1. Introduction

1.1 Overview

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 Smartphone's or other devices.

The smart mirror is a popular project among DIY enthusiasts and it usually consists of a One way mirror with a screen attached to it that displays a static web page. However what we wanted to achieve was something you could interact with. Our goal was to learn how a Raspberry Pi worked and to understand how to combine the software and the hardware components to create a multimedia project.

1.2 Choice of Topic with reasoning

Effective time management is one of the most important factors for success and productivity in a person's day-to-day life. With the increasing integration of technology in our lives, maintaining an efficient schedule has become both easier and more difficult. Keeping up to date with appointments, Twitter, news, social media, and other things is made easier through technology such as tablets, PCs, and Smartphone yet also provide distractions that can interrupt anyone's routine. Technology has become another task in the day that time must be allotted for. In the finite time of the day, technology needs to be designed to work within our schedule and not be an extra piece to it.

The key to effective time management involving technology is multitasking. Anyone in the business or academic world would agree that every second count in the day. This project was formulated through inspiration seen through movies such as Iron Man and tech demos, such as Samsung's transparent LCD Smart Window, seen at the International Consumer Electronics Show in 2012. This extends as well to the continuing trend of integrating touch screens and internet-connectivity into everyday appliances such as ovens and refrigerators. The idea of a smart home is the direction lots of companies are heading and while the kitchen has been getting lots of attention, the bathroom has not. Besides the kitchen, the bathroom is one of the busiest rooms in the home, so it is an excellent place to expand the smart home next.

Constant information and instant access to it drive the current generation. Forget bringing smart phones and tablets into the bathroom and risking damage. The smart mirror

will show you that information with the swipe of a hand. The smart mirror is the result of our team brainstorming on how to solve all these issues and develop something that is functional as well as a showpiece.

The smart mirror is definitely not a true consumer product yet. There are very few truly manufactured and ready for sale smart mirrors in the market. Those that are there are very different in terms of functionality, development, and price. It is certainly going to take a large smart home company to get behind this product and make it main stream to the consumer. Hence this Project can turn into a great Startup Idea.

1.3 Problem Statement

There is plethora of Smart Mirrors in existence by now. Mostly, developed to display time, date, and weather related information. But you will hardly find all these features in a single Smart Mirror. Especially in India, this concept has not made strides yet. Also, voice detection and activation system is not implemented into Smart Mirrors on a large scale. By implementing voice activation module, one can get instant control along with that can save electricity.

1.4 Objectives

We propose to build this smart piece of hardware by keeping in mind all the requirements and drawbacks of existing system. Along with time, date, and weather forecast related information, we also aim at implementing module which can feed news headlines from various sources. More importantly we aim at implementing Voice Detection System. This will display the feeds only when the human voice is detected and will turn off the system if no voice is detected to save electricity and CPU power.

2. Literature Review and Study

2.1 Related Work

The Smart Mirror is designed to perform several functionalities. It will mimic a natural mirror interface through a flat LCD monitor used for the display. A two-way acrylic mirror is used in front of the LCD monitor thereby mimicking the function of a regular mirror. For personalized information services the users will be able to obtain minute updates of latest news and public headlines, weather reports as well as get reports of their interests.

The proposed Smart Mirror represents an interactive interface that enables access to personalized information and services. This is to contribute to this design of a Smart Mirror-like interface as well as the smart environment in which the interface is used for interaction. In the following, we briefly discuss on some related research in this field.
Philips HomeLab is a test bed for creating perspective and context-aware home environments. Among several projects, their work on creating an intelligent personal care environment uses an Interactive Mirror in the bathroom to provide personalized services according to the user's preferences. The mirror can provide live TV feeds, monitor the latest weather, and so on. The mirror is a combination of one or more LCD flat screen displays specifically combined with a mirrored surface and connected with a central processor to provide the intended services. In addition, we use open standards like web services to communicate with the devices and customize various personalized services for the user.

Interactive Mirror is a touch and gesture functional mirror created by Alpay Kasal and Sam Ewen of Lit Studios. The user touches the mirror, which has a built in touch screen, to interact with it. Also, this mirror is less about data and more about artsy visuals. Users in the demo video show off different types of drawing and 2D games that are displayed using a projector. The fact that it emulates a mouse is nice because of the expandability and the range of functions capable. Yet, this still differs from our Smart Mirror since this is used only as a source of entertainment but our Smart Mirror also provides solution to many problems faced by a user on a daily basis.

The HUD Mirror was designed by five students for a course at the Chalmers University of Technology in Sweden . They used a two-way mirror to allow the LEDs they mounted behind to illuminate information through the mirror. This mirror was made for the bathroom and displayed time, weather, outside temperature, and a toothbrush timer by use of the LEDs. Also, instead of using a touch screen for interaction, they used Light Dependent Resistors (LDRs) as buttons behind the mirror. When “touched”, the light changes and can perform a function specified in the Arduino software. Despite being simpler, the HUD Mirror has a lot of the same ideas as the Smart Mirror.

The Magic Mirror was developed by the New York Times Research and Development Lab. It uses a TV with a mirror finish and uses a Microsoft Kinect to track movement and take in voice recognition. Also, it integrated a RFID reader to identify certain

bathroom products. The whole system is run from a Windows PC. The fact it can keep track of prescriptions and use the Kinect to “virtually” put clothes on the user are very inspiring features. The Magic Mirror also allows the ability to check email, calendars, and social media, which are implemented in our Smart Mirror as well.

The Smart Mirror is definitely not a true consumer product yet. There are very few truly manufactured and ready for sale Smart Mirrors in the market. Those that are there are very different in terms of functionality, development, and price. It is certainly going to take a large smart home company to get behind this product and make it mainstream to the consumer. Our Smart Mirror has combined a few features that have been stated in the above paragraphs yet it stands out as we have incorporated voice activation to access said features and their customization through a Remote Access Tool that is designed and developed through the help of web programming and android application.

2.2 Literature Survey

"A Mobile-Programmable Smart mirror for Ambient IoT Environment" published at 5th International Conference on Future Internet of Things and Cloud Workshops in 2017 describes the design and development of Interactive Smart mirror that offers simplified and customizable services to the home environment. The Smart mirror also controls home appliances with very less human intervention using a mobile application. For controlling home appliances the mobile needs to be paired with the smart mirror successfully.

The "Implementation and Customization of a Smart Mirror through a Facial Recognition Authentication and a Personalized News Recommendation Algorithm published at 13th International Conference on Signal- Image Technology & Internet-Based Systems (SITIS) in 2017 includes the above advancement. The daily news recommendation predictive model is implemented through the facial recognition algorithm.

The "SmiWork: An Interactive Smart Mirror Platform for Workplace Health Promotion" describes about a multi-user Smart mirror that promotes wellness and healthier lifestyle. Each user have personalized user-interface which can be accessed using RFID reader in ID card.

"The Smart Mirror" published at International Journal of Advance Research, Ideas and Innovations in Technology and "Design and development of a smart mirror using Raspberry pi" published at International Journal Of Electrical, Electronics And Data Communication gives the design of futuristic mirror and development of mirror using Raspberry pi.

2.3 Study

A. Speech Recantation

Speech recognition is invading our lives. It's built into our phones, our game consoles and our smart watches. It's even automating our homes. But speech recognition has been around.

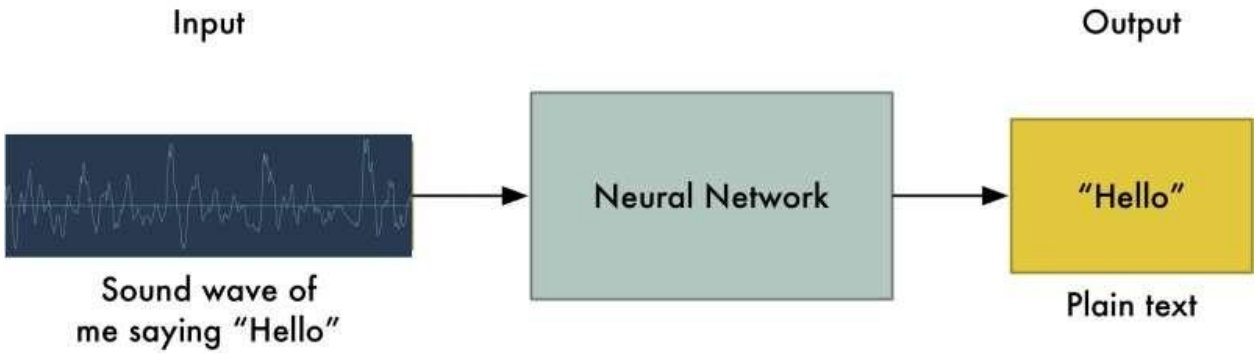


Figure 1: basic structure of Voice Recognition

1. Turning Sounds into Bits

The first step in speech recognition is obvious—we need to feed sound waves into a computer. But sound is transmitted as waves. How do we turn sound waves into numbers? Let's use this sound clip saying "Hello":

