Team 4

Max Goad

Jace Bayless

Tri Pham

Apurva Rai

Meet Kapadia

# Project Proposal Report

## Project Name

- PC Pal

## Project Synopsis

- External touch screen device that controls, adjusts, and manipulates several features within a computer.

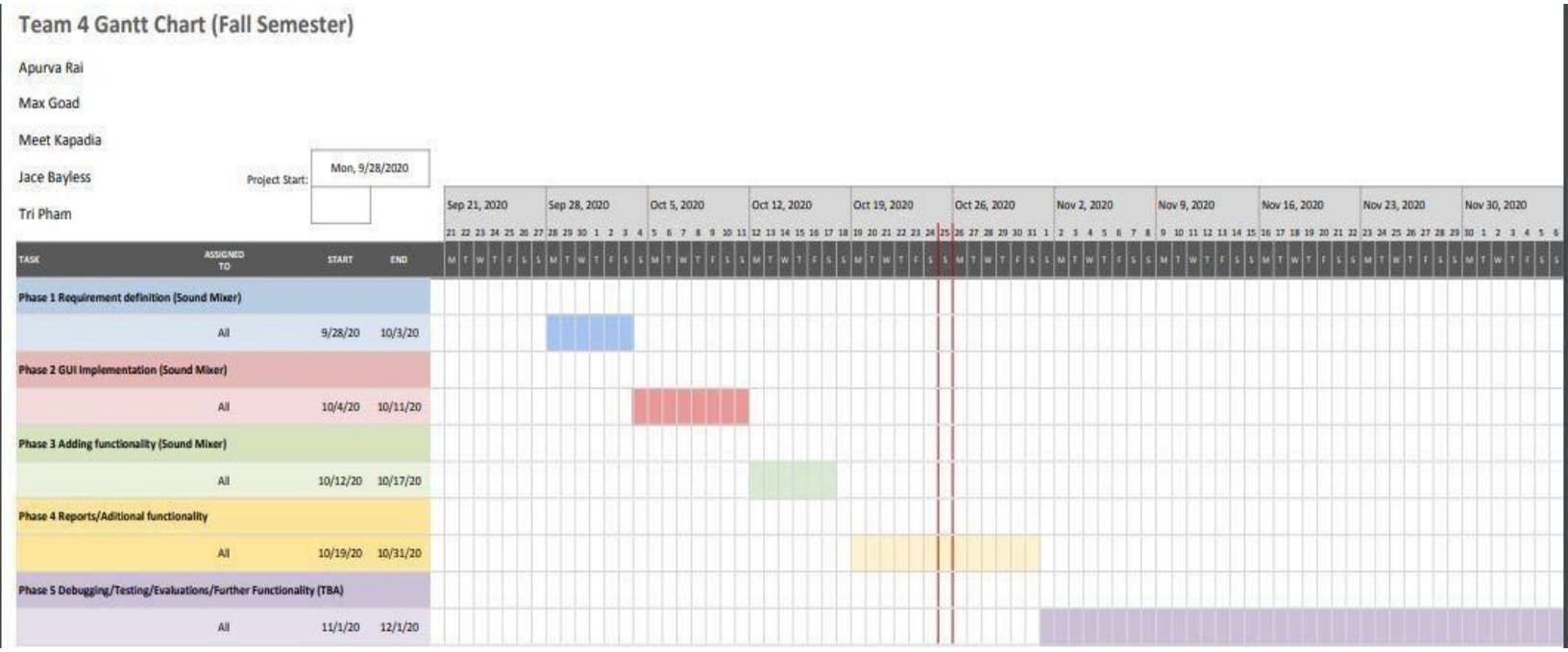
## Project Description

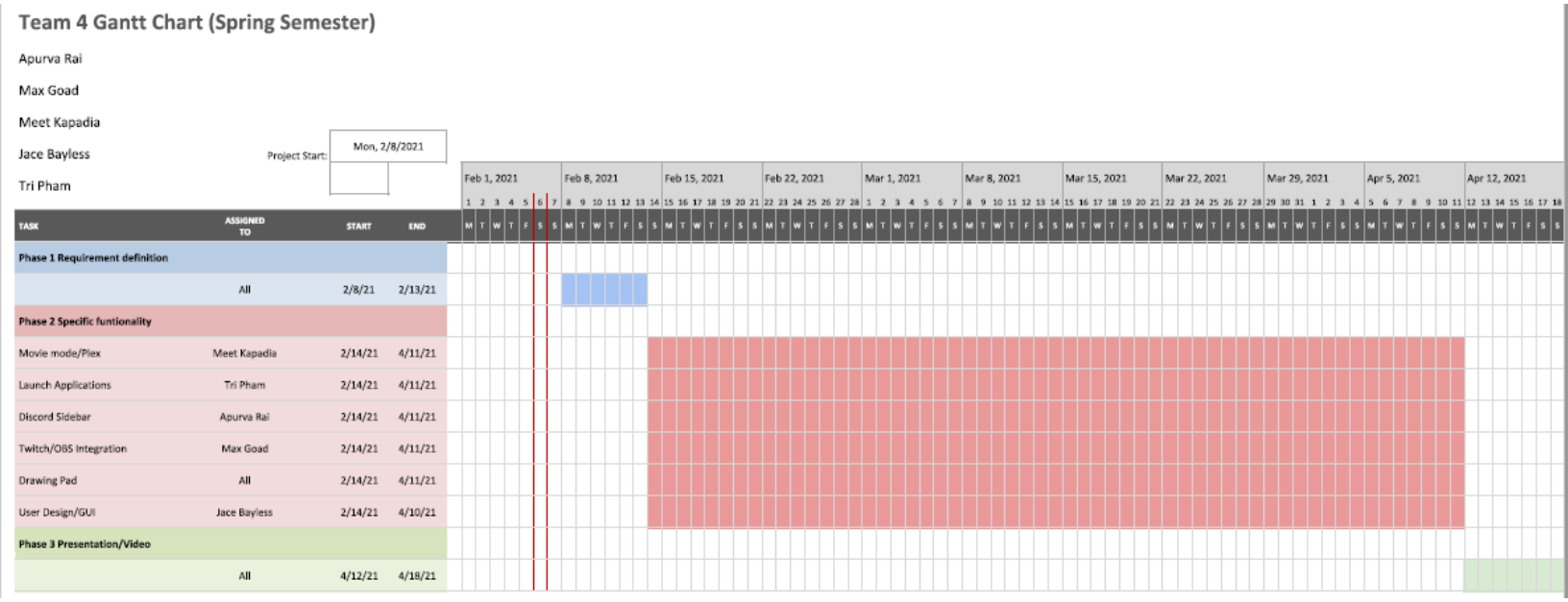
- While in the middle of a game, a full screen, or any other task, accessing features such as the sound mixer or Spotify can be painstakingly tedious. The PC Pal makes those aspects easily accessible without stopping or interrupting the current task. All you have to do is reach over to the touch screen device and manipulate whatever task you want to, such as sound mixer, and it will change that aspect for you without having to alt-tab. In simpler terms, it is almost as if there is a second or third monitor; however, in the end, instead of taking up copious amounts of space and requiring a mouse and keyboard, it is a handheld, touch screen device. The problem of having to switch over or out of a screen, halting that task, will be solved. Some of the features that PC Pal takes into account include but are not limited to controlling the sound mixer, Spotify, switching desktops, etc. The PC Pal will run on a Raspberry Pi with a touch screen that is compatible with that Pi.

## Project Milestones

- For the fall semester, the first milestone was that get all our ideas in a bucket and see which are possible, so we have a rough idea on what we want to implement later on. Right now, we only have the sound mixer done, Spotify integration, and alt-tabbing. As for other features, we have a rough idea on what to do but we are still finalizing what and how to achieve those in the best way. We have delegated who is going to do what (see Gantt charts below)
- Second milestone is to implement the GUI and different functionality to ensure that specific features we want to implement are possible. The GUI for the sound mixer is completed for the most part, but it still needs to be polished. The GUI for Spotify and alt-tabbing at the moment is still very base level and needs to be polished a lot. Last semester our milestone was just to get the GUI to look like something we can use and test and we achieved that.

