# Tidy and Reconfigure a PC

**Easily tidy and reconfigure a pc\***

A common problem when developing software is the need to install development tools, libraries and frameworks so that a developer can compile and debug a project. For some strange reason development tools are often the most brittle and complex software to install and run. More generally, IT departments often need to manage many computers of an organisation and install software on them so that everyone is using up to date software needed to run the organisation. Similarly, users often fill their hard drives with files and programs they don’t need but don’t know how to safely tidy the machine up. The purpose of this project is to create a smart system that understands widely used software and files and can automate the setup of a computer into a manageable state.

Able to take a windows pc and can identify the installed software (by examining the file system, environment variables, autorun applications and running processes), (can easily log before and after an install to learn the changes that an application installs) (possibly try to automatically detect application installs and so automatically learn how they change the computer on install)

Has a plugin architecture with logic for different common programs so it can identify user files so that user files can be backed up, removed locally, restored, and potentially repositioned to ‘tidy’ the pc

Has plugins for automating the installation and removal of a program/driver etc. so that any computer can be automatically configured into a standard state

Has plugins for creating projects e.g. a web app, that installs multiple dev tools if needed, creates some template files and utility scripts e.g. checkin and check out (for github)

Multiple students can work on this project with each student focusing on different applications being automated.

***Ambitious test cases***

Can install and configure python, visual studio code, create a virtual environment for the project, install flask libraries, setup self signed certificate for ssl under flask

Can install Windows subsystem for Linux, cuda and docker gpu to enable research AI programs to be run on a windows machine

***Language and libraries***

Python (including dynamically loaded python scripts)

PyInstaller (To convert a python program into an executable)

PySelenium for automating downloads of drivers etc. from hardware providers

***Further work – system should be able to expand to perform this functionality but doesn’t need to do it for V1 (the main focus of the module)***

Application logic can extract metadata from common files so that existing files can be visualised and searched for easily

Integration with cloud storage for backup (ideally with encryption)

Automated integration with a private github repository for version control of user files

Extended to work with Android devices

Can work with multiple devices so that users can have a single view of multiple computers and devices

Can index and backup emails and other files stored on external cloud systems (like google drive and github)

# Week 1

## 18/01/2021

For the start of the project, I have attempted to break down the areas of the project, by posing parts of the project as questions. I’m using Trello to plan my work. I asked myself, “What makes a computer tidy?”

* No single use screenshots
* No redundant applications, such as multiple browsers
* No unused applications
* No unused files
* User files stored in as few locations as possible to reduce the number of locations that need searched
* No unnecessary duplication of files
* No previous versions of applications left when a new version is installed
* No unnecessary installers left over from installation

Part of the spec says that development tools should be able to be automatically installed, so my next question was “What tools are required at a basic level for each major programming language?”

* Java requires the Java JDK for any project
* Python projects require the installation of python and the Python Package Installer
* As far as I can tell, C# requires visual studio to run
* I have not looked at any of the other major languages at this point

A question that I have asked but not found an answer to is “How to differentiate between user files and system/program files?”, which would allow me to move user files without accidentally removing a text file required by a program

To make uninstallation work properly, you need to be able to discern which files an installer created. One way of figuring this out would be reverse engineering the installer, but this would often be illegal/in breach of T&Cs. The other method is to record before and after state, so I asked the question “How can you record the current state of the file system in Python?”

* Through research, I found out that there is a python library called Watchdog, that can detect changes in a folder and subfolder over a given period of time
  + The library was created because the creator was annoyed that there wasn’t a library that did this, which seems to be why most libraries are created
* Within Watchdog, the dirs\_created, dirs\_modified and dirs\_moved commands, along with their file counterparts appear to be what I need for this project
* The Winreg library can be used to edit the registry, but not to make snapshots of it
* The RegistryChangesView program allows snapshots of the registry, but I would need to ensure that python could parse the output

I researched the question “How can you tell what files have been changed during a install by the installer, and which by the operating system?”. I didn’t find any useful information, but I did find information that is tangentially useful, that I should check the .NET GAC to check which assemblies a program has registered

I asked which UI I should use for the project, and provisionally decided on PyQT, as I have previous experience in its use, which should speed up UI creation

I asked myself “How would you ascertain information from the environment variables?” and found that the python os library handles this under os.environ

Another part of this project is to automate un/installation of programs. Some research showed me that the urllib library allows you to download the exes from the internet, and the subprocess library allows you to create a command shell to un/install the exe

# Week 2

## Supervisor Meeting

At the supervisor meeting, we discussed the aims of the project, and how to go about them. The two main aims are:

* Cleaning and moving programs and projects on a pc
* Setting up a custom installer

We discussed how it is difficult for IT Managers to maintain large numbers of computers in a tidy and usable state, especially if users are allowed to install personal software. These managers need the ability to clean up a machine that someone has been using, ideally remotely. A suggestion was given that the project should be able to audit a system and store the results as a JSON file, as JSON is well supported and documented. It was suggested that some form of pattern matching or rules could be used to ascertain what software belongs to what program/project.

I suggested in the diary that PyQT would be a good UI for the project, but was informed that a WebUI is the easier and more useful, as it allows remote control of a system. It was recommended that I look into python eels to make a functional WebUI for the program. A small program could be created that starts on boot and sits in the system tray, installing and updating/patching the program from the web server. WxServer should be researched to create this, as it uses native windows functionality, making creation easier.

For programming environment, VSCode was suggested, with python as the programming language. I should investigate whether the windirstat library has a python counterpart. Windirstat indexes the file system, enabling you to get a complete overview of the file system. It was also suggested that I should make a list of the most common personal and commercial programs installed on machines, in order to ensure that the project can handle these. The netdata and prometheus projects were suggested. These create a web server that gives realtime stats on the machine it’s running on.

For next week, I should:

* Continue looking into similar programs and libraries that could be helpful for this project
* Get some of those libraries running
* Get a list of the most common programs
* A stretch case for the project would be setting up a software stack on a machine
* Create a virtual machine so that changes can be easily made without risk to my computer

The first deliverable for the project is a how to guide, which should be on some of the programs and libraries that I attempt to install, but have poor documentation. I was cautioned to ensure that concrete progress with tangible results is made, and that obstacles do not prevent me from from doing the work, and that if I have problems that I can’t solve, I should message John on Discord

I downloaded the ISO file for windows 10 so that I can create the virtual machine.

## 21/01/2021

I began by researching WinDirStat and potential python alternatives. I found a tutorial for WinDirStat, [ ]. I found duviz, which is a python alternative, but it works on every os bar windows(typical). Enso portable was recommended on one of the sites, but it is not relevant to the project. I found a PyPI package called SquareMap, which allows you to view data in a hierarchical squaremap format, although I’m not sure how useful this would be, as the process of data removal is supposed to be automatic. At this point, the Virtual Machine finished setting up, so I downloaded WinDirStat to see how it worked. I also began to install the files required to have python work on the virtual machine, so that I could go through the Python Eels tutorial.

I found out that you can use PDQ inventory to remotely call WinDirStat, which may be useful in the future(<https://www.pdq.com/blog/keep-tabs-on-disk-space-with-windirstat-and-pdq-inventory/>)

I started going through the provided eel tutorial, and followed it. However, the program could not find the eel.js file, despite multiple tutorials claiming that it was not required to put the eel file anywhere in particular. Researching it, it was suggested that it would only work when the html file was served by a web server. I downloaded XAMPP, as I had prior experience in it, to see if that was the case. It turned out that the issue was that the eel library was attempting to open chrome, which does not exist on the computer, so when fixed, I did not need XAMPP. However, I decided to leave any software that I didn’t need on the system, so that the virtual machine can be used a test machine for tidying the system. By going through a tutorial, and doing research on how to start the server, I have learnt the basics of the eel package. There is a problem that the execution sometimes times out when running the program from Code, but as it will being executed without an IDE in the production environment, this should not be a problem. I then went through a WxPython basic tutorial, and got it working. Tomorrow, I plan to build a small Wx program that puts itself into the system tray on startup, and find lists of the most common software types.

## 22/01/2021

I found that there was no tutorial(other than a potential tutorial that you had to pay for) that went over how to create a Wx program that uses the system tray. There was an example program with no commenting or explanation, so I went over that and researched what the program was doing. The snippet was programmed with an outdated version of WxPython and was under Python 2, so I had to make some changes to make it work. This could potentially be useful for my how-to guide deliverable. I still had some more work to do to make it fully operational, but I took a break from it to research netdata and prometheus. Both look useful, however, if I need to monitor stats on a computer remotely, prometheus is more customisable as a Web interface.

