CHAPTER 1

INTRODUCTION

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 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. The operating system would support running apps and would show weather, time, calendar, recent email, news and compliments depending on the time. Magic Mirror is a mirror device built with a cardboard frame, a flat screen monitor, a Raspberry Pi 2 and software running on a web browser and a tool called Electron. The module system is very simple and even allows others to develop their own modules. The default installation comes with some basic widgets for time, calendar, weather and news.

CHAPTER 2

LITERATURE SURVEY

[1] The Cybertecture Mirror may look like yet another vapor-rich concept -- what with its translucent overlaid interface and cloud-connected ways -- but it's actually just had its launch in Hong Kong, is set to start taking pre-orders in December, and will (hopefully) be shipped by the middle of next year. The brainchild of one James Law, this reflective renegade measures 800 x 500 x 50mm, comes with stereo speakers totalling 10W of power output, Wi-Fi, IP41 waterproofing, and fog-resistant glass. Before you ask if it runs Android, both the display and operating system are said to be proprietary, with the latter offering access to messaging, weather, calendar, and apps, such as an included fitness-tracking utility. Wen Wei Po reports a 60,000 HKD ($7,733) launch price and a very ambitious expectation that two million Mirrors will be sold over the first three years.

[2]In essence, the magic mirror replaces your standard bathroom mirror. The exact mirror/display being used isn't explicitly stated, but it is probably a Philips Mirror TV. Kinect's movement tracking and voice recognition provides the human-computer interface. Additional interaction is provided through objects (skin care products, prescriptions) with RFID tags that are recognized by the system. On the back end, there is probably a standard Windows PC that uses the free Kinect SDK, and the NYT's developer API to serve up content.

The end result, is an awesome interface that can be used while you brush your teeth, blow dry your hair, or take your meds. Far from being a device that can only be used to surf NYT news, the magic mirror is basically everything you'd want from a bathroom computer. You can use it to check your email or calendar, but you can also use it to go shopping or check on your social networks. In the video, the main functionality that is demonstrated is the ability to recognize an RFID chip in a box of off-the-shelf drugs and then show you their directions for use; handy, and potentially life-saving in some situations. The mirror also has the ability to track your body and overlay clothes, if you want try something on before you buy.

[3] Japanese company Seraku aims to make your network a little more ubiquitous with a prototype design for an Android-powered mirror.

It uses RF proximity sensors to detect where your hands are placed so that you don't have to smudge it all up in order to check those sports scores, and the display unit here at Smartphone and Mobile Expo included a networked scale built into the floor. There's also a meter that displays water flow and temperature information on the mirror. Overall, it reminds us a lot of the Smart Window prototype design we saw from Samsung at CES, although not as responsive or as fully fleshed out. An Android tablet is powering everything behind the scenes, but the display is a separate LCD monitor overlaid with a semi-transparent piece of reflective glass. The Seraku rep says that the company doesn't have a commercial product ready yet, but if and when the product launches, Seraku sees two primary use cases for the mirror: reading the news at salons, and (somewhat inexplicably) filling out questionnaires at drinking establishments.

[4] The magic mirror is basically a one-way mirror made "smart" by a simple LCD display which sits behind the mirror and displays white UI elements with a black background. When the display is on, you can see both your reflection and the white elements, allowing software to present relevant information while you get ready for the day. The screen needs to be readable through the mirrored surface, so the mirror uses a high contrast ratio of pure white on pure black. Lastly and most importantly, the user needs to see their reflection, so the central area of the mirror is kept clear when the user is logged in.

The more-pressing information (weather, time, and a space reserved for alerts) are placed at the top of the mirror near eye-level, and other less-urgent information has been pushed down at the bottom, where it can be ignored or consciously consumed. Using the Hosted Web apps bridge, we turned our web app into a Universal Windows App, which not only give us access to Windows Native APIs but can also run across Windows devices, such as the Raspberry Pi 3 in our case. All the HTML, CSS, and JavaScript comes directly from the server, hence the term hosted. The mirror also has facial recognition capability which is powered by APIs provided by Microsoft's Cognitive Services. Magic Mirror leverages Microsoft's Cognitive Services Face API to match the user's face to their profile. The user creates a profile by adding some personal info and taking a selfie, which is then sent to Cognitive Services to get a unique identifier (a face_id) which is then stored in the Magic Mirror's database.

[5] This is a smart mirror for your home. A few companies are researching them (Samsung, LG) and a few makers have made them. Check back later for more in-depth posts. For now, a

summary: The mirror is made from a deconstructed 40? LED TV behind a one-way mirror, so every black pixel is “replaced” with the reflection. Any image shines through as long as its bright, and the background is black. We used a couple of pieces of software to achieve this: Windows 10 with a desktop background that is black. We also use Cortana and voice recognition for opening apps, talking to mirror, etc. Rain meter lets you put icons, images, widgets, etc. right on your desktop. We used it to create the UI. Desktop Coral forces apps to open only in the small section above the UI but below the user’s face.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 FUNTIONAL REQUIREMENTS

Raspberry Pi: The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation. All models feature a Broad computer system on a chip (SoC), which includes an ARM compatible central processing unit(CPU) and anon-chip graphics processing unit(GPU, aVideo CoreIV). CPU speed ranges from 700 MHz to 1.2 GHz for the Pi 3 and on board memory range from 256 MB to 1 GB RAM.

Secure Digital(SD) cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phone jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols likeI²C. The B-models have an8P8CEthernetport and the Pi 3 has on board Wi-Fi 802.11n and Bluetooth..

Raspbian OS: Raspbian is a Debian-based computer operating system for Raspberry Pi. It is now officially provided by the Raspberry Pi Foundation, as the primary operating system for the family of Raspberry Pi single-board computers.[3]Raspbian was created by Mike Thompson and Peter Green as an independent project.[4]The initial build was completed in June 2012.[5]The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARMCPUs.

Raspbian uses PIXEL, Pi Improved xwindows Environment, Lightweight as its main desktop environment as of the latest update. It is composed of a modified LXDE desktop environment and the Open box stacking window manager with a new theme and few other changes.

LCD Display: A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in colour or monochrome.[1]LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as present words, digits, and 7-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made up of a large number of small pixels, while other displays have larger elements.

Two-way Mirror: A two-way mirror, also called one-way 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.

3.2 NON-FUNCTIONAL REQUIREMENTS

Hardware Requirement

- Raspberry Pi
- Acrylic Two-way mirror
- HDMI cable
- LCD monitor
- Wooden planks.

Software Requirement

- Raspbian OS
- Weather API
- Calendar API
- Map API
- News API

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

System design is the set of operations for precisely stating the meaning of system's architecture, its sub systems, functional modules, interfaces and the required data for satisfying the systems requirements, which have been specified. It is applied for development of system theory to development of a product. Here the design must consist of modules with their definitions, along with the functional abstraction supported by each module.