- Lastly, debug and test functionality. For the sound mixer, we have done this already intensively last semester and we will do this for other features we implement as well. For all these milestones, we worked together already to implement the sound mixer and we worked together as a group for the other features as well, at least last semester. This semester, we are planning on implementing one feature each by ourselves, but last semester we worked together as a group to do most things. Not only that, we have weekly meetings to assess and make progress as well. We still have these weekly meetings.
- For the spring semester, the first milestone we want to achieve is get our ideas set in stone and divide up the work. Again, we have already gotten an idea on what who is responsible for what and that can be seen in the Gantt chart.
- Second milestone we will have is that each person will implement a feature by themselves. See Gantt chart on who is doing what.
- Lastly, our last milestone is to get ready to present this project.
- When it comes to documentation milestones, those are expected to be met when a function is implemented or when they are required. For example, complete creating an initial Gantt Chart by the week of 10/26/2020, and update the Gantt chart every time progress is made. This was last semesters goal; this semester's goal is also to update the Gantt chart as we go along. Not only that, we also need to make a final quad chart by May 5$^{th}$ of this year. Furthermore, finish an initial use case diagram by then as well but make changes to it as changes to the code are made as well. Once more functions/features are implemented (right now we only have one), we are going to make a class diagram as well which we want to get done by sometime this year when more features are added.
- Lastly, in the following Gantt Charts, you can see how the milestones are divided up and how long each might take and when they are due. Some milestones are divided up into several phases, but as you can see, we plan on working together heavily initially and then going on our own. The Gantt Charts are also divided up into the fall semester and spring semester as you can see in the following pictures:





**Team 4 Gantt Chart (Spring Semester)**

Apurva Rai
Max Goad
Meet Kapadia
Jace Bayless
Tri Pham

Project Start: Mon, 2/8/2021

| TASK | ASSIGNED TO | START | END |
|---|---|---|---|
| Phase 1 Requirement definition | | | |
| | All | 2/8/21 | 2/13/21 |
| Phase 2 Specific funtionality | | | |
| Movie mode/Plex | Meet Kapadia | 2/14/21 | 4/11/21 |
| Launch Applications | Tri Pham | 2/14/21 | 4/11/21 |
| Discord Sidebar | Apurva Rai | 2/14/21 | 4/11/21 |
| Twitch/OBS Integration | Max Goad | 2/14/21 | 4/11/21 |
| Drawing Pad | All | 2/14/21 | 4/11/21 |
| User Design/GUI | Jace Bayless | 2/14/21 | 4/10/21 |
| Phase 3 Presentation/Video | | | |
| | All | 4/12/21 | 4/18/21 |



To better see these charts, here are the links for them to view on Google Sheets:

- Graph 1: https://docs.google.com/spreadsheets/d/1NgHov51xTCXuhuXHVR5jusR98QkU1-6VKduKo8gKX48/edit?usp=sharing
- Graph 2: https://docs.google.com/spreadsheets/d/1BfsWX13zPGnAbXvbbHcXJGWIgyS3ny5tg48U5QEn6EI/sedit?usp=sharing

