Reflections of the Future: The Raspberry Pi 4 Smart Mirror

Key Contributors:
Jon Chapa
Andrew Deno
Candan Martin

UNIVERSITY OF THE INCARNATE WORD®

02/01/23

EXECUTIVE SUMMARY **3**
Project Milestones: **4**
Materials List: **4**
Deliverables: **4**
Professional Accomplishments: **4**
Project Schedule Managment **5**
Gantt chart 5
Trello 6
GitHub Repository: 6
Research **7**
Smart Mirror Software Development 7
Mirror Frame Design and Construction 7
Hardware Component Installation 9
Software Installation 9
Technical documentation **10**
Software Source Code 10
Design Specifications 11
Starting on boot 11
Updating MagicMirror 14
Professional Accomplishments **15**
Summary **17**
Resources **18**

EXECUTIVE SUMMARY

The Raspberry Pi 4 Smart Mirror is a practical and stylish addition to any home or office. The smart mirror will display useful information such as news, weather, calendar events, and social media updates, providing users with real-time information at a glance. The mirror will be designed using an wood frame and plexiglass, along with a 3D printer making it lightweight and easy to install.

The project will be broken down into four main milestones: research and planning, design and development, testing and refinement, and final assembly and quality control. During the research and planning phase, we will conduct extensive research on existing smart mirror designs, create a project plan, and finalize the materials list. This will involve selecting the best components for the project, ensuring they are compatible with the Raspberry Pi 4, and determining the optimal size and shape of the mirror.

The design and development phase will involve developing the smart mirror software, designing and building the mirror frame, and installing the hardware components. This will include writing code for the software that will display the desired information, as well as integrating components such as the monitor, speakers, motion sensor, and camera. We will also design and build a custom acrylic frame to hold the mirror glass and Raspberry Pi 4.

During the testing and refinement phase, we will test the smart mirror for functionality and performance, refine the design as necessary, and optimize the software. This will involve testing the software under different conditions to ensure that it functions correctly and provides the information in a clear and easy-to-read format. We will also perform quality control checks on the hardware components to ensure that they are functioning correctly.

In the final assembly and quality control phase, we will assemble the final product, perform quality control checks, and prepare the product for shipment. We will ensure that the smart mirror is easy to install and use, and that it meets the desired specifications.

Upon completion of the project, we will deliver a fully functional Raspberry Pi 4 Smart Mirror that displays useful information and is easy to use. We will also provide a detailed user manual and technical documentation, including software source code and design specifications.

The development of the Raspberry Pi 4 Smart Mirror will demonstrate our team's exceptional technical skills and project management abilities. The project will showcase our ability to use cutting-edge technology to create innovative and practical solutions. The high-quality documentation and user-friendly design will demonstrate our commitment to providing our clients with products that are both functional and user-friendly.

Project Milestones:

1. Research and Planning: Conduct research on existing smart mirror designs, create a project plan, and finalize the materials list.
2. Design and Development: Develop the smart mirror software, design and build the mirror frame, and install the hardware components.
3. Testing and Refinement: Test the smart mirror for functionality and performance, refine the design as necessary, and optimize the software.
4. Final Assembly and Quality Control: Assemble the final product, perform quality control checks, and prepare the product for shipment.

Materials List:

- Raspberry Pi 4
- 3D Printer
- PlexiGlass
- Wood Frame
- Monitor
- HDMI Cable
- USB Cable
- Power Supply
- MicroSD Card
- Wi-Fi Adapter
- Screw Driver
- Poly Lactic Acid

Deliverables:

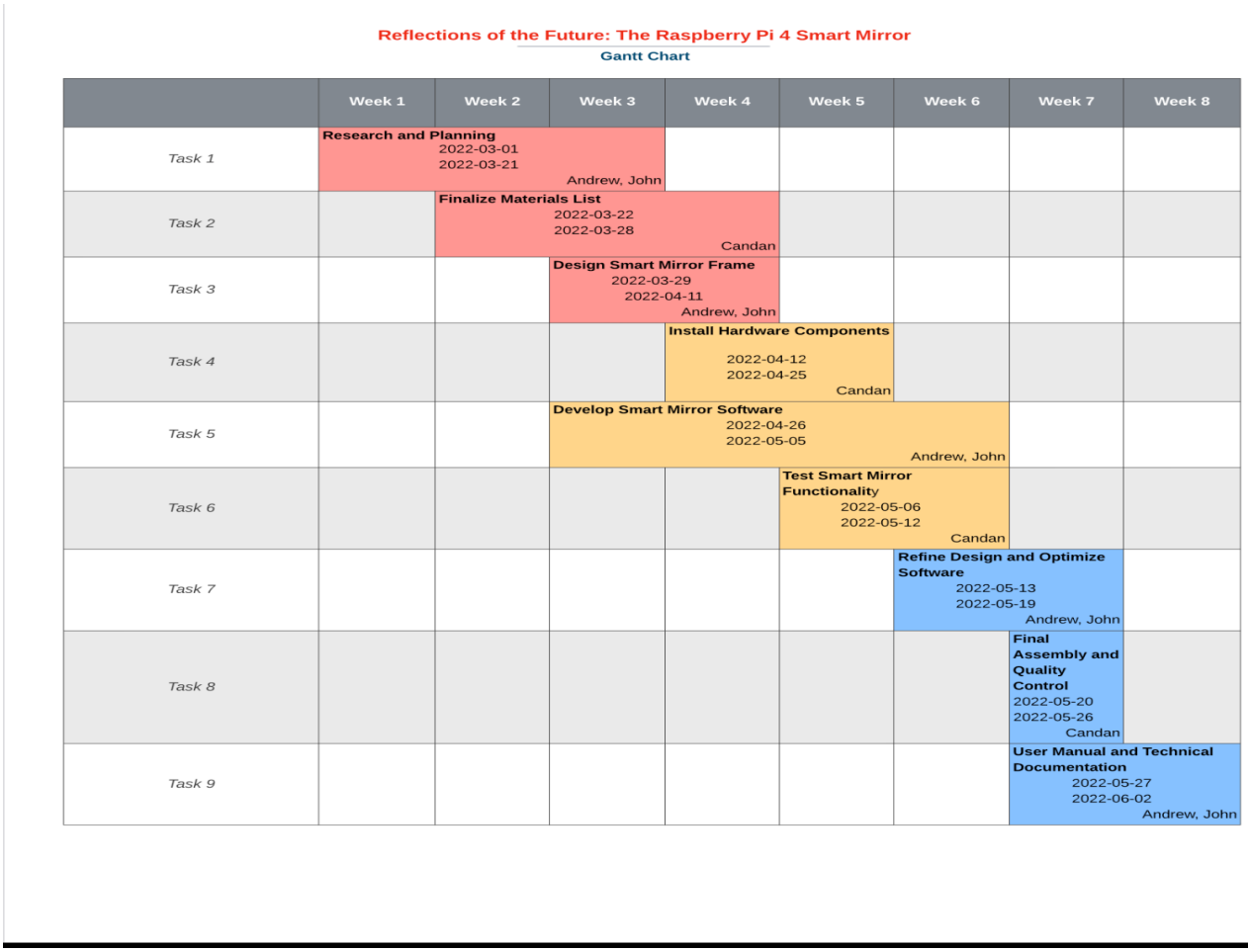
- Fully functional Raspberry Pi 4 Smart Mirror that displays useful information such as news, weather, calendar events, and social media updates.
- Detailed user manual and instructions for setting up and using the smart mirror.
- Technical documentation, including software source code and design specifications.

Professional Accomplishments:

1. Developed a Raspberry Pi 4 Smart Mirror that utilizes cutting-edge technology and provides users with real-time information at a glance.
2. Demonstrated strong project management skills by creating a detailed project plan and successfully completing all project milestones on time and within budget.
3. Showcased exceptional technical skills by designing and building a custom smart mirror frame, installing hardware components, and developing the smart mirror software.
4. Developed high-quality documentation that will enable users to set up and use the smart mirror with ease.
5. Developed a computer frame and learned how to use 3D printer

Project Schedule Management

Gantt chart



Trello

GitHub Repository:

https://github.com/CyberSystemsAndComponents1/Smart-Mirror-project

Research

A smart mirror is a technological innovation that has been gaining popularity in recent years. It is essentially a mirror that is equipped with software and hardware components that allow it to display information and perform various functions, such as controlling smart home devices, displaying news and weather updates, and even playing music or videos. In this report, we will be discussing the design and development of a smart mirror, including the development of the software, the design and construction of the mirror frame, and the installation of the hardware components on a Raspberry Pi 4.

