CENTRE NO. 51427

PC BUILDING TOOLKIT

A LEVEL COMPUTER SCIENCE PROGRAMMING PROJECT

CANDIDATE NAME : BHUPINDER DHOOFER
CANDIDATE NUMBER : 2337
CENTRE NAME : UPTON COURT GRAMMAR SCHOOL

TABLE OF CONTENTS

ANALYSIS

DESCRIPTION OF PROJECT

PROBLEM IDENTIFICATION

Idea Type	Idea	Explanation	Language	Complexity
Booking/management system	• Cinema booking system • Restaurant booking system • PC building toolkit	A system that will allow a user to book movie reservations in a cinema, tables at a restaurant, seating plans.	Python Use Tkinter or pyqt to make GUI desktop applications with features such as dialogue boxes windows and buttons.	Seats can be recommended to a user, such as front or back row seats etc. Databases can be used with lots of data.
Game	• Snake • Monopoly	Allows a user to play a game.	Python Use pygame or pyglet to make interactable games.	If player vs computer, then artificial intelligence to predict moves
Simulation/modelling tool	• Projectile motion • Crowd control • Traffic simulation • Safety checks • Home designer	Allows a user to enter data and see how it is affected by other things, such as gravity.	Python Use pydynamind or SimPy to make simulations	Can be very difficult if lots of situations need to be explored.

PROJECT IDEAS

I started with three ideas, the first being a booking/management style application. Potential ideas would include a Cinema Booking System and Restaurant table seating application. These types of applications would allow a user to book movie/table reservations, with the ability to consider the number of people, ages of people, and personal preference for recommendations, such as window seats in a restaurant, or back row seats in the cinema. The complexity for this will come from these recommendations for movies, seating or times to visit. Permanent databases will also need to be used to allow for multiple bookings, and to manage the bookings available, for example, to prevent a cinema from being overbooked.

The second idea type was a game. Games could include a monopoly or snake style game. These games could have a lot of features included, such as player vs player, player vs computer, or a combination of both. The complexity would come from the game strategically making moves if player vs computer. This would have to use algorithms or machine learning to predict the moves of the player and adjust appropriately.

My final idea type was a simulation/modelling tool. Ideas would include projectile motion, crowd simulations, traffic simulations, risk identifier and home layout designer. The way that these applications would work is with the user entering parameters/data, such as the acceleration of gravity and the mass of an object, which the application processes with an output, such as the weight of an object. The complexity of these applications would come from the large amount of physics required or with the integration of hardware, as many situations would need to be explored.

Of these ideas, I decided to create a PC Building Toolkit. This would follow the first idea type and the complexity would come from an intense use of a database and recommendations from algorithms. The reason for choosing this idea to create is that there is a large community of potential users, with limited existing solutions, which has some missing features which I hope to include. This means that this is a solution to a real problem that could be useful for many people.

I have chosen not to create a game as a lot of market research will be required to make a new game, and there is no reason to recreate an existing game. I will also not be making a simulation tool as the scientists/professionals who would require such software would be able to create their own, suited to their own needs, meaning that this would not be useful for many people.

MY CHOSEN PROJECT

Due to my hobby of tinkering with electronics, I built myself a PC. Eventually, friends and family, along with other clients became interested in the craft and would ask me for advice on what parts to choose and have me assemble it. As fun as it was, clients would go through many iterations of parts lists, consuming a lot my time, especially if they later decided to not go ahead with the build.

Building a toolkit that helps users choose parts for a computer build will allow me to help clients choose what components they want unsupervised, as my program will identify all possible parts and ensure that they are compatible.

The final program should have a Graphical User Interface, with an effortless way to choose parts. It should also have the ability to personalize the build as much as possible. For example, people may want multiple graphics cards or hard drives, so the program must support this. Warnings may appear if it is not easy/not manufacturer supported to do something, such as 5 HDD's in a case supporting 4, as it will still fit, but will have to rest on a panel.

The backbone of the project will be an SQL database integrated into the program, with the ability to make changes, such as adding new parts or changing prices. Depending on time, I may add the ability to use live prices and use links to components, making it easier for the end user to buy.

This project is suited to a computer program as it is the only convenient, quick and straightforward way for inexperienced users to build a PC. Without a computer program, a user would need to learn the basics to choosing components, and then find the components in store. This could require expert help, just to ensure that everything will be compatible. This can easily be achieved with a computer program, which can also be updated when a new component comes out.

THE USER

The user for this PC building toolkit would be anyone looking to build their first PC. When a person wants to build a PC, they must purchase parts, which requires knowledge about what part is compatible with what. This can take a lot of time to research, as there is a magnitude of combinations available. This program will simplify the component choosing process as the worry of returning incompatible parts is removed entirely. From personal experience, the program must be usable by a 14 year old (which was when I started to learn to build a PC) though most enthusiasts would have built their first PC by 18 years old.

COMPUTATIONAL THINKING METHODS

The best way to solve my stakeholder's problem will be to create a computer program. This will require computational thinking methods as they will make it much easier to develop a working solution. I have included examples below.

Thinking Abstractly

Although a lot of details will need to be considered, many details can also be simplified. By using abstraction, it will be a lot easier to solve this problem with a computer program.

- Certain features of a product can be ignored in the database as they will not impact compatibility.
- Multiple variations of a product do not need to be included, e.g. different colours.
- Costs or performance values may not be needed (depending on stakeholder needs).
- Product personalisation may not be needed at this point (e.g. sticker locations).
- Stock images of components may be used instead of the actual one, as it will reduce the size of the program. This is better than having no images as it is more user friendly.

Thinking Ahead

Planning the steps to completing a project will make it easier to gain a successful outcome.

- I will use python due to my current knowledge with the language.
- I will create sketches for approval before creating the GUI.
- Pseudocode and flowcharts will make it easier to program the project correctly.
- I will consider the best ways of inputting and outputting data, e.g. touchscreen or keyboard and mouse.

Decomposition

All problems will be broken into smaller problems which can be solved more easily.

- Displaying the GUI correctly will be solved independently to ensuring compatibility between selections.
- I will create the GUI one page at a time, to prevent complications.

Thinking Concurrently

The program must be able to decide on the importance of various components.

- A user may be advised to change incompatible RAM, rather than the CPU and motherboard.
- The time and space efficiency of the program should be considered

Thinking Logically

The program will need to be run in a certain order.

- The user must select parts in the correct order. For example, they should choose their CPU before their motherboard, as the CPU is usually more important.
- The program should handle errors correctly, e.g. if a user does not select a part, assume they do not want to select one.
- Use flowcharts and pseudocode to plan ahead.

Backtracking

It is important to save all changes made during development.

- By saving the program whenever it is modified, it is easier to find previously working iterations. For this, it is important to add updated comments whenever a change is made.
- If I decide to abandon an aspect of the program, it will be easy to do by backtracking a previous code.

Pattern matching

It is often very easy for a computer to handle and manipulate large amounts of data. This could bring various functionalities.

- If a certain build becomes popular, the program could recognise it and recommend it when a new user starts to build a similar one.

Conclusion

All the above are reasons why a computer program will be the best way to solve this issue. If a computer was not used, additional employees would be needed for the stakeholder, who would require advanced knowledge. This means that a lot of money is spent and often, no profits would be brought in if a customer decides not to build the PC. A computer only has an initial cost and is always available and scalable. It is also more organised and structured, making it a perfect solution to my stakeholder's problem.

CHOICE OF PROGRAMMING LANGUAGE

The language that I will use to create the PC Building Toolkit is Python (3). The reason for this is the large number of libraries and modules available to use with it. This means that I will not be limited to the features I can include in my application. For example, I could use Tkinter or pyQT to create the GUI desktop environment and SQLite3 for managing databases. Python could have also been used for the other ideas, such as pygame or pyglet to make games. Pydynamind or Simpy could have been used to make simulations for the other ideas.

Another language that I considered was JavaScript, for its compatibility with many platforms including Android, iOS and Windows, however, I decided to use python as it has the ability to do everything required in this application, and I have more experience with it.

SOFTWARE DEVELOPMENT METHODOLOGY

Agile development will be used as it will allow for constant communication between the developer and the stakeholder. By creating multiple iterations of the program, the stakeholder will be able to get exactly what they want, in a relatively short time, as with every iteration, you learn from the previous. Using the agile development methodology will also allow for the stakeholder to change what they want, without having wasted a lot of time.

STAKEHOLDER

DETAILS

Arka Roy, the primary stakeholder, is the owner of a local electronic retail and repair shop, who occasionally gets orders for custom PCs. He plans to use this software to save time, allowing the user to choose what they would like, rather than require an appointment to discuss what components are preferred, and risk the user not wanting to continue with the purchase. Arka will provide constant feedback throughout the development of the program, specifying his requirements and preferences when relevant.

REQUIREMENTS

- A program that allows users to choose parts for a computer build
- The program must be easy to use by anyone
- Must run on existing and future hardware (support Windows 10 desktops)
- Ability to make changes after making initial selections
- Ability to print/save selection of hardware
- Provide recommendations and warnings throughout the program
- Have a help feature to support the user if stuck

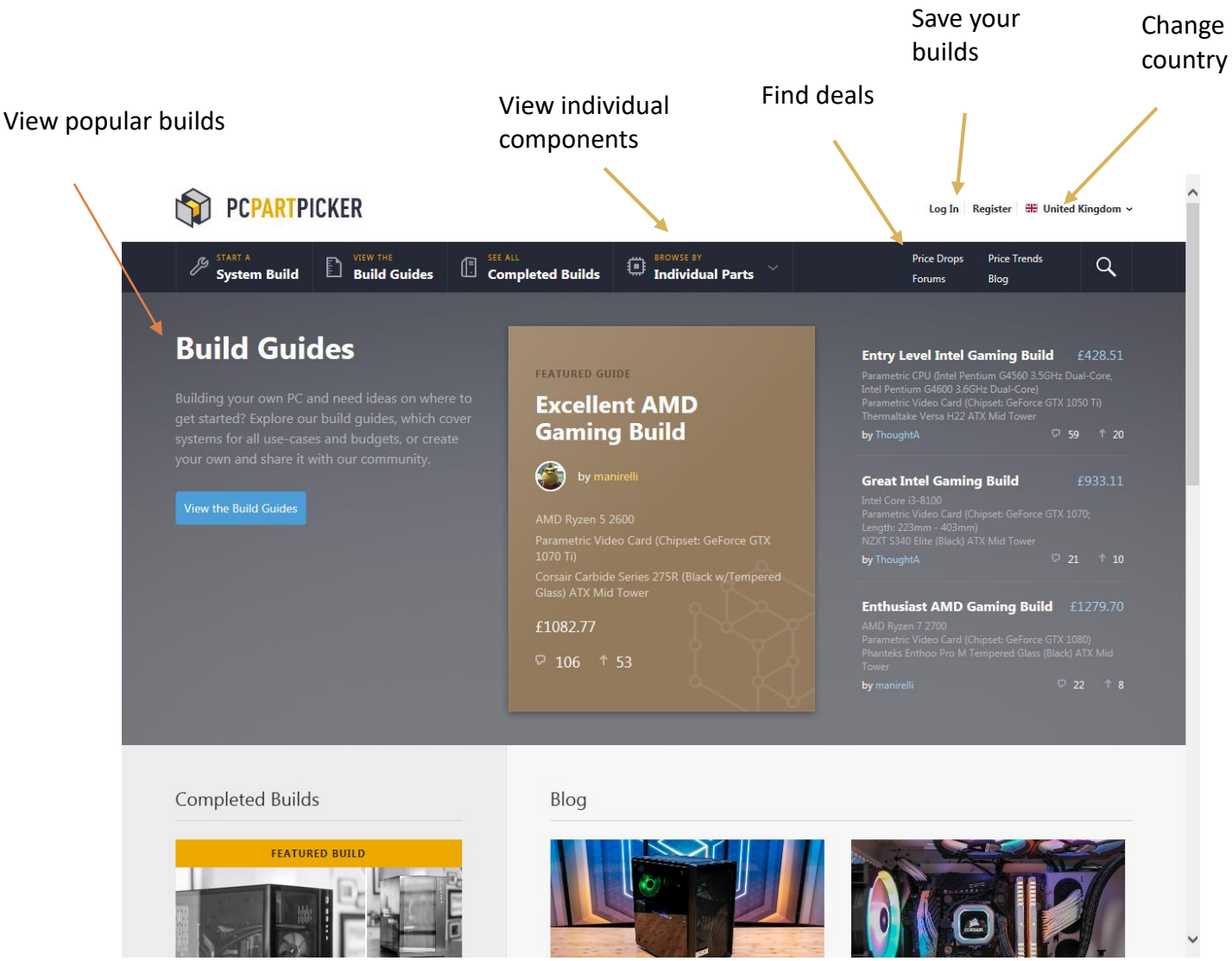
HOW MY SOLUTION WILL MEET THESE REQUIREMENTS

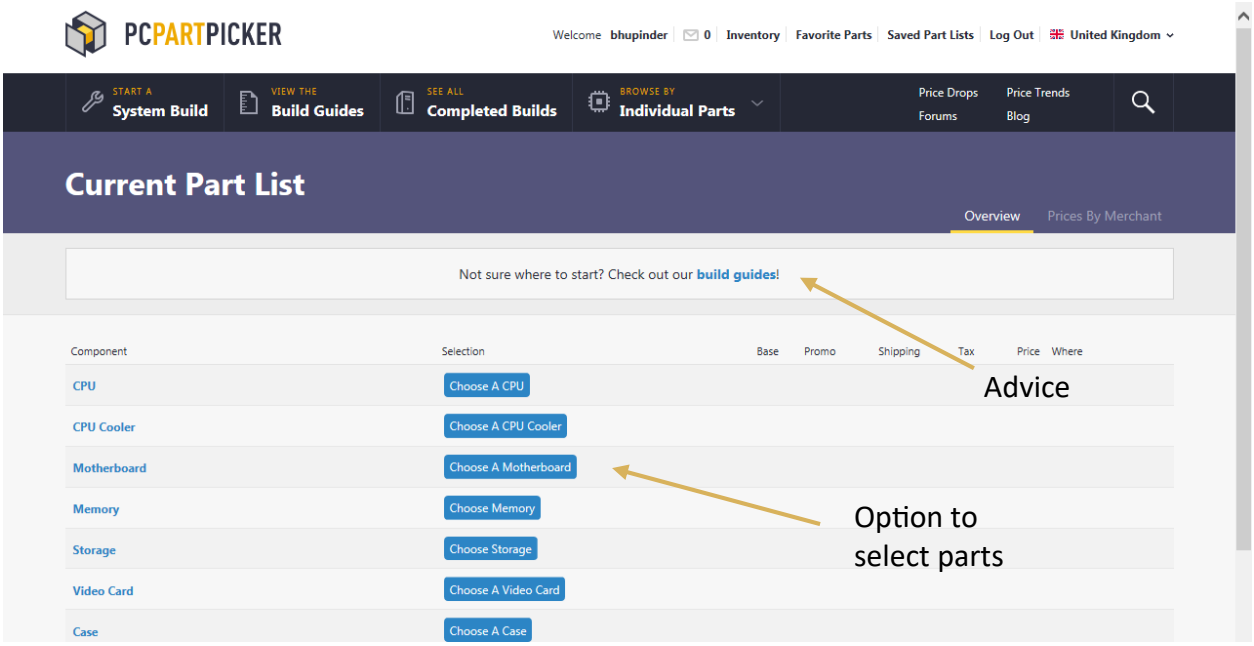
By providing the user with an easily navigable user interface, they will not have problems with using the program its self. This user interface should be graphical, which can be achieved with Tkinter in python. As Tkinter is built into python, which is supported by windows, there will be no compatibility issues with hardware. The program will feature a back button, allowing the user to make amendments to their choices, however, this may remove subsequent choices if not compatible. If an advanced mode is used, it might advise the user of the incompatibility, rather than remove it completely. Though the use of variables, the program will easily be able to print out the contents of the variables in a convenient way, allowing them to be saved to a document. The use of pop-up boxes will allow warnings to be displayed and elements on the GUI can allow for recommendations to be shown. Finally, pop-up boxes can also be used to display advice when requested, which are most likely to be stored in a database/csv.