I looked up the most commonly installed programs on commercial and personal systems, and found the following links:

* <https://smallbusiness.chron.com/list-commonly-used-business-software-59849.html>
* <https://grasshopper.com/blog/small-business-software/>
* <https://www.zdnet.com/article/mapping-the-worlds-most-popular-software-in-users/>
* <https://uk.pcmag.com/software/69541/the-best-free-software-of-2020>
* <http://cleansofts.org/software/popular.html>
* <https://electrons.co/popular-software-downloads/>

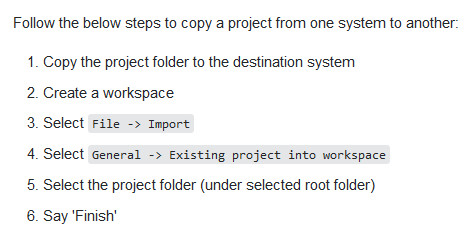
## 25/01/2021

I started trying to finish getting the WxPython systray application to work. I looked into examples to figure out what was going wrong, but before I could try some of the changes from the examples, the Virtual HardDrive ate all the empty space on my C Drive and crashed. It refused to restart on my SATA drive, and I was forced to set up a new virtual machine, which set me back, because it meant that my progress on the system tray application was lost. It took a decent amount of time to create another virtual machine, as windows took longer this time to set up in the guest machine. I did bypass the account creation that Windows 10 requires by disconnecting the internet when the setup service tried to search for the fake email address that I provided([a@b.com](mailto:a@b.com)). Once setup, I then used a variety of websites, including the wx documentation, to get a working tray application(which does nothing). Websites used were:

* <https://stackoverflow.com/questions/6192430/call-a-function-in-another-frame-wxpython>
* <https://wxpython.org/Phoenix/docs/html/wx.Window.html>
* <https://stackoverflow.com/questions/25235833/wxpython-hide-and-show-panel>
* <https://www.blog.pythonlibrary.org/2018/10/19/wxpython-how-to-open-a-second-window-frame/>
* <https://stackoverflow.com/questions/6389580/quick-and-easy-trayicon-with-python?rq=1>
* Others that I forgot to note down from the previous day of working on it

## 26/01/2021

I installed everything from voidtools onto the virtual machine and ran a couple of searches using it. Because the GUI is unlikely to be used in the project, I installed the command line version. I attempted to run searches, but received an error message. I found out that the exe file had to be executed, as no command was registered to the program. Once I found this out, I was able to run searches from the command line, and colour code the results(although I doubt that colour coding will be useful). I then looked into whether similar projects to the one I will be making exist, by seeing if there were ways to automate moving programming projects. I found that in Java, the most common method is just to copy and paste the project folder, and then use eclipse to import it as an existing project, although I’m not sure whether this would be applicable to this project.



However, without an IDE, the code can’t run without being exported to a -jar file, so I may be able to assume that there is an IDE on the destination end of the transfer, and see if I can import the project to it manually

It was suggested during research that code should be maintained using Git, but I am not sure if I can assume that all projects will have been set up with git. If files have been set up with git, I could simply push and pull projects if transfer between computers is required.

For Unity files, you can simply copy the project folder to the destination:<https://answers.unity.com/questions/1448792/how-do-i-move-unity-projects-between-different-pcs.html>

For python, using pipenv to create a project environment, and then copying the environment along with the project should allow the project to be moved:<https://www.reddit.com/r/learnpython/comments/em7ydc/how_to_transfer_a_python_project_to_another_pc/>

In C#, you can copy the project folder to the destination:<https://social.msdn.microsoft.com/Forums/vstudio/en-US/859dbc7a-8fd0-418f-8869-2fa6b8fdab93/moving-c-project-to-another-personal-computer?forum=csharpgeneral>

I was unable to find information on how to change absolute links in a program to relative ones, so I may have to try and get the project to open a program, find absolute references, and change them to relative, but this may not be possible.

# Week 3

## Supervisor Meeting - 27/01/2021

At today’s supervisor meeting, we discussed the previous week’s work, and how to move forward.

It was discussed that the suggested program structure of a small system tray application that maintains a Python Eels program would be a good way forward, as it is easy to maintain, and is the model used by projects such as Discord. This format could also be the basis of my how-to guide.

It was suggested that I should get someone to follow my how-to guide to get an idea of how helpful it is to other people. The how-to guide should be focused, and then iterated on, rather than large untested.

I was told that Eclipse creates a bunch of hidden files that it uses to define the project, and that I should monitor which files it creates. An advanced goal for this project could be to move a java project without having to run the Eclipse import wizard.

The main goal for this week should be to monitor the file system, registry and environment variables before and after an installation or project creation, and use a text editor to compare the before and after images to see what has changed.

It was also said that by opening a project at different places in the file system, I could see which files contain hard-coded paths, and then be able to change them, but that the primary objective should be to monitor and log installs first, before thinking of special cases. Blendr was discussed, as a file type that may not be able to be moved, as a Blendr file is binary. Unity files may be difficult to move, but may be possible.

The next stage could be to see if the project could detect when an install is occuring or that a program is generating files somewhere, and perform the imaging operation automatically. I could then potentially pass that information to a server to improve knowledge on how to handle that program.

## 28/01/2021

I used the quickstart guide for Watchdog to create a basic program that prints file system changes to the console. I changed the directory from the current directory to the “C://” directory, to capture all changes on a system, and let it run passively, without me installing or changin anything. There were approximately 100 file changes during the 45ish second period that I was monitoring. This raises the difficulty of how to discern when a file is being installed by an application. My plan is to let the program run for a few minutes, and then figure out which files the system changes passively, so that I can instruct the program to ignore these file locations. As far as I can tell, Windows should be the only process installing to System32, but I’m not sure, so I’ve asked John.

John has told me that installers can create files anywhere on the system, including System32, so I will have to figure out how to exclude files that windows is creating passively. I currently cannot find anything on the internet or the windows docs that explains what folders windows updates automatically, and considering that every windows feature has 10 years of crap that its balancing on, I doubt that any one person knows every location that windows uses. I will have to consider how to stop the program from monitoring what windows is doing. I could potentially just run the change detector for a long period of time without manually changing the file system, and then instruct the system to remove every file/directory that windows creates from future scans, however, this would not stop the program from picking up any new directories created by windows.

I had the idea that if the computer apparently did not distinguish between the os creating files and an installer creating files, that instead, I could see whether MSI files have to register a system ID to start installing files, and if so, could I filter the file system snapshots to files created by that ID.

From StackOverflow(<https://stackoverflow.com/questions/47623037/how-would-i-get-the-pid-of-the-process-causing-a-file-system-event>), I found that Watchdog cannot access PIDs, and as such cannot detect which file changes are passively done by the OS, and which are the result of a User Program. However, the topic suggested the use of loggedfs-python, a module which creates an overlay file system to allow you to view every operation occurring, which would be perfect for this project, but it only works on Linux systems. I then read through nearly every function in the psutil library. One that seemed encouraging was open\_files(), which returns any files opened by the process, so if I could get the computer to detect when an install is occurring and retrieve the PID of the installer, I could read every file that it was opening, and then use watchdog to check if that file had been edited.

I have created a small python program that iterates through every active ProcessID and attempts to retrieve each file it attempts to access. I plan to test this by installing a program, finding the PID of the installer, and seeing if I can see where the installer is installing files.

I attempted to monitor the installation of OBS Studio using this system, by opening the installation exe, finding its PID from the task manager and then instructing the program to retrieve files opened by the installer. The program returned that I didn’t have access to the specified PID. After I elevated the command prompt to administrator execution, the only processes that I didn’t have access to were the windows processes. I then reran the experiment. I received a few files that the process had accessed, but nowhere near the number that the installer was accessing. As far as I could tell, the installer is passing the actual install to child processes. I attempted to get the program to also search the child processes, but this did not reveal anything, so I’m calling it a day here.

## 29/01/2021

I recompleted the Python Eels tutorial that I lost when the VM crashed. I don’t have much motivation today, so I’m not sure how much more work I’ll get done.

I did some research into how to start the desktop application on startup, but this seems to have different methods for a .py file, and an .exe file. As the application will most likely be an .exe, I will wait until closer to the time where I have a functioning prototype before setting this up. The wxPython program will be calling the Eel program, but the eel program might also be an .exe, so implementing this will depend on decisions made down the line.

## 01/02/2021

To start today’s work, I spun up the virtual machine and opened my WxPython code, which will form the basis of my how-to guide. I took it section by section, and sometimes line by line, and wrote my how-to guide. I decided to make my tutorial on creating a system tray application in WxPython because this is very poorly documented. The only functional example I was able to find was a code block in an answer on StackOverflow, and this was not commented, so I had to figure out what it was doing line by line, using documentation that was pretty bad. For instance, the events that WxPython handles are defined in a table. Every event is defined the same way: “Wx.EventType-Used to refer to events of the Wx.EventType class”, which is about as helpful as saying “food” when someone asks you what dinner will be this evening. I figured out what each event the program used did through seeing what methods were called when I used different functionality of the Icon class. It is also a good topic for my how-to guide because I found two websites(one of which was a question on StackOverflow) that also had code blocks for doing the same thing, but both of them were using code that had been removed or relocated from the Wx module, and as such would waste people’s time when they were trying to build a system tray application, as it would take them quite a while to run through every error caused by functions being deprecated. If I make a tutorial that uses up-to-date functions, this will save the reader time and effort, making it a useful guide. My other main options for the guide would be python eels or Watchdog, but I only had one problem with eels, so it would not make a good how-to guide, and Watchdog comes with its own tutorial that is actually well documented. I used Greenshot to get screenshots of each section of the code and placed these in the correct positions in my guide as I went along, as having each snippet of code that is being discussed next to the paragraph discussing it makes it easier for the reader to understand what the tutorial is teached. At this stage, I was writing out the guide in Google Docs, to allow me to write it without html syntax getting in the way. After explaining step by step how to build the program(including the traditional phrases from coding tutorials, such as “Let’s walk through what this section does”). After this, I took each image from the guide, numbered them based on their position in the guide and placed them in the /resources file for the webpage. I then used VS Code to adapt the guide to web format, in a very basic way. Depending on time, I may import a framework such as Materialize.css, to make the guide look better, but I’m not sure how long that would take.

