardware. Not sticking 5 to just one field has made this project a really fun one and I would recommend it to anyone who is passionate about creating things.

**PROJECT GOALS**

The main goal of this project was to develop a smart mirror device as well as an operating system to run on similar devices. The device was to look like a regular mirror but would have a screen inside and you would be able to interact with it using voice commands, hand gestures and smartphones. The operating system would support running apps and would provide a simple API for third­ party developers to create their own apps for the Smart Mirror. The main features the Smart Mirror would have would be showing basic weather and time information, being able to add alarms, reminders or notes in a similar way we stick post­it notes on a fridge. We would also be able to play music in some way, for example. The software needed to be designed to be modular and responsive in order to fit different hardware.

**CONTEXT**

This project was inspired by a “Magic Home Mirror” device that I found while browsing the DIY section in a popular website called Reddit. The “Magic Home Mirror” is a Nexus 7 Android tablet attached to a one-way mirror. The device has a display with a software that shows time and weather information and top 5 News and it looks very futuristic. I liked that project a lot and I thought I could improve on it by adding some means of interaction to the device. This inspired me to begin this project and develop a Smart Mirror with an operating system that would let you install apps that anyone could develop just like on Android or iOS. The project has a very broad scope covering some current popular topics in the IT sector such as the Internet of Things, Maker culture and home automation.

**INTERNET OF THINGS**

The Internet of Things is a concept defined as a network of connected physical objects (Internet of things, 2016). It’s often viewed as the next step for the internet. Recently it has gained a lot of popularity predicting that in the future most everyday objects will be connected to each other and will be able to interact in smart ways. The Smart Mirror will eventually become one of these connected objects in our households and if we think about it being able to communicate with other objects the possibilities become endless

**MAKER CULTURE**

The maker culture is a contemporary culture derived from DIY culture and hacker culture (Maker culture, 2016). It focuses in the creation of new devices as well as modifying existing ones. It often supports and embraces open­source hardware and software. This culture has been growing rapidly thanks to tools and technology like the Raspberry Pi, 3D printers and other hardware that have become increasingly affordable and accessible. The Internet also plays a 7 big part in the community as it enables people to share their ideas, blueprints and code. The Smart Mirror is a good example of a Maker culture project.

**HOME AUTOMATION**

Home automation has been around for a long time and it is all about turning the house into an intelligent unit with the goal of increasing comfort and efficiency at home. Some of the typical applications are automatic lights, intelligent thermostats, alarms, window blinds (Home automation, 2013). In my project I will not be focusing on home automation since I don't have access to any smart home devices. However it would be very easy to write an application to turn on and off the lights using voice commands or gestures on the mirror or even an application to change the temperature of the room, for example. These examples are just the tip of the iceberg as there are new connected devices emerging everyday that could interact with the mirror.

**MAGIC MIRROR**

Magic Mirror is a mirror device built with a wooden frame, a flat­screen TV, a computer device and running software. The source code for the software and a step-by-step guide to building the hardware. It was one of the first, if not the first popular project of its kind. MagicMirror2 supports small widgets called “modules” and anyone can write one. The default installation comes with some basic widgets for time, calendar, weather and news and there are about 20 third­party modules available.

Magic Mirror is a great device that looks very good and has a clean UI. It also has a big community behind it. There’s a specialised MagicMirror forum helping people build their own hardware and write modules

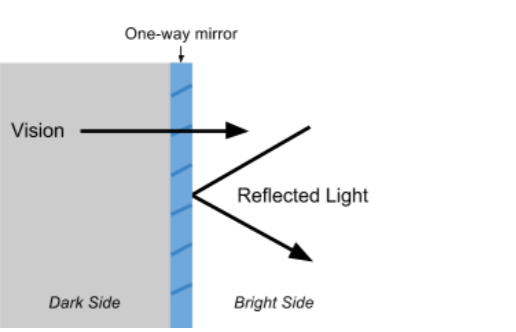
**BUILDING A SMART MIRROR**

**HARDWARE**

For the hardware I used a computer monitor, a 50x90x0.5cm one-­way mirror a monitor screen. Everything was put together in a wooden frame. The device has two wooden parts. The back part holds the display and computer and is used to support the device so that it can be hung on a wall. The frame is attached to the glass 15 by two small wooden slats and it has four holes, two on each side, that contain the ultrasound sensors. The frame can be attached and detached from the back part so it’s easy to change the glass or even the whole frame. See appendices 1 and 2. A breakdown of each of the main parts of the smart mirror (the one­way mirror glass, display, and frame)

