# Analysis part

## Problem Identification

A common issue many people who own houseplants encounter is what houseplant is best for them. The reason this is such a large issue is due to the fact that for many plants, if one aspect of how they’re cared for changes, it could damage them irreversibly and even kill them. That is why as a problem I have identified a need for a programmed system that can identify the best plants to buy for a certain room depending on the needs of that room such as if it’s a kitchen, bedroom or bathroom or based on how able the user is to care for the plants or even the light level of the position of the plant.

## Initial research

To initially research this project I bought a book that I had first bought when using this idea for my Extended Project Qualification. This book is called How to Grow Fresh Air and details many methods for growing and caring for plants, but most importantly it quantifies many of the attributes of plants as well as giving them an overall rating. These attributes include things like air cleansing properties, ease of care and transpiration rate. This data is all originally from NASA and therefore incredibly reliable. From this book I shall create my data structures for my program.

The other research I did was for the UI of my project, there are two direction I went with this, through apps and through online quizzes. In apps these were very streamlined with a huge number of different features, one notable feature was taking a picture of the location to measure light levels and while I don’t intend to implement this it is an interesting idea I would explore with more time. Through quizzes I was a b le to see a much more simplified UI that focused far more on user retention and so it gives me more of an idea of what to do if I need to expand my criteria.

## Computational methods

The problem of a plant recommendation system and calculation would require a large system of databasing because it would only be able to run if it can retrieve the plants. Over the past year in class, we have been learning about databasing and applications of SQL and I think that this problem is a perfect example of the requirements of databasing because although the plants could be stored in a class system, it is far easier to just store it as a database due to the ability to edit and view the raw data. An obvious requirement of databasing would be to ensure that the data meets the 3 normal forms we have learnt about in class.

Another computational method that’s applicable to solving the problem is the use of GUIs to present data, the biggest advantage of such a system is being able to control what the user sees and then present it in a concise way, when the system does the calculation to find what plants to recommend, an open ended SQL query in the command line isn’t how I’d want to do it and instead would want to hide what the user sees, making there a need for a front end GUI and background calculations. The method I will use for my GUI is the tkinter library for python because it has simplistic methods for creating buttons and formatting images as well as the fact I have some prior experience with it from creating a rock paper scissors game on it in class. The way it allows you to put your GUI design inside classes is especially helpful when pertaining to the deconstruction ideal for computational techniques since it breaks down different objects made into methods of a class.

Another vital part of this software will be image representation for the plants shown to the user so they know what they look like, for this I intend to use the BLOB library for SQLite (Binary large objects) because it allows for fairly easy binary representation for media which the python code can easily retrieve

Another thing applicable to solving the problem is the various implementations of computational techniques I will use. For example, abstraction (the act of removing unnecessary data to the point that the only data available is vital for solving the problem) is helpful because the program does not need to know the name of the user or what the plant looks like in order to process calculations, overall increasing the efficiency of the system. Another vital part of computational techniques and thinking is deconstruction (reducing a problem into its constituent parts). This will be used for the problem by implementing object-oriented design, allowing me to break down various parts of the problem into easily manageable class structures. I will also use deconstruction by separating the GUI, backend and database parts of the problem because this means if I change one part I won’t have to change the entirety of the code.

## Stakeholders

The clients for this software range in huge ways, from plant experts looking for reassurance with how to care for their plants, to prospective plant owners looking to start owning houseplants and are in desperate need of advice.

In order to accommodate for this my chosen stakeholders are taking one individual who has never owned a houseplant before as well as someone who has amassed a diverse collection but still needs some tips to stop them from dying as seasons change.

The most important aspect for testing if the software works is if a recommended plant and position actually survives and thrives according to the program. The reason this is convenient is that one of my stakeholders, Erin Campbell, lives in an attic room at the top of her house so all attributes of the room are easily controlled such as the temperature, wind flow and light levels due to the light coming from very controllable sources.