The blueprint of the guide is here: <https://docs.google.com/document/d/1oEmhpcgAlyNd-ZItBrRFeYX56Ba-HMxnXFCkAwxB7J0/edit?usp=sharing>

## Week 4

## 02/02/2021 - Supervisor Meeting

This week, we primarily discussed the issues around recording which files were installed by a program, and the how-to guide.

It was suggested that I should record where specific programs store files, and use that to make templates in a controlled environment. The client program could then check to see if the file system it is running on matches various templates in order to detect which programs are installed.

It was also suggested that I should run the watchdog python library while the computer is idle, and record what changes are made passively by the operating system. I could then filter these out of the snapshots taken during an installation, to ensure that the program only captures files that were created by the installer.

We then moved on to discussing the how-to guide. We were told that a good place to find examples of how to make properly useful how-to guides is to look at the tutorials on the pyimagesearch.com website. We were told that we should be primarily focused on our diary when thinking about the how-to guide, as our thought processes and considerations are more important than the deliverable itself. It was suggested that we should get someone to read our guide and try to follow it, to see if it is generally helpful to other people.

The end of the module was discussed. We were told that we will be making a reddit post detailing the progress we made. We were also told that across this module, we will be primarily marked on thought and effort, rather than what we produce

## 03/02/2021

I will be focusing today on improving the bare bones how-to guide that I made last week, so that it is more fit for use by other people.

I started by going to the website suggested at the supervisor meeting, and looking through a tutorial on the site, to see what made it an engaging tutorial. I noted the following things that made the tutorial more interactive and useful.

* The tutorial starts with an explanation of what the tutorial entails, and the uses you can put it to
  + I think this allows the reader to quickly see if the tutorial meets their needs
* It outlines the steps that the tutorial will go through before starting to go into detail on each step
  + I believe the reason this is effective is because it allows the person viewing the tutorial to quickly grasp whether there is anything in the tutorial that they need, as some readers know how to do what the tutorial teaches, but are looking to find help with a certain part that isn’t working, or they’re not sure about
* The tutorial outlines the tools it is using to solve the problem at the start of the tutorial, which gives the user a chance to research the tools or libraries at the start, if they do not have any knowledge about it beforehand
* Every section title in the tutorial is descriptive, rather than just “Step 7”. This helps alleviate any feelings of tedium, as far as I can tell, as it makes each step feel like its own goal, rather than an item on a checklist.
* The tutorial links every resource it references, as it references it. This makes it easier for the reader to find more information on these resources
* Aesthetically, bullet points are used, and I think this is to break up the monotony of long blocks of text.
* At the end of the tutorial, the steps are summarised. This should help the user to fully understand the importance of each step they went through.

I have now started to update my how-to guide to be more helpful and user friendly. I rewrote the introduction to be more explanatory as to what the tutorial will teach, and put an indirect statement at the start of the introduction to make the tutorial feel less of a checklist and more of an experience. I then installed materialize.css to make the guide look less like it was designed in a basic HTML course. I changed the colours of the titles and subtitles, which should make the guide more engaging to the reader, and center-aligned the subtitles, to make clear where each step began. Within steps, I added dividing lines, to more clearly indicate which text was referring to which images, which should help the reader to understand the tutorial more clearly. I then went through and made every subtitle more descriptive as to what the step would do, which should hopefully help when the reader is looking to reread part of the tutorial, as programmers often do when they can’t remember how to do a specific part of a program. By convention, the constructor for the main class calls methods that are defined later on in the program, which means that I have to inform the reader that we will implement methods later on in the tutorial. To prevent this from confusing the reader, I have included links between the step where I do this and the step where the method is implemented, allowing the reader to jump ahead if they are confused as to the purpose of a method. Halfway through doing this, I realised that I had left out one image from the webpage, and had to change a third of the image links on the page to fix the problem.

## 04/02/2021

I continued to work on my how-to guide today. I added a section to the guide that explains what WxPython is, which will help the reader to better understand what the tutorial is teaching. This will hopefully make the guide more useful to readers who have no prior knowledge of WxPython. In this section, I included a link to the wxPython website, to allow the reader to easily get more information if they require it. II added a more descriptive introduction to the start of the tutorial, that outlines the steps in the tutorial, to allow the reader to see if the tutorial would be useful for them at a glance. I added a section to the conclusion paragraph that summarises what steps the tutorial went through, to ensure that the reader understands what they have programmed. I then went through the entire tutorial and added and removed sections to make it more user friendly. I added a link from the “Create a Taskbar Icon” step as it contains text saying that a method will be defined later, and so I linked it to where it is defined, to allow the reader to understand how the code works together. I realised that “This attribute will be used to keep track of the number of open windows” wasn’t helpful to the reader, as it didn’t explain why the program needed to keep track of the number of open windows, so I added an explanation as to what the window tracking was used for to the paragraph. I added an explanation as to what the popup menu was, as in my first draft of the guide, I assumed that the reader would understand this, but this may not be the case. I realised that my guide did not contain any images of the final project, so I thought about what images would be the most useful to explain at the start of the project what the project was. I decided that a short gif of the program working would be most helpful to the reader, as it shows them exactly what the end result of the tutorial will be. I recorded a gif using gifcap, and added it above the introductory paragraph in my program. Although there are not many blind programmers, there are a few, and so I have added alt text to the first image that explains that the full program is available in plaintext at the bottom of the page, and alt text to the rest of the images to inform the reader that the images are of code.

## 05/02/2021

Today, I tried to get Handle from sysinternals to work. It didn’t take much effort to get it to run, so I had it run while OBS Studio was installing on my VM. It revealed any files that the installer was handling. To check whether it picked up every file being used or created, I then reran the installer while using the watchdog snapshot program, which I know picks up every file change. I accidentally ran the first version of the watchdog program, which only looks at one directory, so I repeated it with the correct version. Unfortunately, although the Handle program showed which files the installer had open, it didn’t show which files are being created, so I am going to have to build up a bank of files that windows passively creates and edits, and then filter them from the watchdog results. Unfortunately, this list will only grow over time, which means that it will take a lot of space. It may be a good idea to have the list be on a server, rather than the client machine.

Yesterday evening, I asked a friend of mine who is familiar with python to read through the tutorial, but he got back to me today, and it turns out the materialize.css framework only works locally on certain versions of Windows, which my friend was not using. There is not enough time to strip Materialize from the webpage and get my friend to look at it, so I am going to have to do without a second opinion, which is unfortunate, but I have hopefully pulled enough information from analysing popular tutorials on PyImageSearch. As the aim of this part of the project is to make a webpage that is useful to other programmers, I am going to include both the materialize version of the webpage(which includes changes that I made to make the webpage more visually appealing), and the plain HTML version, as if we were publishing the webpage, the materialize framework would work, as it is only necessary server side. To accomplish this, I have duplicated the webpage source code, and stripped out the materialize code from it, so that if the materialize version doesn’t work on the examiner’s computer, they will still be able to see the basic version of the program. Hopefully the materialize version does work, because it is the more user friendly version.

I also ran the watchdog program over a period of time to get a list of passively changing files, but the VM fell asleep after a few minutes, so I ran it again and saved the results in a .txt file.

## 08/02/2021

I started today by spinning up the VM and looking at the list of passively changing files. I started making a python function that would load in each line of the list, strip it of characters to leave the path, and then return the list of paths to the calling program. I spent a bit of time thinking about how to deal with the difference between the edit, create and delete events that had been recorded, before realising that since this was files to be ignored, it didn’t matter if they had been updated or created, although as I write this I have realised that if I delete any paths of deleted files(as they can’t be found unless they are recreated), it would speed the process up. I’m sensing a trend, because as I wrote that last sentence, I realised that if Windows deletes a file, it could be a file that it then recreates later, so I shouldn’t delete them from the logs in case it creates false positives. Anyway, after that, since it didn’t matter which file event was occurring, I got the program to find the first “/” character in each line, which is the first “/” in “C://”, and then take two away from it, so that I could build a substring from the “C”, allowing me to isolate the path from each line of Watchdog’s output. I then cross-referenced it with each other substring to ensure that there are no duplicates. I’m now trying to decide how best to implement this filter on the watchdog output, so that it only shows files being actively installed by a program. The way I currently have it, Watchdog is outputting to the command line. I could keep it this way, and pipe the output to a txt file, which would be the template of the installed program. The benefit of this would be that the command line program can be easily piped through the the text file, so it would be more resilient than the other option. However, it would record deletion events, which I probably don’t want to record, as I don’t think that there’s a way for me to recover files that the program has deleted, however, I may test whether the installer sends files to the recycle bin, because in that case, I could restore them when the program is uninstalled. The installer should only be deleting its own temp files during installation though, so if it is deleting files that need to be recovered when it is deleted, the program is probably a virus anyway. I could change how watchdog is configured so that I manually handle each file event, which would do away with the need for the command line, and would make the creation of templates easier, but this would potentially less robust, as the logging method has been tested by Watchdog’s developers. I think I will try this method, and switch over to the other if needs be.