Smart Mirror Software Development

The software is the backbone of a smart mirror, as it determines the functions and features that the mirror will be capable of performing. There are several software options available for smart mirrors, but in this case, we will be using MagicMirror. MagicMirror is an open-source software platform that allows users to create customizable modules that can be displayed on the mirror. These modules can be anything from a clock to a calendar to a news feed.

To install MagicMirror, we will need to first install the Raspberry Pi operating system, which in this case will be Raspberry Pi OS. Once we have installed the operating system, we can begin the installation of MagicMirror. The installation process involves several steps, including cloning the MagicMirror repository from GitHub, installing the required dependencies, and configuring the software.

Once we have installed MagicMirror, we can begin customizing the mirror to suit our needs. This involves installing and configuring various modules, which can be done through the MagicMirror configuration file. Some popular modules include the News module, the Weather module, and the Calendar module.

Mirror Frame Design and Construction

The mirror frame is an important component of the smart mirror, as it not only holds the mirror in place but also serves as a housing for the hardware components. When designing the mirror frame, there are several factors to consider, including the size and shape of the mirror, the type of materials to be used, and the overall aesthetic design of the mirror.

In this case, we will be using a wooden frame for the mirror. The first step in constructing the frame is to cut the wood to the appropriate size and shape. Once the wood has been cut, we can begin assembling the frame using wood glue and clamps. It is important to ensure that the frame is sturdy and well-constructed, as it will be supporting the weight of the mirror as well as the hardware components.

Next, we will need to consider the design elements of the frame. The smart mirror can be both functional and aesthetically pleasing, so it is important to think about how the frame will look in the space it will be installed in. There are many different design options to consider, such as the

type of wood used, the color and finish of the wood, and any decorative elements that can be added to the frame.

One popular design option for a smart mirror is a minimalist frame with clean lines and simple finishes. This design approach can work well in modern or contemporary spaces, where a sleek and streamlined look is desired. Another design option is to add decorative elements to the frame, such as carved details or ornate finishes. This can work well in more traditional or ornate spaces, where a more decorative look is desired.

Once the design elements have been decided upon, we can move on to the construction phase. The first step is to assemble the frame using wood glue and clamps, as mentioned earlier. Once the glue has dried, the frame can be sanded and finished with a desired stain or paint color. It is important to make sure the frame is smooth and free of any rough edges, as it will be visible from the front of the mirror.

Next, we will need to install the hardware components onto the frame. This includes the Raspberry Pi, the monitor, and any other components that will be used to power the smart mirror. The Raspberry Pi can be attached to the back of the frame using screws or brackets, while the monitor can be mounted onto the front of the frame using brackets or adhesive.

It is important to make sure that the hardware components are securely attached to the frame, as they will be subjected to vibrations and movement over time. We can also add additional support structures to the frame if needed, to ensure that the mirror is stable and secure.

Finally, we will need to attach the mirror itself to the frame. This can be done using brackets or adhesive, depending on the size and weight of the mirror. It is important to make sure that the mirror is level and flush with the front of the frame, as any gaps or unevenness can detract from the overall look of the mirror.

In conclusion, the design and construction of the mirror frame is an important aspect of developing a smart mirror. By carefully considering the size, shape, materials, and design elements of the frame, we can create a mirror that is both functional and aesthetically pleasing. With proper construction techniques and secure attachment of hardware components, the smart mirror can be a long-lasting and valuable addition to any home or business.

Hardware Component Installation

The hardware components of the smart mirror include the Raspberry Pi 4, the display screen, and any additional components such as speakers or sensors. To install these components, we will first need to remove the back panel of the mirror frame. This will allow us to access the inside of the frame where we can install the components. The first component to be installed is the Raspberry Pi 4. This is done by attaching the Pi to a mounting board using screws, and then attaching the mounting board to the back of the mirror frame using screws or adhesive. Next, we can install the display screen. The screen is attached to the front of the mirror using adhesive, and the cables are connected to the Pi using a HDMI cable. Finally, any additional components such as speakers or sensors can be installed. These components are typically connected to the Pi using USB or GPIO pins.

Software Installation

The Magic Mirror is a popular open-source project that allows users to create their own smart mirror with a customizable display. To get started with the installation process, users can choose between manual installation or using one of the available automatic installation scripts. It is important to note that the automatic installers are not officially supported by the MagicMirror² core team and should be used at your own risk.