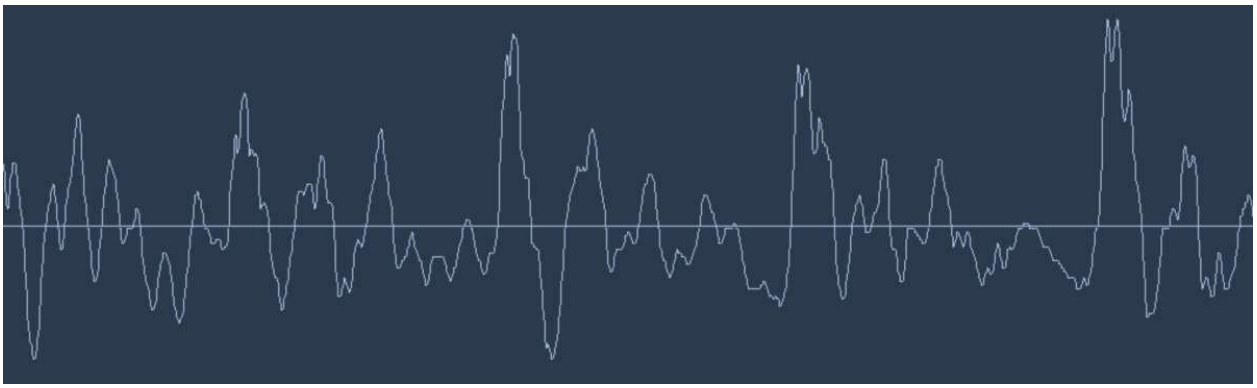


Figure 2 waveform of 'Hello'

Sound waves are one-dimensional. At every moment in time, they have a single value based on the height of the wave. Let's zoom in on one tiny part of the sound wave and take a look:

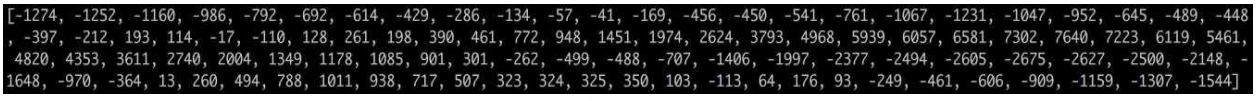


Figure 3 sampling a sound wave

This is called *sampling*. We are taking reading thousands of times a second and recording a number representing the height of the sound wave at that point in time. That's basically all an uncompressed .wav audio file is

2. A Quick Sidebar on Digital Sampling

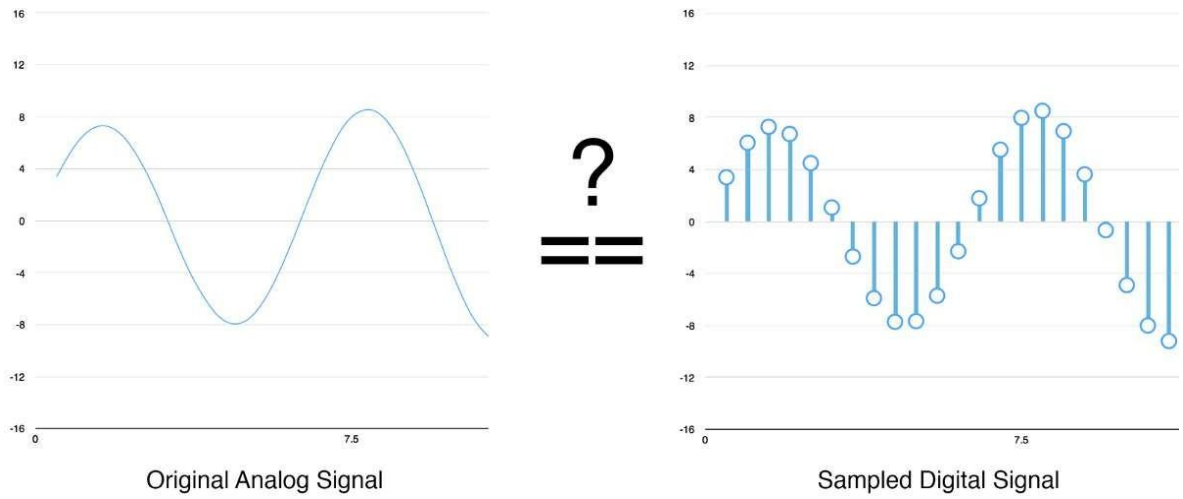


Figure 4: Each number represents the amplitude of the sound wave

You might be thinking that sampling is only creating a rough approximation of the original sound wave because it's only taking occasional readings. There's gaps in between our readings so we must be losing data, right?

Can digital samples perfectly recreate the original analog sound wave? What about those gaps? But thanks to the Nyquist theorem, we know that we can use math to perfectly reconstruct the original sound wave from the spaced-out samples—as long as we sample at least twice as fast as the highest frequency we want to record.

I mention this only because nearly everyone gets this wrong and assumes that using higher sampling rates always leads to better audio quality. It doesn't.

3. Pre-processing our Sampled Sound Data

We now have an array of numbers with each number representing the sound wave's amplitude at 1/16,000th of a second intervals. We *could* feed these numbers right into a neural network. But trying to recognize speech patterns by processing these samples directly is difficult. Instead, we can make the problem easier by doing some pre-processing on the audio data.

Let's start by grouping our sampled audio into 20-millisecond-long chunks. Here are our first 20 milliseconds of audio (i.e., our first 320 samples):

Plotting those numbers as a simple line graph gives us a rough approximation of the original sound wave for that 20 millisecond period of time:

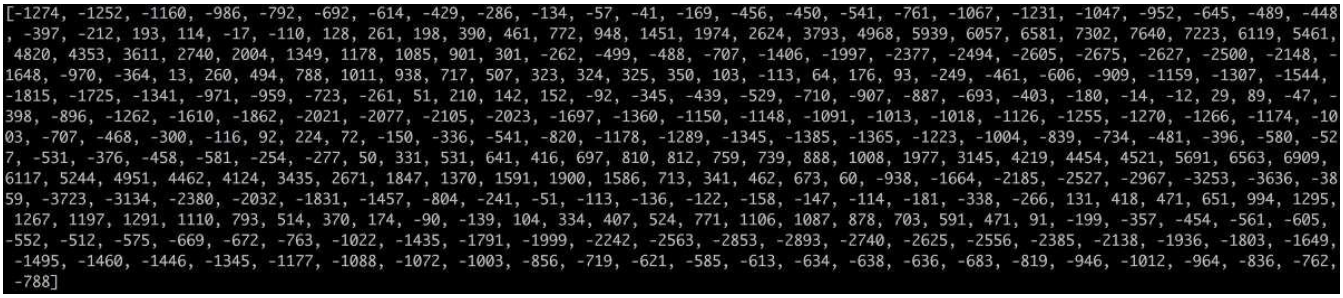


Figure 5: Can digital samples perfectly recreate the original analog sound wave?

This recording is only *1/50th of a second long*. But even this short recording is a complex mish-mash of different frequencies of sound. There's some low sounds, some mid-range sounds, and even some high-pitched sounds sprinkled in. But taken all together, these different frequencies mix together to make up the complex sound of human speech.

To make this data easier for a neural network to process, we are going to break apart this complex sound wave into its component parts. We'll break out the low-pitched parts, the next-lowest- pitched-parts, and so on. Then by adding up how much energy is in each of those frequency bands (from low to high), we create a *fingerprint* of sorts for this audio snippet.

Imagine you had a recording of someone playing a C Major chord on a piano. That sound is the combination of three musical notes— C, E and G—all mixed together into one complex sound. We want to break apart that complex sound into the individual notes to discover that they were C, E and G. This is the exact same idea.

We do this using a mathematic operation called a *Fourier transform*. It breaks apart the complex sound wave into the simple sound waves that make it up. Once we have those individual sound waves, we add up how much energy is contained in each one.

The end result is a score of how important each frequency range is, from low pitch (i.e. bass notes) to high pitch. Each number below represents how much energy was in each 50 Hz band of our 20 millisecond audio clip:

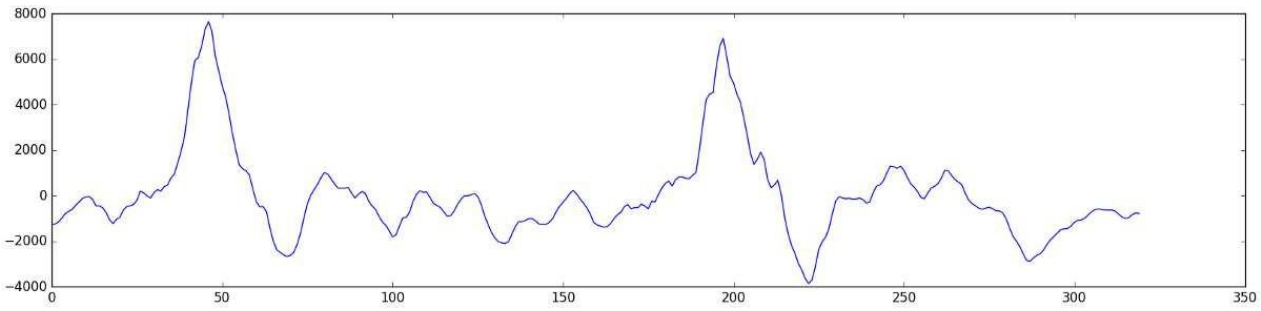


Figure 6: Original sound wave

Each number in the list represents how much energy was in that 50 Hz frequency band but this is a lot easier to see when you draw this as a chart:

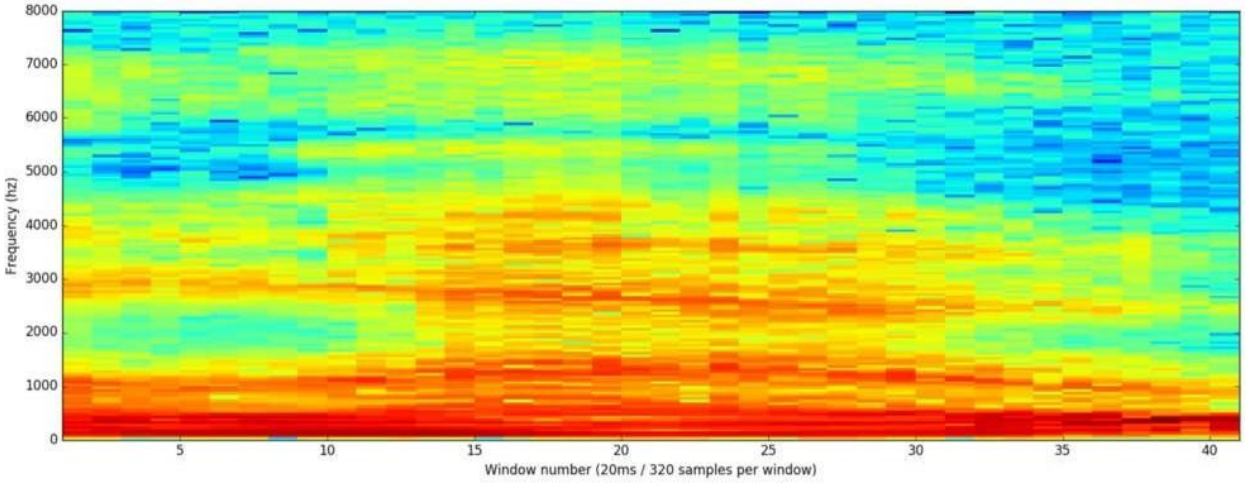


Figure 7: Male Voice

You can see that our 20 millisecond sound snippet has a lot of low-frequency energy and not much energy in the higher frequencies. That's typical of "male" voices.

If we repeat this process on every 20 millisecond chunk of audio, we end up with a spectrogram (each column from left-to-right is one 20ms chunk):

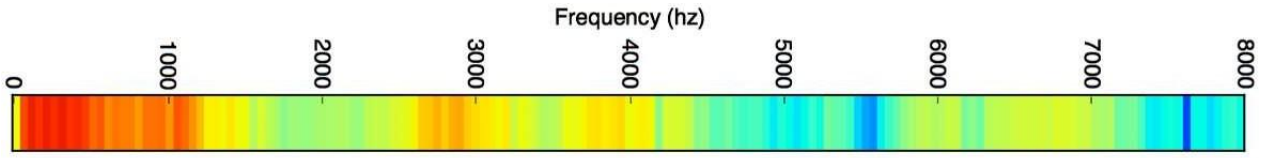


Figure 8: The full spectrogram of the "hello" sound clip

A spectrogram is cool because you can actually *see* musical notes and other pitch

patterns in audio data. A neural network can find patterns in this kind of data more easily than raw sound waves. So this is the data representation we'll actually feed into our neural network.

4. Recognizing Characters from Short Sounds

Now that we have our audio in a format that's easy to process, we will feed it into a deep neural network. The input to the neural network will be 20 millisecond audio chunks. For each little audio slice, it will try to figure out the letter that corresponds the sound currently being spoken.

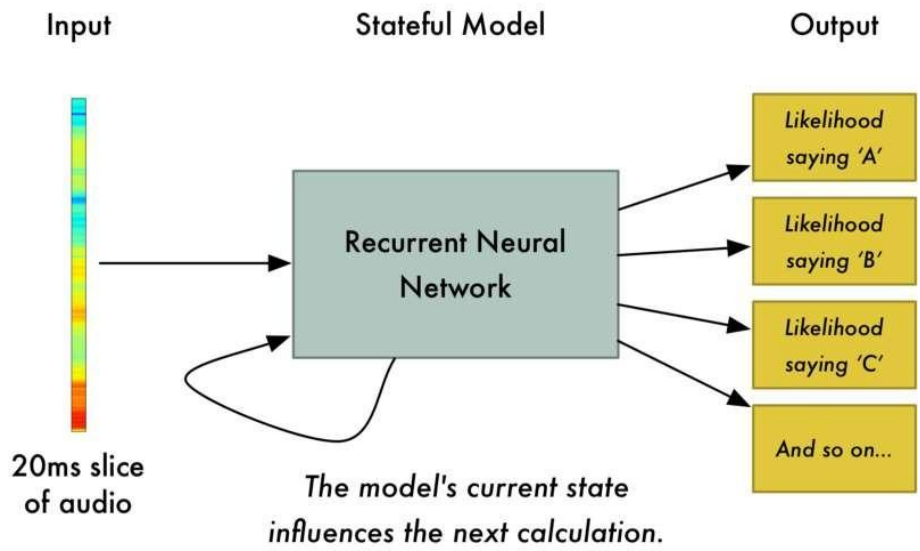


Figure 9: Recognizing Characters

We'll use a recurrent neural network—that is, a neural network that has a memory that influences future predictions. That's because each letter it predicts should affect the likelihood of the next letter it will predict too. For example, if we have said "HEL" so far, it's very likely we will say "LO" next to finish out the word "Hello". It's much less likely that we will say something unpronounceable next like "XYZ". So having that memory of previous predictions helps the neural network make more accurate predictions going forward.

After we run our entire audio clip through the neural network (one chunk at a time), we'll end up with a mapping of each audio chunk to the letters most likely spoken during that chunk. Here's what that mapping looks like for me saying "Hello":

Our neural net is predicting that one likely thing I said was "HHHEE_LL_LLLOOO". But it also thinks that it was possible that I said "HHHUU_LL_LLLOOO" or even

“AAAUU_LL_LLLOOO”.

We have some steps we follow to clean up this output. First, we’ll replace any repeated characters a single character:

- HHHEE_LL_LLLOOO becomes HE_L_LO
- HHHUU_LL_LLLOOO becomes HU_L_LO
- AAAUU_LL_LLLOOO becomes AU_L_LO

Then we’ll remove any blanks:

- HE_L_LO becomes HELLO
- HU_L_LO becomes HULLO
- AU_L_LO becomes AULLO

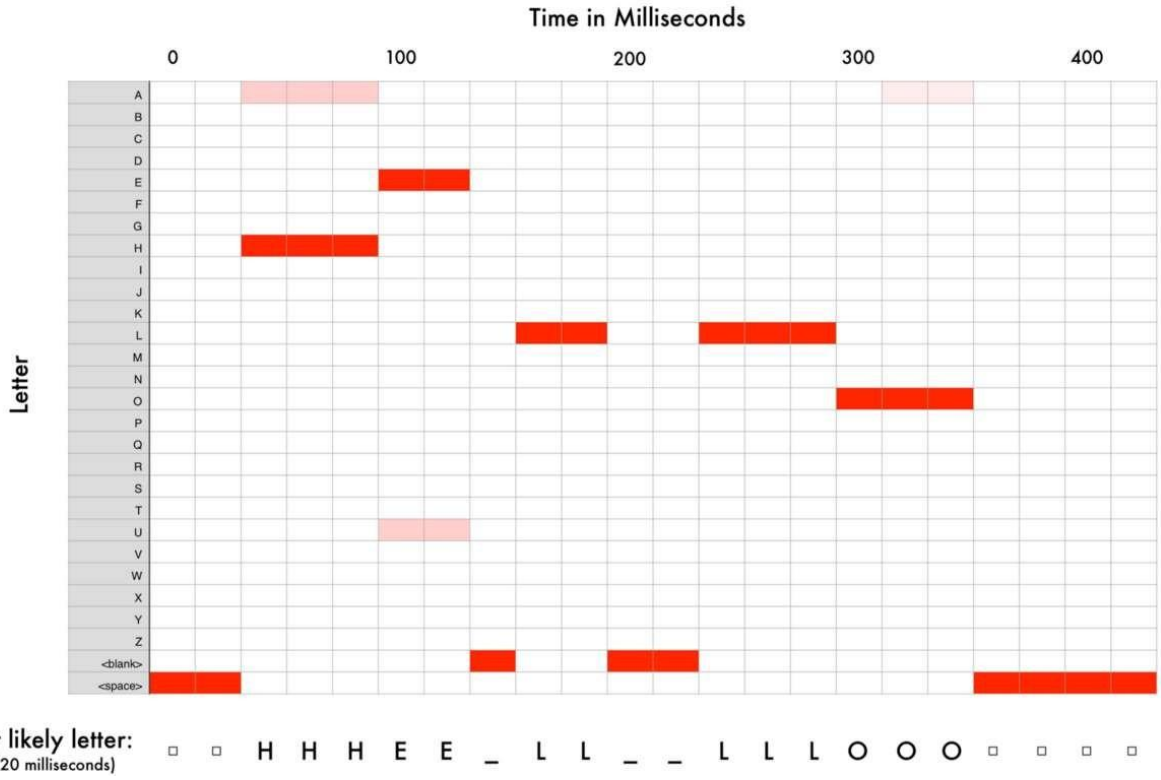


Figure 10: Mapping of 'Hello'

That leaves us with three possible transcriptions—“Hello”, “Hullo” and “Aullo”. If you say them out loud, all of these sound similar to “Hello”. Because it’s predicting one character at a time, the neural network will come up with these very sounded-out transcriptions. For example if you say “He would not go”, it might give one possible transcription as “He wud net go”.

The trick is to combine these pronunciation-based predictions with likelihood scores based on large database of written text (books, news articles, etc.). You throw out transcriptions that seem the least likely to be real and keep the transcription that seems the most realistic.

Of our possible transcriptions “Hello”, “Hullo” and “Aullo”, obviously “Hello” will appear more frequently in a database of text (not to mention in our original audio-based training data) and thus is probably correct. So we’ll pick “Hello” as our final transcription instead of the others. Done!