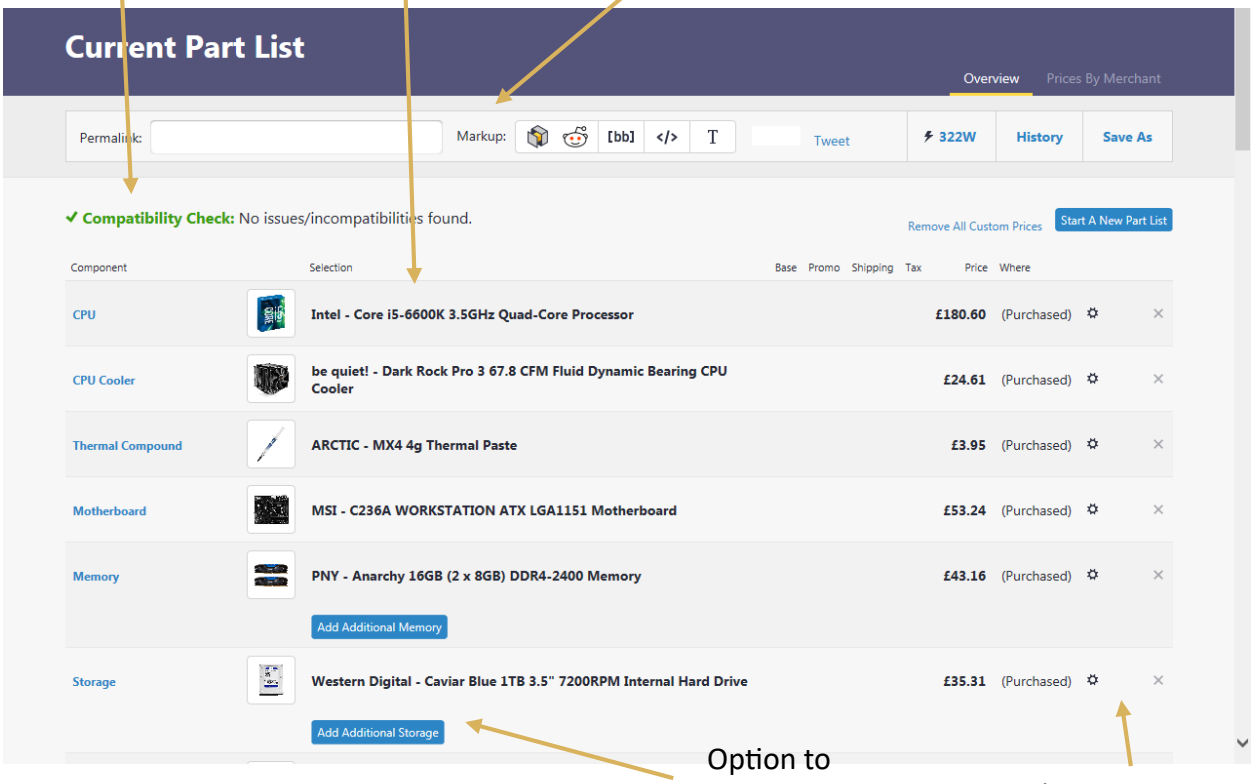
RESEARCH

SIMILAR SOLUTIONS 1

PC Part Picker is a very popular website that allows you to browse for offers on components, as well as build a PC. It does have compatibility checks and makes sure that components will work, however, it does not give advice or show the best components every time. For example, a motherboard could work with a CPU, but not allow overclocking. PC part picker would not inform you of this.

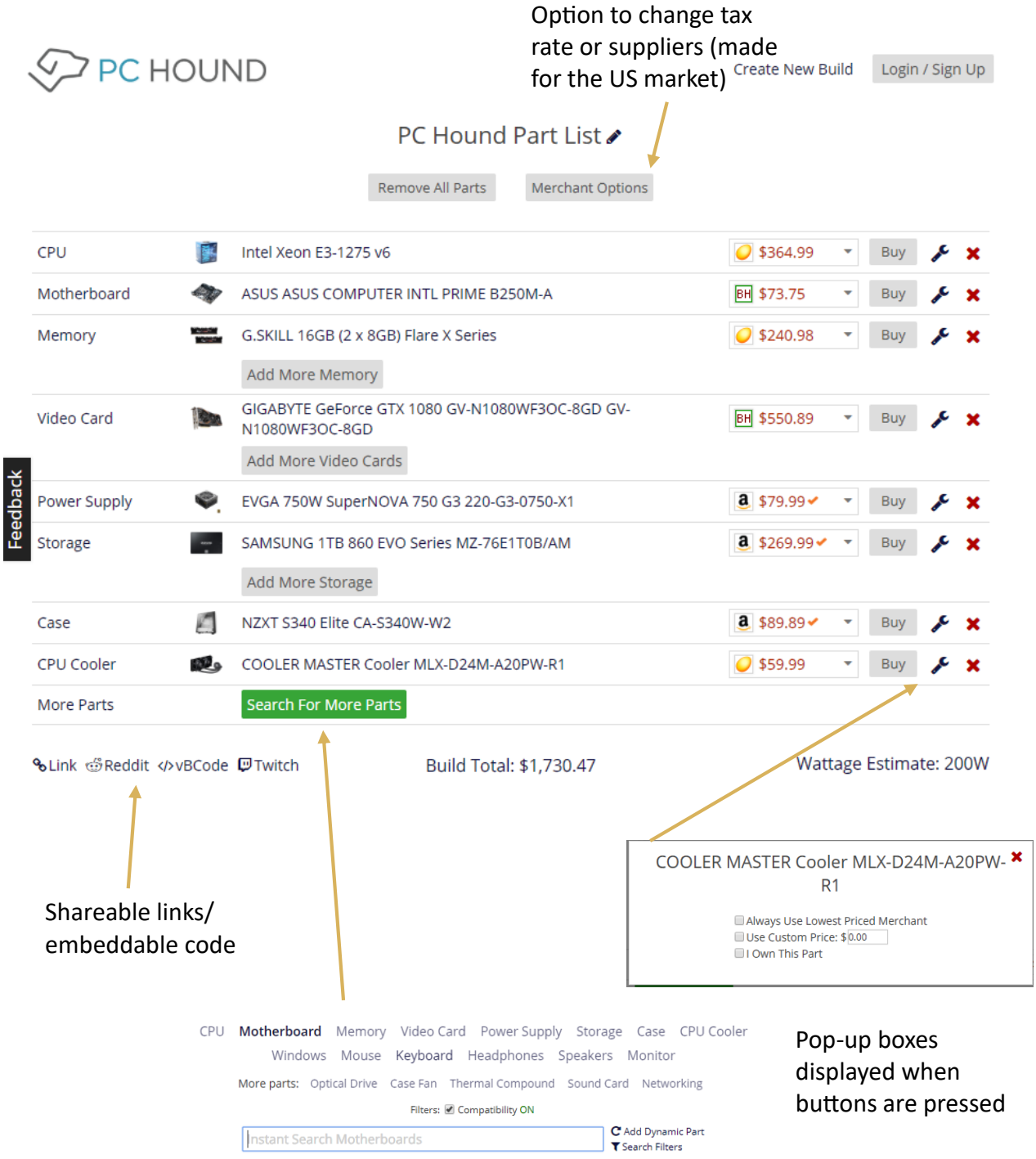



PCPARTPICKER
Welcome bhupinder 0 Inventory Favorite Parts Saved Part Lists Log Out United Kingdom
START A System Build
VIEW THE Build Guides
SEE ALL Completed Builds
BROWSE BY Individual Parts
Price Drops
Price Trends
Forums
Blog
Current Part List
Overview
Prices By Merchant
Not sure where to start? Check out our build guides!
Component
Selection
Base
Promo
Shipping
Tax
Price
Where
CPU
Choose A CPU
CPU Cooler
Choose A CPU Cooler
Motherboard
Choose A Motherboard
Memory
Choose Memory
Storage
Choose Storage
Video Card
Choose A Video Card
Case
Choose A Case
Advice
Option to select parts


Compatibility check
Hyperlink to suppliers
Shareable link
Current Part List
Overview
Prices By Merchant
Permalink:
Markup:
[bb]
</>
T
Tweet
322W
History
Save As
Compatibility Check: No issues/incompatibilities found.
Remove All Custom Prices
Start A New Part List
Component
Selection
Base
Promo
Shipping
Tax
Price
Where
CPU
Intel - Core i5-6600K 3.5GHz Quad-Core Processor
£180.60 (Purchased)
CPU Cooler
be quiet! - Dark Rock Pro 3 67.8 CFM Fluid Dynamic Bearing CPU Cooler
£24.61 (Purchased)
Thermal Compound
ARCTIC - MX4 4g Thermal Paste
£3.95 (Purchased)
Motherboard
MSI - C236A WORKSTATION ATX LGA1151 Motherboard
£53.24 (Purchased)
Memory
PNY - Anarchy 16GB (2 x 8GB) DDR4-2400 Memory
£43.16 (Purchased)
Add Additional Memory
Storage
Western Digital - Caviar Blue 1TB 3.5" 7200RPM Internal Hard Drive
£35.31 (Purchased)
Add Additional Storage
Option to select more parts
Edit price or remove part

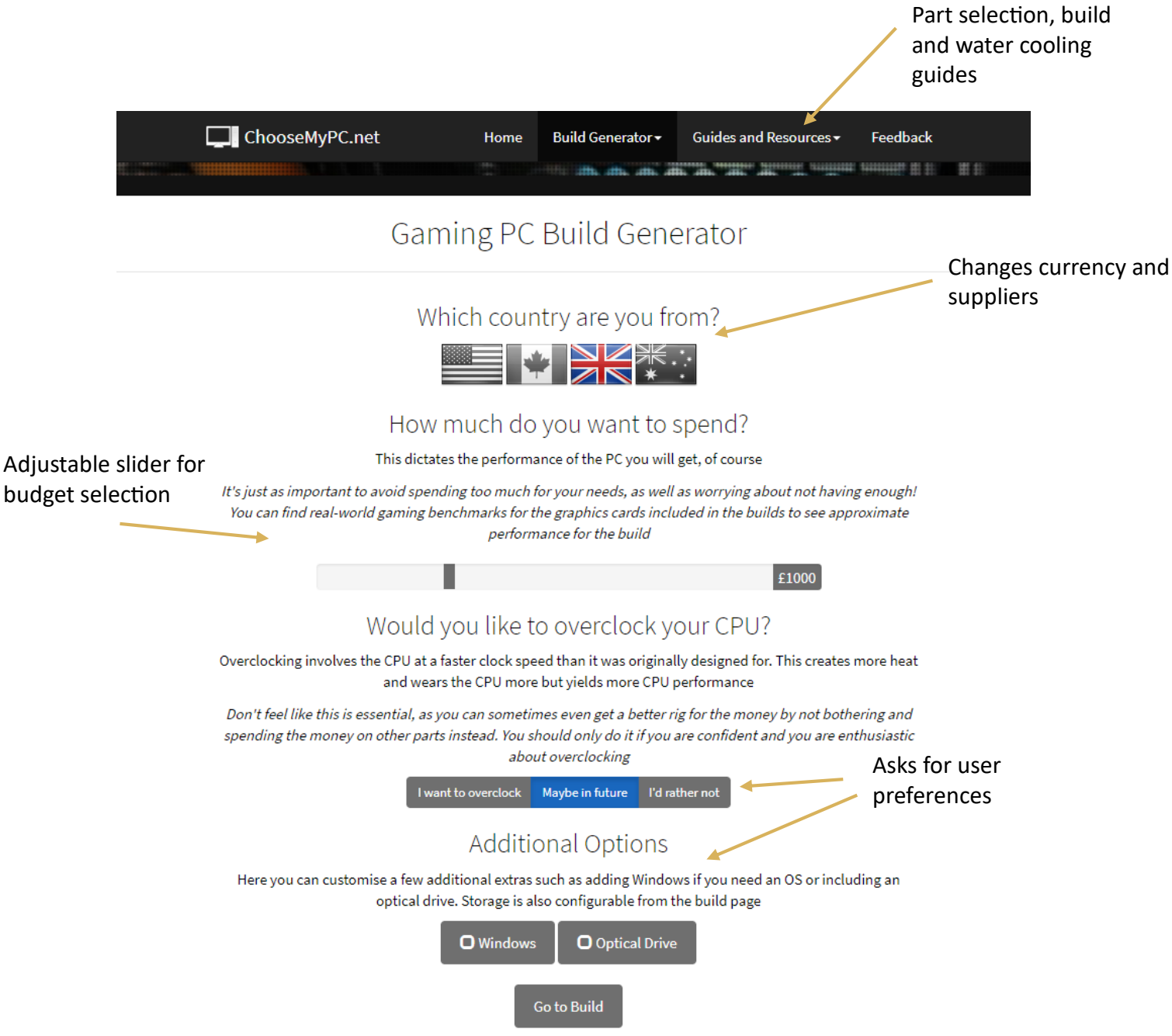
SIMILAR SOLUTIONS 2

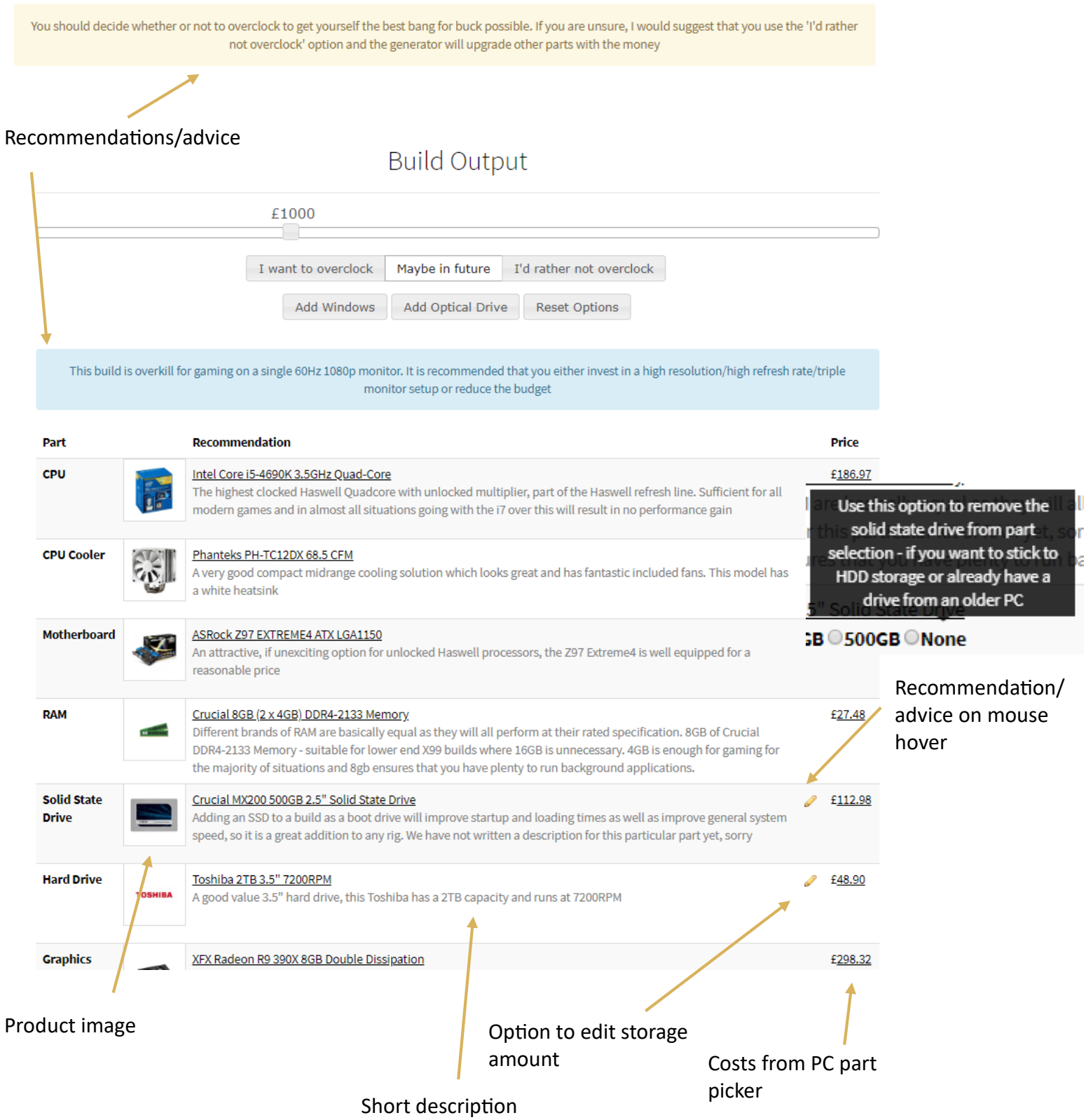
PC hound is a relatively new service, with an easy to use interface, all laid out on one page. It only shows parts that will 'fit' together, but results may not always be relevant. It only considers simple factors such as the CPU socket, but not the chipset, meaning that it will not always work correctly.