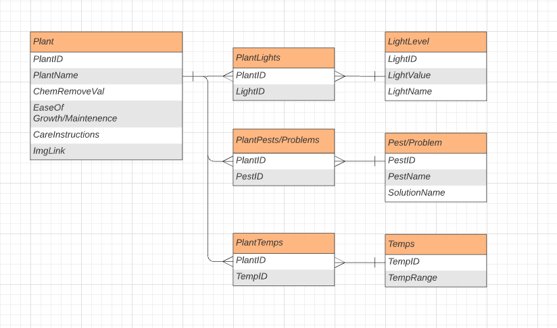
The convenience in the other stakeholder, Matthew Dodd, is that he does not know much about houseplant, which then tests the other vital part of the software, accessibility. By having someone who doesn’t know much about the terminology or the methods, the program will have to be understandable by Matthew.

## Potential limits of the solution

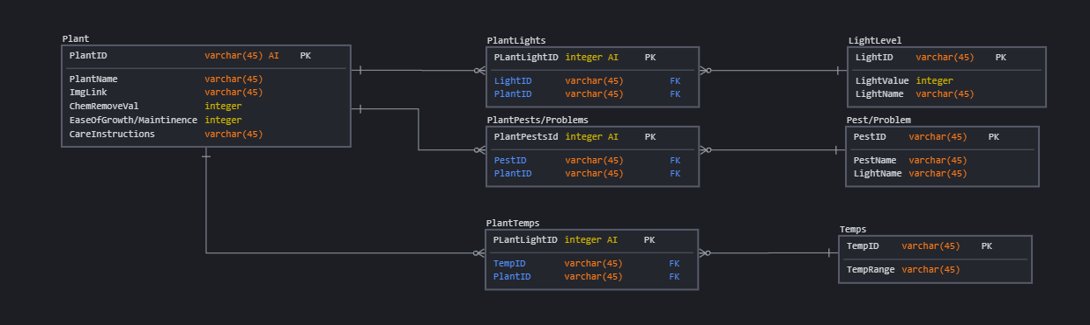
I do not intend to extend more by using AI or connecting the user to a seller of the plant, which is something I’d do if I had more time. There is also and issue of if not enough plants are included it would create a shallow dataset, so a lot of time has to be spent in actually creating the database.

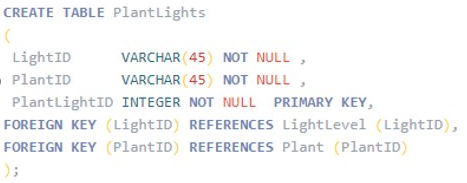
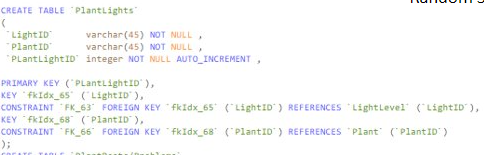
# Development Process

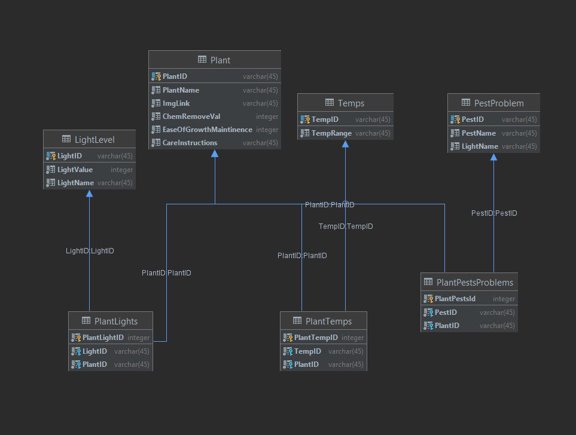
## Databases

The first part of my development process was to construct an initial database, this was in order to understand what my later coding elements would be since the database is the most unchanging part of the system.

I did this by using the book *How to Grow Fresh Air by* B.C. Wolverton since it essentially contains a database of 50 plants with corresponding attributes and diagrams. Form this I was able to decide on my plant attributes as: Name , Rate of chemical vapor removal, ease of growth, care instructions, ideal light level, ideal temperature and common pests and problems.

I then started constructing this database using an online creator called DBSQL which is a website where one replicates a database diagram in MYSQL and then it gives the code used to make the diagram into an actual database, this was extremely convenient because it gave me a baseline for how I wanted my final database to look. Shown below is how the diagram looked in DBSQL. 