**One­way mirror**

This is probably the most important part of the hardware because it’s responsible for creating the futuristic effect and is the biggest part of the smart mirror. Wikipedia provides the following definition: A one­way mirror, sometimes called two­way mirror, is a mirror that is partially reflective and partially transparent. When one side of the mirror is brightly lit and the other is dark, it allows viewing from the darkened side but not vice versa



Schematic diagram of light reflection on a one­way mirror

In the case of this project this essentially means that the dark or black parts of the screen will be seen as a reflection and the light parts will be seen normally. So if there is white text over a black background the white text will be seen as an overlay with the user reflected in the background. 16 This was the most difficult component to find because of these technical requirements, but a one-way mirror was eventually found at a nearby glass store. The one that was bought was unfortunately not very reflective so sometimes you can see the interior of the device. This is not ideal but in the right conditions it works well and it can always be replaced with better quality glass in the future.

**Display**

For the display a 24-inch monitor was bought, which also has built-in speakers and comes with a remote control which is useful to easily turn off the device’s screen. The monitor is much smaller than the mirror so a black sticker was used to cover the parts of the glass which are not covered by the display. An HDMI cable was used to connect the display to the laptop for video and audio.

**SOFTWARE**

**Visual Studio (full version)** is a "full-featured" and "convenient" development environment.

**Visual Studio (free "Express" versions - only until 2017)** are feature-centered and simplified versions of the full version. Feature-centered meaning that there are different versions (Visual Studio Web Developer, Visual Studio C#, etc.) depending on your goal.

**Visual Studio (free Community edition - since 2015)** is a simplified version of the full version and replaces the separated express editions used before 2015.

**Visual Studio Code (VSCode)** is a cross-platform (Linux, Mac OS, Windows) editor that can be extended with plugins to your needs.

**DEVELOPMENT TOOLS**

Taking advantage of the fact that I already had an operating system running on the laptop, I gave myself the challenge of writing all the code for the Smart Mirror on the same device. I installed Visual Studio Code, which is a very lightweight IDE, and I used it to write all the, JavaScript, Python code. In the end, the entire coding for the software was done on the Python and I only used my Windows laptop to create icons and designs with Illustrator and Photoshop. It turned out to be very convenient to be able to easily test the software directly on the Smart Mirror.

**Python**

Python is a high­level, general purpose, interpreted programming language

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**User Interface**

The user interface for the OS is clean and simple. It has an overlaid status bar on the top with the time on the right corner and Weather on Left Side , a status message in the center, Top 5 News on bottom Left Corner , and Mic icon on Bottom right corner . The status bar is dynamic and changes depending in the context: it can be hidden in case we want watch something in fullscreen or expanded to show important information.

The center of the screen shows the current running app and there currently are no guidelines for UI so developers can show anything they want there. In the example we see the Home app. Finally, In the bottom part of the screen there’s an overlaid microphone icon that pops up when the voice recognition is triggered. This user interface is completely responsive so it’s possible to have different sized mirrors and the OS will adapt to it automatically. The included default apps are also responsive but It’s up to app developers to implement this feature. Voice Input The voice recognition feature in MirrorOS uses an online API made by Google. The API is not officially supported and it has a 50 query a­ day limit but it is the best one available.

**Gesture**

Input The initial idea was to implement this feature using a camera. There were two options a regular USB webcam or the dedicated Pi Cam board which has direct access to the Pi’s hardware and provides much higher frame­rate than a USB Cam. However, after doing some tests with OpenCV I found that it was not trivial to detect hands and gestures and it depended a lot on the lighting of the room so I decided to look for alternatives. . The board, however, detected gestures from a very small distance so it wasn’t ideal for my project.