4.1.1 Purpose

The main purpose of this project is to reduce stress placed on an individual when they are grooming themselves. Keeping track of time, the e-mails the user gets at the last minute. These small subtle notifications give us much needed knowledge and is easy on the eye.

4.1.2 Scope of the Project

This project was aimed to be used as a home appliance and intended only for one user. Since it was intended for one user and aimed to produce basic output our model has a fixed working environment but variants of the smart mirror that use more capable processors could possibly incorporate voice and face recognition which gives us a large environment where the smart mirror can be implemented.

4.2 SYSTEM OVERVIEW

This section gives the overview of working of the system. Here in this project we have set the raspberry pi 2B as the processing unit. We have also exploited the property of one way mirror also known as the two-way mirror to only reflect the user's reflection and to allow only the white from the monitor to be visible to the user so all the notifications we use had to be made white to maintain the integrity of the reflection and simultaneously make the notifications visible to the user. The raspberry pi 2B uses a program called magic mirror to run the API's of the time, weather, email, calendar-holidays, news and the compliment modules.

4.2.1 Architectural Design

The architectural design depicts of how the overall architecture of a system is designed. The architecture design varies from one system to another.

The architectural design is specified by viewing the required components, understanding the control and data flow between the components, and stating the functionalities for each of these components, on the type of functions to be performed, data input, data output by effective utilization of resources. The Fig 4.1 defines the architectural design of the system.

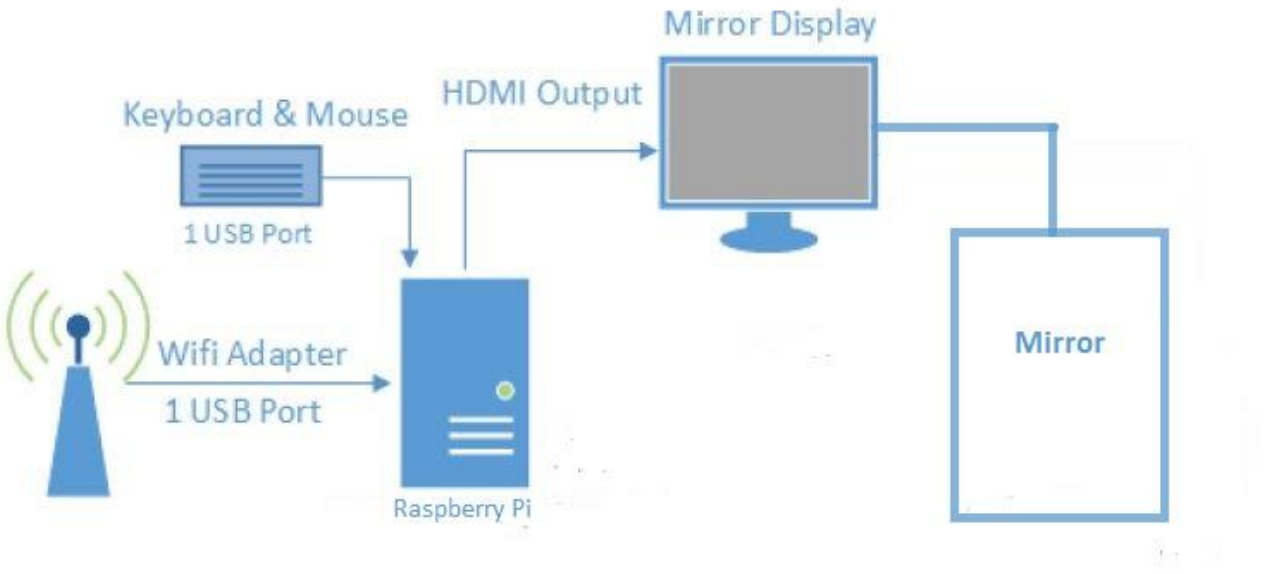

Fig 4.1: System design

The Raspberry Pi forms the heart of the device as it controls all other components. As our modules requires an active internet connection, a Wi-Fi dongle is connected to the Raspberry Pi's USB port. A keyboard and a mouse is required to type in the commands needed to execute the application. The HDMI port of the Raspberry Pi is then connected to the LCD monitor via a VGA converter cable. The display is then mounted on top of the one-way mirror which is attached to the frame.

4.2.2 Dataflow Diagram

It is a graph representing movement of data values between different processes. It shows flow of data, type of input and output data to and from the system. Hence, it is the first step of the design phase that would further functionally filter the requirements into its lowest level of detail. Fig 4.2 defines the dataflow diagram of the system.