3. Software Requirement Specification

3.1. Functional requirements

The requirements were requirements that are imperative to the project's design and objectives. They were designed and implemented before any additional features were worked on. These necessary features are listed below

- The smart mirror is designed to use a 15.5" diagonal display, positioned vertically, which will mounted behind a one-way mirror allowing only elements lit on the screen to be seen by the user.
- The smart mirror contains speakers that allow for application notification sounds and music playback.
- The smart mirror user interface has a set of standard applications that provide important information to the user.
  Following are the proposed applications

1	Weather	Should show weather updates as per area
2	To-do-list	To-do-list and reminders for it
3	Music Player	Should play music from local machine
4	Clock	Should display time and day
5	Quotes	Should show quotes on hourly basis
6	News	Should show local news
7	Contacts	List of user contacts to send email and messages
8	Send Email	To send email to contacts
9	Send WhatsApp Messages	To send user whatsapp messages
10	Voice Assistant	Should answer questions and other system calls
11	Mobile Client	To remotely access and monitor all applications

- The smart mirror has an auto-on and auto-off system via use of a microphone mounted in the housing. The mirror will turn on when it recognizes that there is a user standing in front of the mirror. The mirror will turn-off after 2-minutes of no user present.
- The smart mirror has voice control through a voice recognition system developed into the user interface. The voice control allows for interaction with the music, to-do list, and twitter applications. The voice control is activated through a gesture provided by the user.

3.2. Environment Characteristics

A. Hardware

1. Raspberry Pi

The Raspberry Pi is a series of small single-board computers That Are Used in various IOT projects. We are using Raspberry pi in our project due to its compact size and processing power.

CPU	1.5 GHz quad-core A72 64-bit
RAM	4 GB
Storage	32GB
OS	Linux

2. Computer

Instead of Raspberry pi a normal PC can also be used.

CPU	Pentium Gold Minimum
RAM	4 GB
Storage	100 GB Minimum
OS	Linux
Motherboard	ITX or Mini ITX
Power Supply	250w Minimum

B. Peripherals

1. Speaker

Speakers are needed for playing system sounds as well output from the Voice Assistant.

2. Microphone

Microphones play a big role of capturing the voice data from the user.

3. Keyboard

Keyboard is needed for initial configuration only.

4. Mouse

Mouse is needed for initial configuration only.

C. Operating System

1. Ubuntu (Linux)

Ubuntu is a Linux distribution based on Debian mostly composed of free and open-source software. Ubuntu is officially released in three editions: Desktop, Server, and Core for the internet of things devices and robots. All the editions can run on the computer alone, or in a virtual machine. Ubuntu is a popular operating system for cloud computing, with support for OpenStack.

As Ubuntu is supported on all of the hardware's we choose Ubuntu.

4. System Design

4.1 System Architecture

The System architecture of Smart mirror is shown in below figure. The Smart mirror system mainly consists of three parts a two-way mirror, LCD monitor and Raspberry pi. The two-way mirror is the mirror which is reflective on one side and transparent on the other side. The LCD monitor is used for displaying different widgets on the mirror. The LCD monitor will be connected to the Raspberry pi. The Raspberry pi will be used for programming of different widgets using Python language. The Smart mirror will be switched on using a voice command such as "Hello Mirror!", "Good morning mirror" or any other keyword. The Smart mirror will also give voice as well as text response like greeting the user or give some compliment as response, for which the system will use system compatible microphone and speaker. The process will be firstly programming will be done for displaying images which will be displayed on LCD monitor and user will be able to see those widgets on the mirror when the Smart mirror is switched on using the keyword. The Smart mirror will also display some personal basic information only by recognizing the user's face.

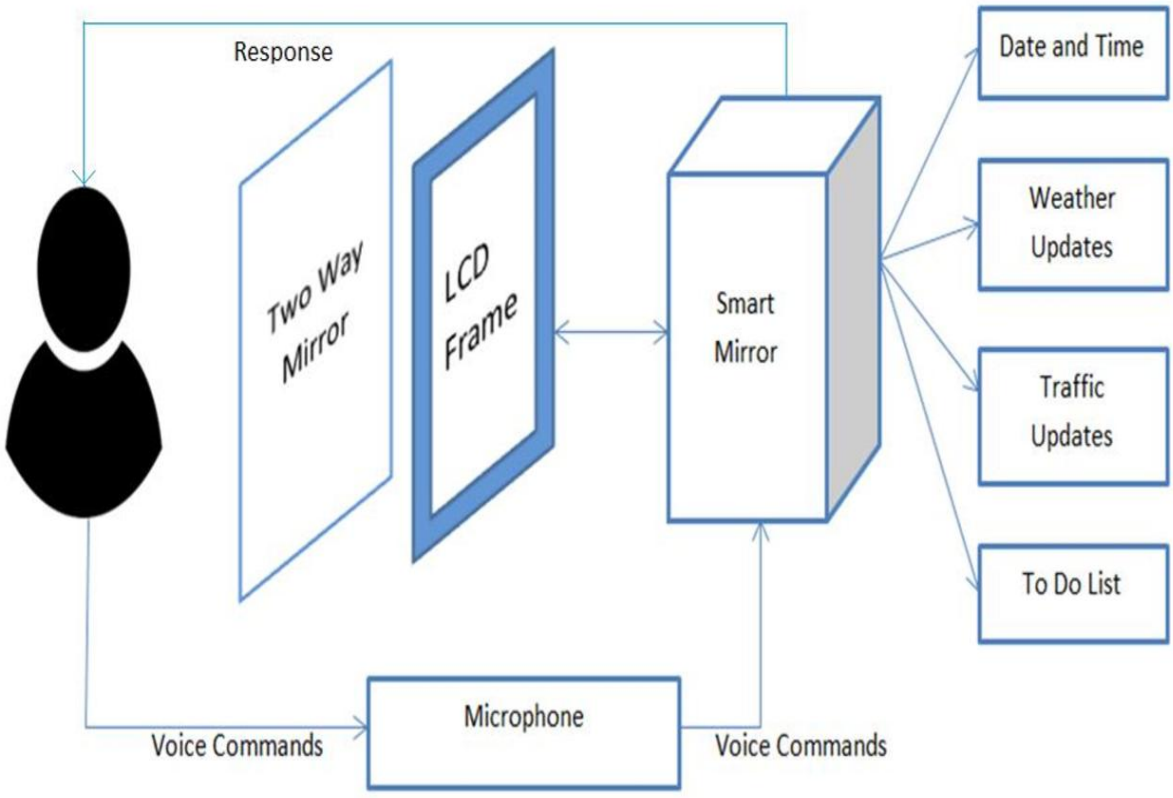


Figure 11: System Architecture

A. Data Collection and Storage unit

Various APIs are one mechanism for users for receiving updated data from data sources. It is used by real-time applications in point to point links and Products on the World Wide Web. Various APIs hold a family of standard web feed formats for publishing frequently updated information like notification in social Medias, news headlines, to-do list. We use Various APIs for retrieving data from the web. APIs allow publishers to associate data automatically. A standard JSON format ensures compatibility with distinct machines/programs. Subscribing to an API discards the need for user to check the website for new content. Alternatively, their browser constantly monitors the site and informs the user if any updates.

Some of APIs we have used

News API	Open News Org
Weather API	Open Weather Org

B. Data Processing Unit

Raspberry Pi is a small low powered single board minicomputer which capable of running an operating system like Linux. We are using Raspberry Pi 4. For making digital devices and interactive objects that can control and sense objects in the physical world, Arduino can be used; it is an open source computer h/w and s/w company that can be used in designing and manufacturing these digital devices.

Components:

- Microcontroller
- Operating Voltage
- Digital I/O Pins
- Analogue Input Pins
- DC Current per I/O Pin
- Flash Memory
- SRAM
- EEPROM

C. Data Visualization Unit

Data visualization is viewed by many disciplines as the modern equivalent of visual communication. This involves the design and study of the visual representation of data, meaning information that abstracted in some schematic form, including attributes or variables for a unit of information.

The prototype of the proposed mirror system itself is an LCD panel mounted with a

two-way mirror, in front of the monitor. If the panel is turned off, the two-way mirror acts as a normal reflective mirror. On the other hand, if the monitor gets turned on, the mirror is transparent to the viewers to see the screen of the monitor. The power of the monitor is controlled by the system based on the state of the operation. When the user comes in front of the mirror, the mirror displays information that is being fed from the web. Raspberry pi: The main task of Raspberry pi is to collect all the data and displays them into the LCD panel. It also provides the means to serve as a controller.

LCD panel: It is the main display of the smart mirror; one-way mirror is attached in front of the LCD panel.

Two-way mirror: It is the mirror in which one side is transparent and other side is reflexive where we sense us. The information that is displayed in the LCD panel can be viewed through the one-way mirror.

D. Python programming

Our programming language is python and our building area is PyCharm. Python is a widely used high level general purpose language. Its design highlights the code readability and the syntax that allows to expose concepts in fewer lines of code than would be possible in languages. Python carry an easy abstract. It features a dynamic system and automatic memory management, has large and comprehensive standard library.

Python interpreters are available for the installation for many operating systems also allowing Python code execution to a wide variety of systems. Python code can be packaged into executable programs for some operating systems, allowing the action of Python based software to use on those environments without having to install Python interpreter.

E. GUI Creation

We use Tkinter library for GUI creation. Tkinter provides a strong object-oriented interface to the Tk GUI Toolkit. Tkinter provides many widget functions on which the user interaction relays. From Tkinter import*, imports every object in Tkinter into the file. import Tkinter imports the "namespace" Tkinter in our namespace also, import Tkinter as tk does the same, but it "renames" it locally to 'tk' to save what we type.