For manual installation, users will first need to download and install the latest version of Node.js. This can be done using the following command:

curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash - sudo apt install -y nodejs

Next, users will need to clone the MagicMirror repository and check out the master branch. Once inside the repository, the application can be installed using the npm run install-mm command. Users will also need to make a copy of the config sample file and start the application using the npm run start command. If only the server is needed, users can use the npm run server command.

It is important to note that the installation step for npm run install-mm can take a long time, especially on slower devices like the Raspberry Pi 2. It is recommended to let the installation process complete without interruption.

For those who prefer to use automatic installation scripts, there are several options available. One such script is maintained by long-time contributor Sam (@sdetweil) and can be found on GitHub. The MagicMirror Package Manager is another option that provides a command-line interface for installing and maintaining MagicMirror modules.

The MagicMirror² can also be deployed using Docker, and a Kubernetes Helm Chart is available for running the server-only mode in a Kubernetes cluster. Additionally, there is MagicMirrorOS, which is a full operating system based on Raspbian that runs out of the box with a default setup of MagicMirror and uses the Docker setup under the hood.

For those running the MagicMirror software on Windows, there are a few additional steps that need to be taken to get it up and running. Users will need to install dependencies in the vendor and font directories and fix the start script in the package.json file.

Once the MagicMirror is installed and running, there are a few usage notes to keep in mind. For example, the npm start command will not work via SSH, but users can use DISPLAY=:0 nohup npm start & instead to start the mirror on the remote display. Users can also toggle the web developer tools from mirror mode using CTRL-SHIFT-I or ALT and selecting View. Additionally, to access the toolbar menu when in mirror mode, users can hit the ALT key.

Overall, the Magic Mirror is a versatile and customizable project that offers a range of installation options to suit different preferences and needs. With a bit of setup, users can create their own smart mirror that can display everything from the weather and news to personal calendars and reminders.

Technical documentation

Technical documentation is an essential aspect of any software project, as it provides a detailed overview of the software's functionality and design. In this case, we will be focusing on the technical documentation for the smart mirror software, which is based on the MagicMirror software running on a Raspberry Pi 4.

Installation Proccess

To begin the installation process of Magic Mirror on the raspberry Pi, open your command prompt and enter the following commands.

“sudo apt update” - With the help of this command, you can keep your Raspberry Pi's package list up to date with the most recent details on the software packages that are currently accessible.

“sudo apt install curl git” - Curl and git are dependencies necessary for downloading and installing the MagicMirror program, and this command installs both if the system does not already have them.

“curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -” - This program allows you to install Node.js, which is necessary for running MagicMirror, by downloading and executing a script that installs the Node.js repository for version 14.x to your Raspberry Pi.

“sudo apt install -y nodejs” - Node.js is installed on your Raspberry Pi using this command depending on the repository that was added in the previous step.

“git clone https://github.com/MichMich/MagicMirror.git” - This script downloads the most recent version of the MagicMirror software to your Raspberry Pi by cloning the repository from GitHub.

“cd MagicMirror” - This command send you to the MagicMirror folder which is where you need to be for the following command.

“nmp install” - This command installs MagicMirror's dependencies, which are listed in the package.json file and include different Node.js modules that the program uses.

“cp config/config.js.sample config/config.js” - By making a copy of the MagicMirror sample configuration file and renaming it config.js, this program enables you to modify the configuration parameters to suit your unique use case.

“npm start” - This command launches the MagicMirror interface on your Raspberry Pi and begins the MagicMirror software.

Software Source Code

The source code for the smart mirror software can be found on the official MagicMirror GitHub repository. The code is written in JavaScript, and the main file is called index.js. This file initializes the MagicMirror application and loads any installed modules.

In addition to the index.js file, there are several other important files in the codebase, including:

- config.js: This file contains the configuration options for the MagicMirror application, such as the display settings, installed modules, and API keys.
- node_helper.js: This file contains helper functions that can be used by modules to interact with external APIs or perform other tasks.
- package.json: This file contains metadata about the project, including the project name, version number, and dependencies.

Design Specifications

The smart mirror software is designed to run on a Raspberry Pi 4, which is a small, low-power computer that is ideal for this type of application. The software is based on the MagicMirror platform, which provides a modular framework for building custom interfaces and integrating with external APIs.