Finally I decided that I would put two ultrasonic distance sensors on each side of the mirror. This would allow me to detect holding a hand in front of the mirror and hopefully detect left and right swipes. I bought the two sensors and once I got them I wrote a python script (see appendix 8) to detect the different gestures. The resulting program was quite good although swipe gestures are not very reliable. To integrate the program with the OS I used the same NodeJS spawn child process feature that I used with the voice recognition scripts.

**HARDWARE REQUIREMENTS**

The hardware looks very good but the glass could be more reflective. In some conditions you can see a bit of the interior of the device.

* System Processors : i3 10 generation.
* Speed : 2.4 GHz
* Hard Disk : 1TB
* Ram : 8GB

**SOFTWARE REQUIREMENTS**

OS and Home app The operating system’s UI simply consists of the top status bar and the bottom microphone icon. The simple Home app works as intended, showing time and weather information and being able to set reminders, timers and notes, latest news, Runtime Date and Time.

* Operating system : 64bit Windows 7 and on words
* Coding Language : Python
* IDE: Visual Studio Code

**Code:**

# smartmirror.py

# requirements

# requests, feedparser, traceback, Pillow

from threading import Thread

from time import sleep

import pyttsx3

import datetime

from speech\_recognition import Microphone, Recognizer, AudioFile, UnknownValueError

from gtts import gTTS

import os

from tkinter import \*

import locale

import threading

import time

import requests

import json

import traceback

import feedparser

import webbrowser

from PIL import Image, ImageTk

from contextlib import contextmanager

from chatterbot import ChatBot

from chatterbot.trainers import ListTrainer

LOCALE\_LOCK = threading.Lock()

ui\_locale = '' # e.g. 'fr\_FR' fro French, '' as default

time\_format = 24 # 12 or 24

date\_format = "%b %d, %Y" # check python doc for strftime() for options

news\_country\_code = 'india'

weather\_api\_token = '<TOKEN>' # create account at https://darksky.net/dev/

weather\_lang = 'en' # see https://darksky.net/dev/docs/forecast for full list of language parameters values

weather\_unit = 'us' # see https://darksky.net/dev/docs/forecast for full list of unit parameters values

latitude = None # Set this if IP location lookup does not work for you (must be a string)

longitude = None # Set this if IP location lookup does not work for you (must be a string)

xlarge\_text\_size = 94

large\_text\_size = 48

medium\_text\_size = 28

small\_text\_size = 18

@contextmanager

def setlocale(name): #thread proof function to work with locale

    with LOCALE\_LOCK:

        saved = locale.setlocale(locale.LC\_ALL)

        try:

            yield locale.setlocale(locale.LC\_ALL, name)

        finally:

            locale.setlocale(locale.LC\_ALL, saved)

# maps open weather icons to

# icon reading is not impacted by the 'lang' parameter

icon\_lookup = {

    'clear-day': "assets/Sun.png",  # clear sky day

    'wind': "assets/Wind.png",   #wind

    'cloudy': "assets/Cloud.png",  # cloudy day

    'partly-cloudy-day': "assets/PartlySunny.png",  # partly cloudy day

    'rain': "assets/Rain.png",  # rain day

    'snow': "assets/Snow.png",  # snow day

    'snow-thin': "assets/Snow.png",  # sleet day

    'fog': "assets/Haze.png",  # fog day

    'clear-night': "assets/Moon.png",  # clear sky night

    'partly-cloudy-night': "assets/PartlyMoon.png",  # scattered clouds night

    'thunderstorm': "assets/Storm.png",  # thunderstorm

    'tornado': "assests/Tornado.png",    # tornado

    'hail': "assests/Hail.png"  # hail

}

class Clock(Frame):

    def \_\_init\_\_(self, parent, \*args, \*\*kwargs):

        Frame.\_\_init\_\_(self, parent, bg='black')

        # initialize time label

        self.time1 = ''

        self.timeLbl = Label(self, font=('Times', large\_text\_size), fg="white", bg="black")

        self.timeLbl.pack(side=TOP, anchor=E)