I set up the new version of my file system scanner. Watchdog has an option that allows you to list paths to ignore, so I passed it the list that my other module created. However, it didn’t seem to recognise every event that it was supposed to, such as when I moved a file from one place to another, and it didn’t show a move event. I made the program more verbose by getting it to print the source file of each change, and it showed that instead of using move operations when you move an object, windows performs a create event and a delete event instead. After I figured this out, the program worked the way it was intended, although it still has a very high false positive rate, because there is still a lot of undocumented background activity going on. I spoke to John about how the next section was going to work, so I now have a better understand of what to do moving forward. I have to get on with coursework for another module now, so I’m going to leave the passive scanner running in the background.

I noticed that when I pressed a key after a long period of not pressing a key, it caused a flurry of file changes. Not sure what to do with that information, but I’m writing it down in case I need it later.

## 09/02/2021

Today I am going to try and get a registry monitor running. I have found a python library called ‘regipy’, that can take registry snapshots and compare them, outputting the results to a csv file. However, this functionality is part of its command line program, and it would be handier to keep the scanner solely in python. The python library allows you to output the date modified for each key in the registry, which I can then compare to the install start time, to see what the installation has changed.

I quickly ran into the issue that this program only works on offline registries, and so I couldn’t use it on the active registry. I looked into a bunch of other tools and libraries to see if they could either take a registry snapshot or detect changes in the active registry. Unfortunately, everything to do with the file system and registry in Windows is terribly documented, with nearly no information about registry access anywhere. I am going to try using the regedit command line program to make a backup of the registry, and see if that can be used as an offline registry.

I started experimenting with the command line program. I found that it won’t let you export an entire hive, but it will let you export an entire subkey, so I can pull each subkey out of the hives and export them. It took quite a bit of faffing around to figure out how the command line worked, because, surprisingly, it wasn’t well documented. Unfortunately, the regipy program crashed because it needs an entire hive to work correctly. So I need to figure out a way to copy the entire registry at once.

After a lot of trial and error, I finally figured out how to provide regipy with an input that it would accept. I initially tried reg.exe’s EXPORT command, which I managed to get to work, but it output’s .reg files, and regipy requires .dat files to work, so I continued researching how to get .dat files. I tried directly searching how to convert from .reg files to .dat files, but .reg files are designed to be human readable and editable, and apparently do not convert to .dat. The main use of registry snapshotting is for cyber forensics, as even when malicious programs delete themselves, they can leave traces in the registry, and even if they wipe the registry, they can still be found in the registry’s changelog files. So I tried looking up how cyber forensic analysts take snapshots of the registry. Unfortunately, the tutorials discussed the topic in a mostly theoretical manner. After a few false starts, I found that the reg.exe’s SAVE command can output .dat files, so I ran it in save mode and got a usable result. So far, all I have done is print the values of every registry key, so I’ll have to work on only monitoring changes to the registry tomorrow. I think I know how to do it, I’ll need to have the program save the time an installation begins, and then compare that with each registry key’s modified date in order to find which keys have changed during installation. This will mean that I will only need to take one copy of the registry, and the timing doesn’t overly matter, which is good, because the hive saving process takes time to complete. I will probably have to thread a number of save processes at once to save time, but that shouldn’t be much of a problem.

# Week 5

## 10/02/2021 - Supervisor Meeting

We discussed the next deliverables for the project today. I was told that my project should be left in a useable state at the end of the course, and that to achieve this, I should stop adding features to it a bit before the last submission, so that I can focus on making sure that my progress has been useful. My deliverables should be genuinely useful to other programmers, and I should try to get other programmers to test my guides to ensure that it is useful. Any feedback that I get from people should be put in my diary. We discussed that my how-to guide could be expanded to explain how to make a discord-style application that launches when run from the system tray.

We discussed my progress in the past week, and the issue of files being deleted during installation. It was suggested that I should treat this as a separate issue. We would want to understand why a program is deleting files, as it could be a virus, or a program like the one this project is creating.

It was suggested that I should go through my diary and figure out how to incorporate sections of it into my next deliverable, as it will cover all knowledge that I have obtained in the course of the project.

We discussed the main goals of the project and how to go about them. John said that the aim would be to take about 20 programs, and install them in different locations, logging what happens when a program is installed in different places. It was suggested that I could then take whichever files change depending on location, and run diff operations on them to see if I could detect file paths that are changing. This could then be an interesting article on analysing popular programs to see what and where they install, and whether they correctly uninstall all of their files.

John discussed where our marks come from. The main place marks come from is the diary of the project, showing that I did steady work throughout the project, was able to research problems to overcome them, and that I asked for help whenever it was necessary. The second section, worth fewer marks, is the remaining assignments, which show that the work I have done has been left in a state where another programmer would like to use it, which is difficult. The third section is whether I can present some part of my work in a way that could advance my career, which is fulfilled by the how-to guide. The main note made was that all thought that goes into the deliverables should be written down in my diary.

We discussed the previous assignment. John said that the main area that I need to improve is communicating what I’ve made, rather than what I’ve made. He said that there is a connection between people who hustle and get the help that they need, and success. I should ask when I need help and perform steady regular work. I was advised to get other students on the module to look over my deliverables and for me to look over theirs so that we can ensure that they are as good as they can be.

## 11/02/2021

Today, I am trying to detect changes in the registry during an installation. My first problem is that the “date modified” field of each key returned by regipy is in ISO format, which python cannot parse. So I am researching how to either get regipy to output datetime objects, or how to convert the ISO format to a datetime object. I am going to see what regipy outputs without being formatted, so see if I could split the string and make it a datetime object.

Attempting to retrieve information using the iter\_keys() method of regipy lead to an “IndexError: pop from empty list” error. I wanted to check if this was a problem with the way I had written the program, so I wanted to try the example straight from the documentation to ensure that it worked. However, the command prompt won’t allow me access to the HKLM hive, even though it will let me access other hives. It doesn’t seem to be a “file in use” problem, because I call pull individual keys out of it with the same method. I found an answer that said that the save command attempts to block the saving of master hives(<https://stackoverflow.com/questions/50572630/why-regsavekey-fails-with-an-error-code-5-access-is-denied-if-the-key-to-be>) but I can still save the other master hives, so I don’t understand why HKLM would be different.

While I can’t save the entire hive for HKLM(or HKU), I can access all four subkeys of the HKLM hive, so not having access to the hive root shouldn’t be a problem.(It seems like HKU only loads keys for the current user, so I shouldn’t need to keep track of this hive). I should now be able to get back to the issue that I started with.

I managed to retrieve an unfiltered date”132560555313370265”. I don’t know how to convert this yet, but I’ll worry about that after I’ve figured out how to get a date from every hive. I am saving every hive now, and then I’m going to figure out how to get the dates.

I found that if you directly printed a subkey returned by regipy, it contained a timestamp in full timezone format. I used python’s re module to go to the position of the timestamp, and then pulled the timestamp. I used strftime to tell the datetime object how to read the string, and create a datetime object. This means that I can now take a timestamp at the start of the install and at the end of the install, and compare each key in the registry to see if it was created or updated by the program.

To make my program more useful to other programmers and to improve readability, I am creating a function for pulling and comparing the timestamps for each registry hive.(Currently, no comparison is happening, because I am focusing on making sure that the whole registry can be scanned)

I have now tested it, and every hive(or subhive for HKLM) is now scanned and returning timestamps. I now need to program a function to open a command prompt and copy each hive after an install, so that the time stamps can be checked, which I will do tomorrow.

The last thing that I did today was to add a section to the program that checks if the program has administrator privileges, and if it doesn’t, reruns the program with admin privilege. This will be necessary because accessing the registry is an administrator action.

## 15/02/2021

I started today by trying to create the function that saves all of the registry hives by opening the command prompt and running the REG SAVE command. I initially ran into a problem that the os library was throwing a unicode error, which I solved by doubling the backslashes in the affected string, as backslash is an escape character in python. Once I had done that, I ran a test save to ensure that the command was being successfully passed, which succeeded, so I had the system pass commands to the system to save every hive. In the future, if there is time, or if the current version is too slow to be useful, I will thread the commands to speed up the process, but at this point, working slowly is fine as long as it’s working.

My plan for the rest of today is to code a function that triggers the file system program to start monitoring, and then a second function that that triggers the registry system copier when the user indicates that an install has finished. (I will have to discuss how the program could recognise when an install is finished with John, but it is not necessary at this stage as we will be manually making the templates). I then intend to connect the two functions to the Eels application, if time permits, if not, I will do it over the next couple of days.

Unfortunately, while I had gotten the registry scanner to pull the times from each hive, I had not actually gotten it to compare each time with the start time of the install, and return matching keys. I fixed this by having the file scanner(which runs when the install starts) create a datetime object of the current time, which is then passed to the registry scanner, allowing it to check each key. I then found that the file scanner was no longer working, so I took a look through the functions it uses, and couldn’t find an error. After a while of searching, I found that the second time I had run the passive scanner, I had changed it to remove the dates from each result that it returned, and this had added blank lines below each result for reasons that are as of yet unclear. I did some research on how to remove blank lines, as each blank line was breaking the program, but no program on my windows VM could do it. Through research, I found that there was a website that could clear out lines that are empty, but as I was writing this, I realised that I could get the program itself to skip lines that had whitespace, so I am going to do that instead. These setbacks have put me slightly behind schedule, so I will have to leave getting the scanner functions to work together until tomorrow or Wednesday.