Different Frontend Technologies as HTML, CSS, JavaScript is also Used.

4.2 Methodology/Algorithm

The proposed Smart Mirror has various functionalities implemented and is designed and developed using a building block architecture. This therefore prepares the Smart Mirror

for future scalability as it can support extended home automation. This means that through the Smart Mirror interface, in the future, other home automation IoT devices can be controlled like ambient lighting systems, centralized air conditioning systems, and even other modular home automation systems like Google Home.

Our Smart Mirror is designed to mimic a natural mirror interface. A two-way acrylic mirror is used for the mirror display, thereby mimicking the function of a regular mirror. A flat LCD monitor is used to provide real time display of the content on the Smart Mirror which is powered by a Raspberry Pi, thereby mimicking the function of a Smart Mirror.

The Smart Mirror has the functionality of many modern day Smartphone applications implemented on it in the form of widgets. Some of the applications incorporated into the working design of the Smart Mirror are those of Bing Maps (Along with real time display of traffic in a particular location), YouTube (Functionality to play videos along with the audio output), a Date and Time widget (Displays the current system date and time. It can also update this information by obtaining it from the internet), a Weather widget (Displays the weather information based on either automatically detected location, or manual location information fed to it in terms of Latitude and Longitude). The temperature units system (Fahrenheit or Celsius) displayed in the widget is based on the popular units system used in the location. This too can manually be changed by the user based on the user's units system of choice.

Users will be able to obtain minute updates of latest news and public headlines, with News widget. Along with the aforementioned widgets and applications, the Smart Mirror also has application functionality. These functionalities are those of a Reminders application which displays the user's manually set reminders. There is also a To-Do List which has a list of to-do items that the user manually inputs.

A Remote Access Tool (RAT) (Mobile Client) was also developed to help provide ease of use of the Smart Mirror to the user. The RAT is a remote web application that provides widget and application customization and visibility options to the user. This RAT is designed to be simple, easy to use, and provide all the functionality of the Smart Mirror that the user has through voice commands, through a text input interface. The RAT is a very useful tool that helps the user overcome the challenges of voice command input like false positives and true negatives.

The RAT has a variety of options for each individual application and widget that has been implemented in the Smart Mirror. The user has the ability to enable/disable any or all of the widgets of the Smart Mirror through the RAT. The RAT is an extension of the Configuration File where the aforementioned settings can be manually edited by programming them accordingly there. The user can also alter the sensitivity of the microphone through the RAT and help improve the Smart Mirror's ability to accept true positives when a voice command is given.

4.3 Data Flow Diagram

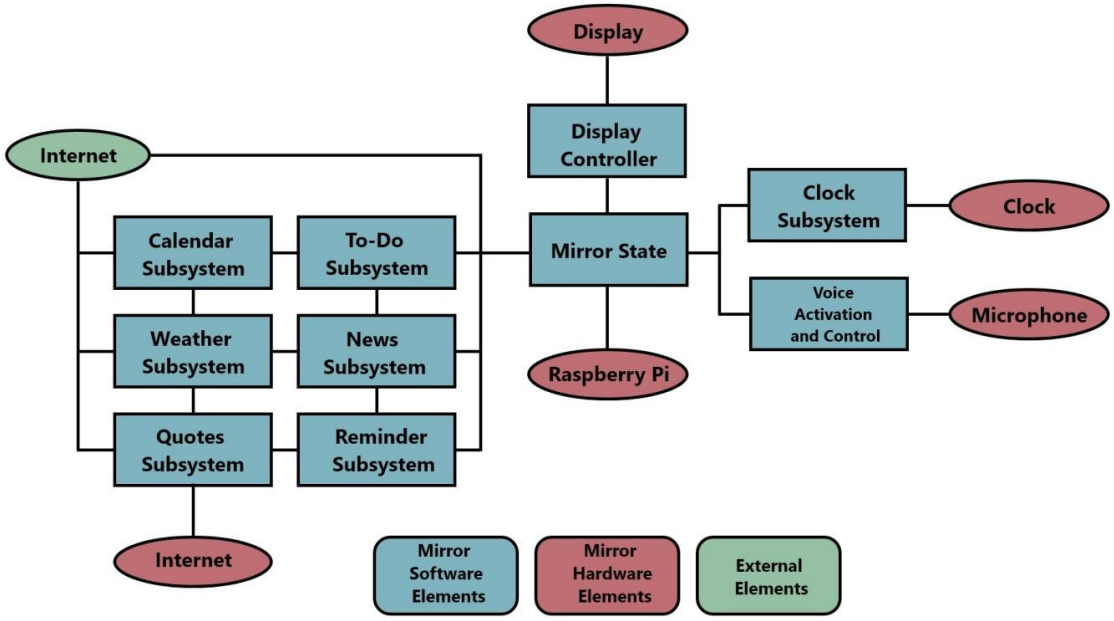


Figure 12: Data Flow Diagram

4.4 Use Case Diagram

A use case is a set of scenarios that describe an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.

