Justin Villalon

Project Proposal

Smarty-Py Mirror

Project Description

A smart mirror is essentially a display behind a two-way mirror that provides an array of information to the user through widgets and enables them to multitask. The display is powered by a Raspberry Pi 3 and allows for interaction and customization by individual users. The mirror also will not turn on until the webcam attached detects a face and will shut off when no one is in front of it for long enough. The widgets will display the local time, weather, and news at the very least and will make suggestions to the user if it is cold enough to need a jacket or sunny enough to need sunglasses.

Competitive Analysis

There have been a lot of DIY Smart Mirror projects in the last few years and I will be incorporating many similar features these previous ones have utilized but I will also be taking a different approach than most. For beginners, my hardware will be very similar although I will be including a web cam which was only sometimes included in projects. This optional feature will allow me to do facial detection and recognition unlike most other projects. While I have seen one other project that successfully integrated facial detection, it did not include facial recognition which is a feature that very few projects have. Regarding software, most of these projects use an open modular platform called MagicMirror2 that installs a lot of modules by default. I on the other hand will be coding from scratch in Python which will allow for a different approach to the device than most people take.

Like other projects, I will be running my program through a Raspberry Pi 3 and include the same basic widgets that display common information like the time, weather, and news. Like many other projects, I will also be storing these widgets as classes and web scraping to update information. I will also just be putting the monitor in a frame and behind a two-way mirror because I do not have the time for a more intricate hardware design for this project.

Structural Plan

Since the display is the most important part of the project, the main file is an animation that calls all other files. Each file outside of this will means it completes a function that is not related to the MVC. For example, there is a file containing a class for each widget that just stores information and there is a file that just returns true if a face is present. All new modules will complete their functions in files separate from the main one. These files would have one function which returns information for either the widgets or the display as a whole to react to.

Algorithmic Plan

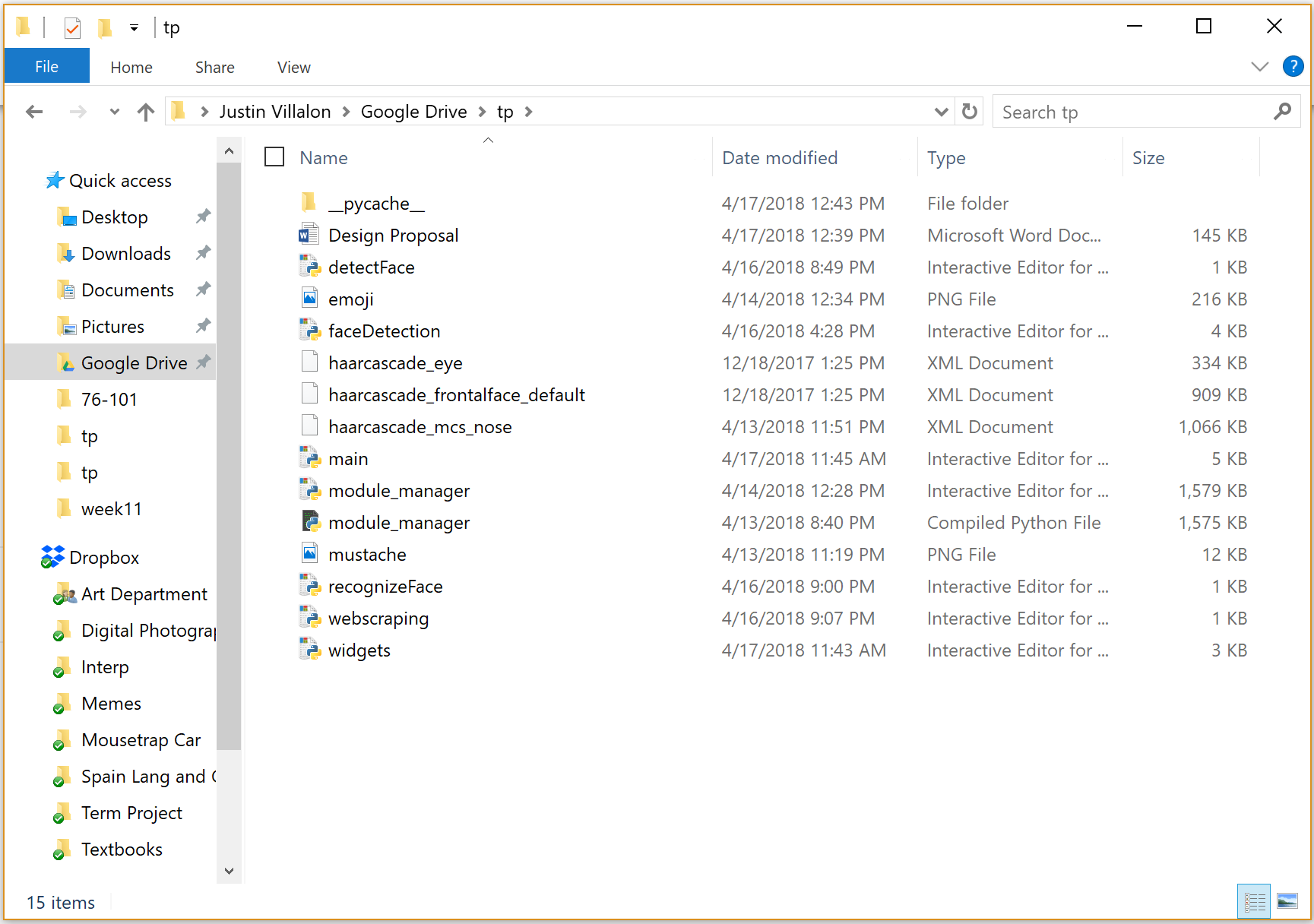
The trickiest part about this project is going to be recognizing certain users and how new users could potentially add themselves to the mirror. This means before running the mirror I would have to train the mirror to certain faces and store the names of the individuals in a list, so the mirror can respond personally to each person. This way if one of those people approach the mirror it will know who it is and respond accordingly. Otherwise, I could have the mirror either respond with a general response that is not personal or ask for input to train it, but the mirror would have no easy way of inputting once it is mounted on a wall. This means I will have to save reference images for the recognition program on the SD card on the raspberry pi before trying to use the program. To begin this process I should also complete the frame of the mirror to mount the webcam and place the raspberry pi since those will be necessary to achieve facial recognition.

Timeline Plan

By Sunday the 22nd I plan to have the hardware completed enough to where I can begin transferring my current code to the raspberry pi. This allows me time before TP2 on the 25th to complete facial recognition and web scraping. By May 3rd I would like to incorporate PyAudio in some way and have hand controls done through OpenCV.

Version Control Plan

For this project, I have installed the Google Drive application for desktop and placed my term project folder in the Google Drive folder which continuously syncs with the cloud. This also enables me to see previous versions if something were to go wrong with my code. In case something to go wrong with laptop, I am able to access the same versions of these files on my desktop.



Module List

I will primarily be using OpenCV, Raspberry Pi, and web scraping for this term project. After MVP I will consider using PyAudio so I can use voice commands and add more interaction with the mirror.

TP2 Update

Since TP1, I have decided to adopt a different way to run my tkinter window rather than the animation template provided in the course notes. I am packing the widgets in the corners instead, so the screen can fit any monitor size. What used to be helper functions in the animation have been moved to their respective class in the widgets file which has become the main file to run on. All OpenCV and webscraping functions are still run within their own respective files.

TP3 Update

Since TP2, no major changes have occurred within my code except for the addition of PyAudio.