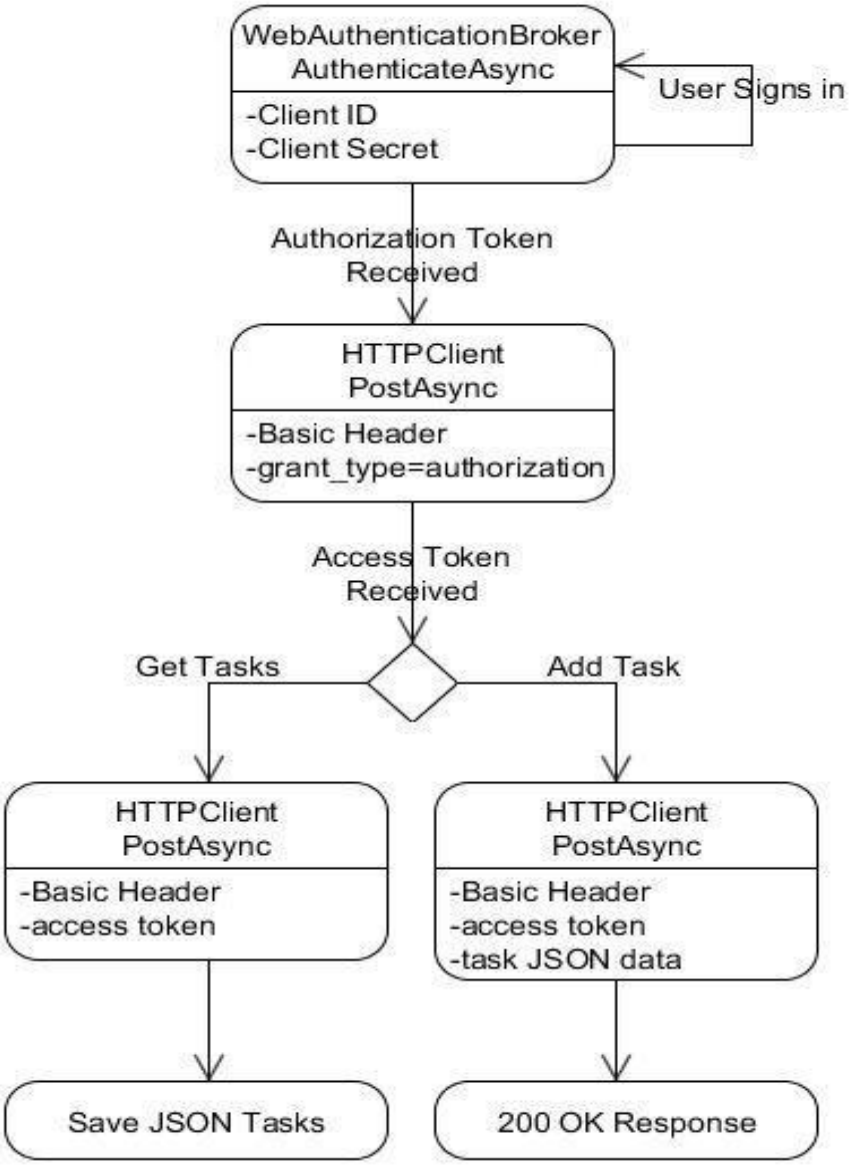

Fig 4.2: Dataflow Diagram

4.2.3 Sequence Diagram

It is represented using parallel vertical lines which represents the various processes or objects that live at the same time, and the horizontal arrows represents the messages exchanged between the objects or processes, in the order of occurrence. Fig 4.3 defines the sequence diagram of the system.

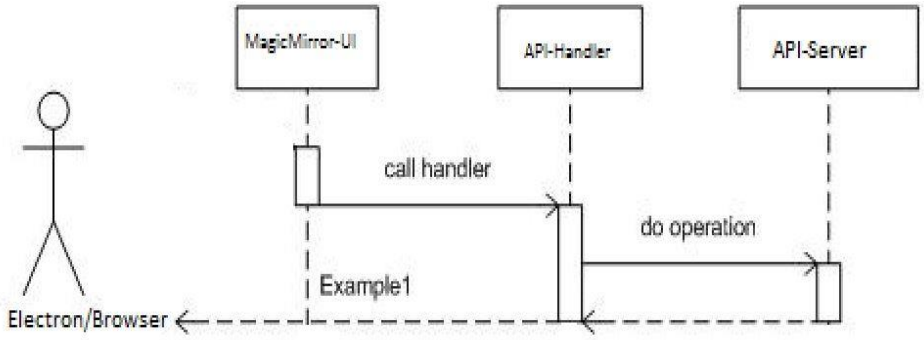

Fig 4.3: Sequence Diagram

4.2.4 Use Case Diagram

It is defined as a set of actions or events, taking place between the system and actors of the system, for a certain purpose. It must contain all the necessary system activities which has user interaction. Use case diagram are often used to depict missions of a system. Fig 4.4 defines the use case diagram of the system.

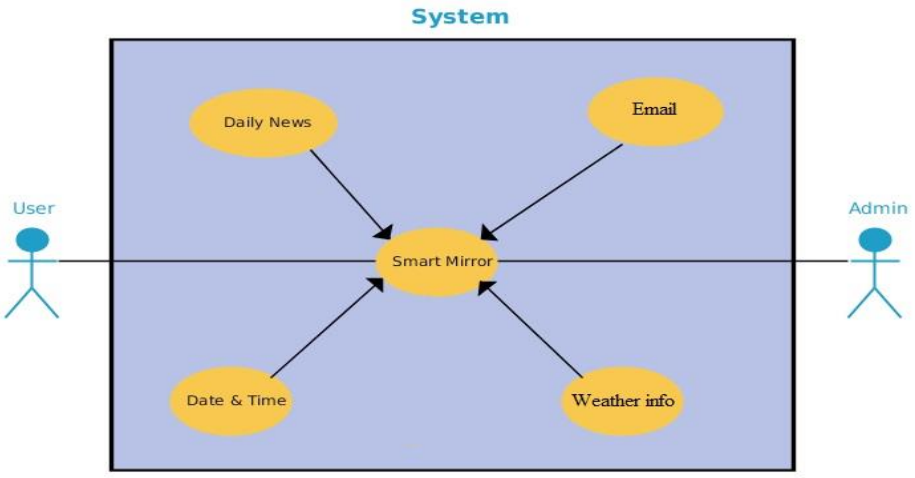

Fig 4.4: Use Case Diagram

CHAPTER 5

IMPLEMENTATION

5.1 HARDWARE

For the hardware,18.5inch Compaq LED monitor is used as display, a 25-inch one-way mirror and a Raspberry Pi 2 Model B. Everything was put together in a cardboard box. Fig 5.1 shows the final sketches for the hardware design.