SIMILAR SOLUTIONS 3

ChooseMyPC.net is a 'website created to help beginners to pick their parts to build their own gaming PC', very similar to what I am going to create. This service works differently to those above, as it asks you questions, including budget and preferences, and provides you with a recommended list. This is great, however, many items were found to be out of stock, removing the convenience of this service.




You should decide whether or not to overclock to get yourself the best bang for buck possible. If you are unsure, I would suggest that you use the 'I'd rather not overclock' option and the generator will upgrade other parts with the money
Recommendations/advice
Build Output
£1000
I want to overclock
Maybe in future
I'd rather not overclock
Add Windows
Add Optical Drive
Reset Options
This build is overkill for gaming on a single 60Hz 1080p monitor. It is recommended that you either invest in a high resolution/high refresh rate/triple monitor setup or reduce the budget
Part
Recommendation
Price
CPU
Intel Core i5-4690K 3.5GHz Quad-Core
The highest clocked Haswell Quadcore with unlocked multiplier, part of the Haswell refresh line. Sufficient for all modern games and in almost all situations going with the i7 over this will result in no performance gain
£186.97
CPU Cooler
Phanteks PH-TC12DX 68.5 CFM
A very good compact midrange cooling solution which looks great and has fantastic included fans. This model has a white heatsink
Use this option to remove the solid state drive from part selection - if you want to stick to HDD storage or already have a drive from an older PC
Motherboard
ASRock Z97 EXTREME4 ATX LGA1150
An attractive, if unexciting option for unlocked Haswell processors, the Z97 Extreme4 is well equipped for a reasonable price
500GB
None
RAM
Crucial 8GB (2 x 4GB) DDR4-2133 Memory
Different brands of RAM are basically equal as they will all perform at their rated specification. 8GB of Crucial DDR4-2133 Memory - suitable for lower end X99 builds where 16GB is unnecessary. 4GB is enough for gaming for the majority of situations and 8gb ensures that you have plenty to run background applications.
£27.48
Recommendation/ advice on mouse hover
Solid State Drive
Crucial MX200 500GB 2.5" Solid State Drive
Adding an SSD to a build as a boot drive will improve startup and loading times as well as improve general system speed, so it is a great addition to any rig. We have not written a description for this particular part yet, sorry
£112.98
Hard Drive
Toshiba 2TB 3.5" 7200RPM
A good value 3.5" hard drive, this Toshiba has a 2TB capacity and runs at 7200RPM
£48.90
Graphics
XFX Radeon R9 390X 8GB Double Dissipation
£298.32
Product image
Short description
Option to edit storage amount
Costs from PC part picker

FEATURES INCLUDED IN SIMILAR SOLUTIONS

Feature	Use
View popular builds	So inexperienced users can learn about what makes a build good
View individual components	So that direct comparisons can be made
***Find deals**	So individual parts can be purchased
Save builds	For reference at a later point in time
***Change country**	To support other markets
Advice	So an inexperienced user can see what an expert would do
Select individual parts	So the user has a choice of the parts they want to include
Compatibility checks	So an inexperienced user does not make a mistake causing their PC to not work as intended
***Hyperlink to suppliers**	So individual parts can be purchased
***Sharable link**	So a user can show others, e.g. friends, their build
Option to select multiple similar parts	E.g. to select 2 hard drives
***Edit product details**	To change the price of a part
Pop-up details	So that screen real-estate is not wasted if the information is not always required
Budget selection	So only parts within the budget can be selected
User preferences	So only parts that match the user's preferences can be selected
Images	So a user can see what the part looks like
Descriptions	So a user knows what the part does

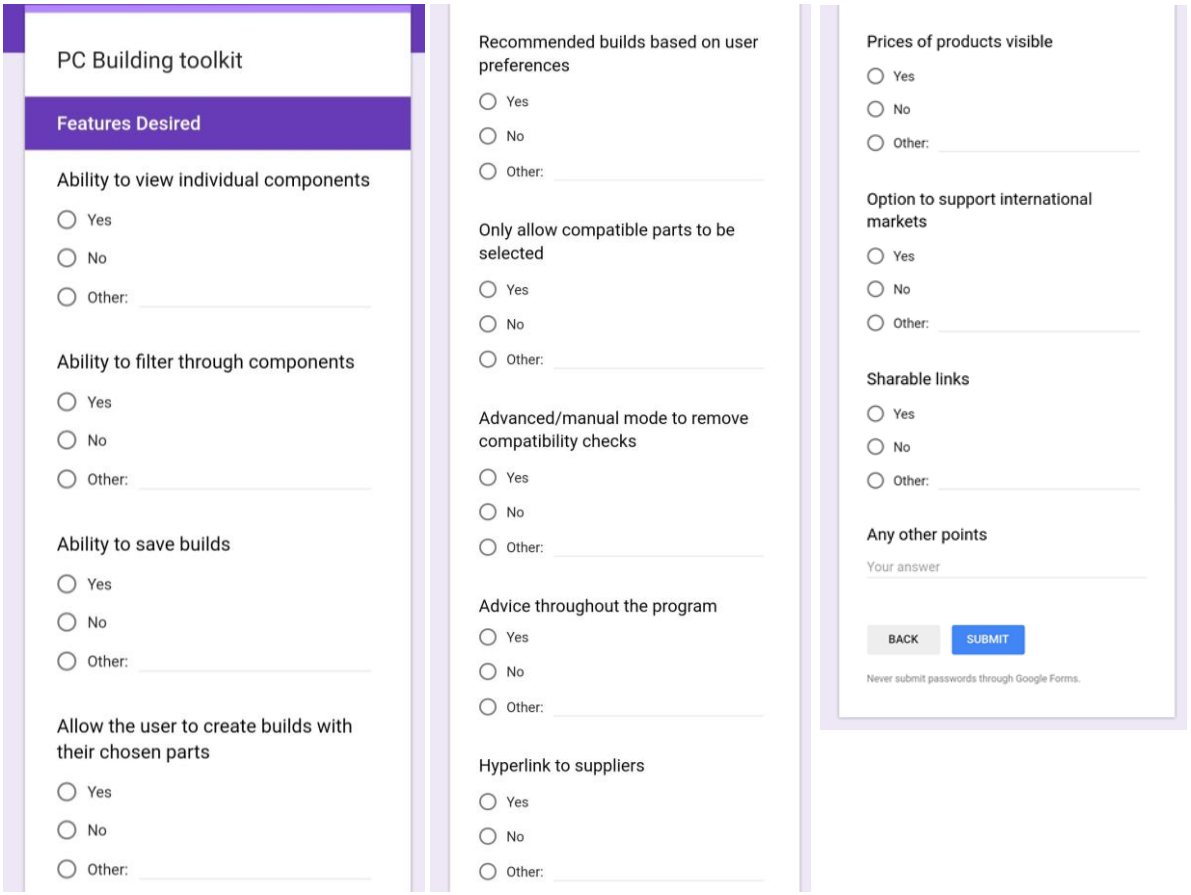
*features that may be irrelevant

These are all the features included in existing solutions. All of these will be presented to the stakeholder for their opinions. Some of these may not be relevant for our program, such as hyperlinks to suppliers, as it would not bring in business for Arka.

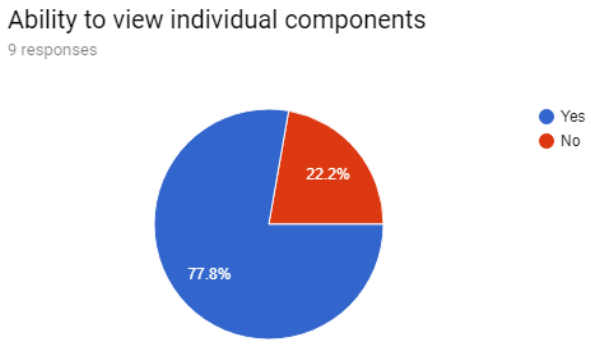
USER RESEARCH

It is important to find out what features users of this program would like to have, and the opinion on the stakeholder will also be very important. For this reason, I will conduct an in-depth interview with the stakeholder Arka Roy, as well as send out questionnaires for selected employees and regular customers to fill out. An IDI will allow me to ask several questions, along with any follow up questions at the time allowing me to gain a lot of highly valuable data. However, an IDI requires dedicated time from Arka and I, which can be difficult to arrange. Sending out electronic surveys, rather than using a focus group will allow me to easily and cheaply gain large amounts of data, without the negative effects of peer pressure or distractions. The disadvantage with this is not everyone will submit a response, and some responses may not be truthful.

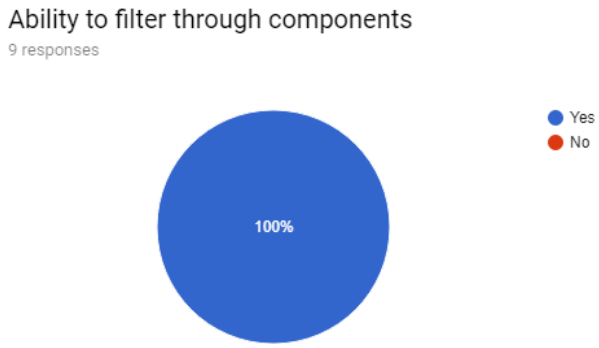
These are the questions that will be asked in the online survey, made on Google Forms.



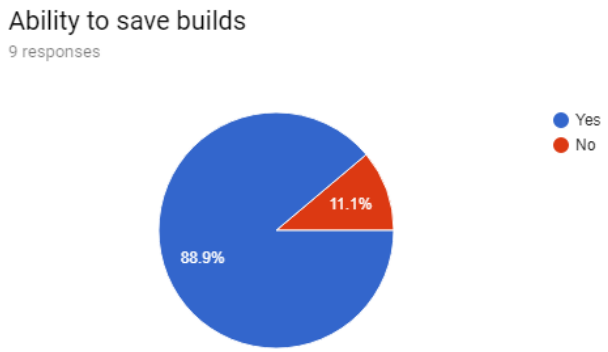
The results can be found below.



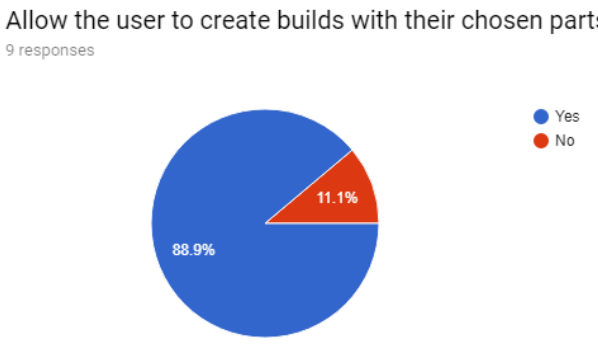
Most people would like to see the different types of components available for their build (e.g. brands and storage options).



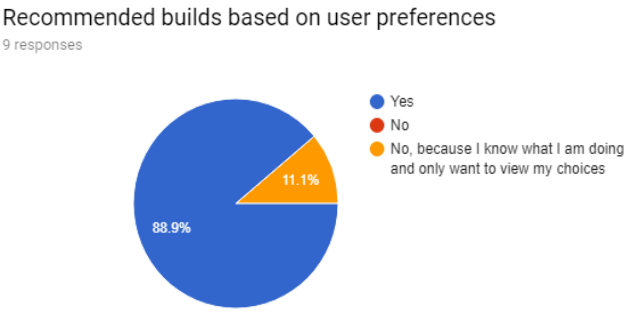
This shows that everyone would like to view components with specified features (e.g. Black or 4 core).



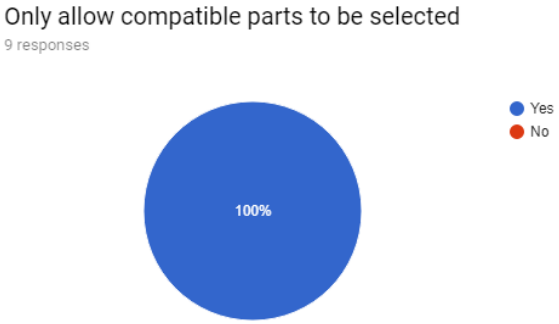
Users would like to access their builds at a later date.



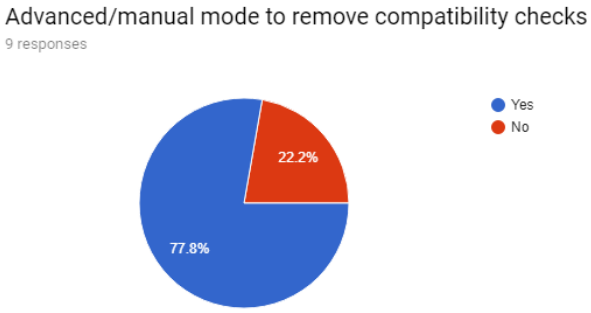
People would rather select the individual components rather than use an automated service, such as ChooseMyPC.net.



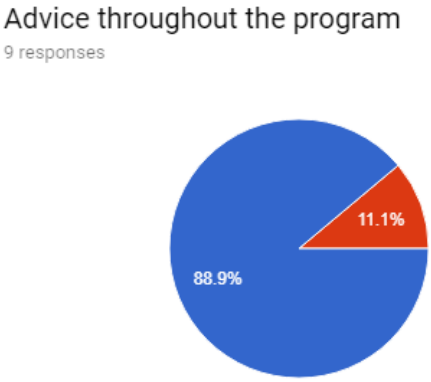
Considering the previous question, users would like to recommendations but not as the primary way to use the service.



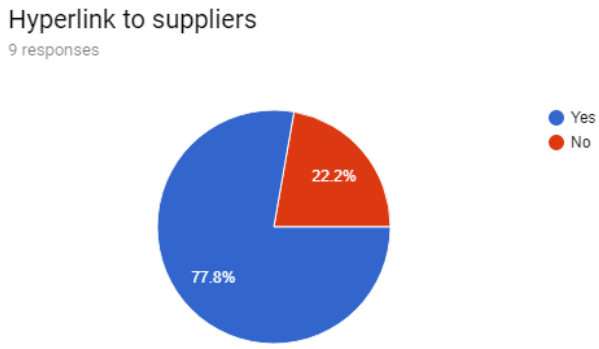
A compatibility checker could be the unique selling point.



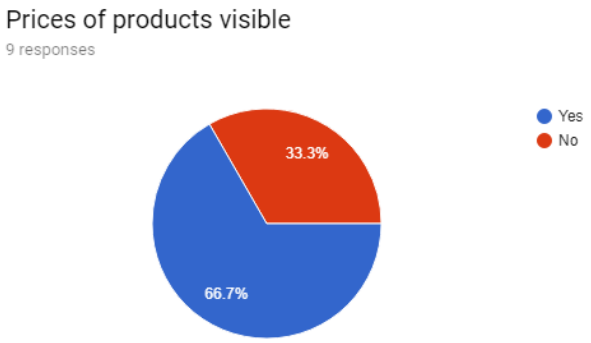
More advanced users may want to disable the compatibility checker, and this should be easy to implement.



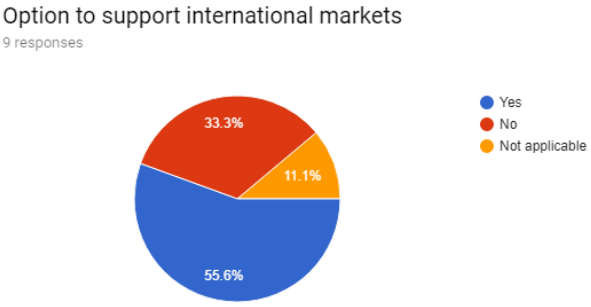
Users are open to using the application for reasons other than choosing parts for the PC. This could include instructions to assemble the PC.