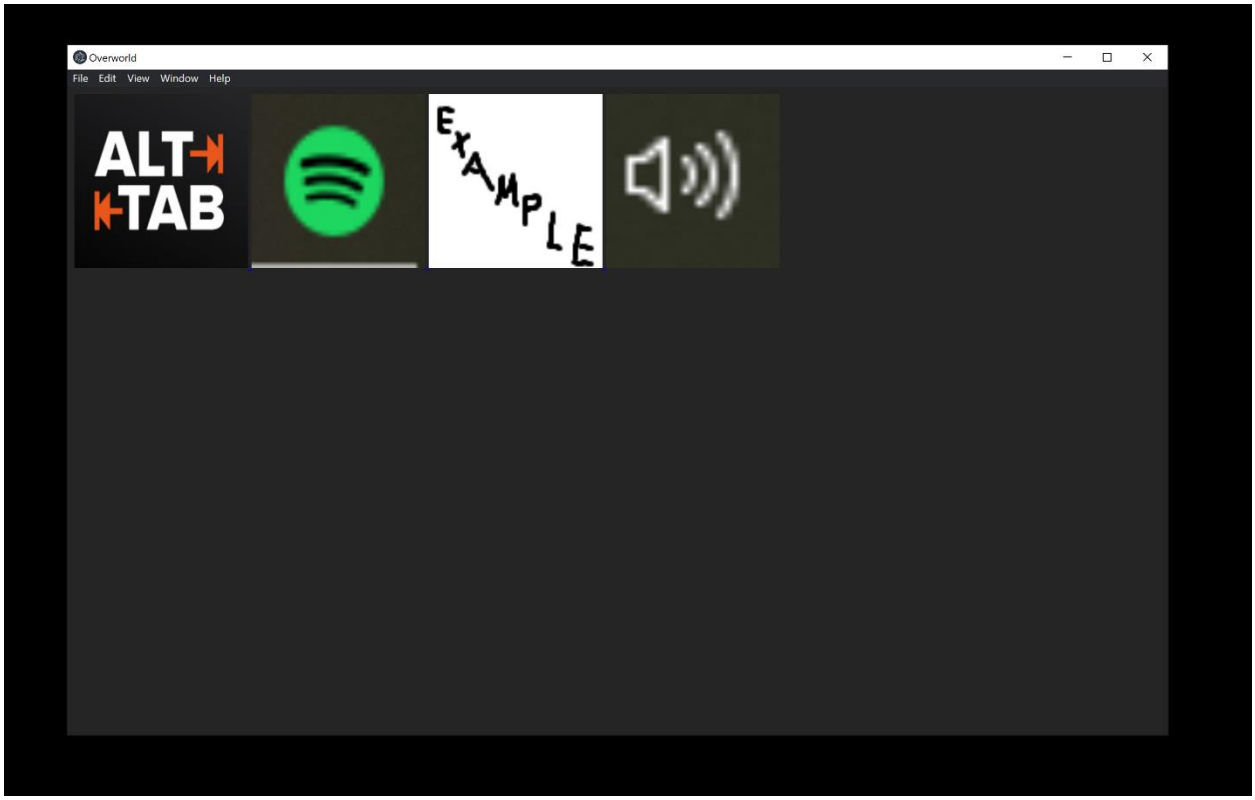
Project budget

- The whole project is dependent on having a Raspberry Pi with a touch screen. The Pi4 starter kit we ordered ended up being $119.99. We looked into the preferred vendors and could not find the specific Pi we wanted so we ended up using Amazon. The touch screen for a Pi4 cost $129.99, we used Amazon for this again. Thus, the cost came out to be $249.98 without tax. The parts have already been picked up by our team and ready to be used to further the project. We even have started testing the product with the Pi and touch screen.

Final Project Design

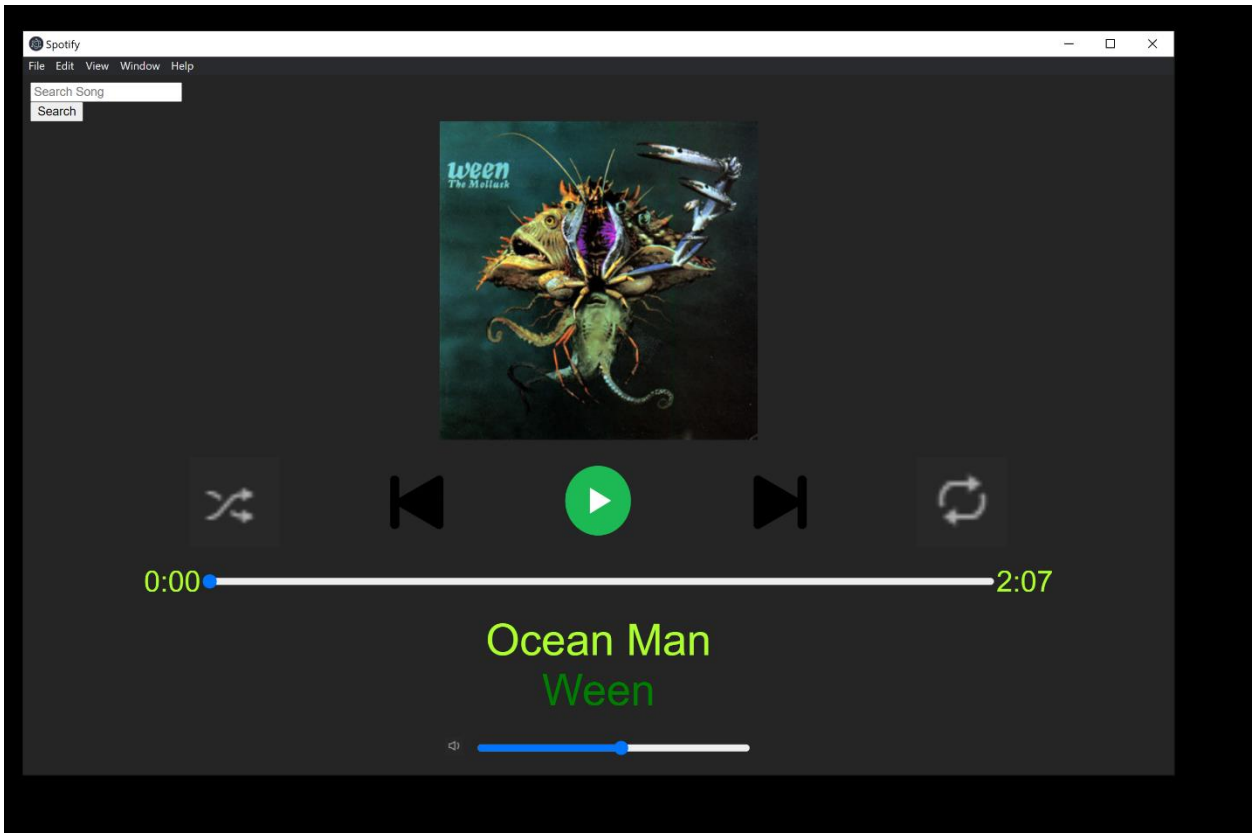
- When it comes to the project, the code is split up in two sections. The first section runs on the PC and the other section runs on the Pi. When it comes to the PC code, it starts with the PC looking at computer COM ports to see if it can detect whether or not the Pi is plugged into the PC or not. If it is not plugged in, it looks for an emulated port. The emulation just tricks the computer into thinking something is plugged into the PC so we can test the code without actually having to plug in the Pi. This makes testing the code easier for everyone considering we will only have two Raspberry Pis and five members on the team. If someone is not in possession of a Pi, that person can just use the emulation to test the code. Regardless, if the port has been found or if the emulation is