I have got the program to ignore lines that are not results from watchdog. I forgot to mention that the reason that I was running the watchdog code was to see whether the program could properly handle interrupts and how it was currently outputting results. The answer was that interrupts had to be entered twice, which I believe I know the reason for(from examining the except statement, the program is restarting the scanner on an interrupt instead of ending its process) and the results are output to the command line, so I will need to figure out how to pipe these to a file where they will be useful.

## 17/02/2021

Today, I am going to try and get the registry and file scanners to save their results to arrays and to work in tandem. The watchdog program is routing its results to the command line as well as to my arrays, but that should be fine because the command line will not be open for very long. In order to stop the file scanner and start the registry scanner, I need an external input, so I am going to try and create a function with the Eels program that will serve this function.

Trying to get all of the functions to work together is quite confusing, as I am having to decide which variables need to be global, and which variables need to be returned from functions, as well as trying to incorporate Eels functions into the program. I have settled upon two major functions that will be triggered from the Eels page, one which will trigger the file scan, causing watchdog to start passing file changes to the program, and a second that will terminate the file scan and start the registry scan. In later versions, the second function will save all of the acquired data to a file, but for testing purposes, I will have the data output to the console, so that I can check that the correct files and keys are located. Once I have got it working, I plan to test it using OBS Studio, as its installer prints to the installer each change that it makes, allowing me to compare my program’s results with the actual results.

I have linking the file scanning function to the Eel server, and after I fixed an error about the fact that Eel functions have to be called with two sets of parentheses, it appears to be working, as file system events are being logged.

I added the registry function, which appears to be copying the registry when pressed, as it should, but it does not terminate the watchdog process, which is also only intermittently responsive to Keyboard Interrupts. I need to figure how to terminate its process from the program, so that the registry scanner can stop it.

I have researched the problem, and the watchdog listener should be stopped by the observer.stop() method, which I am calling in the second function. I have tried switching where the variable is declared as global, removing the global definition entirely, and editing the observer’s exit conditions, but it hasn’t worked so far. I am going to look through the documentation to see if there’s a specific way that you’re supposed to reference the observer that I am unaware of.

I didn’t find exactly what I was looking for in the documentation, but I did realise that the problem might be that because the program is single thread, the second function can’t activate until the first finishes. To test this, I changed the second function so that it just passed text for the Eel webpage to echo. It didn’t echo it, meaning that the second function isn’t being executed. I will attempt to put the file system scanner in a separate thread.

I put the function in a separate function, which initially didn’t work, as the program still couldn’t use the other function that stops the scanner. I realised that I was putting the parentheses in the thread call, which was calling the function before it could be threaded. Once I removed them, the file process was killed by the registry process, although the registry scanner then threw an error about not finding functions. I fixed that problem, but a problem about passing datetimes came up. I moved the time capture function out of the threaded function in case there was a problem with passing global variables out of a thread. This worked, but unfortunately, this meant that I couldn’t pass the observer out of the thread either, so I moved the watchdog creation code out of the thread, and just had the thread start the observer.

After fixing problems stopping the datetime from being passed to the registry scanner, I came across the problem that the scanner couldn’t compare “offset-naive and offset-aware” datetimes, because the registry timestamps include the UTC offset, whereas the datetime.now() method didn’t. I added timezone.utc() (found at https://stackoverflow.com/questions/796008/cant-subtract-offset-naive-and-offset-aware-datetimes) to the datetime.now call.

The functions now both execute, but the actual checks of the registry against install time seem to be taking a long time to run, which I may have to consider later on.

## 17/02/2021 - Supervisor Meeting

We discussed how the project could look moving forward this week.

It was suggested that the program could reference a txt file, or a shell script could trigger the program, in a way that automates the process of scanning the file system during an installation. As part of this, the Eels website could have two pages, one that triggers the scanner to start, and one that triggers the end process. Curl could be used to access these to automate the process.

It was also suggested that I should add a second drive to my Virtual Machine, to see what happens when an install occurs on another drive.

John suggested that I should ask on a Windows forum whether anyone knows how to detect an installation occurring, to ensure that a simple solution isn’t being missed.

It was suggested that a program could monitor the active processes on the computer, and look for any new exes, as these would indicate an installer being run. It could then automatically take the template.

Looking forward, we discussed that once I have templates for various programs, I could attempt to backup and restore programs to see if they work after restoration, which would indicate that the scanner had picked up all of their files. Then I could attempt to move programs from one position to another using the scanner, and see if it was possible to change hard-coded paths to make this possible

John mentioned that the next deliverable is mainly a first attempt at something that you would give to your past self to prevent them from having to do the research that the project required, as well as a potential update to the how-to guide.

## 18/02/2021

My plan for today is to finish testing whether the program is successfully finding changes and storing them in the program, and then to save the changes to a file that will be a template file. I plan to use json for this purpose, as it seems to be a good way of structuring data files, and it is commonly used, so it will make the project more usable. The first step today is running the program with print statements at various points to ensure that the whole program is being executed.

From printing status, I found that the system was hanging on the HKCR hive, preventing the rest of the hives from being scanned, as was the software hive and system hive. Unfortunately, I don’t believe that there is a way to speed up the scanning, as threading will only reduce the time the faster scans will take, as the length of the longest scan(at least 5 minutes) is the limiting factor. For the current testing, I will simply omit those hives, so that I can test that the underlying code is functioning correctly. At a later stage, I can then see about threading subhives, but for now, the system will just be slow.

I have now had the program run skipping the large keys for testing purposes. It is functioning, I got it to print out the files that had been changed, and the registry keys that had been updated. However, there was a large number of registry keys changed even though I was running the program without installing anything, meaning that I have come across the same problem as previously, where windows is passively changing large amounts of data in the background. I am going to see if there are any easy ways to filter changes made by the OS.

I have done some research, and read articles on the topic, and the registry does not seem to track who makes changes to it, so I might have to manually filter out passive results from the registry scanner. The main problem with this is that they is no way to be sure that a filter program has found every registry key and every file that windows changes, so the scanner program could always be picking up false positives. I am going to ask on a Windows forum whether there is a more reliable method of filtering Windows inputs, and also whether it is possible to detect installations occurring.

I will check the question again later, but currently the only helpful answer about detecting installations talks about using the procmon to find the msiexec process. Unfortunately, this only works for programs being installed using the proper installer, although it might be something to try. I can’t ask the other question yet because apparently you can only ask 1 question every 90 minutes, and they froze my ability to ask the second question before telling me that, so I have to wait until tomorrow to ask about filtering windows inputs.

I realised that I might be able to speed up the registry scanning if regipy had an option to order keys by date modified, but from searching the documentation, that doesn’t seem to be an option.

I have tried researching if there is any way to tell which registry events are done by the system, but so far, I haven’t been able to find anything. Tomorrow, I plan to start either making the filter that removes passive registry updates, or to output the scanner results to a file, as I would have done today if a number of issues hadn’t come up.

## 19/02/2021

The plan for today is to output the results from the scanner to a file. I plan to use a json file for ease of access, and the results of the scanner are already being stored to data structures, so it should work well. After that, if there’s time, I plan to build the registry filter, to filter out the passive registry changes.

I had noticed that the command line would often hang until I pressed a key, stopping the program from being automatic. I have now found the source of the problem, which is that the cmd window’s “quick edit” function sometimes pauses execution when it loses focus. I have turned it off in properties, and the program now appears to run without me having to press enter every so often.

I have made the json file, and while it correctly stores the file changes, it is storing empty lists for the registry changes, despite changes having been made. There must be an issue with how I’m passing the lists between functions, as the original list is successfully output to the command line before the file is made.

I have followed the flow of the data through, and it is all accounted for until it enters the dictionary object, and then it disappears. I’m going to research whether this is a problem with how I am assigning the data, although as far as I can tell, it is input the same way as the ones that work. I tried removing the one object that was a dictionary, but that didn’t solve anything. I am going to install PyCharm on my machine, so that I can step through the program to see where the error is.

Annoyingly, the PyCharm debugger is really slow, so it is taking a while to reach the stepthrough point.

Dammit!!! All that effort and it turns out I had misassigned a variable. It now runs, but the JSON library can’t serialise registry subkeys, so I need to figure out how to store them. I thought that registry key names are unique, so if I pulled the name out of the key, I should be able to find it later, but upon reading the documentation, value names are only unique in relation to the key directly above them. This means that I need to collect enough data about each key and value to identify it, as opposed to a single unique identifier.

From reading the sparse documentation of regipy, I think that if I can pull the path from each subkey, I can use that to identify each subkey. There is no way using the current method that I can figure out which values within the key have been changed, so I’m going to have to make two copies of the registry, so that I can pull the subkeys that have changed from the previous registry and check which values have changed. I’m really starting to dislike the registry. I then need to save each subkey,the values that were changed and the old values in the json file, so that they can be reverted if they need to be. So I am going to have to change a lot of my registry scanner code, which is annoying.

I have an exam on Monday and a coursework submission this weekend, so I might not be able to do as much project work this week.