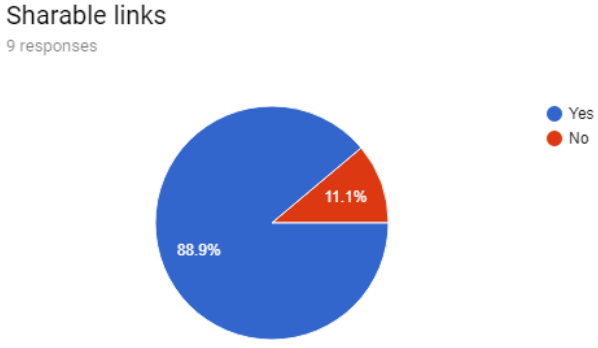
Most people would like to see current sale prices for components, but this might be a feature Arka would like to exclude.



More people would like to see the price of components. Arka may not like live prices to be used but possibly his own values.



Customers would rather have features than not, and it may be useful to have multiple languages available to choose from.



Users would like the option to share their builds. This could include a ‘Look at what I bought’ message on social media.

Any other points

> *"Compare the prices of certain products with other sellers and have a forum page so I can submit my build. "*

This is a good idea, where we could compare our service with a similar one, such as PCSpecialists, who would be a competitor to Arka's business.

> *"It should have the option to add personal custom parts to the pc build. "*

This could be implemented through a feedback or notes sections, where requests could be made. This would allow for unlimited customizability.

> *"Allow me to complete several builds at a time and compare them against different statistics (e.g. FPS in certain game)."*

The first part would be possible by creating and saving builds one at a time. We could add the ability to 'compare with your previous builds' to make it easier to view. It may be difficult to give accurate FPS values for certain games due to the number of variables, such as game settings, resolution, etc. However, by implementing a forum, users could share their values. Or an algorithm could be created to place the build on a scale from 'light browsing' to 'quantum computer'

> *"I do not need support as I am a professional and this will only be a tool to make it easy. A manual mode will be very useful, and having tips everywhere could get in the way"*

Overloading the main purpose of the application could be a problem, so the design should be well thought out.

> *"Though I will be getting the store to build the computer, guides may be interesting to read. I do not want to view the prices of components as it may make me fill like I am spending too much money."*

It should be easy to implement build guides. For example, we could video record building one, and have a link to it in the application.

> *"Could it also allow me to think about alternative options to the ones that are given anyway? For example, whilst discussing hard drives, could I have an option to incorporate an SSD"*

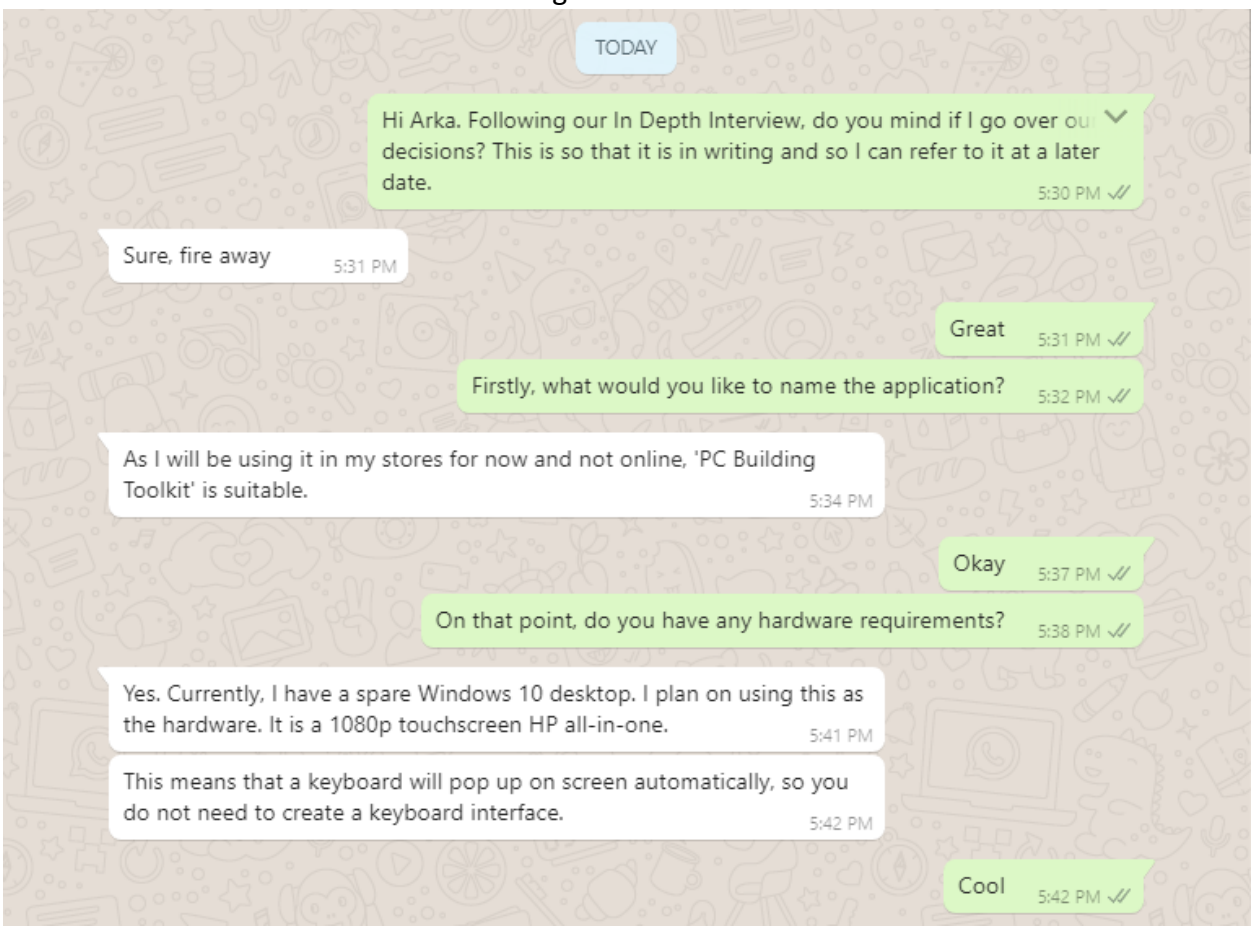
An 'other' or 'add your own' section may be useful for a user looking for something specific.

From this data, we learn a lot. Firstly, a lot of the inexperienced users would like an application that can give them the best PC for their needs and are not concerned about the details such as independent component cost. The other type of user is the advanced user, only using this application for choosing their parts, or possibly just to assemble them. From the comments, we find out that guides and blogs would be an interesting read but are not important for most people. One interesting comment was one user wanting to view benchmarks of PC build, with the frame rate for games with the potential PC.

STAKEHOLDER RESEARCH

Sat
30 Jun 2018 10:30 – 12:30 IDI with Arka Roy

After conducting the user research, we decided to conduct the IDI. In this in person interview, we discussed all the points in the survey, as well as what (other) features we would implement and those that we would not. The following summarises what we discussed and decided on.




However, any buttons should be big enough to press easily, without a mouse.
5:43 PM
I will connect a kb&m for diagnostics and installation though.
5:44 PM
Sure
5:44 PM
Who is your target market?
5:45 PM
The main customers that I get to build a PC are over 16 years old.
5:46 PM
I often get customers looking to buy a PC for a child or friend. This means that the application should be usable by novices.
5:48 PM
Okay. Now to the desired features. Do you want prices to be visible?
5:50 PM
No
5:51 PM
This will make it hard to operate the business correctly. It will be better for me to do this part manually, as it can allow for easier negotiations and to account for other circumstances (such as used parts to be considered).
5:53 PM
Okay. I guess I should not add links to the product either (e.g. Amazon)?
10:16 PM
Correct. If needed, the user is free to use the internet, but it is better to keep them within the application as much as possible
10:17 PM
Do you want to allow the user to export their builds?
E.G. as a PDF or email
10:18 PM
Yes. It will just be a lot more convenient for everyone. Also, I would like to save all builds created on the application to its own database.
10:20 PM
Okay. Are you aware that this may breach the Data Protection Act?
10:22 PM
I could add a pop up notification making the user aware.
10:22 PM
I am aware of this. There is no need for the pop-up. I will only be using this database for analytics. I will not need any personal information, and will include a physical notice mentioning the collection of data. In a future version of the app, we may remove this.
10:24 PM

Sure 10:24 PM

Do you want to include a compatibility checking algorithm? 10:26 PM

Yes, this will be one of the key features of the program. 10:26 PM

Do you want to include an 'advanced mode'? 10:27 PM

Sure, but make sure it is a less obvious option than the standard one. 10:27 PM

Should I include 'tips and tricks' within the program? 10:28 PM

If you have time, then you can. It is not important though as I will be building the PC anyway. It could also clutter the application. 10:29 PM

Should I include recommended builds and guides? 10:30 PM

+44 7484 799915 ~BSD

If you have time, then you can. It is not important though as I will be building the PC anyway. It could also clutter the application.

Same as this. 10:30 PM

As it is not the main function of the program. 10:30 PM

Are you happy to be contacted throughout the development of the program? 10:31 PM

I ask as I am following the agile software development methodology, which includes constant feedback and communication. 10:32 PM

Sure. I would like to be as involved as possible. This should also allow for easy changes of requirements. 10:33 PM

Exactly 10:33 PM

That is all of my questions for now 10:33 PM

Thank you for your time. 10:33 PM

No problem. 10:34 PM

ESSENTIAL FEATURES

Feature	Reason
Start page	Title 'PC Building Toolkit', with the option for a user to enable or disable the compatibility check from user research.
Create a build	So a user can configure components for a useable system after using the program. Thinking logically will be required for efficiency, as pages could be reused multiple times. From Stakeholder research.
Edit parts in the build	If a user chooses to change a component after selecting one, they should be able to. I would need to think logically to ensure this is executed properly. For example, it would need to remove the old component and add in the new one. From Stakeholder research.
Save the build	All builds created must be saved locally, for future analysis. Performance modelling would ensure a lot of storage space is not taken up. From Stakeholder research.
Export the build	So that a user can access and possibly edit their build at a later date. This could be an email, pdf etc and is from the user research.
Components database	A database will all components stored properly. The user must not be able to edit this database. Abstraction could be used here so relevant attributes are stored in the database. Performance modelling would make sure data is found quickly. From Stakeholder research.
Relevant advice	So an inexperienced user knows they are looking at and what should be done with that information. From Stakeholder research.
Recommendations*	So an inexperienced user knows what the best choice would be in a situation. Data mining could be used here to find the best recommendations. From user research.
Compatibility checks*	So that the user does not make accidental purchases for components that cannot be used together. From Stakeholder requirements.
Graphical user interface	To make it easy to use the program. This will be the best way to interact with a touchscreen device. Abstraction could be used here to create the best usable interface. From Stakeholder requirements.
Advanced mode*	So advanced users can make edits and override the compatibility checks. Thinking procedurally will allow for this, as some functions will not need to be run. From user research.
How to Guides and recommended builds*	So users can view popular builds and learn how to assemble the computer. From user research.

*if possible, due to limitations

LIMITATIONS

As this is an A Level project, the main issue is with time. With enough time, my expertise with a language could improve, and any unavailable resources could be created. However, as time is finite, I will have to make the most of it with my available experiences and resources.

The main limitation due to time will be with the SQL database. Due to the extremely large choice of products available, the database will need to be robust and reliable. Any mistakes could cause major problems. Creating this database could take a long time to create, so preferably, an API (a predefined set of functions and procedures) could be imported and used, which will save a lot of time. This API would be connected to a web service and would fetch the required data automatically when called. If this resource is not available, a limited dataset could be used to demonstrate the program, and any remaining time could be used to populate the database. The benefit of this is that it is all offline, and the PC will not need to be connected to a network. Implementing the database into the program could also take a lot of time, as it is complex, but I feel that it can be achieved within the time frame.

Another time limitation could be the recommendations, as advanced algorithms will need to be created and used for a working product. This could also take a long time to create and may not be achievable within the provided time frame. An alternative to this could be to create generic builds that fill most budget ranges.

The final limitation will be my expertise. Despite experience with python, every project has different requirements, which can create its own problems. For example, I will need to learn to use SQL within python. I will also need to find an efficient way for the program to create recommendations.

Requirement	Ideal	Limitation
Components database	Will be full of components representing the whole market.	If a suitable (official) API cannot be found, it will be difficult to manually fill the database in the required time, especially because new components are constantly coming out.
Relevant advice	There is enough information for someone with a basic understanding to use the program.	If there is too much information, it could detract from the main purpose of the application. Also, it will be difficult to answer all the questions a user may have. This could be solved with a Q&A section but will take a lot of time to research and create.
Recommendations	If a user is configuring a commonly built PC, the application could suggest an autocomplete.	This would be difficult to implement and may require the use of AI or machine learning (neither of which I have experience with).
Compatibility checks	The application will account for every aspect of the build.	Most attributes can be confirmed easily, however, features such as colours or dimensions of items can be hard to account for. E.G. A GPU may be too large for a small PC case. This will be hard to check for. This is another resource or time limitation.
Graphical user interface	It will be quick and easy to use.	I have chosen to use Python, due to my current experiences with it. Unfortunately, it is not the best for GUIs, which can lack flexibility and features, such as transition animations. If there is not enough time, I will not be able to add a lot of graphical features, such as images on buttons.
How to Guides and recommended builds	Some information will be available to learn how to build a PC and common builds.	A lot of useful content is readily available online, and it will be hard to make anything better in the given time. However, a solution could involve using pre-recorded videos as screensavers.

SOLUTIONS REQUIREMENTS

FINAL SOLUTION REQUIREMENTS

Design and output requirements

1. Display a Graphical User Interface to operate the program -from user research
2. Large buttons to support a touchscreen input -from stakeholder IDI
3. Easy to read text -from stakeholder IDI
4. Suitable advice and recommendations displayed -from user research
5. All features of the GUI must be labelled and easy to use -from stakeholder IDI
6. A suitable colour scheme must be used -from stakeholder IDI
7. Allow users to filter through components -from user research
8. Allow users to toggle advanced mode -from user research
9. It must save the build for later access by the same user -from user research
10. The user can choose to save their build for personal use. -from user research
11. All builds created must be saved for stakeholder analysis -from stakeholder IDI

Input requirements

12. Must support touchscreen inputs and touch -from stakeholder IDI
13. Must be able to amend a database -from stakeholder IDI
14. Text must be entered as a string
15. Any data must be stored in a variable
16. Each user must enter their name and a method of contact - from stakeholder IDI
17. Each user must be assigned a new CustomerNumber -from stakeholder IDI
18. Allow users to ‘make a note’/leave feedback -from stakeholder IDI

Process requirements

19. Able to create a useable system for a user and their preferences -from stakeholder IDI
20. Use an SQL database, which holds the properties of components -from stakeholder IDI
21. Give advice to the user on what things are in a minimalistic way -from user research
22. Ensure parts are compatible -from user research
23. Uses system resources efficiently -from stakeholder IDI
24. Gives relevant recommendations -from stakeholder IDI

SOFTWARE REQUIREMENTS

- Operating systems: Windows 7 or later, macOS, and Linux
- Onscreen keyboard and touchscreen support
- Python version: 3.6
- All required python modules (which could include SQLite3 (for the database), pandas (for an aesthetic output of the data), Tkinter (to create the GUI) etc.)

HARDWARE REQUIREMENTS

As this program will be made in python, it will be supported across many platforms and devices. However, these are the minimum system requirement.

- Processors: Intel Atom® processor, 1GHz or faster
- Memory: 1 GB
- Storage: 16 GB
- Display: 1080p or above, touchscreen
- Mouse and Keyboard: None (onscreen)

Any system that can support the listed operating systems will have enough resources to run the program. For this reason, the system requirements are based on the Windows 7 requirements. Because concurrent processing/pipelining is not taking place, a single core single thread CPU such as an Intel Atom or equivalent sufficient. As Windows 7 launched in 2009, most laptops and PCs from the last decade would be able to support this program.