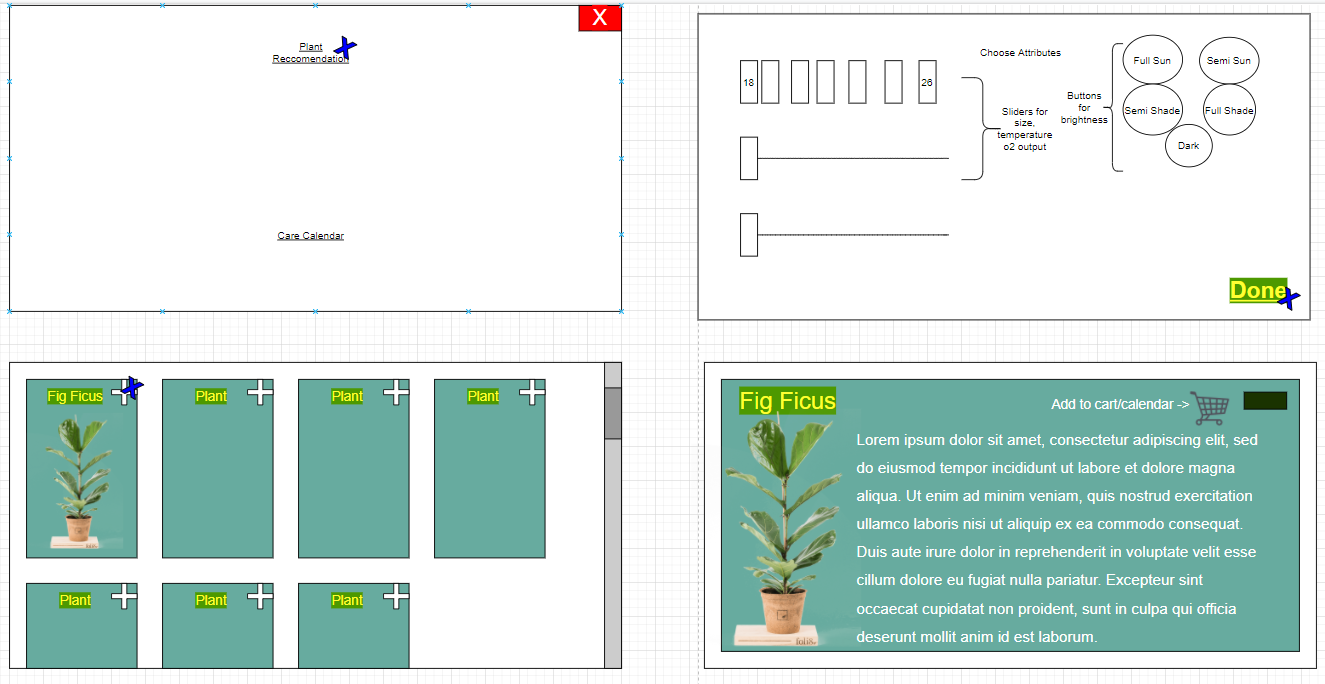
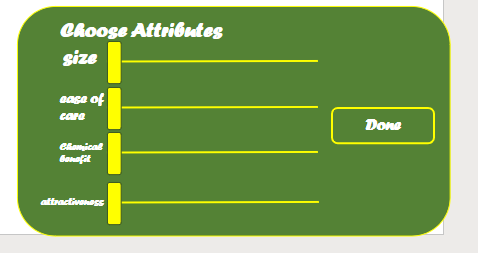
Following this I imported the code into PyCharm and into my main project file from GitHub, but I soon realised the issue with my use of DBSQL which was the fact that the database created was in MySQL and not in SQLite, the optimal formatting for integrating with python, luckily the two formats are very similar so all I had to do was change the code in an online IDE from MySQL to SQLite. As shown by these two pictures both coding for the same table, PlantLights. On the left is it in MYSQL and even though they are similar pieces of code, the defining of constraints was something I was not familiar with, which is why I then converted them into SQLite as seen on the right.