## 22/02/2021

Today, I need to try and rewrite my registry scanner so that it can detect changes in individual key values. The first thing I need to do is write a function to create a copy of the registry before installation, so that I can compare the hives.

I have written the copy function, and started into the hive comparison functionality. I have used substringing and string searches to pull the key path from the key. I should now be able to use the getKey() method by passing the path for both hives, and then iterate through the JSON objects to see which key values have been updated. I can then create a dictionary of the values that have changed, and then save them to a dictionary of changes per key. In its initial configuration, this scanner will only be able to detect when keys are modified rather than created or deleted, but I should be able to include that functionality once the base version is created.

I have now created the key comparison code, which should be able to store any key values changed within a hive. Currently, there is an error in the program, but setting up the new code took almost an hour, so I’ll have to do debugging tomorrow, as Monday is my busy day for the other modules.

## 23/02/2021

Today I am going to try and figure out why the registry scanner isn’t correctly passing values to the main program, and then hopefully finish outputting the scanner output to a JSON file.

Initially, after debugging the program, I realised that my test hive wasn’t being updated, which explained why no results were being obtained. After I expanded to include a hive that I knew was frequently updated, I received a different error, telling me that the scanner could not find the subkey being passed to it. I am going to look into it to see if there are any issues with the program, but it could be that the subkey was created rather than modified, which would explain why it didn’t show up in the initial registry.

I checked the registry, and the key being referenced existed, so I went back over the error message again, and saw that the path was being double backslashed, I’m assuming Python automatically inserted escape characters into the string. I’m going to try using the string replacement method to reformat the string. I might need to declare it as a raw string, but I’ll try without it first to see if it's needed.

For some reason, the code worked when one of the strings was raw and one was regular for python. This allowed me to progress to the next error(yay) which was that the program couldn’t access the objects within the subkey. I think I can fix this by adding a JSON argument to the assignment operation.

The JSON argument did not fix the error, so I searched for value methods in the github repository for regipy, but that did not seem to work. Luckily, for some unknown reason, PyCharm decided that code completion was something that I would only need after PyCharm had already running for well over an hour, and the autocomplete revealed how to retrieve the value from the Regipy object. The program was then able to proceed to its next error, which was that there was an index out of range. I can’t see a problem with the code, so I think the problem is that keys had been deleted, so I have added a catch statement that prints that keys have been deleted, as a placeholder until I can figure out whether that is the case. The scanner now passes output back to the main program, which then stores it in a json format file.

## 24/02/2021

Today, I am running the scanner passively for a long period of time to get a file of passive changes. There’s not much I can write about this, but it had to be done at some point, so it might as well be now.

## 24/02/2021 - Supervisor Meeting

In this meeting we discussed the upcoming blog post deliverable. John said that it should lay out the longer term vision for the project, as well as the work that has been done. He suggested imagining that we’re handing the project to someone who was starting the same project, and making it useful for them, by explaining the aims, the progress and the work still to do, as well as linking the code repository. John stressed the importance of writing down the thought process that goes into the blog post, especially how I am making it fit for its audience. We discussed the timeline for the deliverable, and it was suggested that I lay out the goals of the project before next week. The project should have a clearly articulated high level goal, and clear short term goals.

John suggested that I should talk to Conor about Eels, to give my thoughts on how it works as a program structure.

John pointed out that the output file for the scanner should be human readable, as it is likely that there will be a human readthrough of the program signature files to ensure that they are accurate before they are used in other parts of the project. He talked about the 80/20 rule, and how human input can minimise the 20% taking up the longest amount of time.

John suggested that a stretch goal of the project is to put a computer into a clean state, with just the programs that the user requires, and that an alternative goal could be to provide overall information on the status of the file system and registry.

## 24/02/2021

I used the debugger to step through the code handling the registry object to get better insight into how the keys worked. I think I now know how to ascertain which key values have been deleted, so that they can be stored separately. I need to check the length of the hive keys and if they are different, iterate through the objects to figure out which values have been deleted or added.

I made a separate function to handle this, as it was a separate process. I’ve made it so that the function detects key names that are the same, created or deleted during installation, and passes that information back to the calling function. I can then use this information to remove the index out of range error from before, as I can now manipulate the key comparison loop to only check common keys.

I have now altered the comparison code. Previously, it iterated through both keys at the same time, assuming that the keys had identical names within them, which was causing an out of range error if a key had been deleted in one of the keys. I now have it so that it searches both keys for each value name found to be the same in the function mentioned above, and then compares them, saving any modified keys to a dictionary to uniquely identify them. Tomorrow, I will need to check that the program is working, and then save the deleted and created key values to the return dictionary for the registry scanner, and then change the other 7 registry scanner functions to match the one I’ve updated.

## 25/02/2021

Today I need to test the functions that I created yesterday to ensure that they are working as intended, and then update the returns on the scanner to pass all required information to the main program.

I created three new functions, one to retrieve created key values, one for deleted key values and one for modified key values. They use the output from the function that I created yesterday evening, that finds which key values are different than they were, and then they grab the values from each key value and store them against the key value name in a dictionary. I should now be able to test that the code is correctly detecting the changes. I realised that I have six functions that differ by only two lines of code(these are the actual scanner functions, and they differ by the registry hive assignment), so I should make a master function and pass the registry hives to it, to declutter the code and make it easier to maintain for anyone who takes on this project after me.

I have run the program and checked its output, and it seems to be correctly finding the changed keys, so I’m going to create the master function to move the whole program over to this version.

I went back over the input before changing the rest of the functions, and the registry scanner is returning the same key names for modified, deleted and modified, but with different values. I’m going to step through the program to see what’s going wrong.

I figured out through stepping through that the line where I checked if a name was in a key was always turning up false because the object I was iterating through was a Value object rather than a list. I’m in the process of rewriting the registry key comparison function, but just wanted to take a second to ask who’s idea it was to name the values within a registry key “values” but to also give them “names” and “values”, which is making naming variables and commenting difficult, as I can’t talk about value values, even though that is technically what they’re called.

I’ve updated it, but I’m now having to chase down a bug that’s somehow populating lists that I know to be empty. Once I get it working, the registry scanner should be nearly functional, and after I build the filter and pass it the passive changes, it should be complete.

I found the problem with the code. The way I was updating the dictionaries that stored the created and deleted keys, if there were no keys values different, they would store the key path with an empty dictionary, making it look as though that key had been deleted. By checking to see if the dictionaries I am passing are empty, I should get the correct output.

The output seems to show that the scanner is now functioning correctly. I’m going to rewrite the scanner functions to remove clutter and bring the scanner to full functionality(other than the registry hives that take up more than 2 minutes to scan, which are still omitted for testing purposes).

I have now refactored the program so that it is more readable and easier to modify. The scanner is now almost complete, save for building the registry filter to remove system changes to the registry. For simplicity in the creation of the registry filter, I am going to assume that individual keys that windows uses are not changed by installations. This may not be the case, but it should be true enough of the time to make this work, considering that I will be manually inspecting the output. And if it wasn’t the case, then even filtering the individual key values wouldn’t work, as any of them could be rewritten. I am currently getting the program to print just the paths of keys that passively change, so that I can gain an understanding of how much filtering the program will have to do.

I had to go through my code line by line to figure out why it had broken, because it wouldn’t run the registry scanner. I figured out that PyCharm had suggested a function refactor to make my function name match PEP8 formatting and I had agreed, but I hadn’t changed the function in the Eels webpage. Once I sorted that, I found that Windows changes approximately 4 keys per minute, so it should be easy to make manual passes over the data.

After a lot of effort trying to pull the correct information from the JSON object, and issues with how file strings work, I managed to get a functioning filter for the registry, that pulls key paths to ignore from a file. I tested it against OBS Studio, and it showed the file changes that OBS was making. It didn’t show the changes OBS was making to the registry, but that is most likely because I’m still omitting the Software Registry Hive, where most of the changes are being made. The scanner program is now technically functional, and so my aims tomorrow are to update the passive file filter as there are still too many false positives, and then to start on my blog post, laying out the aims and targets for the project.

## 26/02/2021

My first task for the day is to update the file filter to include the hour of passive files that I recorded the other day. I should be able to pull the file paths out of the JSON object and add them to the list in the passive file program.

I pulled the paths and put them into the scanner’s no scan list, which worked for the most part. Files being created by microsoft edge were still being shown because they all had unique names. There doesn’t seem to be a way to tell watchdog to ignore this without breaking the scanner, so instead, I told the handler functions that handler file events to only save the events if they weren’t from edge. I tried to do this with a regex of the path to Edge’s temp files, but the re module doesn’t like backslashes, so I created two regexes, one for “Edge” and one for “User Files”(which is the directory edge saves temp files to), and this got rid of the Edge events from the scanner.

## 27/02/2021

Blog post link: <https://docs.google.com/document/d/1Ftp9i_hJ3xI7kwaxYT94dlILOKMwcEUxbGlnEJn4Az4/edit?usp=sharing>

Today, I focused on the blog post for the project. I decided to start it with an “If you are reading this, I am dead” trope, to break the ice with the reader and put them at ease. This is a lot more likely to engage them with the guide than starting with something like “In this Project, we…”, and puts a bit of humour into the post.