        # initialize day of week

        self.day\_of\_week1 = ''

        self.dayOWLbl = Label(self, text=self.day\_of\_week1, font=('Times', small\_text\_size), fg="white", bg="black")

        self.dayOWLbl.pack(side=TOP, anchor=E)

        # initialize date label

        self.date1 = ''

        self.dateLbl = Label(self, text=self.date1, font=('Times', small\_text\_size), fg="white", bg="black")

        self.dateLbl.pack(side=TOP, anchor=E)

        self.tick()

    def tick(self):

        with setlocale(ui\_locale):

            if time\_format == 12:

                time2 = time.strftime('%I:%M %p') #hour in 12h format

            else:

                time2 = time.strftime('%H:%M') #hour in 24h format

            day\_of\_week2 = time.strftime('%A')

            date2 = time.strftime(date\_format)

            # if time string has changed, update it

            if time2 != self.time1:

                self.time1 = time2

                self.timeLbl.config(text=time2)

            if day\_of\_week2 != self.day\_of\_week1:

                self.day\_of\_week1 = day\_of\_week2

                self.dayOWLbl.config(text=day\_of\_week2)

            if date2 != self.date1:

                self.date1 = date2

                self.dateLbl.config(text=date2)

            # calls itself every 200 milliseconds

            # to update the time display as needed

            # could use >200 ms, but display gets jerky

            self.timeLbl.after(200, self.tick)

class News(Frame):

    def \_\_init\_\_(self, parent, \*args, \*\*kwargs):

        Frame.\_\_init\_\_(self, parent, \*args, \*\*kwargs)

        self.config(bg='black')

        self.title = 'News' # 'News' is more internationally generic

        self.newsLbl = Label(self, text=self.title, font=('Times', medium\_text\_size), fg="white", bg="black")

        self.newsLbl.pack(side=TOP, anchor=W)

        self.headlinesContainer = Frame(self, bg="black")

        self.headlinesContainer.pack(side=TOP)

        self.get\_headlines()

        self.image = Image. open('assets/mic.png')

        # The (450, 350) is (height, width)

        self.image = self.image.resize((50,50), Image. ANTIALIAS)

        self.my\_img = ImageTk. PhotoImage(self.image)

        '''def lis():

            recog = Recognizer()

            mic = Microphone()

            with mic:

                print("Talk")

                audio = recog.record(mic, 4)

            try:

                recognized = recog.recognize\_google(audio)

                print("you said: ",recognized)

            except UnknownValueError:

                print("Unable to recognize")

                speak("please retry")

            if recognized == "hello" or recognized == "hai" :

                speak("how are you")

            elif recognized=="what's a time" or recognized=="what's the time" or recognized=="whats a time" or recognized=="whats the time":

                l()

            elif recognized == "no thanks":

                speak("ok")

            elif recognized == "open YouTube":

                speak("ok")

                webbrowser.open('https://www.youtube.com/')

            elif recognized == "open Instagram":

                speak("ok")

                webbrowser.open('https://www.instagram.com/')

            elif recognized == "open Facebook":

                speak("ok")

                webbrowser.open('https://www.facebook.com/')

            elif recognized == "open Twitter":

                speak("ok")

                webbrowser.open('https://www.twitter.com/')

            else:

                answer = bot.get\_response(recognized)

                speak(answer)'''

        self.mi = Button(self,image=self.my\_img,command=lis,state=NORMAL).pack(side=BOTTOM,anchor=SE)

    def get\_headlines(self):

        try:

            # remove all children

            for widget in self.headlinesContainer.winfo\_children():

                widget.destroy()

            if news\_country\_code == None:

                headlines\_url = "https://news.google.com/news?ned=us&output=rss"

            else:

                headlines\_url = "https://news.google.com/news?ned=%s&output=rss" % news\_country\_code

            feed = feedparser.parse(headlines\_url)

            for post in feed.entries[0:5]:

                headline = NewsHeadline(self.headlinesContainer, post.title)

                headline.pack(side=TOP, anchor=W)

        except Exception as e:

            traceback.print\_exc()