Converting to SQLite made the database work inside of PyCharm meaning I was able to create a sufficient database diagram as seen to the left. Unfortunately after creating this I was told that it would be easier to create and manipulate the whole database in SQLalchemy, the integrator I was using the manipulate the data in python, than by bringing the data into the SQLalchemy module, manipulating it and then sending it to the main python code. Whilst telling me this my teacher also let me know that some parts of my database were sloppy and I did not need as many link tables. I subsequently used a template for my new database which I had used previously in my A-Level and produced what is shown below in SQLAlchemy. On the left is my base tables being made and on the right is my link table that joins the two together.

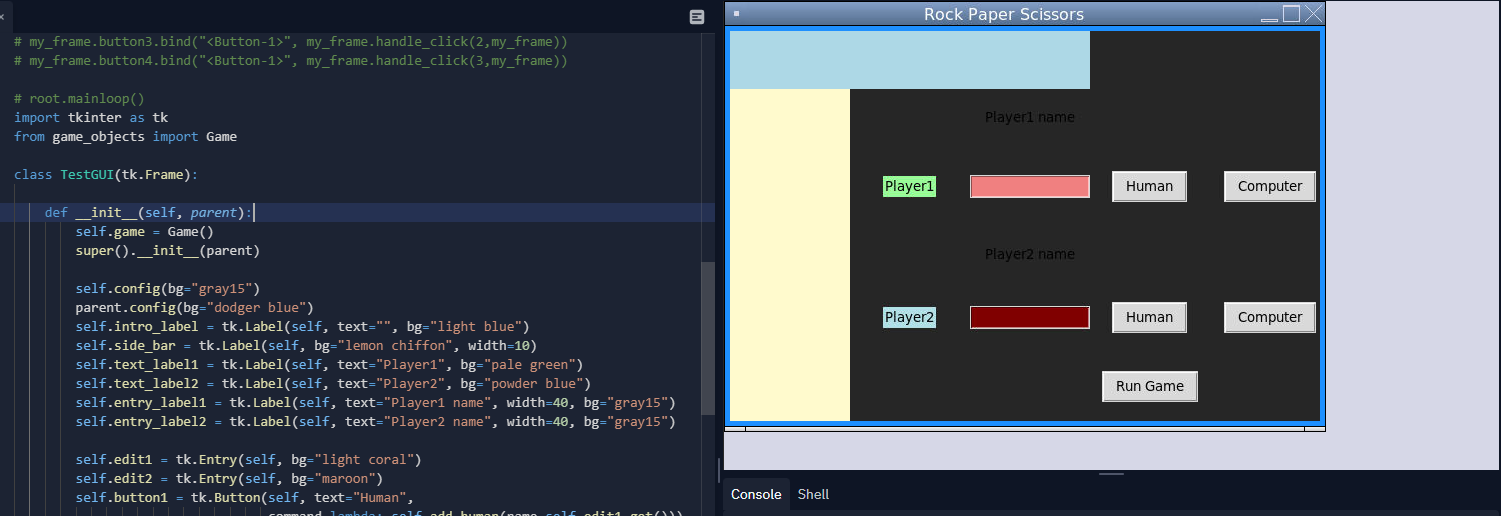
## GUI

After reaching this point of my database creation I decided to leave it for a while and start to work on my other parts of code in order to know what my calculations will require as well as needing a break from the rest of my code. As such I started to deign my GUI so I will know what I need from the processing code.

I did this by initially creating a design in an online program called Draw.io which I had previously used to construct the database diagram as well as some flowcharts for a previous task. It was really convenient because I was able to make a simplistic layout of the GUI without having to code anything. I then set out an goal GUI in PowerPoint because of the ease of manipulation it provides, essentially constructing a series of screens which I’d use once the program was complete just for aesthetic.



 From these digrams I had decided on the design of my GUI and began to code it, I initially started by using an old Tkinter project designed to construct a rock paper scissors game in order to get back into using Tkinter again since this had a very simple frame and grid setup. I realsied quite quickly that this was ineffective and simply bad code but at the same time it worked as a good start point.

After this I started actually coding by creating a placeholder plant class which would act as my databse inputs that the GUI would take. I used object oreinted programming because that’s how it would be retrieved and allows me to easily refactor code by only changing parts at a time. At the same time I created a User Inputs class in the same file which determined how the inputs of the user would change in order to save them and edit what the reccomened plant list was off these values.