I then went into discussing the overall aims of the project. I included the original title of the project brief in the first paragraph, because this allows the reader to quickly understand what the project is trying to do. I then went into what this meant, but not in too much detail, because I don’t want to overload the reader with information and cause them to stop reading the guide because they can’t take in what they’re reading. I then broke down the overall problem into the questions it raised, and how the project solved/planned to solve them. By asking questions in this introduction section, I can hopefully get the reader to begin thinking about how to take the project forward right from the outset, meaning that they’ll be going into the more technical sections of the blog post with an attitude of problem solving, which is the frame of mind that should be most beneficial to them as a person trying to use this project to solve problems. Because I was only briefly discussing the overall aspects of the program, I mentioned that I would go into more detail later on in the post. I’m hoping that by reassuring the reader that they don’t need to fully understand everything right at the outset, they are more likely to continue reading, and less likely to be preoccupied with trying to fully understand the entire project right at the outset, because if they do that, they are likely to ignore anything I write that would go against how they would have tackled the problems, and won’t gain as much help from the guide as they should. I also gave small examples of the problems that would need to be overcome, such as moving a file without causing a program to lose track of it. This should give the reader a better understanding of the issues that I am talking about, which is necessary so that the reader understands the project properly. To further prevent information overload, I have included the line “the program is going to get annoyed and go on strike”, which should take the user out of the guide for a second, allowing them to go over what they’ve just read. It also stops the intro from being monotonous. I have included the line “there are a few million different ways that a program could install itself onto the file system” which should cause the reader to take a few seconds to think about different ways a program could install itself, which will help them to understand the approach I have taken with the project, and potentially spark some ideas in the reader for how to improve the project moving forward. I very briefly outlined the UI that the project uses, and how the scanner works, to allow the reader to understand the concepts at a very basic level before they have to try and understand them at a more advanced level. This gradual gradient of explanation should provide the best understanding as to the function of the project. I finished the intro section with how what I have explained ties into the main aim of the project, which should help the project to click with the reader.

Next, I went into more detail on each subsection of the module, explaining the libraries used and what they did, and why that was necessary for the project. This section is designed to show the reader the tools best suited to the project, and why they are good choices. I have included links to the documentation of the libraries in this section, to allow the reader to easily get more information on the libraries if they need them, to help them understand them better. I also plan to put the link to my updated how-to guide into this section once I have written it, to allow the reader to get all the relevant information for the libraries in one section. I included a fact about pressing Ctrl-Shift-I on discord, which is tangentially relevant to the reader, as it allows them to see a real world example of the UI archetype that the project uses, building their understanding in a way that should be entertaining to the reader.

## 01/03/2021

Today, I am continuing with my blog post. I have started with the non-technical explanation of the registry scanner. The purpose of the non-technical explanations is to allow the reader to understand the concepts behind the project work before having to understand the ins and outs of how it actually functions. It will hopefully make the blog post most useful by easing the reader into the topics, preventing information overload which would make the blog less useful. I have included a line about how the registry scanner was “the bane of my existence” and that ”hopefully my explanation later in this blog should save you a lot of googling and reading documentation”. The first part should hopefully make the reader feel more comfortable, as annoyingly complex parts of programming projects are a near universal experience for programmers. This should hopefully allow them to focus on the blog more, as they feel more connected with it. The second part will help reiterate the purpose of the blog post, to save the reader time. This should make the reader want to keep reading the post, as there’s little programmers like more than shortcuts, and I am mentioning that I’m saving them a good amount of research and trial and error. I have explained the first problem with creating a registry scanner at a general level, because if the reader understands up front the problems that the project has tried to solve, they will better understand the design decisions that I have made in the project, which will make it easier for them to carry the project on. An additional benefit of explaining the problems is that they may be able to think of easier or less computationally expensive methods of overcoming the problems, which would make the project more useful and enable it to be run on a wider majority of devices. I then explained how I solved the problem briefly, and mentioned that I would explain it more completely later on. This encourages the reader to continue reading, and again allows them to understand the basics before getting into the specifics. I inserted a link to the windows REG command, while I explained that it was part of the solution to the problem. This gives them the opportunity to research the module before I explain the specifics of how it was used, which ensures that they have a good grasp of the command before they see how it is used in the project. I then explained that the project uses Regipy, and provided a link to it. I mentioned that it is only sparsely documented, and that this guide should “hopefully fill in the gaps.” This should hopefully get the reader to understand that this blog is providing information that is difficult to find, ensuring that they continue reading in order to gain the full benefit of my research. I explained how regipy was used in the project, and added a line saying not to worry if it didn’t make sense at this point, because we’d be going over it later on. This line should prevent the reader from getting worried that they don’t understand the project, and lets them know that this is a section that we will be focusing on.

## 02/03/2021

Today, I started with the brief description of the program template files. I started this by reminding the reader of the purpose of scanning the file system and registry. This provides the context for the template files to the reader, helping them to understand what the template files are used for.

I explained that JSON was used for the template files, and why this was the case. By explaining why design decisions were made as well as what design decisions are made, the reader should be able to more easily put themselves in the headspace to understand the project more thoroughly, and to allow them to discern whether they know of better design decisions that would better suit the project. I gave examples of the sort of data being stored, which should help them understand the files in a practical as well as theoretical manner. I then moved onto a description about the install process, explaining to the reader why our scanner is operated manually rather than automatically. This is not intuitive, as most people don’t know how program installations work, so explaining how it works here helps avoid confusion for the reader. I explained to the reader how we would make sure that the project was as useful as possible by ensuring that it works with the most popular programs, which should give the reader a better insight into the overall goals of the project. Following this, I explained the next area of the project, moving files on the file system. Right at the start of this description, I mentioned that it was an area that I had not begun work on yet. Hopefully, this should get the reader into the right mindset to consider how to take this section of the project forward. To cement this, I have mentioned to the reader that this section could be a good starting point for their continuation of the project. I then went through the next part of the project, removing unnecessary files from the computer. I again mentioned that this could be a section of the project that the reader could take forward. In this section, I included the phrase “You may think that…”. This gives me a chance to address misconceptions that the reader has regarding this section of the project, allowing me to further explain the design decisions behind this project. I used the word “babysit” to describe how we would handle this section of the program, which should help the reader understand that installations need monitoring because they often place files in “immature” locations that do not make sense. For the final part of the non-technical introduction, I described the last goal of the project, setting up a custom installer that would allow our project to keep the file system in a tidy state. I explained the reason for this goal to the reader, to ensure that they understood how it linked to the overarching goal of the project.

## 03/03/2021

Today, I focused on writing out the technical section of the blog post. I decided to split this into sections, where each section corresponded to one aspect of the system, such as Registry scanner or UI. This allows the reader to understand what will be discussed before it is discussed, which should allow them to put themselves into the correct frame of mind for understanding the technical explanation of each section.

At the start of the technical guide, I will put a link to the GitHub repo for this project. Placing this at the start allows the reader to try to read through the project files on their own if they would prefer, and then they could navigate to whichever section of the technical guide that they don’t understand from the code itself. I have thoroughly documented the source code for the project, which should facilitate this type of project readthrough.

The first section that I broke down into its technical aspects was the UI. I decided to start with this as I already have a guide for part of the UI that I plan to update on Friday to cover the full UI. By putting this first, the reader can take a break from reading the main blog post, and attempt to follow the UI guide. This provides a natural break point in the blog post, to prevent information overload, and the reader is more likely to understand how this project’s UI works specifically if they understand how this type of UI works generally. After explaining the basics of the UI, I then took the user through each section of the UI, explaining specific technical aspects. Before I did this, I explained to the reader that I had included Materialize within the UI to make it look nicer, and that this had no bearing on the function of the UI itself. To ensure that the reader didn’t get confused as to what code was for aesthetics and what was for function, I highlighted the code that was from Materialize in yellow. This should increase readability and understanding for the reader. I included the line “as you can see” in the guide at various points. This phrase should cause the reader to look back at the screenshot at the section I am referring to if they hadn’t already, which ensures that they will understand what is being discussed. I have included the phrase “you will notice” for similar reasons. I drew the reader’s attention to certain lines within the screenshot that exist for debugging purposes. This increases the reader’s understanding of the code, and also lets them know that these lines would need to be removed before the project was finished, but that they are currently useful to the reader to reduce the amount of manual debugging that they have to do. I have not yet connected the system tray application to the Eel application, and so I cannot discuss that within the guide until they have been connected, which I plan to do on Friday.

That finished the UI section, and so I moved on to describing the scanner systems themselves. I decided to start this by going over the imports required for the project. This allows the reader to understand the libraries at the start, so that they are not confused by the use of the libraries during the tutorial. In case the reader has not installed the correct version of Python on their system, or doesn’t have pip installed, I have linked to a guide for installing python 3 with pip. I have included installation commands for the required python libraries in bold, clearly labelled, to prevent confusion for the reader.

I then moved on to the technical explanation for the File Scanner. I started with a reminder as to the purpose of the file scanner, and where it fits into the project, to ensure that the reader fully understands the purpose of the scanner. I tried to go through the functions in an order that reduced the amount of code that I have to say will be explained later, as the reader is more likely to understand the code if the explanation does not jump around. While I remember to mention it, throughout the guide, I have used screenshots to aid the reader’s understanding, and have included different types of section breaks(dotted lines, solid lines etc) to denote where sections end and where subsections end. This should provide natural stopping points for the reader, so that they do not become overwhelmed reading the guide. I have also used the collective pronoun “we” instead of “I”, because this helps the reader to think of themselves as connected to the project.