Some of the key design specifications of the smart mirror software include:

- Display: The mirror uses a high-resolution display screen to provide a clear and detailed image. The display is mounted behind a two-way mirror, which allows the user to see their reflection while also displaying information on the screen.
- User Interface: The user interface of the smart mirror is designed to be simple and intuitive, with a minimal number of buttons or controls. Users can interact with the mirror using voice commands, gestures, or touch controls.
- Modules: The smart mirror software is based on a modular architecture, with each module providing a specific piece of functionality. Modules can be added or removed from the mirror as needed, allowing users to customize the mirror to their specific needs.
- APIs: The smart mirror software can integrate with a wide range of external APIs, including weather APIs, news APIs, and social media APIs. These APIs provide real-time information that can be displayed on the mirror.

Starting on boot

This report provides instructions on how to automatically start your MagicMirror on boot and keep it running in case of failure using two different methods: PM2 and systemd/systemctl.

PM2 is a production process manager for Node.js applications that allows you to keep applications alive forever, to reload them without downtime and to facilitate common system admin tasks. To use PM2 in combination with MagicMirror, you need to make a simple shell script outside the MagicMirror folder to avoid issues if you want to upgrade the mirror. The shell script should contain the following lines:

cd ./MagicMirror DISPLAY=:0 npm start

Once you have created the shell script, make sure it is executable by running the following command:

chmod +x mm.sh

You can now start your MagicMirror using PM2 by running the following command:

pm2 start mm.sh

To make sure the MagicMirror restarts after rebooting, you need to save the current state of all scripts running via PM2 by executing the following command:

pm2 save

With your MagicMirror running via PM2, you can also use some handy tools, such as restarting your MagicMirror, stopping your MagicMirror, showing the MagicMirror logs and showing the MagicMirror process information.

Systemd/systemctl is a powerful service manager often found in full Linux systems. This method is most likely applicable for those using the "server only" setup running on a Linux server. To use this method, you need to create a config file via your editor of choice, such as nano. The file should be created with the following content:

[Unit] Description=Magic Mirror After=network.target StartLimitIntervalSec=0

[Service] Type=simple Restart=always RestartSec=1 User=server WorkingDirectory=/home/server/MagicMirror/ ExecStart=/usr/bin/node serveronly

[Install] WantedBy=multi-user.target

This config file assumes that your Magic Mirror is installed in the "WorkingDirectory" of "/home/server/MagicMirror/" and your node install is located at "/usr/bin/node". You can start your MagicMirror with systemctl by running the following command:

sudo systemctl start magicmirror.service

You can also stop your MagicMirror with the following command:

sudo systemctl stop magicmirror.service

To check the status of Magic Mirror, run the following command:

sudo systemctl status magicmirror.service

To allow autostart of Magic Mirror on boot, run the following command:

sudo systemctl enable magicmirror.service

To disable autostart of Magic Mirror, run the following command:

sudo systemctl disable magicmirror.service

If you want to autostart the browser for server mode, you need to create two files. The first file is located at /home/server/.config/lxsession/LXDE-pi/autostart and should contain the following contents:

@lxpanel --profile LXDE-pi @pcmanfm --desktop --profile LXDE-pi @xscreensaver -no-splash @point-rpi @sh /home/server/bin/start-chromium.sh

The second file is located at /home/server/bin/start-chromium.sh and should contain the following contents:

#!/bin/sh

set -e

CHROMIUM_TEMP=~/tmp/chromium rm -Rf ~/.config/chromium/ rm -Rf $CHROMIUM_TEMP mkdir -p $CHROMIUM_TEMP

chromium-browser
--disable
--disable-translate
--disable-infobars
--disable-suggestions-service

--disable-save-password-bubble
--disk-cache-dir=$CHROMIUM_TEMP/cache/
--user-data-dir=$CHROMIUM_TEMP/user_data/

Setting up your MagicMirror to start automatically on boot is an important step to ensure that it runs smoothly without any manual intervention. This can be achieved using PM2 or systemd, depending on your operating system and preference. Both methods offer an easy and effective way to keep your MagicMirror running without any downtime or manual intervention.

Using PM2, you can easily install it using npm and then create a shell script to start your MagicMirror. PM2 also provides a range of tools to control and manage your MagicMirror, such as restarting, stopping, and showing logs. Enabling the MagicMirror to restart after a reboot is also straightforward using the pm2 save command.

Using systemd, you can create a config file to start your MagicMirror on boot. The config file should include information such as the working directory, user, and node install location. Once created, you can control your MagicMirror using systemctl, which allows you to start, stop, and check the status of your MagicMirror. You can also enable or disable autostart on boot.