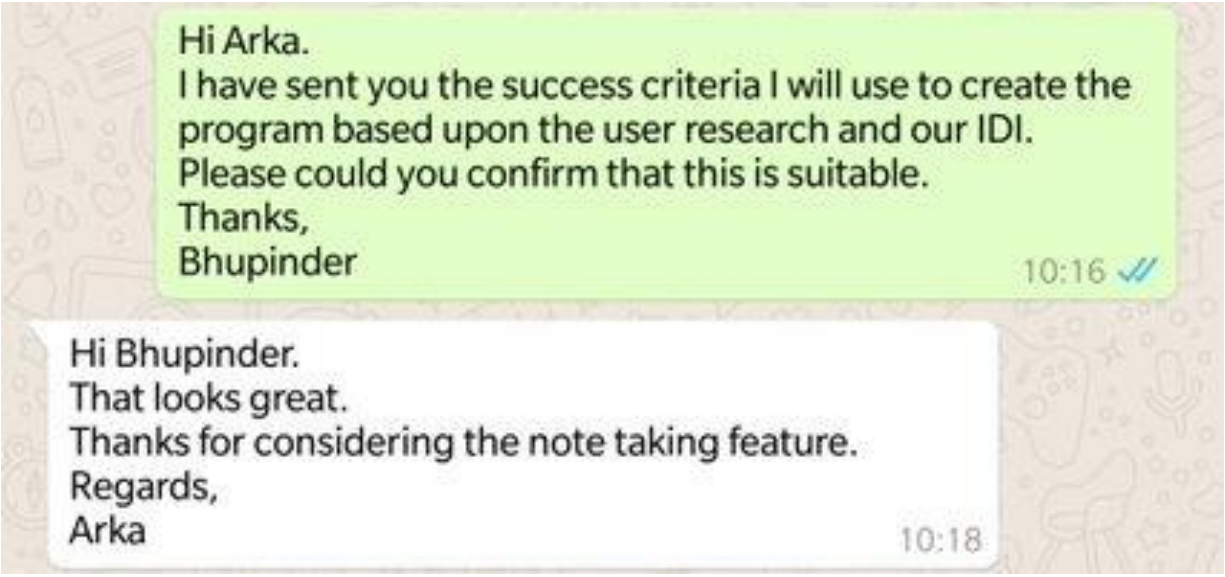
SUCCESS CRITERIA

For my project to be successful, it must meet the following requirements. These requirements have been decided on from various sources of information (indicated in the final solution requirements). The success requirement will identify the final tests that will take place.

No.	**Requirement**	**Justification**
1	All buttons for navigation must be easy to access	It should be easy to find forward and back buttons for the user to operate the program easily.
2	Large buttons to support a touchscreen input	So people can easily click a button without pressing the wrong one.
3	All buttons should be labelled accurately	So a user knows what each button does and can easily operate the program for their required function.
4	Suitable advice and recommendations displayed	This will help with understanding more complex options in the program. This requirement came from the user research.
5	The parts can be chosen in any order	Allows a user to add in the components of highest priority first, from the stakeholder IDI.
6	Allow users to toggle advanced mode	From the user research, it should be easy to find the advanced mode toggle to use the program quickly.
7	It must save the build for later access by the stakeholder	From the stakeholder IDI, this will allow the use of the program to be analysed. All data must be up to date and saved correctly.
8	The user can choose to save their build for personal use	From the user research, so the user can easily see what they have selected for later reference or research.
9	There should be suitable error checking at all data entry points.	The program should always operate and advice a user if their input is invalid. A pop-up could be used.
10	Each user must enter their name	So that it can be found by the stakeholder if they choose to build it, from the stakeholder requirement.
11	Allow users to 'make a note'	From user research, so a user can mention preferences or any feedback.
12	Use an SQL database, which holds the properties of components and user selections	So that the program is robust and can work with other applications. It will allow for easy data analysis and updates, which was a stakeholder requirement.
13	Ensure parts are compatible	This is one of the major aspects of the program and will ensure a user does not make mistakes in the selection process. It will stop Arka from needing to contact the client about their mistake later on.
14	Uses system resources efficiently	The program should not use unnecessary storage space, RAM or processing power. This will allow it to run for as long as possible on the hardware.

This criterion will be assessed at the end of every major development as a percentage (with 100% as fully achieved). I will know if my program is successful as each point will be assessed on a scale of 1-3, where 1 is ‘fully achieved’, 2 is ‘acceptable’ and 3 is ‘needs improvement’.

STAKEHOLDER APPROVAL



DESIGN

DECOMPOSING THE PROBLEM AND DEFINING THE STRUCTURE

My program must use a database to gather parts, which must be presented in a graphical user interface for the user to filter through or view recommendations to create a final build. By splitting all these sections up, it will become much easier to create the program, as problems can be solved independently and put together afterwards.

The database will be my first problem to solve. I will do this by using SQL to make a relational database and test that data can be entered properly and that results are summoned correctly.

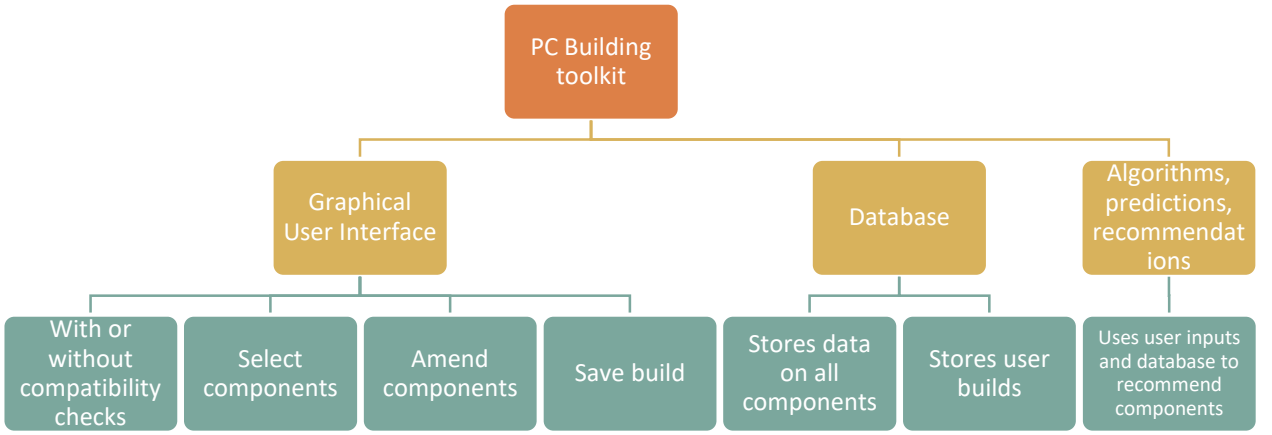
The second issue will be the GUI. I will try to lay out all sections of the page correctly, and with support and communication from the stakeholder, will make the user interface. After creating this, I will add all the functions of the program, and assign functions to button presses.

Finally, the user must be able to filter through parts from the database using the program to create a final build. This will also require the use of SQL, but in more depth compared to the first. At this point, the database will be very large and could take a long time to get results. I will need to make sure that the database is optimized, and that results can be gained efficiently.

This is a very quick overview of how my program will function, but I decided to break down my program even more (stepwise refinement) to gain a further understanding of the program. I decided to create a top down diagram as it allows me to clearly show the decomposition of the program into smaller modules. This also allows me to think ahead and consider all aspects of the program before beginning the programming. Creating a flowchart will also help with thinking logically and deciding how the modules will be assembled.

TOP DOWN DIAGRAM

This top down diagram shows the individual modules that the project will be broken into.

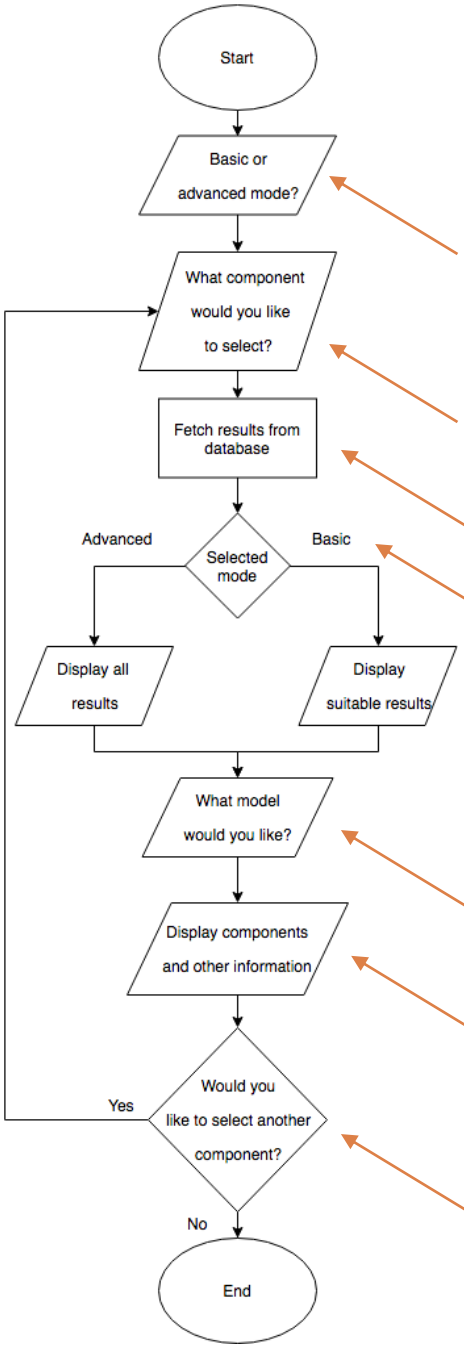


My toolkit will be broken into 3 main modules. The first will be the GUI made using Tkinter. Tkinter is a python module which allows you to create graphical user interfaces. I chose to use Tkinter over another module such as PyQt as it is standard in python and comes preinstalled. This means that it will be natively supported on Arka's hardware and will make it easy to deploy the program on computers easily. I will need to map all the buttons, text and other visual components so that they are displayed correctly. It must also be compatible with Arka's hardware and follow the design specification. The GUI must consider all inputs made by the user, and all outputs that need to be displayed to the user. The colour scheme must also follow the stakeholder's requirements and if there is enough time, there should be an accessibilities menu, so anyone who requires visual aids is able to use the program, such as larger text and buttons or a contrast between the text and the background. With the GUI, I must also consider how errors are shown (such as input or compatibility errors). One solution would be to use pop-up notifications.

The next main module would be the databases. One sub module would need to be able to add data to the components database. This will only be done on occasion when a new product is launched and in stock. The next sub module would need to fetch data from the components database. Suitable SQL queries need to be used so only compatible components are displayed. The final sub module would need to add a user's final build into a database for the stakeholder to view. If there are any errors with this, such as a duplicate user name, it must handle it correctly.

The final modules will be all the algorithms responsible for the logical operation of the program. This includes allowing the user to select more components until then stop. It also includes the exporting of the build for the user to keep as well as the compatibility checker.

FLOWCHART



This flowchart shows the order in which the program will run, where it will make any check and take inputs, and any loops it will use. This shows the logic that the program should use.

By allowing the user to choose between a basic and advanced mode, the program will be more suited to all users. It will also enable the compatibility checker algorithm.

The user can then choose a component. This is the first part of the large loop and will allow the user to eventually create a complete build in any order.

After selecting the preferred component category, the program will connect to the SQL database and fetch the relevant products, using the compatibility checker algorithm.

The user can view the features and specifications of the products and can select one.

The user is then shown their current build, so they know what is remaining and if they want to check anything.

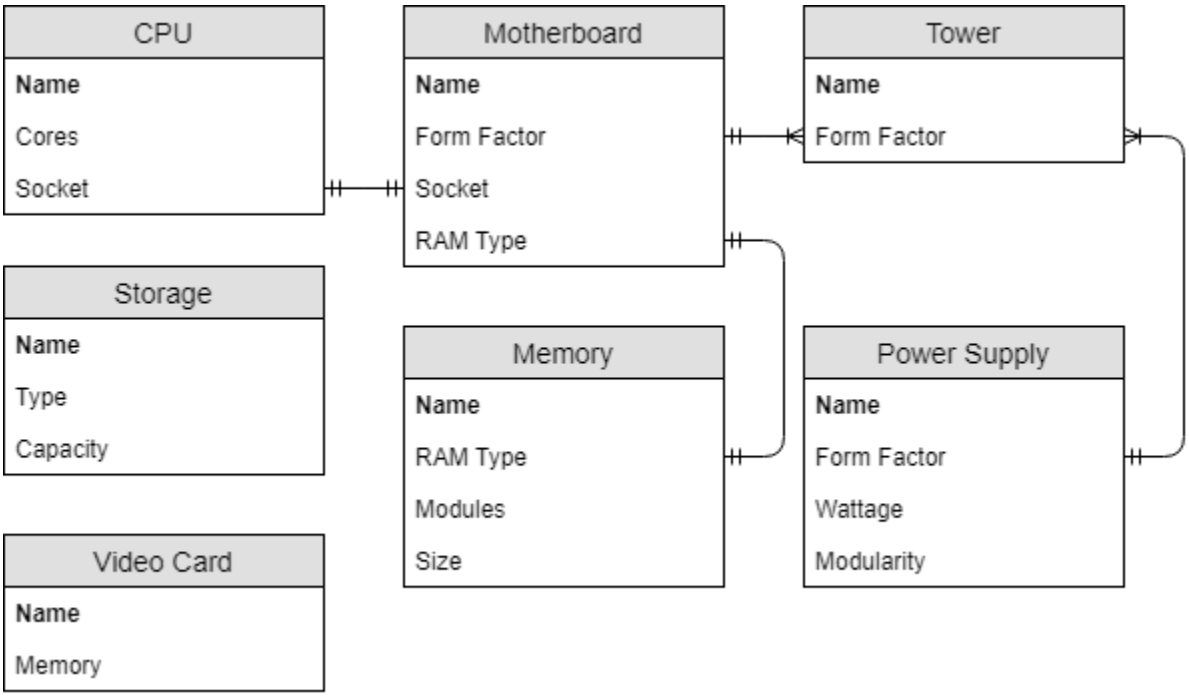
They can then choose to select another component or end the program.

Before the program is ended, the chosen components are saved in the database and the user is presented with the option to export their build using the exporter module.

ALGORITHMS

ENTITY RELATIONSHIP MODEL

This model is used to represent how the relational database will operate diagrammatically. The name will be the primary key.



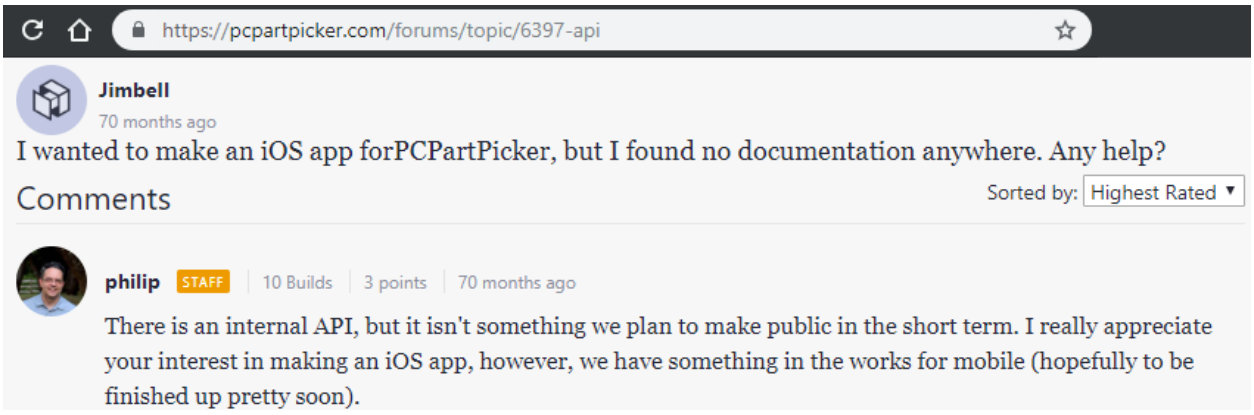
These are all the tables in the components database. For testing purposes, the database will be in the unnormalized form. If there is enough time, I will try to move the database to 3NF. I have created the database in this way as it will be easy for the compatibility checker module to function. I have also used abstraction to only add the key attributes of the components in the database

TEST DATA IN DATABASE

Table Name				
CPU	Name	Cores	Socket	
	AMD Ryzen 2600	6	AM4	
	Intel Core i5 6600k	4	LGA1151	
	Intel Core i5 8600k	6	LGA1151	
	Intel Core i7 8700k	6	LGA1151	
	AMD Ryzen 2700X	8	AM4	
Motherboard	Name	Form Factor	Socket	RAM type
	MSI Z370 A PRO	ATX	LGA1151	DDR4
	Gigabyte B360M	Micro ATX	LGA1151	DDR4
	Asus STRIX B350-F	ATX	AM4	DDR4
	MSI Z270 Gaming Plus	ATX	LGA1151	DDR4
	ASRock AB350M Pro4	Micro ATX	AM4	DDR4
Memory	Name	RAM Type	Modules	Size
	Corsair Vengeance LPX	DDR4	1x8GB	8GB
	Team Vulcan T-Force	DDR4	2x8GB	16GB
	Patriot Viper White LED	DDR4	2x8GB	16GB
	G.Skill Ripjaws V Series	DDR4	2x4GB	8GB
	Kingston HyperX Fury	DDR4	2x16GB	32GB
Storage	Name	Type	Capacity	
	Western Digital Blue	7200RPM	1TB	
	Seagate Barracuda	7200RPM	2TB	
	Toshiba X300	7200RPM	5TB	
	Kingston A400	SSD	120GB	
	Adata XPG M.2 NVMe	SSD	480GB	
Video Card	Name	Memory		
	MSI GTX 1050 Ti	4GB		
	EVGA GTX 1060	6GB		
	Gigabyte GTX 1080	8GB		
	MSI RX 580	8GB		
	Asus RX VEGA 64	8GB		

Power Supply	Name	Form Factor	Wattage	Modularity
	EVGA SuperNova	ATX	650W	Full
	Corsair CX550M	ATX	550W	Semi
	Seasonic	ATX	750W	Full
	Thermaltake	ATX	650W	Full
	Corsair SF600	(SFX) Micro ATX	600W	Full
Tower	Name	Form Factor		
	NZXT S340	ATX		
	Corsair 200R	ATX		
	Cooler Master Lite 5	ATX		
	Fractal Design Focus G	ATX		
	Thermaltake Core V21	Micro ATX		

I have chosen to fill the database with these components for many reasons. Firstly, they represent a large percentile of the market. These could allow the user to build a budget PC (using slightly older products, such as the i5 6600k), all the way up to an enthusiast build with the latest and greatest products. Secondly, these are all popular products, which would be easy for Arka to source and have a proven record of reliability. This will allow Arka to build the highest quality PCs.