            #print "Error: %s. Cannot get news." % e

        self.after(600000, self.get\_headlines)

class NewsHeadline(Frame):

    def \_\_init\_\_(self, parent, event\_name=""):

        Frame.\_\_init\_\_(self, parent, bg='black')

        image = Image.open("assets/Newspaper.png")

        image = image.resize((25, 25), Image.ANTIALIAS)

        image = image.convert('RGB')

        photo = ImageTk.PhotoImage(image)

        self.iconLbl = Label(self, bg='black', image=photo)

        self.iconLbl.image = photo

        self.iconLbl.pack(side=LEFT, anchor=N)

        self.eventName = event\_name

        self.eventNameLbl = Label(self, text=self.eventName, font=('Times', small\_text\_size), fg="white", bg="black")

        self.eventNameLbl.pack(side=LEFT, anchor=N)

class Calendar(Frame):

    def \_\_init\_\_(self, parent, \*args, \*\*kwargs):

        Frame.\_\_init\_\_(self, parent, bg='black')

        self.title = 'Calendar Events'

        self.calendarLbl = Label(self, text=self.title, font=('Times', medium\_text\_size), fg="white", bg="black")

        self.calendarLbl.pack(side=TOP, anchor=E)

        self.calendarEventContainer = Frame(self, bg='black')

        self.calendarEventContainer.pack(side=TOP, anchor=E)

        self.get\_events()

    def get\_events(self):

        #TODO: implement this method

        # reference https://developers.google.com/google-apps/calendar/quickstart/python

        # remove all children

        for widget in self.calendarEventContainer.winfo\_children():

            widget.destroy()

        calendar\_event = CalendarEvent(self.calendarEventContainer)

        calendar\_event.pack(side=TOP, anchor=E)

        pass

class CalendarEvent(Frame):

    def \_\_init\_\_(self, parent, event\_name="Event 1"):

        Frame.\_\_init\_\_(self, parent, bg='black')

        self.eventName = event\_name

        self.eventNameLbl = Label(self, text=self.eventName, font=('Helvetica', small\_text\_size), fg="white", bg="black")

        self.eventNameLbl.pack(side=TOP, anchor=E)

class Weather(Frame):

    def \_\_init\_\_(self, parent, \*args, \*\*kwargs):

        Frame.\_\_init\_\_(self, parent, bg='black')

        api\_address='http://api.openweathermap.org/data/2.5/weather?appid=0053e9bbb2858993046d610bd0b72d89&q=chinchwad'

        json\_data = requests.get(api\_address).json()

        format\_add = json\_data['main']

        temp1 = int(float(format\_add["temp"])-273.15)

        w = json\_data['weather']

        wt = w[0]

        weather = wt["main"]

        dec = wt["description"]

        print(weather)

        print(temp1)

        temp = str(temp1)+"°C"

        if weather == "Cloud":

            self.image = Image. open('assets/Cloud.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            #print(temp)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather == "Rain":

            self.image = Image. open('assets/Rain.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            #print(temp)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Moon":

            self.image = Image. open('assets/Moon.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

            #print(temp)

        elif weather =="Sun":

            self.image = Image. open('assets/Sun.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Hail":

            self.image = Image. open('assets/Hail.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Wind":

            self.image = Image. open('assets/Wind.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Sunrise":

            self.image = Image. open('assets/Sunrise.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="PartlyMoon":

            self.image = Image. open('assets/PartlyMoon.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="PartlySunny":

            self.image = Image. open('assets/PartlySunny.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Snow":

            self.image = Image. open('assets/Snow.png')

            # The (450, d350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Storm":

            self.image = Image. open('assets/Storm.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Tornado":

            self.image = Image. open('assets/Tornado.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Haze":

            self.image = Image. open('assets/Haze.png')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        elif weather =="Clear":

            self.image = Image. open('assets/Clear.jpg')