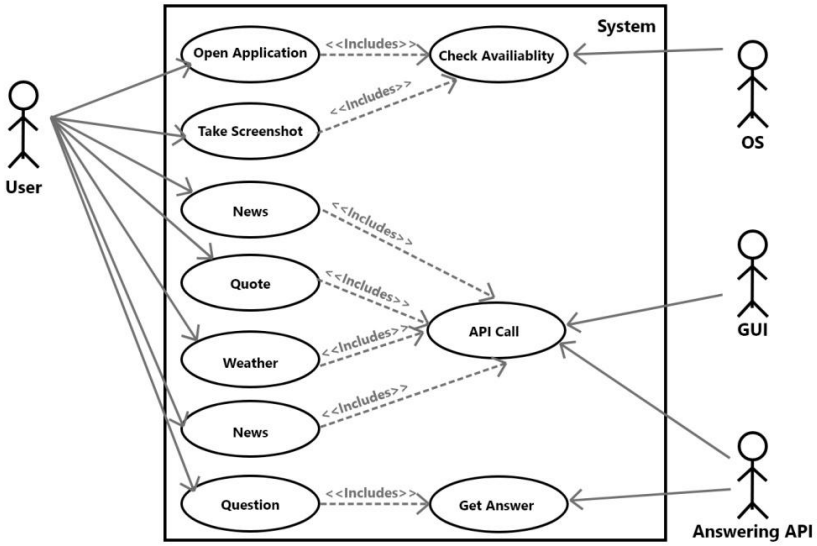


Figure 13: Use Case Diagram

4.5 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

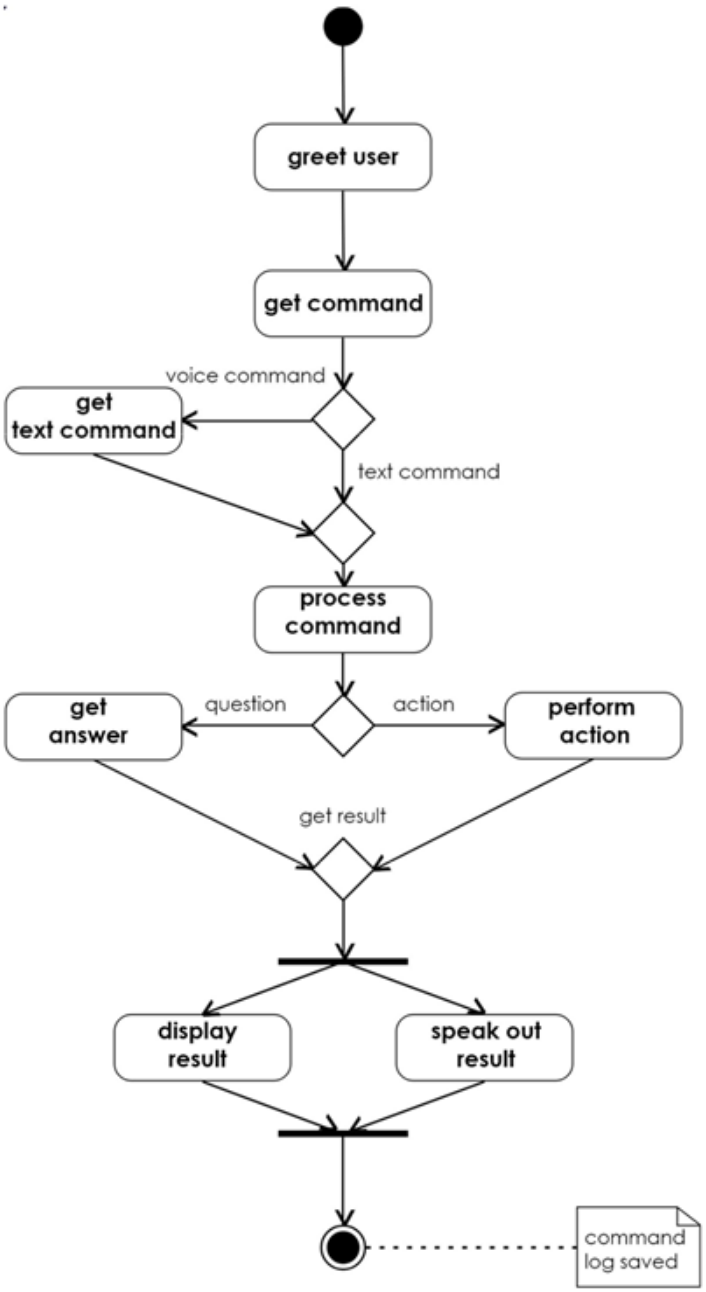


Figure 14: Activity Diagram

4.6 Sequence Diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

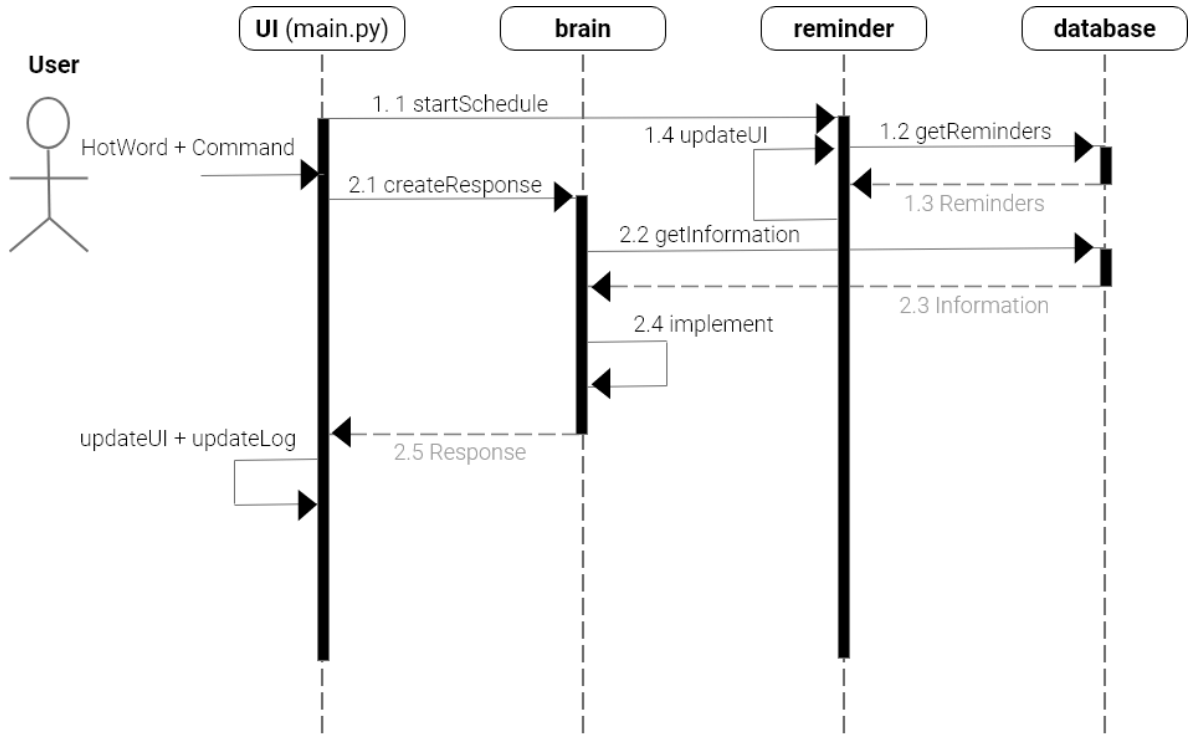


Figure 15: Sequence Diagram

5. Coding Techniques and Implementation details

5.1 Tools and Technologies Used

A. Git and Github:

Git is an example of a distributed version control system (DVCS) commonly used for open source and commercial software development. DVCSs allow full access to every file, branch, and iteration of a project, and allows every user access to a full and self-contained history of all changes. Unlike once popular centralized version control systems, DVCSs like Git don't need a constant connection to a central repository. Developers can work anywhere and collaborate asynchronously from any time zone.

Without version control, team members are subject to redundant tasks, slower timelines, and multiple copies of a single project. To eliminate unnecessary work, Git and other VCSs give each contributor a unified and consistent view of a project, surfacing work that's already in progress. Seeing a transparent history of changes, who made them, and how they contribute to the development of a project helps team members stay aligned while working independently.

GitHub is a Git hosting repository that provides developers with tools to ship better code through command line features, issues (threaded discussions), pull requests, code review, or the use of a collection of free and for-purchase apps in the GitHub Marketplace. With collaboration layers like the GitHub flow, a community of 15 million developers, and an ecosystem with hundreds of integrations, GitHub changes the way software is built.

B. Python:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatics (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum's long influence on Python is reflected in the title given to him by the Python community: Benevolent Dictator for Life (BDFL) – a post from which he gave himself permanent vacation on July 12, 2018

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.Van Rossum led the language community until stepping down as leader in July 2018.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural. It also has a comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and Metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

C. IDEs:

1. PyCharm

JetBrains has developed PyCharm as a cross-platform IDE for Python. In addition to supporting versions 2.x and 3.x of Python, PyCharm is also compatible with Windows, Linux, and macOS. At the same time, the tools and features provided by PyCharm help programmers to write a variety of software applications in Python quickly and efficiently. The developers can even customize the PyCharm UI according to their specific needs and preferences. Also, they can extend the IDE by choosing from over 50 plug-ins to meet complex project requirements.

The intelligent code editor provided by PyCharm enables programmers to write high quality Python code. The editor enables programmers to read code easily through colour schemes, insert indents on new lines automatically, pick the appropriate coding style, and avail context-aware code completion suggestions. At the same time, the programmers can also use the editor to expand a code block to an expression or logical block, avail code snippets, format the code base, identify errors and misspellings, detect duplicate code, and auto-generate code. Also, the editor makes it easier for developers to analyze the code and identify the errors while writing code.

4. Visual Studio Code

Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The source code is free and open-source, released under the permissive MIT License. The compiled binaries are freeware for any use.

D. Font End Technologies

1. HTML5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and last major version of HTML that is a World Wide Web Consortium (W3C) recommendation. The current specification is known as the HTML Living Standard and is maintained by a consortium of the major browser vendors (Apple, Google, Mozilla, and Microsoft), the Web Hypertext Application Technology Working Group (WHATWG).

HTML5 was first released in public-facing form on 22 January 2008, with a major update and "W3C Recommendation" status in October 2014. Its goals were to improve the language with support for the latest multimedia and other new features; to keep the language both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc., without XHTML's rigidity; and to remain backward-compatible with older software. HTML5 is intended to subsume not only HTML 4 but also XHTML 1 and DOM Level 2 HTML

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

2. CSS3

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable

multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

3. JavaScript

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMA Script specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). However, the language itself does not include any input/output (I/O), such as networking, storage, or graphics facilities, as the host environment (usually a web browser) provides those APIs.

JavaScript engines were originally used only in web browsers, but they are now embedded in some servers, usually via Node.js. They are also embedded in a variety of applications created with frameworks such as Electron and Cordova.

E. Mobile client (Android) (Remote Access Tool):

1. Java

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them.

2. Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development

3. Firebase

Firebase provides a real-time database and back-end as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud.

F. Python Libraries:

1. **SpeechRecognition 3.8.1**
   https://pypi.org/project/SpeechRecognition/pip
   pip install Speech Recognition
   Library for performing speech recognition, with support for several engines and APIs, online and offline. Google Speech Recognition
   Google Cloud Speech API etc.

2. **wolframalpha 3.0.1**
   https://pypi.org/project/wolframalpha
   pip install wolframalpha
   Python Client built against the Wolfram|Alpha v2.0 API. Basic usage is pretty simple. Create the client with your App ID (request from Wolfram Alpha)

3. **JSON**
   https://docs.python.org/2/library/json.html
   Preinstalled
   JSON (JavaScript Object Notation), specified by RFC 7159 json exposes an API familiar to users of the standard library marshal and pickle modules.
   JSON

Object	Python
array	dict
string	list
number (int)	unicode
number (real)	int, long
true	float

false	True
object	False
null	None

4. **PyAudio 0.2.11**
   https://pypi.org/project/PyAudio/
   pip install PyAudio
   Bindings for PortAudio v19, the cross-platform audio input/output stream library.

5. **Wikipedia**
   https://pypi.org/project/wikipedia/
   pip install Wikipedia
   Wikipedia is a Python library that makes it easy to access and parse data from Wikipedia.
   Search Wikipedia, get article summaries, get data like links and images from a page, and more. Wikipedia wraps the MediaWiki API so you can focus on using Wikipedia data, not getting it.

6. **smtplib**
   https://docs.python.org/2/library/smtplib.html
   Preinstalled
   Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.
   Python provides smtplib module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP.

7. **requests2 2.16.0**
   https://pypi.org/project/requests2/
   pip install requests2
   Requests is the only Non-GMO HTTP library for Python, safe for human consumption.

8. **OS** https://docs.python.org/2/library/os.html
   Preinstall
   This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.
   Python os.system() function
   We can execute system command by using os.system() function. According to the official document, it has been said that This is implemented by calling the Standard C function system(), and has the same limitations.

9. **Eel**

   https://pypi.org/project/Eel/

   pip install eel

   Eel is a little Python library for making simple Electron-like offline HTML/JS GUI apps, with full access to Python capabilities and libraries. Eel hosts a local webserver, then lets you annotate functions in Python so that they can be called from Javascript, and vice versa.

Other:

playsound: https://pypi.org/project/playsound/

twilio: https://pypi.org/project/twilio/

gtts: https://pypi.org/project/gTTS/

firebase: https://pypi.org/project/python-firebase/

5.2 Implementation

Smart mirror is implemented in such a way that it displays information retrieved from the internet. Retrieved data includes weather condition, time, calendar, notifications from social media. The procedure for implementing Smart Mirror is realized in the following steps:

1. The idea and the mirror
2. The monitor
3. The casing
4. Hardware installation
5. Installing raspberry pi
6. Production of interface

A. The idea and mirror

A regular mirror would not work. The mirror should be semi transparent or to be more accurate, it has to behave like a mirror when the screen behind it is black and should behave like a glass window when information is displayed on the screen.