I started by explaining that the first four functions we would discuss were very similar in function, and so we would look at one of them, as opposed to all four. This cuts out redundant information for the reader, ensuring that they are only given information that is relevant. When I discussed the file filter that I have built into the scanner, I explained why it was necessary, and the edge cases that it did not work for. I mentioned that I had coded a solution for the edge cases, but that my solution was not the most suitable solution. This allows the reader to consider whether they could improve the file scanner if they took the project forward.

The triggerFileScan() function is one of the biggest in the program, and as such could be confusing to the reader. So instead of describing the function as a whole, I broke it down into sections, each of which contained a screenshot of the lines of the function I was discussing. This should help ensure that the reader gains the most understanding possible, without being confused as to where we have gotten to in the function.

To ensure that the section isn’t monotonous, I have included the line “**\*Flight Attendant voice\*** If I could direct your attention to the sixth line of this screenshot” which should provide some levity to the reader, helping them to enjoy the guide.

In this section, there is a watchdog function that takes four parameters, which would be confusing for the reader. To ensure that they understand what the parameters do, I have added a bullet point list where each point explains one of the parameters. I have placed the parameter names in bold, because it is more aesthetically pleasing for the reader.

After I had finished writing the File Scanner section, I moved onto the Registry Scanner portion of the guide. Because the registry scanner spans multiple python programs(the main file and the registryScanner.py file), I started this section of the guide by explaining that we would go through the registry functions in the main program, and then the registryScanner.py file. This should prevent confusion over where in the code the functions I am discussing are situated.

In the registry section, I decided to include a brief description as to the layout of the registry, as I know from experience that this can be confusing. This should allow the reader to better understand the function of the scanner. I mentioned at this point that I would link resources to help understand the registry at the end of the post, in case the reader does not understand my explanation and requires additional information.

To make the move from the main file to the registryScanner.py file more interesting for the user, I used the phrase “If you’ll follow me out of the main program, and through the door marked “registryScanner.py”, we’ll take a look at how we scan the registry in this project.” This should make the blog more fun to read for the reader.

The next function of the registryScanner is quite large, and I did not want to confuse the reader, so I used Paint to highlight each section of the function in a different colour, and then referred to each section by its colour. This should ensure that the reader always understands where we are in the function, improving its readability and reducing the chance that the reader will have to reread large sections of the project. It was within this function that the terminology for the registry keys became important, as for some reason, as I have mentioned before in this diary, multiple separate things in the registry are called by the same name. To ensure that there was no confusion for the reader, I have created a text box that stands out from the website,. It explains the terminology that the Registry uses, and what we will refer to things by in this blog to avoid confusion.

To add some humour to this section, I made fun of my own naming conventions within the project, mentioning that “The function that I have, in an amazing blaze of creativity, called check\_for\_key\_creation\_and\_deletion().” This should help the reader to maintain focus on the project, as well as ensuring that they pay attention to the functions as we go along.

I understood that there was a loop in one of the functions that was quite complex, so I made sure to thoroughly explain what it was doing, so that the reader could understand its function without having to try and figure it out on their own.

To let the reader know that I understand that the registry seems unnecessarily confusing, I have included the line “Windows doesn’t keep a list of registry keys that it changes, and I’m nearly certain that at this point, Windows is so abstracted that no one knows exactly what its doing in the background”. This should get the reader to understand why the program is complex, and why this project is necessary.

Because of the subject matter, the blog post is necessarily quite long. I have kept the conclusion short, which should be beneficial to the reader, who will not want to be reading a long winded conclusion paragraph. After the conclusion, I have given a number of links that I believe will be helpful to the reader taking the project forward. To ensure that the reader can navigate directly to the links they want, I have grouped the links under headings such as “Registry” or “Eel”.

## 04/03/2021

Today, I worked on converting the draft blog post to a website, editing it to be more useful to the reader as I went along. I decided to title the blog post “Tidying the Computer - Describing my File and Registry Scanner Project”, as it is descriptive, gives the reader the aim of the project right at the start, and multi-part titles are frequently used on the internet. I decided that I would use dotted line dividers to separate different parts of the same section in the blog, and full line dividers to separate the sections. This should make it easy for the reader to understand where each section begins at a glance.

I decided to title the section that introduces each section of the project “parts of the project” as this is simple, making it easier for the reader to understand its purpose, and it is also alliteration, which draws a reader’s attention.

I removed the word “but” from the start of the second section’s introduction, as it was portraying a lack of confidence in the project. I also removed “if you want to”, as this also showed a lack of confidence.

I have changed some wording within the install process description because it sounded confusing when I read it back. The new version should be easier for the reader to understand.

To add some more humour to the guide, I have included the line “without breaking the program and the law”. This should prevent reading the guide from becoming monotonous for the reader.

I noticed that I didn’t have much of an introduction for the technical section of the program, which would be off putting for the reader, so I added a brief description to show the reader what the section will be about.

After I had created the blog post, changing phrasing and sentence structure to aid understanding, I tried to figure out how to connect the WxPython system tray application to the Eel application. I read through the Eel source code, but there does not appear to be a way to hide the control webpage using Eel. I thought about trying to query the procmon to find the browser, but Eel works with any browser, and there wouldn’t be an easy way to discern what was an Eel browser window and what was a regular browser window. It would also be complex to access the browser settings from python to try and hide it. Through playing about with the eel code and reading the eel docs, I found that you could start the Eel server without opening a browser, and then navigate to the control page on your own. I am now trying to set up selenium to allow the system tray application to open its own browser window, navigate to the Eel command address, and then control the window. To get Selenium to work, I have had to install the edge webdriver, unzip it, and place it in the same directory as the python file.

<https://stackoverflow.com/questions/29858752/error-message-chromedriver-executable-needs-to-be-available-in-the-path>

<https://towardsdatascience.com/controlling-the-web-with-python-6fceb22c5f08>

https://www.reddit.com/r/learnpython/comments/6kh7nn/selenium\_hide\_then\_show\_window/

It doesn’t seem like you can hide and reveal the browser through this, but you can set its position to something wayyyyyyyyyy off of the screen, and bring it back when you need it, which is what I plan to do.

I GOT IT TO WORK. I’ll probably explain tomorrow, but I’m too tired this evening.

## 05/03/2021

Today, I am going to start by expanding the how-to guide to explain how to make an Electron.js style web application. Previously, the guide only covered creating a system tray application, but it will now cover an Eel web app and using Selenium to control one from the other.

I’ve started by creating a new index.html file, and changing the existing one to “tray.html”. I have started the guide by asking a couple of rhetorical questions, designed to get the reader to think about how Discord works. This should put the reader in the correct frame of mind for the tutorial.

After I summarised the steps in the guide, I added a section to the guide explaining how to install and set up the libraries that the reader will need for the tutorial. I gave each library its own subsection, to aid readability for the reader. I made sure to explain the Selenium setup in detail, because it is the most confusing of the installs. I included the phrase “Okay, here’s where things get slightly more difficult”. This lets the reader know that they will find the section difficult, so they are less likely to be concerned with the increase in difficult and stop reading.

The next heading for the guide is “Making a system tray application”. Here, I provided a link to my wxPython tutorial that this how-to guide is an extension of, and explained to the reader that once they had completed it, we would be updating it to control the web app, but that they would need to understand the basics first. By having them build a basic system tray application first, the reader will better understand the more advanced version that the tutorial uses.

I then moved onto the Web Application tutorial. I started this with a screenshot of what the top level of the file structure for the web app should be, as this was a section that tripped me up initially. I briefly went through how the web app functioned, as it functions similarly to a regular HTML website. By not going over every single detail of the application, I prevent the reader from giving up due to being giving useless information, ensuring that the reader continues to read the guide.

After I finished explaining the web app, I moved onto the “Putting it all Together” section of the program. I included the line “Okay, I need you to pay very careful to what I am about to show you, I'm going to throw a lot of information at you at once, and I need you to keep up. Ready? Okay, here goes nothing. The changes you need to make you your server code are as follows: Go to the last line of the server.py file, and change mode="Edge" to mode=None.”. This humour should keep the reader engaged in the tutorial, and pretending that something is complicated and then making it easy is a good method of relaxing the reader, as it puts the problem into perspective.

I have ended the tutorial with “I hope you have found this tutorial useful, good luck with whatever project you are working on.” as this is a good way to ensure rapport with the reader.

I now need to focus on the CSS for both the How-To Guide and the Project Blog Post, so that the webpages look appealing to the reader. I am not going to use Materialize.css this time, as it caused problems the last time. Unfortunately, because of time constraints and the fact that I have two other active pieces of coursework at the minute that I need to do over the weekend, I am not going to be able to get a second opinion on my guide and blog.

In order to speed up the CSS side of things, I am using a themed CSS schema from templated.co. This makes the website look professional to the reader, which encourages them to read it. I also added some royalty free images to the blog post, as well as a screenshot of the program in action, because the first section of the blog post was just a wall of text, which isn’t appealing to the reader. I have now finished the second hand-in, so I am going to upload it to canvas, and then that’s me done for now.