detected, it starts listening on the port for serial input, which will come in the form of Json. Json is a minimal, readable format for structuring data. Json is mainly used to transmit data between a server and web application, as an alternative to XML. This is how the Pi or emulation communicates with the PC. When the input is found, it passes it over to a function that will decode the Json and make the appropriate function call, such as mute master volume for Windows volume mixer or the lowering of volume of a specific application. Once the appropriate function call is completed, it will start to listen again for serial input again to see what other functions to call. In other words, it starts to detect again if more manipulations are being made by the user. All of this code is done in Python, and the backend code running on the Raspberry Pi is also in Python. The other half of the code only runs on the Pi, and at the moment we only have a single feature that runs on it. The Pi-side code works by running an electron app (for the GUI) which communicates using local Rest calls to a Flask Python server also running on the Pi, which then sends Json data to the windows machine over the serial port.
On this webpage, we are able to design a front-end and this allows us to make python calls to send serial data to the PC through the webpage. Using Electron (HTML and JS) for the front-end allows us to create a better user-friendly UI. At the moment, we only have the ability to manipulate the Windows volume mixer, Spotify, and alt-tab using either the Pi or emulation, but more features will be added later on (the code now is very generic so more apps can be created easily). In the below pictures, you can see what the current code does in regard to the sound mixer. This is what the home page of the product currently looks like:

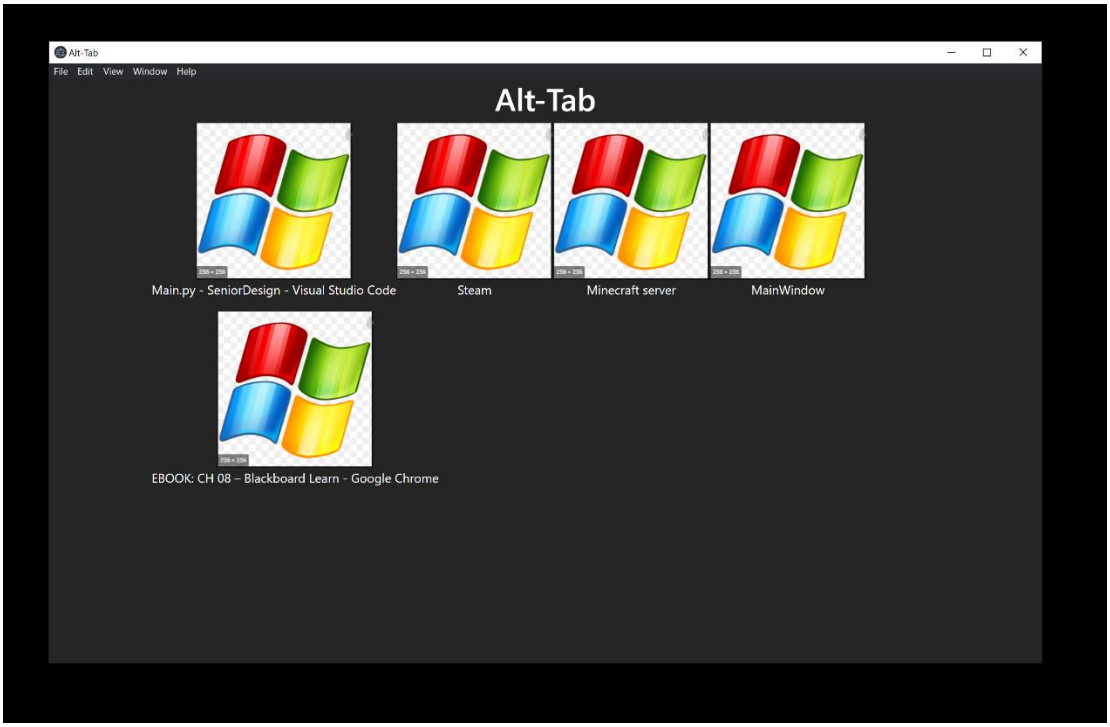


The first picture is the picture of what screen looks like the second picture is illustrating how manipulating the sliders within the program manipulates the Windows volume mixer.


In the below pictures, you will see how the Spotify integration currently looks and operates:



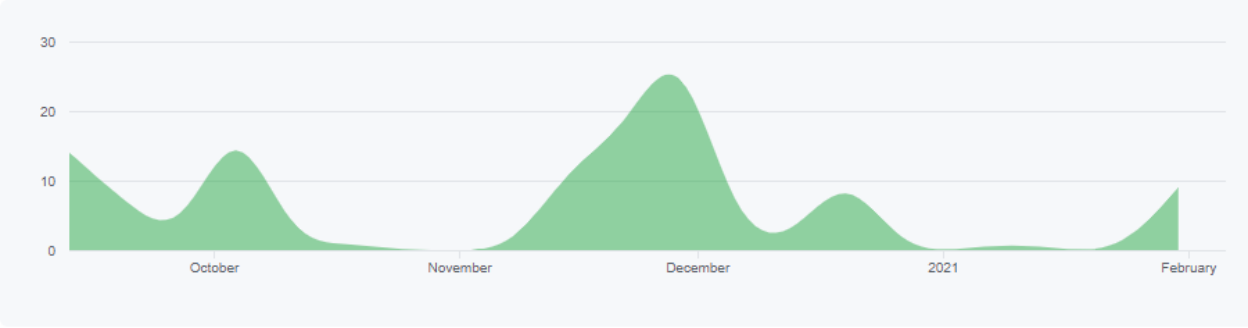

In the below pictures, you will see what the alt-tab currently looks like:



As you can tell by the pictures, a lot of redesigning needs to be done to make these apps look good; however, all the functionality is there. Some of the other features we are thinking for at the moment are plex integration, a Discord sidebar, Twitch/OBS integration, a drawing pad, launching applications, etc. In the below chart you can see our current milestones and what we have completed and what we are looking to complete when it comes to these features this semester:

| Date | Milestone | Status | Changed (*) |
|---|---|---|---|
| 2/5/21 | Implement alt-tab, Spotify, and sound mixer (from break till now) | Completed | |
| 2/12/21 | Finish up final project design. Distribute individual apps | Completed | * |
| 2/19/21 | Work on individual apps | In-Progress | * |
| 2/26/21 | Work on individual apps | Planned | |
| 3/5/21 | Work on individual apps | Planned | |
| 3/12/21 | Test each other's individual apps | Planned | |
| 3/19/21 | Implement individual apps into the main project | Planned | |
| 3/26/21 | Clean up the UI and remaining bugs. | Planned | |
| 4/2/21 | Stress test the product on Raspberry Pi | Planned | |
| 4/9/21 | Finish up the project video. | Planned | |

We have been using GitHub to keep track of progress as well. After every edit that was made and thoroughly tested, it is pushed onto the GitHub to be reviewed by other team members. Once it is approved by the members, it officially gets added to the repository and further gets tested by the other team members. After this if anything else is changed, the progress is repeated. In the below graph, you can see all the progress we have made and tracked through GitHub:




The main design constraint we have with our product is that a consumer or anyone has to have a Raspberry Pi and a touch screen that goes along with it. Without these two parts, there is nowhere the code can even be run or implemented. Because of this, without these two products, PC Pal would not exist and all the features that were implemented within the Pi cannot be used. You cannot replace the Pi with anything else either, for example an Arduino, that serves a similar purpose as the Pi. Furthermore, the touch screen or a

monitor is necessary as well because one cannot manipulate different settings with just the Pi. This is a huge business constraint for us as well because in order to sell our product, we would need to sell the Pi and the touch screen with it. In order to do that, we would have to get permission from the creators of these products to sell them, which probably cannot be done. This means we have to go a different route of just selling the code that users can buy and apply to a Pi or using custom hardware which seems out of the scope of this project. So, the user has to buy the Pi and touch screen separately and buy our code separately as well. Another design constraint we have is that the product cannot be wireless, it has to be plugged in to a computer. The PC must have a port for the Raspberry Pi to be plugged into as well, without it the Pi is unusable. As you can see in the picture below, the Pi has to be plugged into a PC for the code to run and work.

We started testing with the touch screen recently; however since we only have one of those, emulation still plays a huge part in our project. The images of the sound mixer from up above were taken on a PC, however, the official product will be a touch screen, that screen will be the touchscreen connected to the Raspberry Pi instead. Speaking of a computer, another constraint is that this product does not work with any Apple products, it has to be Windows. It is not compatible for MacBook, iMac, etc. so the user must own a Windows OS for PC Pal to be usable. Not only that, some of the Macs don't even have the port that is required for the Pi to be plugged in, making our product unusable.

Ethical Issues

- A major ethical issue we need to take into account is privacy. The product will be run on a Raspberry Pi, which has its own OS, it will be connected to a Windows OS, and it also has the ability to be connected to the internet. Based on these facts, we have to make sure as a company that we do not include any code that extracts any type of data that is private from the user because that is unethical. Being connected to so many things and interacting with so many things, a lot of data is generated, data such as what sites does the user frequently visit, etc. Because of this, our product will not extract any data other than what is necessary for the functionality and, thus, will be ethical.
- Since this product cannot exist without the Raspberry Pi, we as a company have to make sure that we are meeting all the ethics code that the creators of Raspberry Pi set out themselves. The reason for this is that they can easily ban our product from being run

within their OS or they can simply sue as well if they feel like we are in any way causing harm to the user or the product itself. To avoid that, we have to make sure we are abiding by all their rules, legal and ethical. We have to meet their standards and maybe even go beyond them to be able to continue selling our code. While we cannot control what the users individually do with the Raspberry Pi, we can control what the code we sell does to the Raspberry Pi.

- Since we are selling code, we have to make sure all of our programmers are writing their own code from scratch. Furthermore, if they are using someone else's code for anything, they must ask the original creator of the code permission to use it within our product. If the person does not give permission, we do not use it within our product and try to find alternative ways to solve the issue. Stealing code is extremely unethical and can even lead to legal issues as well. Because of this, our company has to make sure the code is actually being written by the programmers instead of being stolen from somewhere else.

Intellectual Property Issues

- Since the product requires a Raspberry Pi and a touchscreen to work, the only thing we can copyright, patent, trademark, etc. is our code. This means the consumers can buy our code, and they will get a program that they can download onto the Pi and run it there. The issue here is that the Raspberry Pi is the main key to our product, and we do not own the rights to this product. Because of this, we open ourselves to the owners of Raspberry Pi to sue us and claim our product is harmful to their product; furthermore, they can ban our product from being able to be run on the Pi as well. While the source code, which is our product, is protected by intellectual property, the inclusion of the Raspberry Pi isn't.
- A huge issue with selling just code is that it is really hard to prevent someone from just copying the code for their own use. Not only that, it is even more difficult to prevent others from editing your code once they buy it. Because of this, we potentially run into some intellectual property issues since a person can buy the repository and they themselves can sell it to others without us finding out if they do it on a small scale. Not only that, if users start editing the code, it can lead to our whole product being jeopardized. While intellectual property does protect us from a lot of things, it is almost impossible for it to cover these scenarios, which can make our product less profitable.

Change Log

- Made changes to the "final project design" to match all the progress we have made. Not only that, updated it to make it more specific now that we know what we are doing more. Updated our "project milestones" as well because we have a clearer idea on what we want to accomplish. Everything else stayed the same for the most part.