B. The monitor

After a few measurements and some tryouts by tape on the wall where we planned to eventually mount the mirror, we figured an appropriate measurement that would give the perfect monitor size. Eventually we choose to use LCD monitors that met most of the expectations. They are relatively cheap simple touch buttons and the right connector orientation. This control panel of monitor is to be connected and mounted within the casing.

C. The casing

Measured the dimensions needed for the new casing and we decided to make a wood casing that would create a strong and steady frame. This casing acts as a shelf where the things can be kept. Since the prototype would probably generate some heat, air ventilation holes were provided. Also, a nice and firm mounting point was added on the backside of the casing.

D. Installing Hardware

Installing hardware required the following components:

1. The Monitor
2. A Raspberry Pi
3. A HDMI Cable (to connect the Raspberry to the Monitor)
4. A USB to micro USB cable (to power the Raspberry Pi)
5. A power cable to power the monitor

Installing hardware is just required to simply connect all the components, plugged in the power cable and then provide power to the monitor. The Raspberry is booted and the system didn't create any significant heat. The hardware installation part included mounting the panel behind the mirror and attaching the raspberry pi to it using HDMI cable. We make use of a micro USB cable to power the Raspberry Pi.

E. Installing the Raspberry Pi

We had chosen the operating system Raspbian, due to its flexibility and wide open-source community support. It provides a platform for installation. Since, additional cables would reduce the flexibility of the Smart Mirror, we preferred Wi-Fi to connect the smart Mirror tithe internet.

F. Production of the interface

The interface we built on top the Raspberry desktop is not a mysterious application. It is simply a full-screen web that allows us to use Python scripting. And as an added bonus, it allows to develop and test the interface on the usual PC before pushing it to the Smart Mirror.

5.3 Output Screens

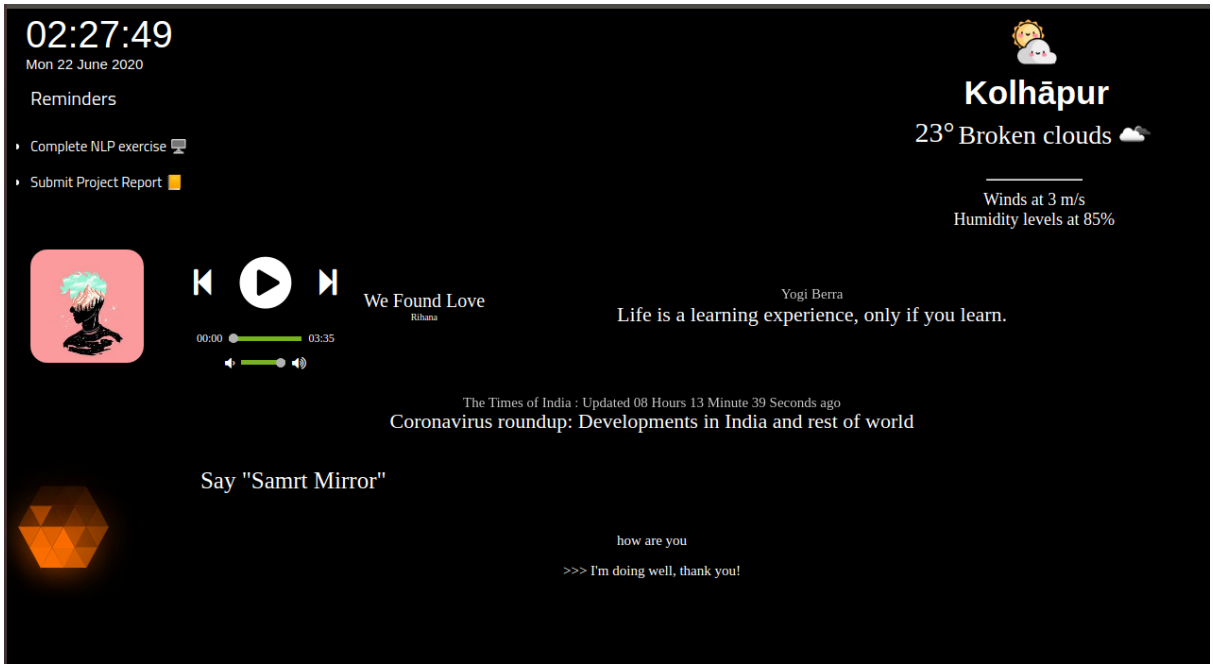


Figure 16: Mirror UI-1

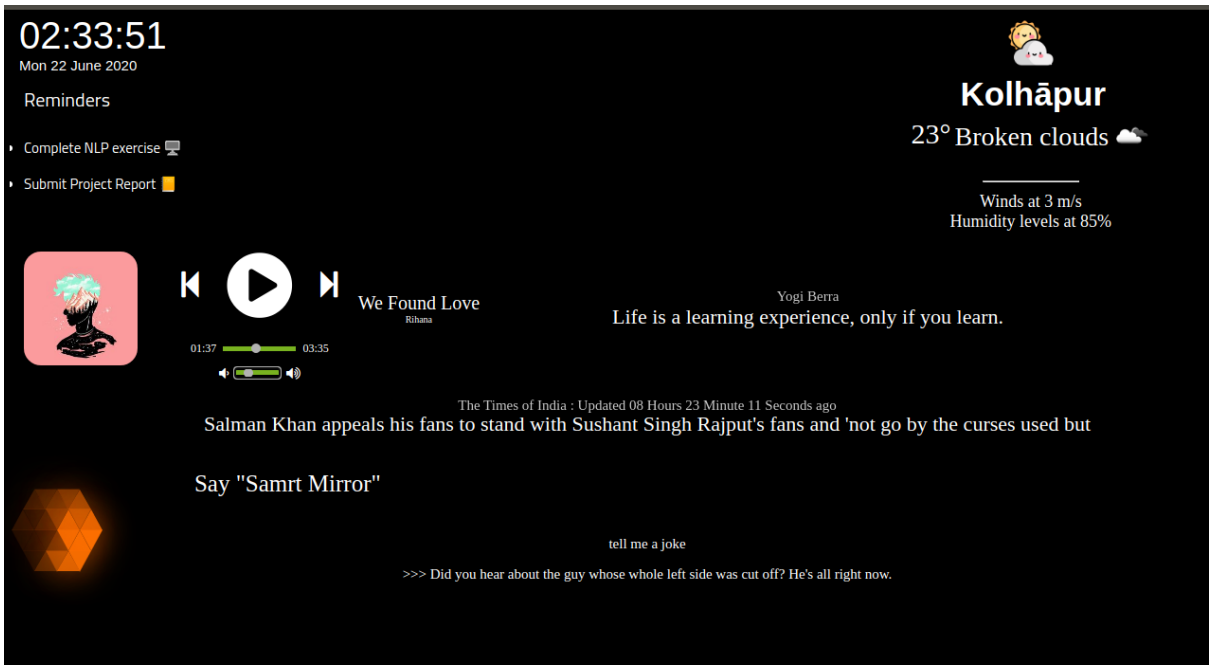


Figure 17: Mirror UI-2

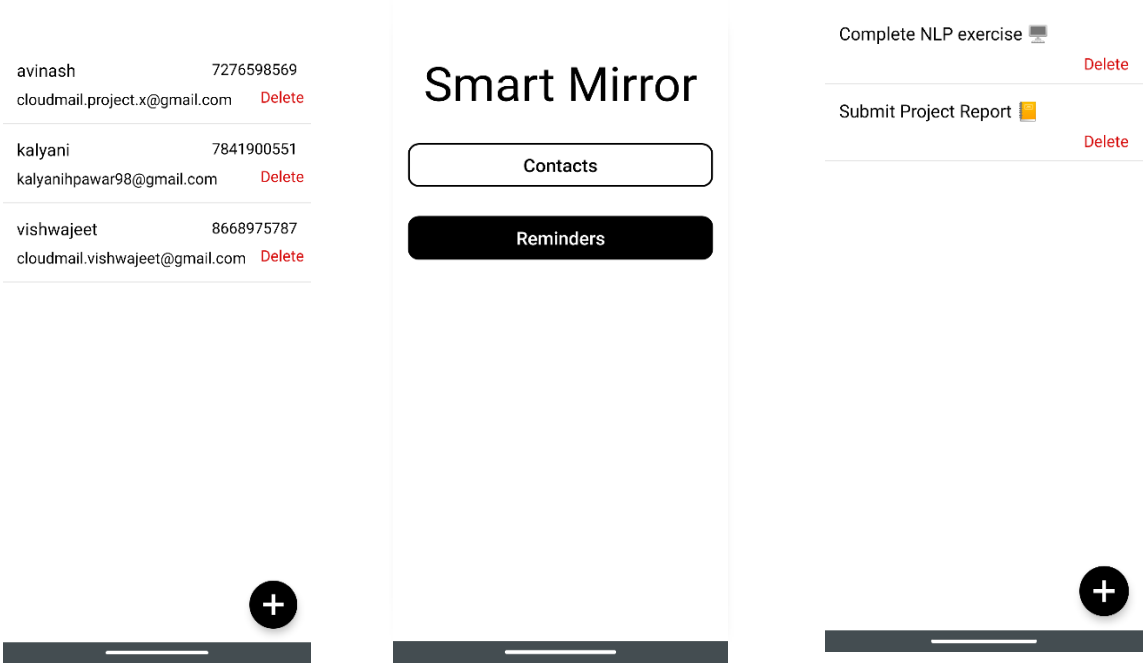
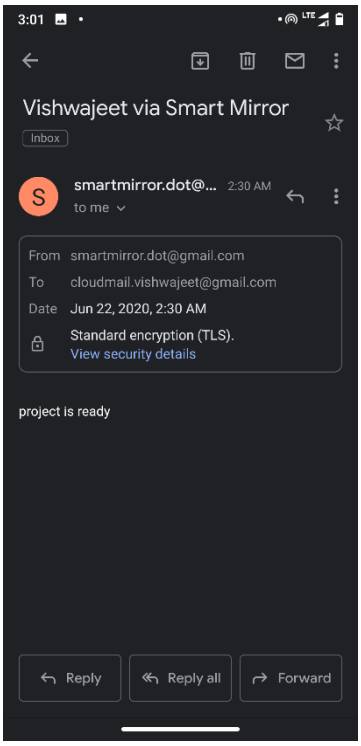