            # The (450, 350) is (height, width)

            self.image = self.image.resize((50,50), Image. ANTIALIAS)

            self.my\_img = ImageTk. PhotoImage(self.image)

            self.t = Label(self, image=self.my\_img,justify=LEFT,padx = 0, pady = 0).pack(side="left")

        self.tl = Label(self,text="Chinchwad, Pune \n"+weather+",\n"+dec,bg="black",width=15,fg="white",font=("Times", 10)).pack(side=LEFT, anchor=NW)

        self.tl = Label(self,text=temp,bg="black",width=4,fg="white",font=("Helvetica", 44)).pack(side=LEFT, anchor=NW)

class FullscreenWindow:

    def \_\_init\_\_(self):

        self.tk = Tk()

        self.tk.configure(background='black')

        self.topFrame = Frame(self.tk, background = 'black')

        self.bottomFrame = Frame(self.tk, background = 'black')

        self.topFrame.pack(side = TOP, fill=BOTH, expand = YES)

        self.bottomFrame.pack(side = BOTTOM, fill=BOTH, expand = YES)

        self.state = False

        self.tk.bind("<Return>", self.toggle\_fullscreen)

        self.tk.bind("<Escape>", self.end\_fullscreen)

        # clock

        self.clock = Clock(self.topFrame)

        self.clock.pack(side=RIGHT, anchor=N, padx=100, pady=60)

        # weather

        self.weather = Weather(self.topFrame)

        self.weather.pack(side=LEFT, anchor=N, padx=100, pady=60)

        # news

        self.news = News(self.bottomFrame)

        self.news.pack(side=LEFT, anchor=S, padx=100, pady=60)

        # calender - removing for now

        self.calender = Calendar(self.bottomFrame)

        self.calender.pack(side = RIGHT, anchor=S, padx=100, pady=60)

    def toggle\_fullscreen(self, event=None):

        self.state = not self.state  # Just toggling the boolean

        self.tk.attributes("-fullscreen", self.state)

        return "break"

    def end\_fullscreen(self, event=None):

        self.state = False

        self.tk.attributes("-fullscreen", False)

        return "break"

    #w = FullscreenWindow()

    #w.tk.mainloop()

engine = pyttsx3.init('sapi5')

voices = engine.getProperty('voices')

print(voices[1].id)

engine.setProperty('voice', voices[1].id)

engine. setProperty("rate", 150)

def speak(audio):

    engine.say(audio)

    engine.runAndWait()

    pass

def l():

    hour = str(datetime.datetime.now().hour)

    minuites = str(datetime.datetime.now().minute)

    sec = str(datetime.datetime.now().second)

    time = hour+"hour"+minuites+"minuites"+sec+"seconds"

    speak(time)

    pass

def wishMe():

    hour = int(datetime.datetime.now().hour)

    if hour>=0 and hour<12:

        speak("Good Morning!")

    elif hour>=12 and hour<18:

        speak("Good Afternoon!")

    else:

        speak("Good Evening!")

    pass

def lis():

    recog = Recognizer()

    mic = Microphone()

    with mic:

        print("Talk")

        audio = recog.record(mic, 4)

    try:

        recognized = recog.recognize\_google(audio)

        print("you said: ",recognized)

    except UnknownValueError:

        print("Unable to recognize")

        speak("please retry")

    if recognized == "hello" or recognized == "hai" :

        speak("how are you")

    elif recognized=="what's a time" or recognized=="what's the time" or recognized=="whats a time" or recognized=="whats the time":

        l()

    elif recognized == "no thanks":

        speak("ok")

    elif recognized == "open YouTube":

        speak("ok")

        webbrowser.open('https://www.youtube.com/')

    elif recognized == "open Instagram":

        speak("ok")

        webbrowser.open('https://www.instagram.com/')

    elif recognized == "open Facebook":

        speak("ok")

        webbrowser.open('https://www.facebook.com/')

    elif recognized == "open Twitter":

        speak("ok")

        webbrowser.open('https://www.twitter.com/')

    else:

        answer = bot.get\_response(recognized)

        speak(answer)

bot = ChatBot("My Bot")

convo = [

    'hello',

    'hi there !',

    'What’s your name?',

    'I am your Personal AI',

    'Where are you from?',

    'I’m in your device',

    'What is your surname?',

    'I dont have Surname',

    'good morning?',

    'Good morning to you too',

    'how do you do?',

    'how are you?',

    'i am cool.'