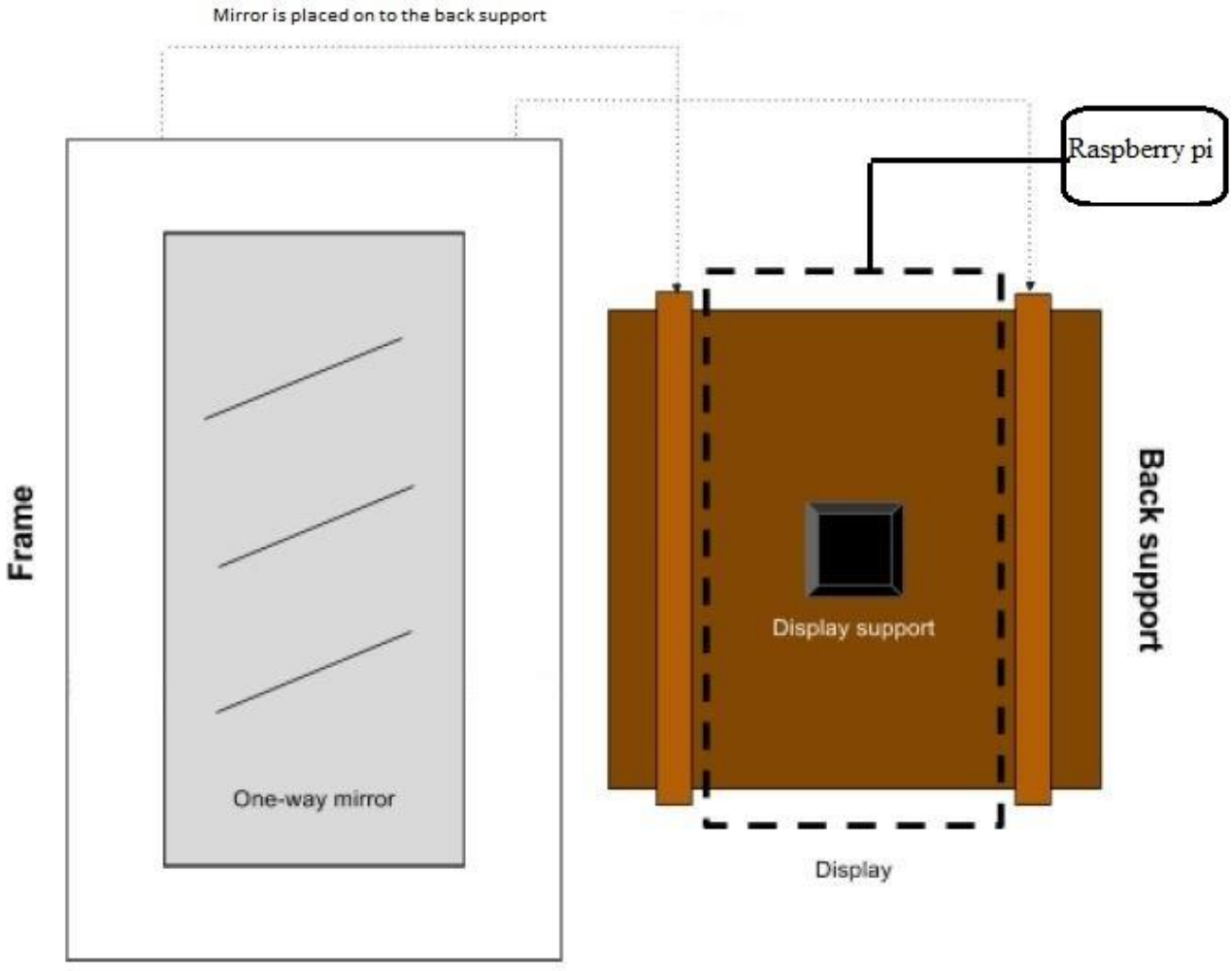

Fig 5.1. Sketch of the hardware design required for the smart mirror

We 1st cut a window like slit on one side of the cardboard box. then we placed the mirror on this side by using double sided tape. We took apart the monitor casing and placed

the monitor over the mirror. The monitor and the mirror was glued together by using duct tape to ensure durability. Then we made a slit so that the VGA cable and the power supply cable could be accessible. Finally, we taped the entire box together in duct tape for durability and to make it look a bit cooler.

5.1.1 One-way 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 one-way mirror, sometimes called two-way 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. Fig 5.2 depicts the schematic diagram of light reflection on a one - way mirror.

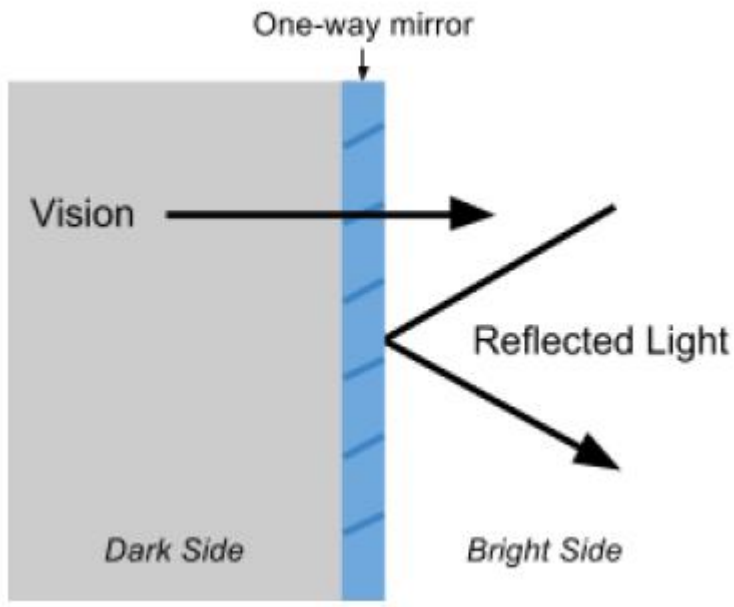

Fig 5.2. Schematic diagram of light reflection on a one-way mirror

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. 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.

5.1.2 Display

For the display, an 18.5 inch COMPAQ LED monitor was bought. Black insulation tape was used to cover the parts of the glass which are not covered by the display. An VGA converter cable was used to connect the display's VGA cable to the Raspberry Pi for video.

5.1.3 Raspberry Pi 2

The Raspberry Pi is a single board computer developed by the Raspberry Pi foundation in the UK. It has become the most popular computer of its kind thanks to great support and a big community behind it as well as an inexpensive price. The Pi does not work out of the box. It lacks a hard drive and it does not come with a preinstalled operating system. To install an OS you need a microSD card prepared with an OS image. And because the software that will be running on the mirror will be coded on the same device at least a screen, a keyboard and a mouse are required.

5.1.4 Frame and support

The frame is made of cardboard and it provides the support for the mirror and all the other components. It frames the glass and provides a possible way for hanging the mirror on a wall or we could device a stand suing it. It has two parts: the front is painted white and has four holes for the ultrasonic sensors.

5.2 SOFTWARE

All the software runs on the Raspberry Pi 2. We chose to use Raspbian which is the official Linux distribution from the Raspberry Pi Foundation because it has a lot of support and documentation. It was downloaded from the official Raspberry Pi website and we copied it on a microSD card. Then we inserted the card on the Raspberry Pi, started it and followed the setup instructions, which are quite simple. Once Raspbian was installed, the distribution was updated with the latest packages and then we configured the basics of the OS and everything was ready to go.

5.2.1 Development Tools

Taking advantage of the fact that there was already an operating system running on the Pi, we gave ourselves the challenge of writing all the code for the Smart Mirror on the same device. In the end, the entire coding for the software was done on the Raspberry Pi and we only used our 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 Magic Mirror

Electron

Electron is a software based on Chromium, the open source version of Google's Chrome, that includes NodeJS and several improvements to make it easy to develop web based software for desktop computers. The OS was built on top of Electron using web technologies.

NodeJS

NodeJS is a JavaScript engine for server side applications. It comes included with Electron and we used it to launch processes to control things that are not available in web APIs. It can be also used it access the filesystem and read the app files.

5.2.2 MagicMirror UI

The Smart Mirror's interface and it runs on top of Raspbian and on top of Electron. In the following figure, you can see the layers of the software stack. Fig 5.3 show the layers of software stack of MagicMirror UI.

Fig 5.3. Layers of software stack of Magicmirror UI

Architecture and features

The OS was designed to be very simple and lightweight as it already runs on top of many layers of software. It's written in HTML, JavaScript and Python and it is basically a framework for web apps that provides APIs for displaying messages to the user in a consistent way. All apps run on a single process. Currently, the current software includes 8 modules by default and others can also add their own modules very easily.

The user interface for the OS is clean and simple. We display the clock and calendar on the top-right part, the weather details in the top-left part and the email notification in the bottom-left part. On the bottom side, there the news is shown dynamically and there is a compliment message at about two-thirds of the screen. Fig 5.4 shows the user interface of the magicmirror.

Fig 5.4. User interface for Magicmirror

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.

Workflow

MagicMirror boots on top of Raspbian. After the initialization, the weather, email, calendar and all other APIs are started are all started. Fig 5.5 shows the boot sequence and basic operation of magicmirror.