After conducting some research, I discovered that I would not be able to use an API to get data for the components (as shown above). This would have been the preferred method, as it would display all relevant products without requiring me to create a database. Other methods were available, such as creating a data scraper algorithm, but this could cause legal issues and would require large amounts of maintenance.

PSEUDOCODE FOR CREATING THE SQL DATABASE

```
START
        IMPORT sqlite3
        CONNECT to database
        CREATE TABLE CPU (ID INTEGER, Name TEXT PRIMARY KEY, Cores TEXT, Socket TEXT)
        ARRAY(CPU) = [
        ["AMD Ryzen 2600","6","AM4"],
        ["Intel Core i5 6600k","4","LGA1151"],
        ["Intel Core i5 8600k","6","LGA1151"],
        ["Intel Core i7 8700k","6","LGA1151"],
        ["AMD Ryzen 2700X","8","AM4"]]
        EXECUTE INTO TABLE CPU (Name, Cores, Socket) VALUES FROM ARRAY
        SAVE CHANGES TO DATABASE
END
```

This is an example of how to initially create a table in the components database. It uses the python module SQLite3 (which I chose as it is native to python). I did not use a DBMS for this, as this would be an additional piece of software that would be required, increasing the time for installing the application on Arka's hardware.

PSEUDOCODE FOR THE GRAPHICAL USER INTERFACE

```
START
        Complexity = null
        DISPLAY TKinter(homepage)
        IF UserSelection == Basic Mode:
                Complexity = Basic
        IF Userselection == Advanced Mode:
                Complexity = Advanced

        DISPLAY TKinter(components)
        SelectedComponent = UserSelection
        SELECT ID, NAME FROM UserSelection;
        SelectedModel = UserSelction

        DISPLAY TKinter(repeat)
        PRINT("Would you like to select another component?")
        IF UserSelection == YES:
                DISPLAY TKinter(components)
        IF UserSelection == NO:
                DISPLAY TKinter(end)
END
```

This is the pseudocode for the GUI. On the first page, it will ask if you want to enable the compatibility checker. Once that is done, you will progress to the next page. Here, you can select the category of components, and the compatibility checker will ensure the suitable components are displayed. Once one is selected, you will go to the next page where you can loop over to add more components or go to the final page with the final build and export option.

PSEUDOCODE FOR THE MAIN PROGRAM

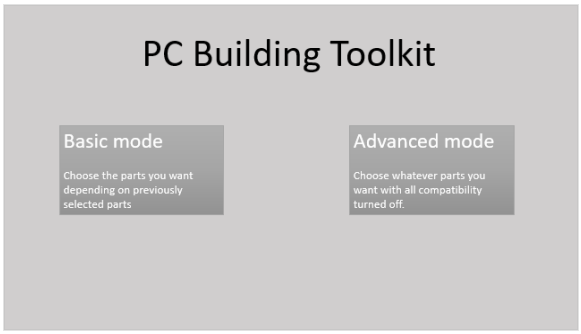
```
START
        DEFINE FUNCTION Mode():
                UserName = INPUT("What is your name?")
                ModeInput = INPUT("Would you like to use basic or advanced mode?")

        DEFINE FUNCTION ComponentSelection():
                ComponentInput = INPUT("What component would you like to select?")

        DEFINE FUNCTION ItemSelection():
                Component Selection()
                CONNECT TO SQL DATABASE
                PRINT 'ComponentSelection' FROM DATABASE
                ItemChoice = INPUT("What item would you like?")
                ARRAY(ComponentList).append(ItemChoice)
                AnotherComponent = INPUT("Do you want to select another component?")
                IF AnotherComponent == "Yes":
                        ItemSelection()

        DEFINE FUNCTION BasicMode():
                IF Socket[1] != Socket[2]:
                        PRINT("Error, Socket Incompatible")
                        ItemSelection()
                IF FormFactor[1] != FormFactor[2]:
                        PRINT("Error, FormFactor Incompatible")
                        ItemSelection()
                IF FormFactor[3] != FormFactor[2]:
                        PRINT("Error, FormFactor Incompatible")
                        ItemSelection()
                IF FormFactor[1] != FormFactor[3]:
                        PRINT("Error, FormFactor Incompatible")
                        ItemSelection()

        DEFINE FUNCTION UserComponents():
                CONNECT TO SQL DATABASE
                CREATE TABLE 'UserName' (ID INTEGER PRIMARY KEY, Component TEXT, Name TEXT)
                EXECUTE INTO TABLE 'UserName' (Component, Name) VALUES FROM ARRAY(ComponentList)
                SAVE CHANGES TO DATABASE
                PRINT("Here are your components")
                PRINT(SELECT ALL FROM 'UserName')
        Mode()
        ItemSelection()
        IF ModeInput == "Basic":
                BasicMode()
        UserComponents()
END
```

This pseudocode shows how everything will fit together. The function Mode() will be located on the first page of the GUI, shown above as Tkinter(homepage). The function ComponentSelection() will run on GUI page Tkinter(components). The function ItemSelection() will run on the same page. BasicMode() will also run here and will prevent the user from selecting an incompatible component. Finally, UserComponents() will run Tkinter(end) and will save the build to the stakeholder's database. All these functions were decomposed in the top down diagram and allow the program to be modular, easy to update and easy to diagnose.

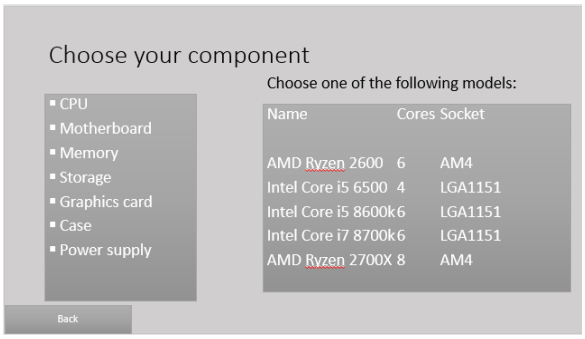
USABILITY FEATURES

SKETCHES 1

I decided to create digital sketches of how the program would be laid out and what it would include. These will be presented to Arka for his opinions and suggestions.



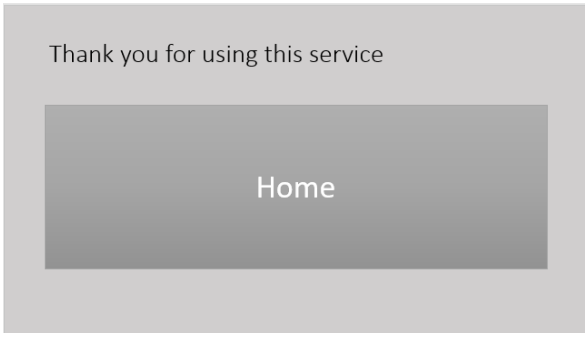
This would be the home screen. It would allow a user to choose if they want compatibility enabled or not. Descriptions have also been added so users know that the buttons do. Clicking a button will move you to the next page automatically.



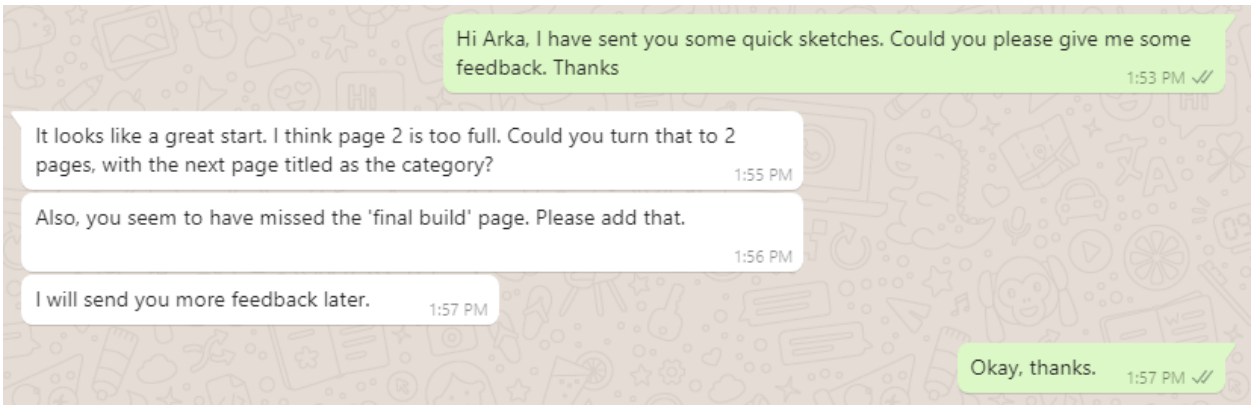
Upon selection, the user will be presented with a list of components to choose from. A back button is present if they want to return to the main menu.

When they select a component, they will get a list of components to choose from.

Once selected, they will be presented with information on the component, as well as a prompt to select another.

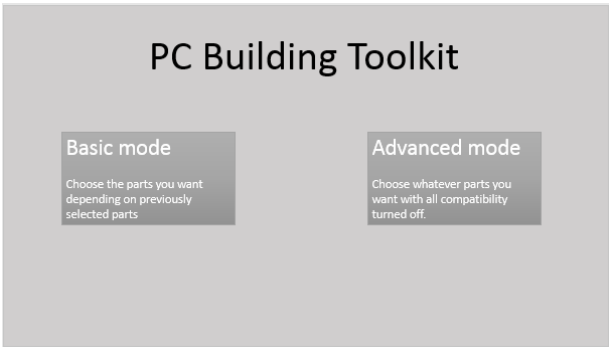


This is the final page shown to the user, allowing the next user to use the program.

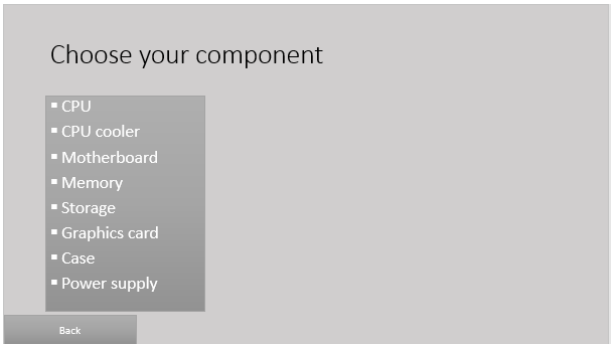


This was my conversation with Arka. We have focused on the actual pages for now and will edit the content on the pages in our next conversation.

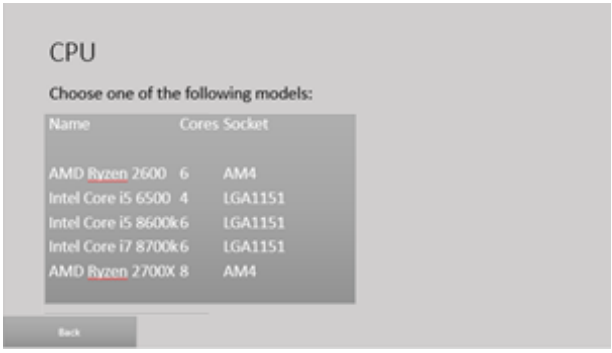
SKETCHES 2



This has remained the same.



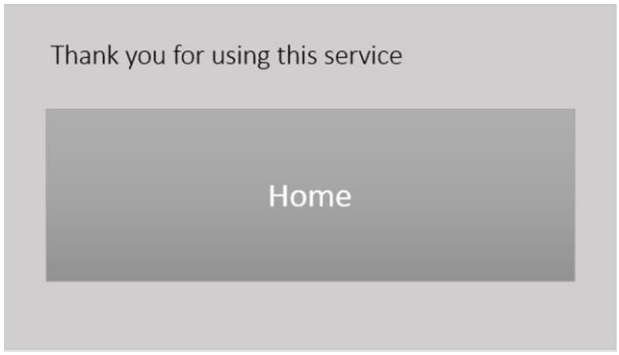
I have separated the components to the next page.



The program is more visually appealing and easier to look at.

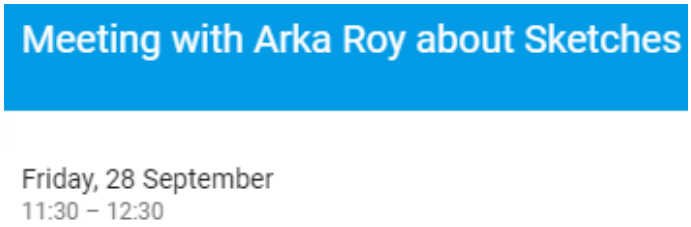
This has remained the same.

When they have chosen to stop adding more components to their build, they are shown a summary page and the option to save their build.



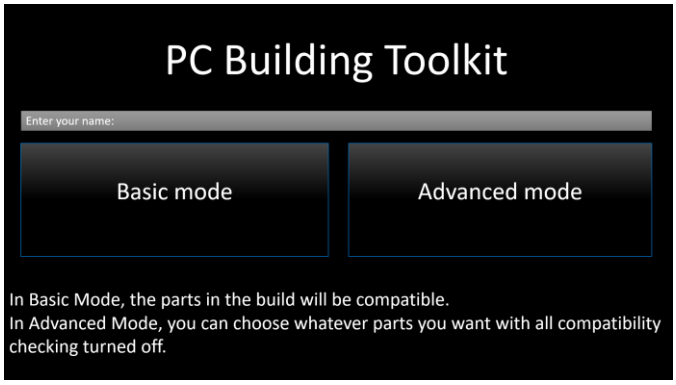
This has remained the same.

After setting a meeting with Arka Roy, we discussed these sketches. The following was what was decided.