    'fine, you?',

    'always cool.',

    'i am ok',

    'glad to hear that.',

    'i am fine',

    'glad to hear that.',

    'i feel awesome'

    'excellent, glad to hear that.',

    'not so good',

    'sorry to hear that.',

    'how are you doing?',

    'I am doing very well thank you for asking',

    'thank you?',

    'my pleasure'

]

trainer = ListTrainer(bot)

# now training the bot with the help of trainer

trainer.train(convo)

w = FullscreenWindow()

ws = w.tk.winfo\_screenwidth()

h = w.tk.winfo\_screenheight()

w.tk.geometry("%dx%d+0+0" % (ws, h))

photo = PhotoImage(file = "assets/logo.png")

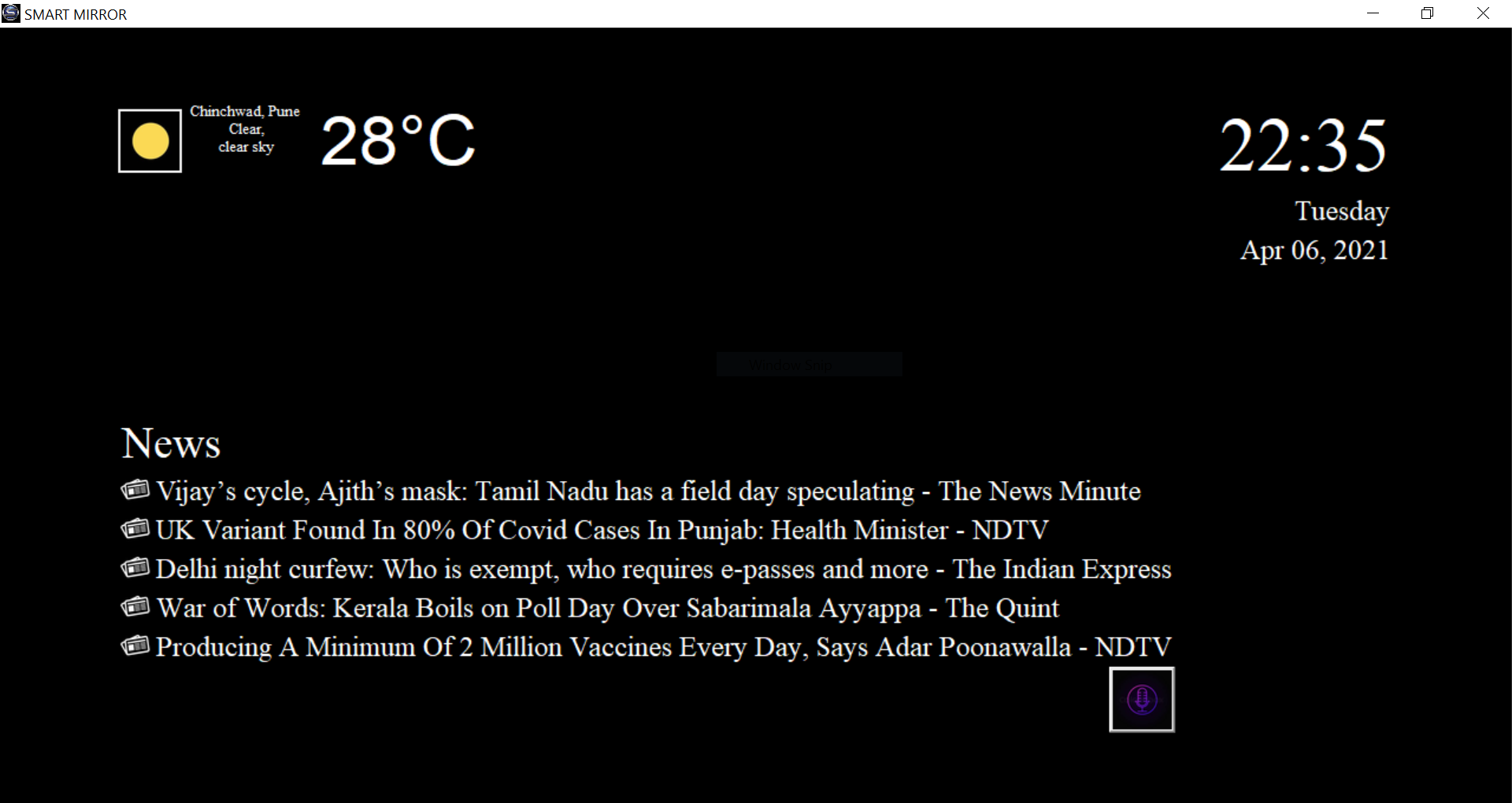
w.tk.iconphoto(False, photo)