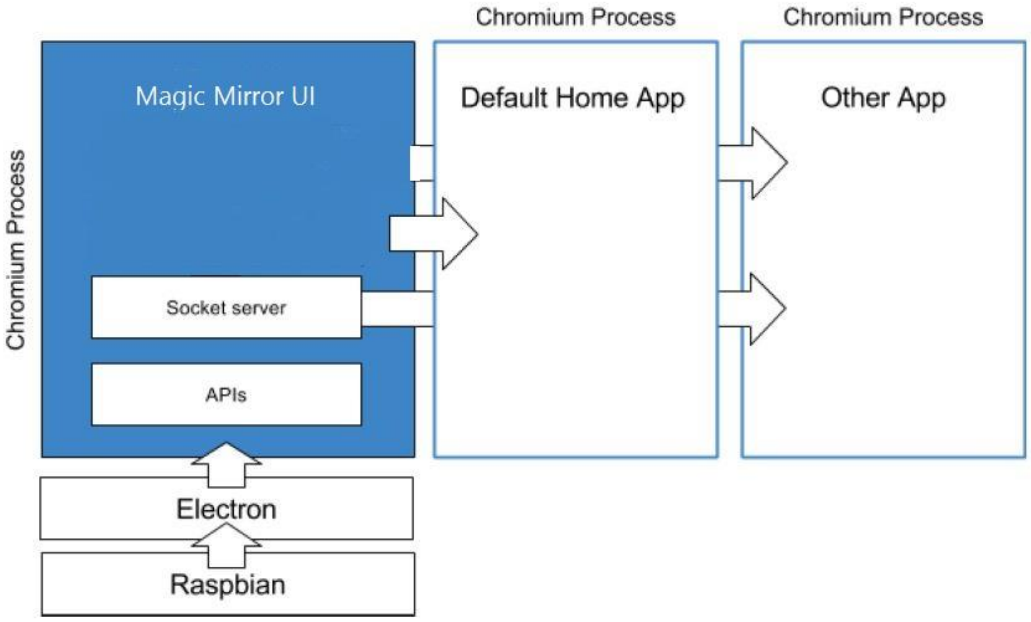

Fig 5.5. Magicmirror boot sequence and basic operation

5.2.3 Developing Apps for MirrorOS

A typical MirrorOS app should have the following file structure:

Index.html

Main html file for the app. Contains references to js and css.

js/app.js

Main jsloigc file

css/main.css

styles for the app

package.json

App description.

5.3 Modules

5.3.1 Module: Clock

The clock module is one of the default modules of the MagicMirror. This module displays the current date and time. The information will be updated real time.

Using the module

To use this module, add it to the modules array in the config/config.js file:

```
modules: [
        {
                module:"clock",
                position:"top_left",        // This can be any of the regions.
                config: {
                        // The config property is optional.
                        // See 'Configuration options' for more information.
                }
        }
]
```

Configuration options

The following properties can be configured:

Option	Description
timeFormat	Use 12 or 24 hour format. **Possible values:** 12 or 24 **Default value:** uses value

Option	Description
	of *config.timeFormat*
displaySeconds	Display seconds. **Possible values:** true or false **Default value:** true
showPeriod	Show the period (am/pm) with 12 hour format. **Possible values:** true or false **Default value:** true
showPeriodUpper	Show the period (AM/PM) with 12 hour format as uppercase. **Possible values:** true or false **Default value:** false
clockBold	Remove the colon and bold the minutes to make a more modern look. **Possible values:** true or false **Default value:** false
showDate	Turn off or on the Date section. **Possible values:** true or false **Default value:** true
showWeek	Turn off or on the Week section. **Possible values:** true or false **Default value:** false

Option	Description
dateFormat	Configure the date format as you like. **Possible values:** Docs **Default value:** "dddd, LL"
displayType	Display a digital clock, analog clock, or both together. **Possible values:** digital, analog, or both **Default value:** digital
analogSize	**Specific to the analog clock.** Defines how large the analog display is. **Possible values:** A positive number of pixels **Default value:**200px`
analogFace	**Specific to the analog clock.** Specifies which clock face to use. **Possible values:** simple for a simple border, none for no face or border, or face-### (where ### is currently a value between 001 and 012, inclusive) **Default value:** simple
secondsColor	**Specific to the analog clock.** Specifies what color to make the 'seconds' hand. **Possible values:** any HTML RGB Color **Default value:** #888888
analogPlacement	**Specific to the analog clock.** ***(requires displayType set to 'both')*** Specifies where the analog clock is in relation to the digital clock

Option	Description
	Possible values: top, right, bottom, or left **Default value:** bottom
analogShowDate	**Specific to the analog clock.** If the clock is used as a separate module and set to analog only, this configures whether a date is also displayed with the clock. **Possible values:** false, top, or bottom **Default value:** top
Timezone	Specific a timezone to show clock. **Possible examples** **values:** America/New_York, America/Santiago, Etc/GMT+10 **Default value:** none.

Table 5.1: Configuration for clock API

5.3.2 Module: Calendar

The calendar module is one of the default modules of the MagicMirror. This module displays events from a public calendar. It can combine multiple calendars.

Using the module

To use this module, add it to the modules array in the config/config.js file:

```
modules: [
        {
                module:"calendar",
                position:"top_left",        // This can be any of the regions. Best results in left or right regions.
                config: {
                        // The config property is optional.
```

```
                        // If no config is set, an example calendar is shown.
                        // See 'Configuration options' for more information.
                }
        }
]
```

Configuration options

The following properties can be configured:

Option	Description
maximumEntries	The maximum number of events shown. / **Possible values:** 0 - 100 **Default value:** 10
maximumNumberOfDays	The maximum number of days in the future. **Default value:** 365
displaySymbol	Display a symbol in front of an entry. **Possible values:** true or false **Default value:** true
defaultSymbol	The default symbol. **Possible values:** See Font Awsome website. **Default value:** calendar
maxTitleLength	The maximum title length. **Possible values:** 10 - 50

Option	Description
	Default value: 25
wrapEvents	Wrap event titles to multiple lines. Breaks lines at the length defined by maxTitleLength. **Possible values:** true or false **Default value:** false
fetchInterval	How often does the content needs to be fetched? (Milliseconds) **Possible values:** 1000 - 86400000 **Default value:** 300000 (5 minutes)
animationSpeed	Speed of the update animation. (Milliseconds) **Possible values:**0 - 5000 **Default value:** 2000 (2 seconds)
fade	Fade the future events to black. (Gradient) **Possible values:** true or false **Default value:** true
fadePoint	Where to start fade? **Possible values:** 0 (top of the list) - 1 (bottom of list) **Default value:** 0.25
calendars	The list of calendars. **Possible values:** An array, see *calendar configuration* below.