Figure 18: Android client



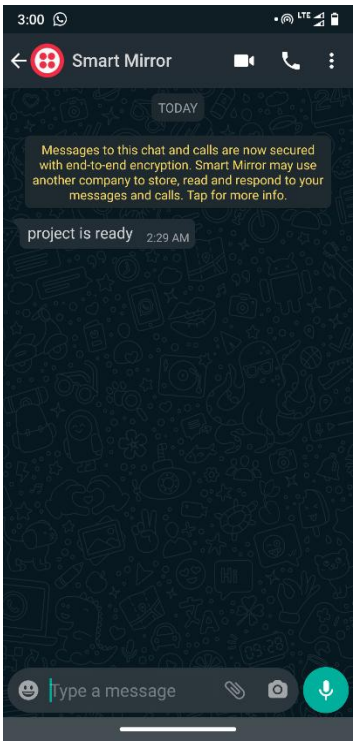


Figure 19 (a) *Figure 19 (b)*

(a): WhatsApp message sent by Smart Mirror
(b): Email sent by Smart Mirror

6. Applications

Smart mirrors have many potential applications in both personal and social settings. In personal settings, smart mirrors can be used to display relevant information, control household appliances, and provide emotional support to users In bathrooms, smart mirrors could prove to be a valuable application for many people. As people prepare for their day, their hands are typically busy, but they typically stand in one spot performing low cognitive tasks. People could have their email messages, trending tweets or Facebook posts, and breaking news show up on their smart mirrors seamlessly.

Smart mirrors can facilitate communication within a family. Many homes have a mirror in their entrance or foyer. A smart mirror in this location could act as a central hub for family scheduling, which would increase awareness of each other's activities within the family. Built right into the mirror, a digital calendar can be synced with users' calendar on their phone. Digital notes left for other users would not fall off from people walking by or opening the front door.

In public locations, smart mirror applications would be generalized, perhaps acting as conduits for information much like information kiosks, but provide physical and design benefits over implementing a kiosk. For example, there is limited space in mall washrooms to place a kiosk. A smart mirror, however, could display kiosk information, advertisements, and emergency alerts, all while utilizing the space that is already allocated for mirrors. Installed in fitting rooms, a smart mirror could revolutionize clothes shopping. No longer would some-one need to physically try on all the clothes they are interested in. Using augmented reality, people could view themselves in digital versions of the available store wardrobe or experiment with various design options . Such system would allow customers to browse a clothes catalog at a faster rate, filters down their selection of clothes, and only try on clothes to measure fit and comfort. The fitting room itself provides an excellent, low distraction environment for the processing of augmented reality, and could facilitate sensors for determining the correct size for customers.

7. Conclusion.

The Smart Mirror system we have developed combines the concepts and methodologies that have been implemented in other systems like smart phones. It is a novel method of creating a smart interactive system that is reliable and easy to use. We have concentrated on an interactive and user friendly home appliance that is beneficial to the user. The architecture has been adapted for the development and deployment of various services which all use web service communication mechanisms. These services of the Smart Mirror can be further extended to support future enhancements and developments. This means that features such as extended home automation (control ambient lighting systems, centralized air conditioning systems, and other modular home automation systems/devices like Google Home) and additional applications and widgets can be implemented.

A user's personalized data such as calendar, news feeds, and other information relevant to their lifestyle have been implemented and displayed . The Smart Mirror uses voice commands to switch between each view, and performs the functions corresponding to the input voice command. Rather than become confined to a home, we can implement the functionality of our Smart Mirror system to commercial work, and public environments.

Each widget and application has been developed using a building block approach, keeping the aspects of future enhancement and scalability in mind. The main advantage of using this building block approach is that any new application or widget can be implemented and put to functional working by developing them as plugins and using them as required to meet the environment and user's needs.

The Smart Mirror we have developed takes an average of 15 seconds to begin execution once the command to initiate its process has been input into the Terminal window in the Raspberry Pi. On initiation, when the Raspberry Pi is connected to the internet, a unique IP address with the port "8080" suffixed to it, is generated. This IP address is displayed in the terminal window and can also be viewed in the Smart Mirror process by saying the command "Show Remote Link". The IP address is used to connect the user's laptops, smartphones, or any other device capable of connecting to the internet, remotely to the Smart Mirror. That is, the Remote Configuration Tool (RCT) is initiated whenever a device uses the generated IP address to connect to the Smart Mirror. As a prerequisite, the device running the RCT needs to be connected to the same network connection as that to which the Raspberry Pi of the Smart Mirror is connected.

When the Smart Mirror is running, voice commands can be masked by background noise that is present in the environment in which the Smart Mirror is placed. This therefore can result in false positives and true negatives of voice command input to the Smart Mirror. In order to overcome this, the user can either manually alter the Configuration File or use the RCT to alter the sensitivity of the microphone. The voice command to activate the Smart Mirror is user defined and was set to "Smart Mirror" in our implementation.

8. Future Work

As for future work, many new plug-in opportunities are now available with the ability to access external hardware. It would be nice to explore various plug-in ideas using motion detectors, temperature and light sensors, gesture recognition, voice commands, and some form of proximity detection, such as detecting the closest phone in range.

There are features that can improve the usability of Smart Mirror. One feature would be the ability to install and uninstall plug-in through the Server API. Currently, to install new plug-in, users have to login to their smart mirror servers and put the plugin's files to the plug-in root directory. If the Server API supports the installation and removal of a plug-in, we can use these features and develop a web GUI, so that users do not have to install/uninstall plug-in manually.

Another enhancement would be to add a priority system for processing API calls. Currently, with wireless internet and network interference, there can be a delay in sending API calls. This delay causes the API calls to get cached on the server and then executed as the call queue is processed. With a priority system in place, non-important API calls could be dropped completely as to clear up the API call queue faster and try to alleviate the delay. Having prioritized API calls would benefit interactive processes that require real-time user input/output, such as rearranging a plugin's location in the mirror.

Finally, research into the appropriate uses of a smart mirror is warranted. Mirror usage usually involves a local presence, with motion and visual interaction, and depends on the context and location of the mirror. Some locations may allow for more personal applications, while others to more generalized purposes for a large user base. People may develop different mental models about how a smart mirror works, which may shape their behavior while interacting with it and raise issues, such as privacy and security. Understanding of user needs and expectations is key to successful deployment of smart mirrors.

9. Bibliography / References

- Alpay Kasal and Sam Ewen . 2008 A project of interactive mirror with artsy visuals in Lit Studios
- Chalmers University of Technology in Sweden . 2009 Two-way mirror to allow the LEDs and to display information on screen
- Ivette Cristina Araujo Garcia, Eduardo Rodrigo Linares Salmon, Rosario Villalta Riega, Alfredo Barrientos Padilla, "Implementation and Customization of a Smart Mirror through a Facial Recognition Authentication and a Personalized News Recommendation Algorithm", in 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 2017.
- Mohammed Ghazal, Tara al Hadithy,Yyasmina al Khalil, Muhammad Akmal and Hassan Hajjdiab, " a Mobile-programmable smart mirror for ambient IoT environments", in 5th international conference on future internet of things and cloud workshops, 2017.
- New York Times Research and Development Lab. 2011 A Magic mirror using Microsoft Kinect to track movements
- Oihane Gomez-Carmona, Diego Casado-Mansilla, "SmiWork: An Interactive Smart Mirror Platform or Workplace Health Promotion", 2017.
- Philips Homelab
  http://www.research.philips.com/technologies/misc/homelab/index.html
- Ramya .S , Saranya. S , Yuvamalini. M, "The Smart Mirror", in International Journal of Advanced Research, Ideas and Innovations in Technology, 2018.
- Tatiana Lashina. Intelligent bathroom. In European Symposium on Ambient Intelligence (EUSAI'04), Eindhoven, Netherlands, 2004

10. Appendix A:

Glossary [Define terms, acronyms, and abbreviations used]

1. CSS

   Cascading Style Sheets (**CSS**) is a simple mechanism for adding style (e.g., fonts, colours, spacing) to Web documents

2. HTML

   Hyper Text Markup Language (HTML) is the most basic building block of the Web. It defines the meaning and structure of web content

3. IDE

   Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program

4. IOT

   The internet of things, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

5. JS

   JavaScript is the programming language of HTML and the Web.

6. OS

   An operating system is the program that, after being initially loaded into the computer by a boot program, manages all of the other application programs in a computer. The application programs make use of the operating system by making requests for services through a defined application program interface (API).

7. PC

   A personal computer (PC) is a multi-purpose computer whose size, capabilities, and price make it feasible for individual use.

8. API

   An application programming interface (API) is a computing interface which defines interactions between multiple software intermediaries.