w.tk.title("SMART MIRROR")

w.tk.overrideredirect(1)

w.tk.mainloop()

**Output:**



**EXISTING SYSTEM:**

Several efforts have been made pursuing the objective of adding special capabilities to mirrors, both commercial and research-based approaches.

But such systems are Infeasible to implement such applications in a normally owing to large space requirement.

**DISADVANTAGES OF EXISTING SYSTEM:**

* Infeasible/ illogical to implement such applications in a normal house owing to large space requirement.
* Secondly, an overlapped image is formed between the project and mirror.
* The project performs poorly in a relatively bright environment due to the influence of reflected light.

**PROPOSED SYSTEM:**

* The proposed smart mirror system aims to provide users with an interactive interface for simplified and personalized services in the comfort of the user’s home. It is a smart and user friendly solution presented in the form of a mirror that also acts as a gateway to interactive services, particularly those of information oriented nature, such as multimedia and news feed among others. Hence, the proposed system allows users access to customizable services, all while they are performing other tasks (i.e. grooming). As such, it serves as a convenient time-saver using web scraping.

**ADVANTAGE OF PROPOSED SYSTEM:**

## The system act as an interactive mirror interface.

## Distinguish between all users and provide the corresponding customized services.

## Allow for custom user profile management where the user creates his/her own profile that is to be stored on the database server of the proposed system.

## The Magic Mirror dashboard shows the weather, calendar you choose, current time by using python tools.

## This product would be useful for busy individuals that want to multitask and stay informed while on the go.

* The big advantage of a smart mirror is the ability to display useful information without needing to open apps or do anything.
* You simply look at your smart mirror and the information is there.
* Smart Mirrors are interactive devices that helps you check updates easily with voice control.

**APPLICATIONS**

Besides our homes, Smart Mirrors can be used in various business like clothing shops, beauty shops, barber shops, kiosk mall, and many more.

**CONCLUSIONS**

The main strengths of this project are that this is a new kind of smart device that people don’t see every day and it looks very spectacular. The platform has a very simple API that makes it very easy for developers to make apps. The voice recognition is very accurate. The smartphone integration works very well and it is something that hasn’t been done with smart mirrors before. Of course there are also weaknesses: the app ecosystem is currently very small, the glass could be more reflective but it can be easily changed, the swipe gestures are sometimes unreliable and finally I would have liked to have the hardware and software more decoupled because currently the sensors and microphones are tied to the software and it can be difficult to make the OS work with different hardware. However, this can also be solved given enough time by making the software more modular.

There are many future possibilities for this project and hopefully it will be continued. For the software, It would also be good to make some changes to make it truly multiplatform. The companion app needs a new UI, maybe an app repository and also the ability to easily change settings for the mirror. A community around the OS and the hardware should be created so people can help each other build and evolve these devices and create apps for them.

Once polished, the software could be made open­source. Finally, for the hardware part, the glass panel could be replaced for a more reflective.

**FINAL THOUGHTS**

This has been a very satisfying project to work on. It has been enriching and most importantly, I had fun doing everything and for me this is very important because it motivated me to continue developing it and to keep adding features. Developing this project I have learned many fascinating things. I feel like this list would be endless but I will try to explain some of the most important ones. Electron is a very interesting project that is growing very fast and I think it will soon become the main tool for developing desktop apps.

As you can see the list of positive results is very long and I think I even left out lots of things but in general my conclusion is that I really enjoyed this project and I hope that I can continue building similar things and I’m very excited about what the future of the Maker community will bring us.