Option	Description
	Default value: *An example calendar.*
titleReplace	An object of textual replacements applied to the tile of the event. This allow to remove or replace certains words in the title. **Example:** {'Birthday of ' : '', 'foo':'bar'} **Default value:** { "De verjaardag van ": "", "'s birthday": "" }
displayRepeatingCountTitle	Show count title for yearly repeating events (e.g. "X. Birthday", "X. Anniversary") **Possible values:** true or false **Default value:** false
dateFormat	Format to use for the date of events (when using absolute dates) **Possible values:** See Moment.js formats **Default value:** MMM Do (e.g. Jan 18th)
timeFormat	Display event times as absolute dates, or relative time **Possible values:** absolute or relative **Default value:** relative
getRelative	How much time (in hours) should be left until calendar events start getting relative? **Possible values:** 0 (events stay absolute) - 48 (48 hours before the event starts) **Default value:** 6

Option	Description
urgency	When using a timeFormat of absolute, the urgency setting allows you to display events within a specific time frame as relative. This allows events within a certain time frame to be displayed as relative (in xx days) while others are displayed as absolute dates **Possible values:** a positive integer representing the number of days for which you want a relative date, for example 7 (for 7 days) **Default value:** 7
broadcastEvents	If this property is set to true, the calendar will broadcast all the events to all other modules with the notification message: CALENDAR_EVENTS. The event objects are stored in an array and contain the following fields: title, startDate, endDate, fullDayEvent, location and geo. **Possible values:** true, false **Default value:** true
hidePrivate	Hides private calendar events. **Possible values:** true or false **Default value:** false
excludedEvents	An array of words / phrases from event titles that will be excluded from being shown.

Option	Description
	Example: ['Birthday', 'Hide This Event'] **Default value:** []

Table 5.2: Property values of calendar

Calendar configuration

The calendars property contains an array of the configured calendars. The colored property gives the option for an individual color for each calendar.

Default value:

```
config: {
        colored:false,
        calendars: [
                {
                        url:'http://www.calendarlabs.com/templates/ical/US-
Holidays.ics',
                        symbol:'calendar',
                        auth: {
                           user:'username',
                           pass:'superstrongpassword',
                           method:'basic'
                        }
                },
        ],
}
```

Calendar configuration options:

Option	Description

Option	Description
url	The url of the calendar .ical. This property is required. **Possible values:** Any public accessble .ical calendar.
symbol	The symbol to show in front of an event. This property is optional. **Possible values:** See Font Awesome website. To have multiple symbols you can define them in an array e.g. ["calendar", "plane"]
color	The font color of an event from this calendar. This property should be set if the config is set to colored: true. **Possible values:** HEX, RGB or RGBA values (#efefef, rgb(242,242,242), rgba(242,242,242,0.5)).
repeatingCountTitle	The count title for yearly repating events in this calendar. **Example:** 'Birthday'
maximumEntries	The maximum number of events shown. Overrides global setting. **Possible values:** 0 - 100
maximumNumberOfDays	The maximum number of days in the future. Overrides global setting
auth	The object containing options for authentication against the calendar.

Table 5.3: Configuration options for calendar

Calendar authentication options:

Option	Description
user	The username for HTTP authentication.
pass	The password for HTTP authentication. (If you use Bearer authentication, this should be your BearerToken.)
method	Which authentication method should be used. HTTP Basic, Digest and Bearer authentication methods are supported. Basic authentication is used by default if this option is omitted. **Possible values:** digest, basic, bearer **Default value:** basic

Table 5.4: Authentication options for calendar

5.3.3 Module: Weather Forecast

The weather forecast module is one of the default modules of the MagicMirror. This module displays the weather forecast for the coming week, including an an icon to display the current conditions, the minimum temperature and the maximum temperature.

Configuration Options

The following properties can be configured:

Option	Description
Location	The location used for weather information. **Example:** 'Amsterdam,Netherlands' **Default value:** false **Note:** When

Option	Description
	the location and locationID are both not set, the location will be based on the information provided by the calendar module. The first upcoming event with location data will be used.
locationID	Location ID from OpenWeatherMap **This will override anything you put in location.** Leave blank if you want to use location. **Example:** 1234567 **Default value:** false **Note:** When the location and locationID are both not set, the location will be based on the information provided by the calendar module. The first upcoming event with location data will be used.
Appid	The OpenWeatherMap API key, which can be obtained by creating an OpenWeatherMap account. This value is **REQUIRED**
Units	What units to use. Specified by config.js **Possible values:** config.units = Specified by config.js, default =

Option	Description
	Kelvin, metric = Celsius, imperial =Fahrenheit **Default value:** config.units
roundTemp	Round temperature values to nearest integer. **Possible values:** true (round to integer) or false (display exact value with decimal point) **Default value:** false
maxNumberOfDays	How many days of forecast to return. Specified by config.js **Possible values:** 1 - 16 **Default value:** 7 (7 days) This value is optional. By default the weatherforecast module will return 7 days.
showRainAmount	Should the predicted rain amount be displayed? **Possible values:** true or false **Default value:** false This value is optional. By default the weatherforecast module will not display the predicted amount of rain.

Option	Description
updateInterval	How often does the content needs to be fetched? (Milliseconds) **Possible values:** 1000 - 86400000 **Default value:** 600000 (10 minutes)
animationSpeed	Speed of the update animation. (Milliseconds) **Possible values:**0 - 5000 **Default value:** 1000 (1 second)
Lang	The language of the days. **Possible values:** en, nl, ru, etc ... **Default value:** uses value of *config.language*
Fade	Fade the future events to black. (Gradient) **Possible values:** true or false **Default value:** true
fadePoint	Where to start fade? **Possible values:** 0 (top of the list) - 1 (bottom of list) **Default value:** 0.25
initialLoadDelay	The initial delay before loading. If you

Option	Description
	have multiple modules that use the same API key, you might want to delay one of the requests. (Milliseconds) **Possible values:** 1000 - 5000 **Default value:** 2500 (2.5 seconds delay. This delay is used to keep the OpenWeather API happy.)
retryDelay	The delay before retrying after a request failure. (Milliseconds) **Possible values:** 1000 - 60000 **Default value:** 2500
apiVersion	The OpenWeatherMap API version to use. **Default value:** 2.5
apiBase	The OpenWeatherMap base URL. **Default value:** 'http://api.openweathermap.org/data/'
forecastEndpoint	The OpenWeatherMap API endPoint. **Default value:** 'forecast/daily'
appendLocationNameToHeade	If set to true, the returned location name

Option	Description
r	will be appended to the header of the module, if the header is enabled. This is mainly intresting when using calender based weather. **Default value:** true
calendarClass	The class for the calender module to base the event based weather information on. **Default value:** 'calendar'
iconTable	The conversion table to convert the weather conditions to weather-icons. **Default value:** view table below
Colored	If set 'colored' to true the min-temp get a blue tone and the max-temp get a red tone. **Default value:** 'false'

Table 5.5: Configuration for Weather Forecast

5.3.4 Module: Alert

The alert module is one of the default modules of the MagicMirror. This module displays notifications from other modules.