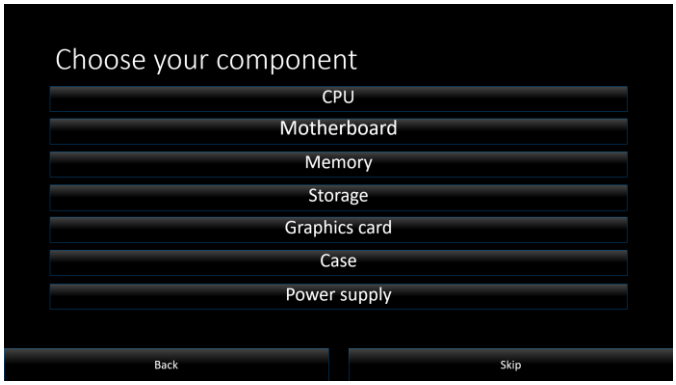


- Remove the cost of products
- Change the colour scheme, white text on a black background with blue accents
- Add a ‘skip’ button on relevant pages
- Add an ‘optional user feedback’ page at the end to get the user’s opinion
- Errors must be presented with a pop-up message and the screen required for the fix
- Remove button descriptions from buttons
- Make buttons bigger and separated
- Add user name

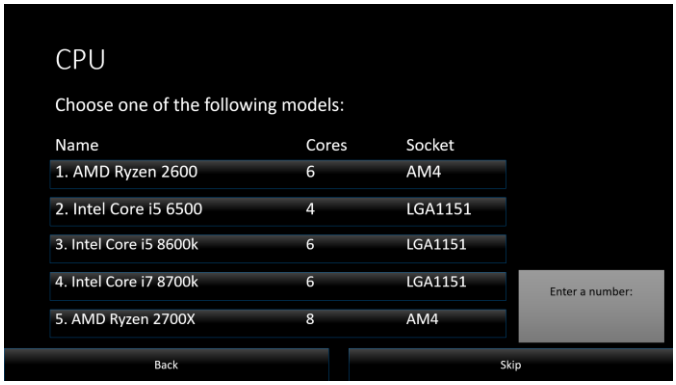
SKETCHES 3



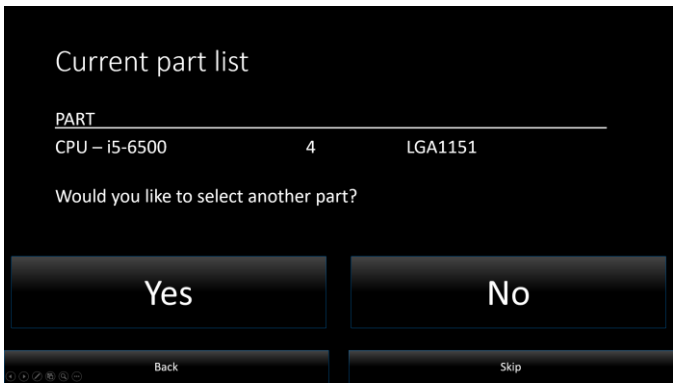
The buttons have been made bigger, with the descriptions on the bottom of the page. The new colour scheme has been added. Overall, this is a much cleaner look with emphasis on the main aspects. The user name entry box has also been added so a build can be found easily.



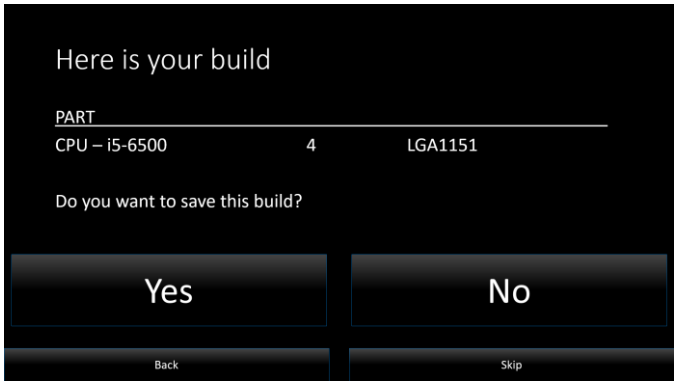
Each component category is on its own clearly visible button. There is also a skip button if required. Buttons are also a lot larger, making it a lot more suitable for touchscreen inputs.



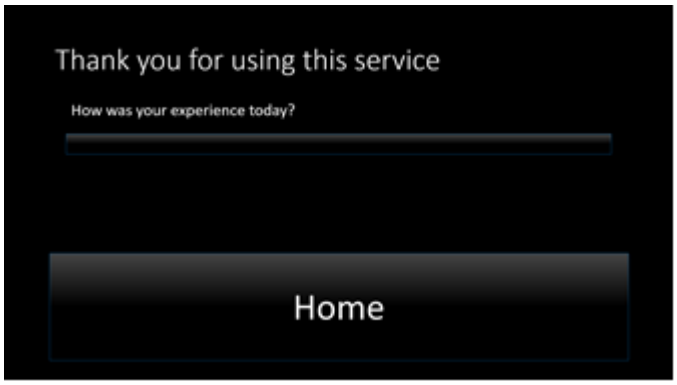
The component category is the title of this page. All components are on separate labels along with their details. By using an entry box to add a component, the wrong one is not selected.



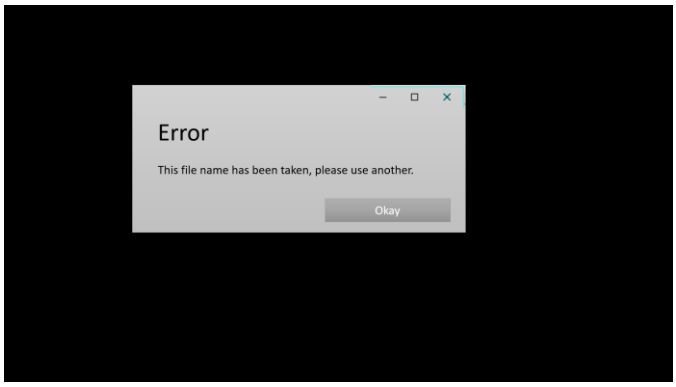
The prices for components have been removed, as per stakeholder requirements.



Once the user has stopped adding components, they can see their final build, along with the option to save everything for their own use, as per user research.

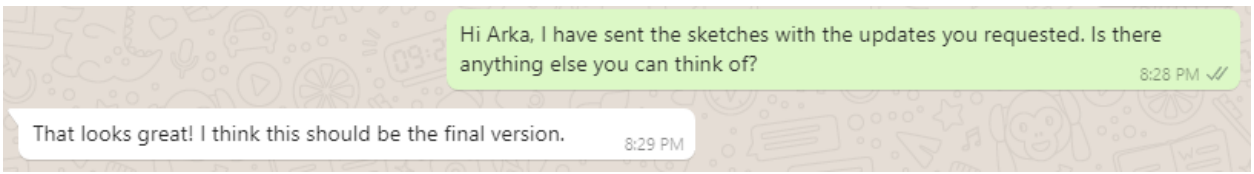


A feedback section has been added allowing a user to type a comment. This could be used for comments on how to improve the program, or if the user wants to make any customisations.



This is an example of the pop-up box. Here, it is being used to show an error, and pressing 'Okay' in the bottom right will move the user to the page of the error.

STAKEHOLDER APPROVAL



I also feel that this version is great. The colour scheme is very consistent, as are the sizes of the buttons (such as the back and skip buttons). It is a very linear layout and it will be very easy to find any part of the program. Pop-up boxes are a perfect way to present issues and will prevent issues from arising later in the program.

VARIABLES, DATA STRUCTURES AND VALIDATIONS

VARIABLES

Variable name	Use	Data Type
UserName	Stores the user name and file name	String
ModeInput	Allows the user to choose basic or advanced mode	Integer
ComponentInput	Stores the component category the user will choose	Integer
Database	Represents to the SQL database	sqlite3.Connection
cursor	Used for fetching data from the database	sqlite3.Cursor
ItemChoice	Stores the productID that the user chooses	String
Check	Stores the product from the database for validations	sqlite3.Cursor
AnotherComponent	Asks user if they want to continue adding components and stores the response	Integer
ItemText	Stores fetched product name	String
loops	Stores the number of times the while statement has run	Integer
CPUCheck	Stores details used for compatibility checks	String
MotherboardCheck	Stores details used for compatibility checks	String
PowerSupplyCheck	Stores details used for compatibility checks	String
TowerCheck	Stores details used for compatibility checks	String

DATA STRUCTURES

Name	Use and example	Data Type
ComponentList	Stores component categories for the user to select from. ComponentList = ["CPU","Motherboard","Memory", "Storage","VideoCard","PowerSupply","Tower"]	List
UserChoices	Stores user's products in 2d array. This will fill up as the user chooses components and will be used to export the data for the user. UserChoices = [["CPU",""], ["Motherboard",""], ["Memory",""], ["Storage",""], ["Video Card",""], ["Power Supply",""], ["Case",""]]	List
ComponentChoice	Stores chosen products and will fill up as the user chooses components ComponentChoice = ["","","","","","","",""]	List

For all of my tables, the primary key will be id, as an incrementing INTEGER. Everything else will be if data type TEXT.

Components Database

Table Name

CPU	Name	Cores	Socket	
	AMD Ryzen 2600	6	AM4	
	Intel Core i5 6600k	4	LGA1151	
	Intel Core i5 8600k	6	LGA1151	
	Intel Core i7 8700k	6	LGA1151	
	AMD Ryzen 2700X	8	AM4	
Motherboard	Name	Form Factor	Socket	RAM type
	MSI Z370 A PRO	ATX	LGA1151	DDR4
	Gigabyte B360M	Micro ATX	LGA1151	DDR4
	Asus STRIX B350-F	ATX	AM4	DDR4
	MSI Z270 Gaming Plus	ATX	LGA1151	DDR4
	ASRock AB350M Pro4	Micro ATX	AM4	DDR4

Memory	Name	RAM Type	Modules	Size
	Corsair Vengeance LPX	DDR4	1x8GB	8GB
	Team Vulcan T-Force	DDR4	2x8GB	16GB
	Patriot Viper White LED	DDR4	2x8GB	16GB
	G.Skill Ripjaws V Series	DDR4	2x4GB	8GB
	Kingston HyperX Fury	DDR4	2x16GB	32GB
Storage	Name	Type	Capacity	
	Western Digital Blue	7200RPM	1TB	
	Seagate Barracuda	7200RPM	2TB	
	Toshiba X300	7200RPM	5TB	
	Kingston A400	SSD	120GB	
	Adata XPG M.2 NVMe	SSD	480GB	
Video Card	Name	Memory		
	MSI GTX 1050 Ti	4GB		
	EVGA GTX 1060	6GB		
	Gigabyte GTX 1080	8GB		
	MSI RX 580	8GB		
	Asus RX VEGA 64	8GB		
Power Supply	Name	Form Factor	Wattage	Modularity
	EVGA SuperNova	ATX	650W	Full
	Corsair CX550M	ATX	550W	Semi
	Seasonic	ATX	750W	Full
	Thermaltake	ATX	650W	Full
	Corsair SF600	(SFX) Micro ATX	600W	Full
Tower	Name	Form Factor		
	NZXT S340	ATX		
	Corsair 200R	ATX		
	Cooler Master Lite 5	ATX		
	Fractal Design Focus G	ATX		
	Thermaltake Core V21	Micro ATX		

I have chosen to fill the database with these components for many reasons. Firstly, they represent a large percentile of the market. These could allow the user to build a budget PC (using slightly older products, such as the i5 6600k), all the way up to an enthusiast build with the latest and greatest products. Secondly, these are all popular products, which would be easy for Arka to source and have a proven record of reliability. This will allow Arka to build the highest quality PCs.

Customers Database

Table Name

FileName	Component	Name
	CPU	
	Motherboard	
	Memory	
	Storage	
	Video Card	
	Power Supply	
	Case	
FileName+Feedback	Stars	Comment

Whenever a user creates a build, two tables are created. The first stores the components in their build. This is very similar to what the user will receive if they choose to export the build. In order to keep the database as space efficient as possible, I do not include the attributes of the components selected. Firstly, Arka would know the specifications of the products. Secondly, he will be able to see the specs when using the admin program.

The second table stores the feedback provided by a user. This will be very useful for the stakeholder as they can get useful feedback from real users as the program is being used and request new features or updates in the future.

LIBRARIES

Library	Justification	Example
SQLite3	This is a very lightweight module standard in python which allows for the use of a database without the need for another application.	It will be used in storing the components, the builds created, and the user feedback.
Tkinter	This is python's standard GUI package, with lots of tutorials and documentation.	It will be used to create the GUI and pop-ups.
Pandas	Makes it easy to present large amounts of data and to interact with an SQL database. It is also very high performance and efficient.	Presenting the components, its index and its attributes on the GUI.

VALIDATIONS

Validation	Justification	Example
Is data entered in the correct format	So that a TypeError does not arise, are any issues with SQLite3.	A special character could be used in the user name field, causing an error with saving the build.
That data been entered	So that a NameError does not occur, due to missing data.	The user may not add their name, meaning that the build cannot be saved.
Does the index exist	So that a component is selected from the database, and an out of range error does not occur.	Selecting productID 6 when there are only 5 available when choosing a component.
Is the file name unique	So that data is not overwritten when saving a file with the same name.	If the same table is written to a database, it would overwrite the old one when saving it.
Compatibility checker	This will be an algorithm ensuring components are compatible.	If a user selected an LGA 1151 CPU with an AM4 motherboard, they would not be able to build the PC.

VALIDATIONS PSEUDOCODE

```
START
        TRY:
                ModeInput = INT(INPUT("Would you like to use:
    1. Basic Mode
    2. Advanced Mode"))
            IF ModeInput == 1 OR ModeInput == 2:
                break
            ELSE:
                PRINT("***Please enter a valid number***")
        EXCEPT ValueError:
            print("***Please enter the valid number***")
END
```

If creating the program using a Command Line Interface, this would ensure a suitable input has been provided. Until the user has given a suitable input, the program will not continue. This is similar in the case of the GUI, if a button has not been clicked, you cannot go onto the next page. This would be suitable for Validations 1, 2 and 3.

```
START
       WHILE TRUE:
              TRY:
                     CREATE TABLE 'UserName'(ID INTEGER PRIMARY KEY, Component TEXT, Name TEXT)
                     BREAK
              EXCEPT sqlite3.OperationalError:
                     UserName = INPUT("A file with this name may already exists or this may
                                   contain special characters. Please enter another name.")
END
```

This pseudocode would be suitable for Validations 1, 2 and 4. If an error arises when creating a new table with the given file name, it will ask for a new one, reminding the user of the requirements.

CLASSES

Due to the nature of this program, Object Orientated Programming does not need to be used much. The only place where it could be used is in the GUI programming, as it is used in Tkinter.

Tkinter Class	Description
Top Level	It is a container used to contain multiple frames.
Frame	Contains and groups widgets such as buttons and labels. This will ensure it works on Arka's 1080p displays
Label	Displays text or images and will be used for titles and descriptions.
Button	Displays text or images and can execute a command or function, e.g. the back button or basic mode button to enable compatibility checking.
Entry	Allows text to be entered, such as to take the user's filename or component.

ITERATIVE DEVELOPMENT TEST DATA

Test	Reason	Test data	Expected result
Can a user enter a file name	To ensure builds are saved correctly and not overwritten	‘Arka’ - valid ‘Arka.com’ - invalid ‘2001’ - extreme	Program continues Pop-up error due to SQL Pop-up error due to SQL
Can the user choose a component	To ensure the correct components are selected correctly	‘1’ - valid ‘6’ - invalid ‘two’ - extreme	Component selected Not found error Not found error
Does the compatibility checker work	To ensure the program works as expected	2600 + MSI Z370 - invalid 6600k + MSI Z270 – valid B350-F + Core V21 - valid	Choose diff CPU or Mobo Saved Saved
Does the GUI load	To ensure the program can be used	Running the program	GUI displaying
Does the GUI display correctly	To ensure the program is aesthetically pleasing	Running the program	GUI displaying correctly
Do the buttons work	To ensure the functions of the buttons run	Basic mode – valid Skip – valid 3 stars - valid	Checking enables Next page shown 3 Stars in feedback
Does the build save and export	To ensure the databases are saved correctly	Export – valid Don’t export – valid	Database and export saved Database saved
Do the results from the database show correctly	So the correct components can be displayed	CPU - valid Case – valid NONE - extreme	Components shown Components shown Please select a component
Does the feedback save	So that the stakeholder can learn from feedback	‘good, 4 star’ -valid ‘bad!How?’ - extreme ‘’ - extreme	Saves Saves No save as nothing to save

POST DEVELOPMENT TEST DATA

Requirement	Test Data	Expected Outcome
All buttons for navigation must be easy to access	Press Back Button - valid	Goes to the previous page
	Press CPU - valid	Displays all CPUs
	Press Skip - extreme	Asks user to choose category
Large buttons to support a touchscreen input	Stylus input – valid	Works as expected
	Input from finger – valid	Works as expected
	Input from elbow - extreme	Works despite obstructed view
All buttons should be labelled accurately	Click basic mode – valid	Easy to find and goes to next page
	Click Skip - valid	Easy to find and goes to next page
	Click CPU - valid	Easy to find and goes to next page
Suitable advice and recommendations displayed	Press basic mode – valid	Compatibility check enabled
	Press advanced mode- valid	Compatibility check disabled
	Press Neither -valid	Waits for input
The parts can be chosen in any order	Case then CPU – valid	Saves both
	GPU then PSU – valid	Saves both
	SSD then SSD - extreme	Saves last one
Allow users to toggle advanced mode	Press basic mode – valid	Compatibility check enabled
	Press advanced mode- valid	Compatibility check disabled
	Press Neither -valid	Waits for input
It must save the build for later access by the stakeholder	Force close at end – extreme	Saved
	Create build and finish – valid	Saved
	Save empty build - extreme	Saved
The user can choose to save their build for personal use	Save complete build – valid	Exported
	Save partial build – valid	Exported
	Don’t save - valid	Nothing

Test	Test Data	Expected Outcome
There should be suitable error checking at all data entry points.	“BSD.2” – invalid “Roy” – valid Id 7 selected - invalid	SQL Error, try again Continues Invalid Id, try again
Each user must enter their name	‘Arka then Arka’ - invalid ‘ ’ – invalid ‘Bhupinder’ - valid	Table exists, use another Nothing entered, try again Continues
Allow users to ‘make a note’/leave feedback	‘Can you add Xeon E3/1200V5?’ - valid ‘ ’ ‘😁’	saves Nothing to save Saves
Use an SQL database, which holds the properties of components and user selections	View build saved Bhupinder – valid View all components – valid View all CPUs with 4 cores- extreme	View saved build View all components I5 6600k shown
Ensure parts are compatible	2600 + MSI Z370 - invalid 6600k + MSI Z270 – valid B350-F + Core V21 - valid	Choose diff CPU or Mobo Continue Continue
Uses system resources efficiently	Empty file size Complete build size	0 bytes <1000 bytes

DEVELOPMENT

I have decomposed this problem. To make it easier to test and organise, I will use numbers to name them. The first number will represent the prototype number. The number after a dot (.) will be the module number. Finally, the last number will be version of that module. For example, 2.5.3 is the third iteration of the 5th module on the second prototype.