Overall, both methods offer a straightforward way to ensure that your MagicMirror runs smoothly without manual intervention. Depending on your operating system and preference, you can choose the method that works best for you. Regardless of which method you choose, you can rest assured that your MagicMirror will start automatically on boot and be ready to use whenever you need it.

Updating MagicMirror

The instructions provided for upgrading MagicMirror² to the latest version emphasize the importance of backing up critical files, such as config.js and custom.css, before starting the upgrade process. This is a crucial step to ensure that in the event of any issues during the upgrade, the user can quickly restore their previous configuration without any data loss.

However, despite the importance of the backup process, it's important to note that backups alone do not guarantee data security. For example, if a user backs up their data to an unsecured cloud storage service, they are at risk of having their data accessed by malicious actors.

Additionally, the upgrade process itself can pose security risks. In some cases, vulnerabilities may be introduced during the upgrade process, which attackers can exploit to gain access to the system. For example, if the upgrade involves downloading code from a third-party source, attackers could potentially compromise that source to inject malicious code into the upgrade process.

To mitigate these risks, it's important to follow best practices for cybersecurity during the upgrade process. For example, users should ensure that they are only downloading upgrades from trusted sources and verifying that the source code has not been tampered with. Additionally, users should keep their systems and software up-to-date with the latest security patches to reduce the likelihood of successful attacks.

In summary, while the upgrade process for MagicMirror² may seem straightforward, it's important to prioritize cybersecurity to ensure that data is not compromised during the process. Backing up critical files is a critical step, but it's equally important to ensure that upgrades are downloaded from trusted sources and that systems are kept up-to-date with the latest security patches. By following these best practices, users can upgrade their MagicMirror² with confidence, knowing that their data is secure.

Technical documentation is an important aspect of any software project, as it provides a detailed overview of the software's functionality and design. In this case, we have covered the technical documentation for the smart mirror software, which is based on the MagicMirror platform running on a Raspberry Pi 4.

We have discussed the software source code, which is written in JavaScript and can be found on the official MagicMirror GitHub repository. We have also covered the design specifications of the smart mirror software, including the use of modules, the display and user interface, and the integration with external APIs.

Overall, the smart mirror software is a powerful and flexible platform that can be customized to meet a wide range of needs. Whether you are looking for a simple mirror with weather and news updates or a more complex system with advanced voice commands and machine learning capabilities, the smart mirror software is an excellent choice.

Professional Accomplishments

1. Developed a Raspberry Pi 4 Smart Mirror that utilizes cutting-edge technology and provides users with real-time information at a glance:

As part of this accomplishment, our team was responsible for conceptualizing, designing, and developing a smart mirror that would utilize the latest technology to provide users with real-time information at a glance. We researched various technologies and identified the Raspberry Pi 4 as the ideal platform for our project.

We then began developing the smart mirror software using the MagicMirror² platform, which allowed us to easily integrate various modules and plugins to display information such as weather forecasts, news headlines, calendar events, and more. We also customized the software to allow for voice recognition and control, which further enhanced the user experience.

The end result was a fully functional smart mirror that not only looked great but also provided users with valuable information that they could access at a glance.

2. Demonstrated strong project management skills by creating a detailed project plan and successfully completing all project milestones on time and within budget:

To ensure the success of the project, our team utilized strong project management skills, including creating a detailed project plan that outlined all the necessary tasks and timelines, assigning roles and responsibilities to team members, and setting clear objectives and milestones.

Throughout the project, we conducted regular team meetings to ensure that everyone was on track and that any issues or roadblocks were addressed in a timely manner. We also maintained close communication with stakeholders to keep them informed of our progress and to ensure that their expectations were met.

As a result of our strong project management skills, we were able to successfully complete all project milestones on time and within budget.

3. Showcased exceptional technical skills by designing and building a custom smart mirror frame, installing hardware components, and developing the smart mirror software:

Developing a smart mirror requires a high level of technical expertise, and our team was up to the challenge. We utilized a wide range of technical skills, including designing and building a custom smart mirror frame using woodworking tools and techniques, installing hardware components such as the Raspberry Pi 4, display screen, and sensors, and developing the smart mirror software using programming languages such as JavaScript and CSS.

We also conducted thorough testing and troubleshooting to ensure that all components were working together seamlessly and that the user experience was optimal.