Usage

To use this module, add it to the modules array in the config/config.js file:

```
modules: [
        {
                module: "alert",
                config: {
                        // The config property is optional.
                        // See 'Configuration options' for more information.
                }
        }
]
```

Configuration Options

The following properties can be configured:

Option	Description
Effect	The animation effect to use for notifications. **Possible values:** scale slide genie jelly flip exploader bouncyflip **Default value:** slide
alert_effect	The animation effect to use for alerts. **Possible values:** scale slide genie jelly flip exploader bouncyflip **Default value:** jelly
display_time	Time a notification is displayed in milliseconds. **Possible values:** int

Option	Description
	Default value: 3500
Position	Position where the notifications should be displayed. **Possible values:** left center right **Default value:** center
welcome_message	Message shown at startup. **Possible values:** string false **Default value:** false (no message at startup)

Table 5.6: Configuration Option for Alert

Developer notes

For notifications use:

```
self.sendNotification("SHOW_ALERT", {type: "notification"});
```

For alerts use:

```
self.sendNotification("SHOW_ALERT", {});
```

Notification params

Option	Description
Title	The title of the notification. **Possible values:** text or html
Message	The message of the notification.

Option	Description
	Possible values: text or html

Table 5.7: Notification Parameters

Alert params

Option	Description
Title	The title of the alert. **Possible values:** text or html
Message	The message of the alert. **Possible values:** text or html
imageUrl (optional)	Image to show in the alert **Possible values:** url path **Default value:** none
imageFA (optional)	Font Awesome icon to show in the alert **Possible values:** See Font Awsome website. **Default value:** none
imageHeight (optional even with imageUrl set)	Height of the image **Possible values:** intpx **Default value:** 80px
timer (optional)	How long the alert should stay visible in ms. **Important:** If you do not use the timer, it is

Option	Description
	your duty to hide the alert by using self.sendNotification("HIDE_ALERT");! **Possible values:** int float **Default value:** none

Table 5.8: Alert Parameters

5.3.5 Module: Compliments

The compliments module is one of the default modules of the MagicMirror. This module displays a random compliment.

Using the module

To use this module, add it to the modules array in the config/config.js file:

```
modules: [
        {
                module:"compliments",
                position:"lower_third",        // This can be any of the regions.
                                                                        // Best results in one of the middle regions like: lower_third
                config: { // The config property is optional.
                        // If no config is set, an example calendar is shown.
                        // See 'Configuration options' for more information.
                }       }       ]
```

Configuration Options

The following properties can be configured:

Option	Description

Option	Description
updateInterval	How often does the compliment have to change? (Milliseconds) **Possible values:** 1000 - 86400000 **Default value:** 30000 (30 seconds)
fadeSpeed	Speed of the update animation. (Milliseconds) **Possible values:**0 - 5000 **Default value:** 4000 (4 seconds)
compliments	The list of compliments. **Possible values:** An object with four arrays: morning, afternoon, evening and anytime. See *compliment configuration* below. **Default value:** See *compliment configuration* below.
remoteFile	External file from which to load the compliments **Possible values:** Path to a JSON file containing compliments, configured as per the value of the *compliments configuration* (see below). An object with four arrays: morning, afternoon, evening and anytime. - compliments.json **Default value:** null (Do not load from file)

Table 10.9: Compliments Configuration Options

Compliment Configuration

The compliments property contains an object with four
arrays: morning, afternoon, evening and anytime. Based on the time of the day, the compliments will be picked out of one of these arrays. The arrays contain one or multiple compliments.

If use the currentweather is possible use a actual weather for set compliments. The availables properties are:

- day_sunny
- day_cloudy
- cloudy
- cloudy_windy
- showers
- rain
- thunderstorm
- snow
- fog
- night_clear
- night_cloudy
- night_showers
- night_rain
- night_thunderstorm
- night_snow
- night_alt_cloudy_windy

External Compliment File

You may specify an external file that contains the three compliment arrays. This is particularly useful if you have a large number of compliments and do not wish to crowd your config.js file with a large array of compliments. Adding the remoteFile variable will override an array you specify in the configuration file.
This file must be straight JSON. Note that the array names need quotes around them ("morning", "afternoon", "evening", "snow", "rain", etc.).

5.3.6 Module: Email

Using the module

To use this module, add it to the modules array in the config/config.js file:

```
modules: [
```

```
            {
                        module:'email',
          position:'bottom_left',
          header:'Email',
config:{
              user:'johndoe@gmail.com',
              password:'xxx',
              host:'imap.gmail.com',
              port:993,
tls:true,
authTimeout:10000,
numberOfEmails:5,
              fade:true,
                        maxCharacters:30
       },
              }
]
```

Option	Description
user	Full email address of the user
password	Email password
host	IMAP hostname
post	Port that imap uses **Default value:** 993
tls	Is TLS being used? **Possible values:** true or false **Default value:** true
authTimeout	Number of milliseconds to wait to be authenticated after a

Option	Description
	connection has been established **Default value:** 10000 (10 seconds)
numberOfEmails	Number of emails to display at a time **Default value:** 5
maxCharacters	Maximum number of characters to display **Default value:** 30
fade	Fade older emails to black. (Gradient) **Possible values:** true or false **Default value:** true

Table 5.10: Configuration Options for Email

This module displays emails on Mirror and listens for new incoming emails. When a new email is received, the mirror is updated to display it.

CHAPTER 6

SYSTEM TESTING

API testing was completed for each API in priority order. Testing was first done in the API console provided by the third-party company.

All API calls necessary for the widget were tested in the console. Next, all API calls were made from a simple JavaScript program on a computer with an internet connection. Once all calls are validated, the JavaScript program was run from the embedded computer and all of the outputs are then validated. The API calls were then integrated into widgets that were first tested on the computer, and then the embedded computer. Development was always completed on the computer first because more robust development tools are available, and it left the embedded computer available for other testing. The API tests can be seen in the Appendix under API Tests. Testing software is one of the important stages in development cycle of any product. Testing is the final step of verification and validation which contains series of test cases which makes the system error free. Testing is carried out to make sure that the product works exactly as it supposed to be.

6.1 TESTING METHODOLOGIES

Different types of testing are made in order to test the software. Here the system is tested with unit testing, integration testing and finally system testing. Testing goals at measuring the quality of project eliminate errors of project and provide operational reliability of system. The following section tells the various test cases of different type of testing.

6.1.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business

process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

- **Clock:** We tested the clock by changing the system clock with various time zones and regions. We found out that the Magic Mirror's clock was also updated in real time. No errors were found.
- **Calendar:** We tested the calendar by changing the number of entries to be displayed and changing the fading options. All testes were successful and no errors occurred.
- **Weather Forecast:** For this module we entered an invalid API key and started the Magic Mirror application. There was no forecast displayed on the screen. Next, we tried changing the location to other regions. All the forecast and wind speed about the regions were displayed properly.
- **Compliments:** We changed the compliment text that was going to be displayed on the screen. We also tried adding more compliments to see if that worked. All tests were successful.
- **Email:** We changed the configuration file and typed an invalid email and password. The module showed "Loading" in the Magic Mirror screen and didn't change. The log file indicated that there was an authentication problem. Next we tried changing the number of mail that was displayed. For each number of mails, we specified, the same number of emails was displayed.
- **New York Times News:** To check the correctness of the content we cross referenced the shown new and the official site news. Both were same.

6.1.2 System Test

System testing ensures that the entire integrated software system meets requirement. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

We combined all the modules together and tested the system. Each module worked perfectly as expected and displayed accurate information. We continued the test up to an hour to check the real time updating of all the modules. All modules updated in real time.

We also turned off the network connectivity to see how the modules would respond. Only the clock and compliments were visible, as expected. All other network dependant modules was not shown on the screen.

6.1.3 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

In this testing the configuration file of each module was edited and supplied with various pre-defined values. All modules behaved as expected without any errors.

6.1.4 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

Test Objectives:

• All APIs must work properly.

• User must be identified from the identified API-key.

• The Magic MirrorUI screen and API responses must not be delayed.

All testes were successfully completed with no errors.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 7

RESULTS AND DISCUSSION

Calendar

Below fig shows upcoming holiday events.

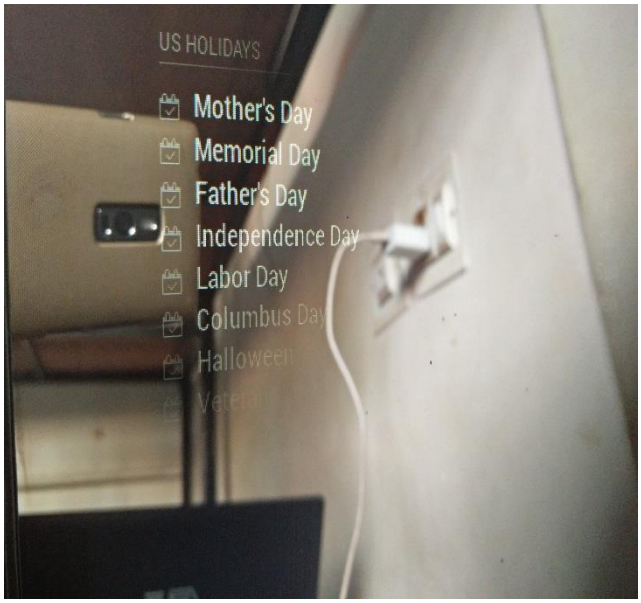

Fig 7.1: List of events

Weather Forecast

Below fig shows the weather forecast for Delhi,India.

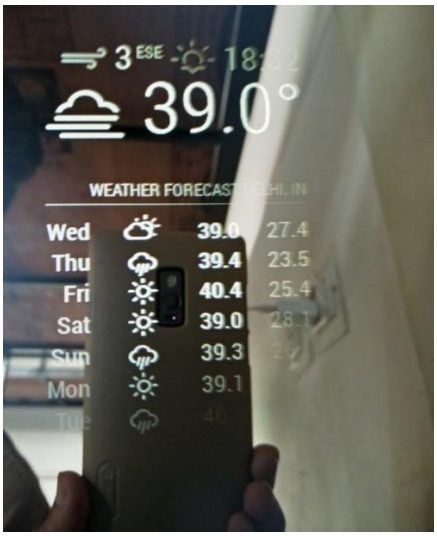

Fig 7.2: Weather forecasting

Email

Below fig shows the recent email notification of the user.

Fig 7.3: Email notification

New York Times News

Below fig shows the latest headlines from New York Times.

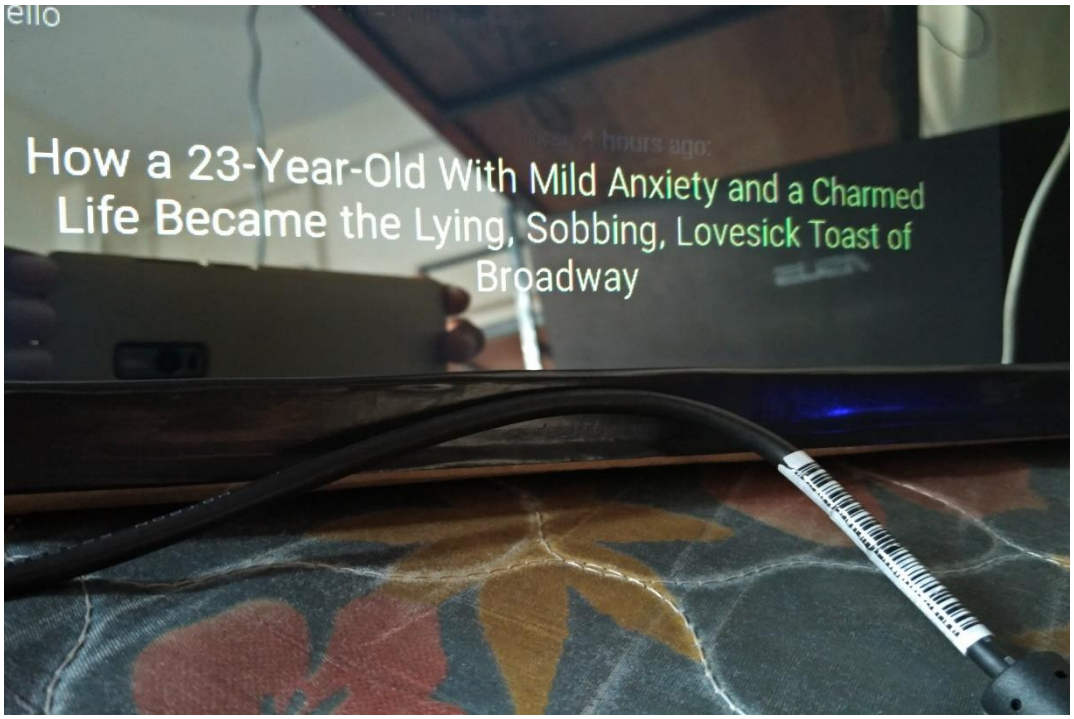

Fig 7.4: News feeds

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

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 smart mirror idea was created to give instant access to information in a convenient and time-saving environment, the bathroom. All other aspects of the mirror's design developed from these ideas and inspirations. The goals of the smart mirror were to aim to reduce time needed in a user's daily routine and provide a merger of user and technology that becomes an enhancement, not a new burden. The functionality must meet these descriptions in the design. The smart mirror did the thinking for the user with intelligent, commonly used applications. Modules like their calendar, news, mail and weather will be available. These modules are unobtrusively displayed on the screen, hidden by the two-way mirror, as to look like a seamless experience.

8.2 FUTURE ENHANCEMENTS

- A more reflective glass can be used.
- A more powerful device than the Raspberry Pi 2 can be used.
- Voice integration like Amazon's Alexa can be used.
- Facial recognition and motion detection sensors can be integrated.