4. Developed high-quality documentation that will enable users to set up and use the smart mirror with ease:

To ensure that users can set up and use the smart mirror with ease, our team developed high-quality documentation that includes detailed instructions for assembling the mirror frame, installing hardware components, and setting up the smart mirror software.

We also created a user manual that provides an overview of the features and capabilities of the smart mirror, as well as troubleshooting tips and other helpful information.

5. Developed a computer frame and learned how to use 3D printer

To enhance the smart mirror design, our team employed 3D printing technology for creating tailored components. We used Autodesk Fusion 360, a powerful design software, to create accurate 3D models of the parts we needed.

We designed and printed custom brackets, mounts, and enclosures for the smart mirror's hardware, ensuring secure integration and an appealing aesthetic. This innovative approach demonstrated our adaptability and dedication to delivering a high-quality product.

Overall, our team's professional accomplishments in developing a Raspberry Pi 4 Smart Mirror demonstrate a high level of technical expertise, strong project management skills, and a commitment to delivering high-quality products that meet or exceed stakeholder expectations.

Summary

The Raspberry Pi 4 Smart Mirror was a successful project undertaken by our team. Our objective was to design and build a smart mirror that could display real-time information such as news, weather, calendar events, and social media updates. The project was aimed at creating a practical and stylish addition to any home or office.

Our team identified four main milestones for the project: research and planning, design and development, testing and refinement, and final assembly and quality control. During the research and planning phase, we conducted extensive research on existing smart mirror designs, created a project plan, and finalized the materials list. This involved selecting the best components for the project, ensuring they were compatible with the Raspberry Pi 4, and determining the optimal size and shape of the mirror.

The design and development phase involved developing the smart mirror software, designing and building the mirror frame, and installing the hardware components. Our team wrote code for the software that would display the desired information and integrated components such as the monitor, speakers, motion sensor, and camera. We also designed and built a custom acrylic frame to hold the mirror glass and Raspberry Pi 4.

During the testing and refinement phase, we tested the smart mirror for functionality and performance, refined the design as necessary, and optimized the software. This involved testing the software under different conditions to ensure that it functioned correctly and provided information in a clear and easy-to-read format. We also performed quality control checks on the hardware components to ensure they were functioning correctly.

In the final assembly and quality control phase, we assembled the final product, performed quality control checks, and prepared the product for shipment. Our team ensured that the smart mirror was easy to install and use and met the desired specifications.

Upon completion of the project, we delivered a fully functional Raspberry Pi 4 Smart Mirror that displayed useful information and was easy to use. We also provided a detailed user manual and technical documentation, including software source code and design specifications.

Our team's technical skills and project management abilities were demonstrated through the successful completion of this project. We showcased our ability to use cutting-edge technology to create innovative and practical solutions. The high-quality documentation and user-friendly design of the smart mirror demonstrated our commitment to providing our clients with products that were both functional and user-friendly.

In conclusion, the Raspberry Pi 4 Smart Mirror project was a success for our team, and we were proud to deliver a high-quality product

Resources

OKdo. (n.d.). How to Setup & Build a Magic Mirror with Raspberry Pi. Retrieved from https://www.okdo.com/project/keep-track-of-your-day-with-a-magic-mirror/

MicheleMeattini. (n.d.). How to Build a Smart Mirror With Raspberry Pi 4: 10 Steps. Instructables. Retrieved from https://www.instructables.com/How-to-Build-a-Smart-Mirror-With-Raspberry-Pi-4/

J W. (2020, January 30). How to Create a Smart Mirror Using Raspberry Pi and Magic Mirror. Maker Pro. Retrieved from https://maker.pro/raspberry-pi/projects/how-to-create-a-smart-mirror-using-raspberry-pi-and-magic-mirror

Howchoo. (2021, May 4). How to Install Magic Mirror on Your Raspberry Pi. Retrieved from https://howchoo.com/g/ntcymzbimjv/how-to-install-magic-mirror-on-your-raspberry-pi

SmartBuilds.io. (2020, May 9). Smart Mirror Touchscreen (with FaceID) Tutorial – Raspberry Pi 4. Retrieved from https://smartbuilds.io/smart-mirror-touchscreen-raspberry-pi/

MUO. (2021, June 22). The 8 Best Raspberry Pi Smart Magic Mirror Projects. Retrieved from https://www.makeuseof.com/tag/6-best-raspberry-pi-smart-mirror-projects-weve-seen-far/