PROTOTYPE 1

In my initial prototype, I would ensure that the program is logically sound. This means that it must run in the correct order, take in user inputs, and use these inputs to save chosen components into a file or database. This means that the program must interact with the database correctly, and the compatibility checking algorithm must also work. I will not include a GUI in this prototype.

X.1.X is the module Mode(), which allows a user to enter their name and select their mode.

X.2.X is the module ComponentSelection(), which allows the user to select a component.

X.3.X is the module ItemSelection() which tries to fetch a individual component.

X.4.X is the module UserComponents() which saves the build to the database

X.5.X is the algorithm BasicMode() which ensures components are compatible

Test No.	**1.1.1**
Test	Can the user enter a file name
Reason	So that their choices are saved and exported
Test Data	Arka
Expected Result	Program continues
Code being tested	`UserName = input("What is your name (Will be used as the file name)? \n")`

Actual Result	`What is your name (Will be used as the file name)?`
	`Arka`
	
	`Would you like to use:`
	`1. Basic Mode`
	`2. Advanced Mode`
	`3. More information`

Other Notes	-pass. Validation in 1.4.1

Test No.	**1.1.2**
Test	Can the user select the mode correctly
Reason	So the program works as the user wants
Test Data	one
Expected Result	Program continues

Code being tested	`ModeInput = input("""Would you like to use:`
	`1. Basic Mode`
	`2. Advanced Mode`
	`""")`

Actual Result	Compatibility check does not run
Other Notes	Extreme valid test data, -failed

Test No.	**1.1.3**
Test	Can the user select the mode correctly
Reason	So the program works as the user wants
Test Data	one
Expected Result	Program continues

Code being tested	`while True:`
	`  try:`
	`  ModeInput = int(input("""Would you like to use:`
	`  1. Basic Mode`
	`  2. Advanced Mode`
	
	`"""))`
	`  break`
	`  except ValueError:`
	`  print("Please enter the number\n")`

Actual Result	`Would you like to use:`
	`  1. Basic Mode`
	`  2. Advanced Mode`
	`one`
	`***Please enter the number***`
	
	`Would you like to use:`

Other Notes	Extreme valid test data, prompts correct input method. The whole true loop only breaks once a number has been given. -pass

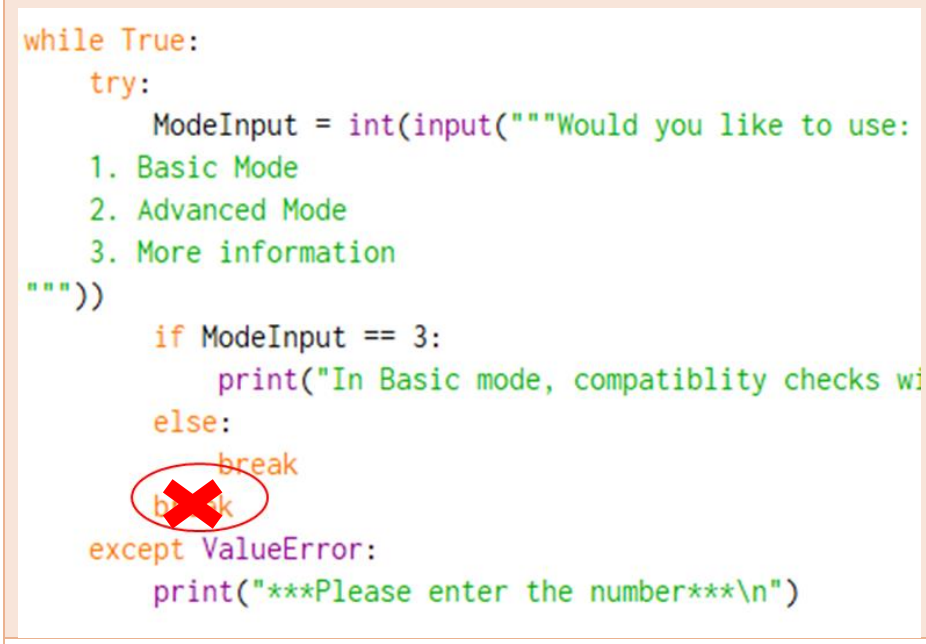
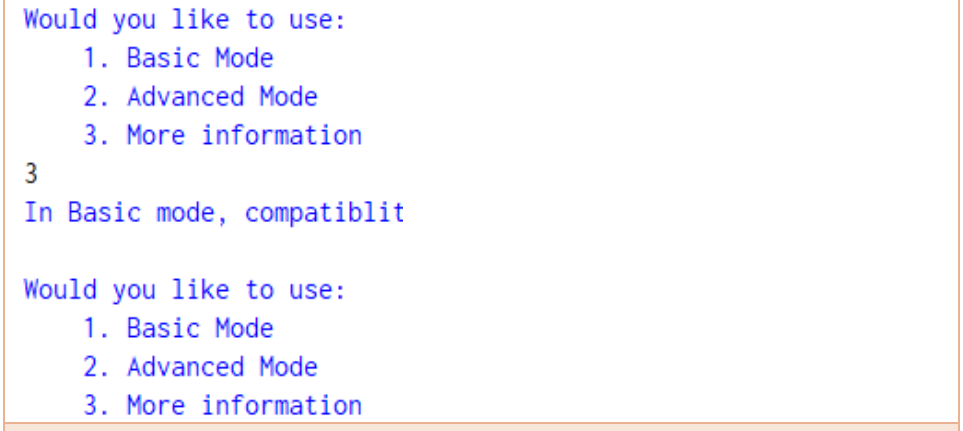
Test No.	**1.1.4**
Test	Can the user select the mode correctly
Reason	So the program works as the user wants and understands the options
Test Data	3
Expected Result	Program continues
Code being tested	

```
while True:
    try:
        ModeInput = int(input("""Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information

"""))
        if ModeInput == 3:
            print("In Basic mode, compatiblity checks will be
        else:
            break
        break
    except ValueError:
        print("Please enter the number\n")
```

Actual Result

```
Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
3
In Basic mode, compatibility checks will be enabled throughout the program
, meaning that the components that you select will work with each other. A
dvanced mode is only for professionals as they will not have any compatibi
lity checks.
```

Other Notes	valid test data, added ‘more information’ option for user’s understanding, however, the program does not ask again for the mode, -failed

Test No.	**1.1.5**
Test	Can the user select the mode correctly
Reason	So the program works as the user wants and understands the options
Test Data	3
Expected Result	Program asks for mode
Code being tested	
Actual Result	
Other Notes	Removed the extra break which would stop the program from asking the user for their input again, -pass

```
while True:
    try:
        ModeInput = int(input("""Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
"""))
        if ModeInput == 3:
            print("In Basic mode, compatiblity checks wi
        else:
            break
            break
    except ValueError:
        print("***Please enter the number***\n")
```

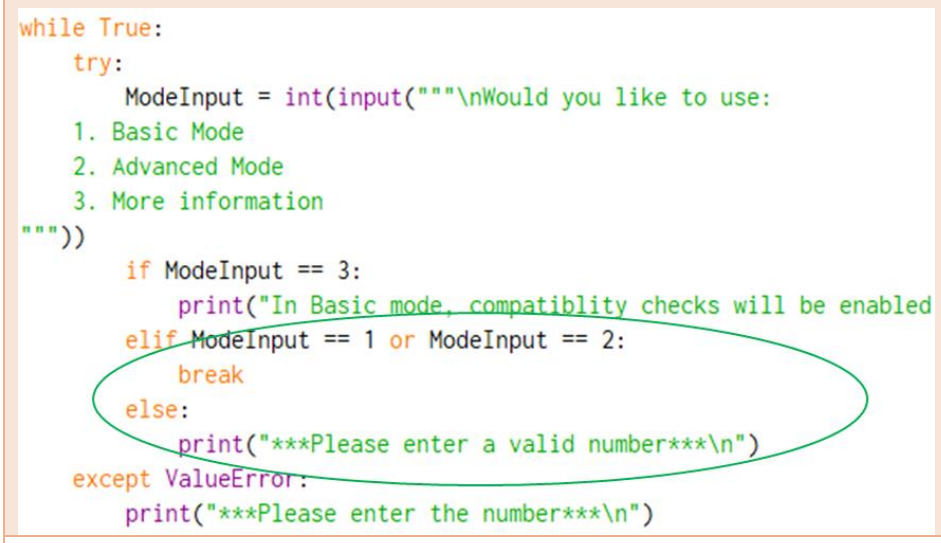
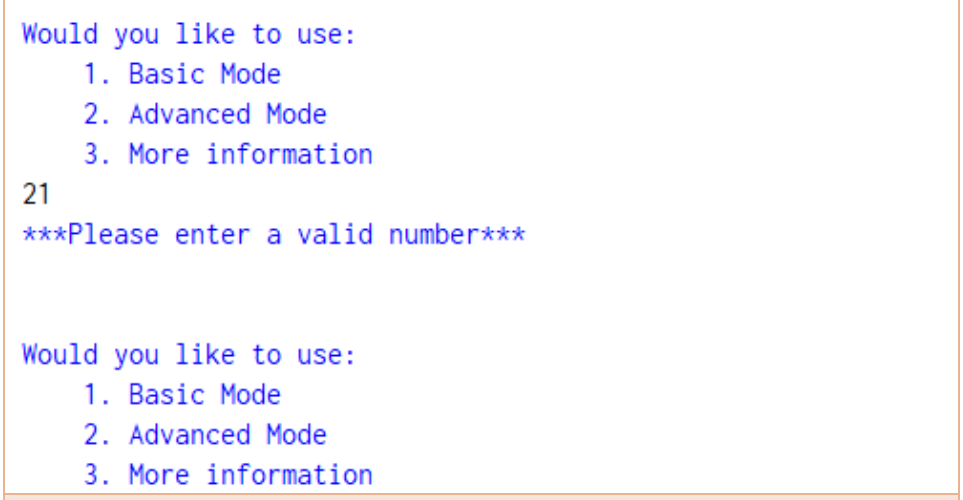
```
Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
3
In Basic mode, compatiblit

Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
```

Test No.	**1.1.6**
Test	Can the user select the mode correctly
Reason	So the program works as the user wants and understands the options
Test Data	21
Expected Result	Program reports error

Code being tested	`while True:`
	`  try:`
	`  ModeInput = int(input("""Would you like to use:`
	`  1. Basic Mode`
	`  2. Advanced Mode`
	`  3. More information`
	
	`"""))`
	`  if ModeInput == 3:`
	`  print("In Basic mode, compatiblity checks will be`
	`  else:`
	`  break`
	`  break`
	`  except ValueError:`
	`  print("Please enter the number\n")`

Actual Result	`Would you like to use:`
	`  1. Basic Mode`
	`  2. Advanced Mode`
	`  3. More information`
	`21`
	
	`What component would yo`
	`1. CPU`
	`2. Motherboard`
	`3. Memory`
	program continues

Other Notes	Invalid test data, -failed

Test No.	**1.1.7**
Test	Can the user select the mode correctly
Reason	So the program works as the user wants and understands the options
Test Data	21
Expected Result	Program reports error
Code being tested	
Actual Result	
Other Notes	The program checks if 1 or 2 have been entered (the only options other than 3). If something other has been entered, and error is given and they must try again.-pass

```
while True:
    try:
        ModeInput = int(input("""\nWould you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
"""))
        if ModeInput == 3:
            print("In Basic mode, compatiblity checks will be enabled
        elif ModeInput == 1 or ModeInput == 2:
            break
        else:
            print("***Please enter a valid number***\n")
    except ValueError:
        print("***Please enter the number***\n")
```

```
Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
21
***Please enter a valid number***


Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
```

FINAL CODE FOR MODULE 1.1 MODE()

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def Mode():
    global UserName, ModeInput                                              #Declares these variables as global
    UserName = input("What is your name (Will be used as the file name)? \n")  #Takes an input from a user, \n is a new line
    while True:                                                             #stars a loop till made false
        try:                                                                #Tests the next block of code for errors
            ModeInput = int(input("""\nWould you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
"""))                                                                       #Takes an input from the user as an interger
            if ModeInput == 3:                                              #If the user asks for more info
                print("In Basic mode, compatibility checks will be enabled throughout the program, meaning that the components
            elif ModeInput == 1 or ModeInput == 2:                          #If a valid input is received, break loop
                break
            else:                                                           #If a number out of range is provided
                print("***Please enter a valid number***\n")                #Tells the user to give a valid number
        except ValueError:                                                  #If the input from the user is not an integer
            print("***Please enter the number***\n")                        #Tell the user to enter an integer
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

I have decided to use global variables as this is a fairly small project with a single developer (myself). If there were more programmers or someone else to do maintenance in the future, I would have ensured the use of passing by reference, as this is much better practice and can also be easier to diagnose problems, as these variables can accidentally be changed in runtime.

Also, I will not be providing the same annotations where a similar line has been annotated.

Test No.	**1.2.1**
Test	Can the user select a component type
Reason	So a user can select the category correctly
Test Data	8
Expected Result	'Please enter a valid number'
Code being tested	(see code below)
Actual Result	8 ***Please enter a valid number***
Other Notes	invalid test data, -pass

Code being tested:

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def ComponentSelection():
    global ComponentInput
    while True:
        try:
            ComponentInput = int(input("""\nWhat component would you like to select?
    1. CPU
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
"""))
            if ComponentInput <1 or ComponentInput >7:      #If the number is out of the range 1-7
                print("***Please enter a valid number***")
            else:
                break                                       #Stops when a valid number has been given
        except ValueError:
            print("***Please enter the number***\n")
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Test No.	**1.2.2**
Test	Can the user select a component type
Reason	So a user can select the category correctly
Test Data	5.5
Expected Result	'Please enter a valid number'
Code being tested	See code below
Actual Result	5.5 ***Please enter the number***
Other Notes	invalid test data. As I am asking for an input as an integer, a float will cause an error, -pass

Code being tested:

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def ComponentSelection():
    global ComponentInput
    while True:
        try:
            ComponentInput = int(input("""\nWhat component would you like to select?
    1. CPU
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
"""))
            if ComponentInput <1 or ComponentInput >7:      #If the number is out of the range 1-7
                print("***Please enter a valid number***")
            else:
                break                                       #Stops when a valid number has been given
        except ValueError:
            print("***Please enter the number***\n")
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

FINAL CODE FOR MODULE 1.2 COMPONENTSELECTION()

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def ComponentSelection():
    global ComponentInput
    while True:
        try:
            ComponentInput = int(input("""\nWhat component would you like to select?
    1. CPU
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
"""))
            if ComponentInput <1 or ComponentInput >7:      #If the number is out of the range 1-7
                print("***Please enter a valid number***")
            else:
                break                                       #Stops when a valid number has been given
        except ValueError:
            print("***Please enter the number***\n")
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Test No.	**1.3.1**
Test	Can the program access the database

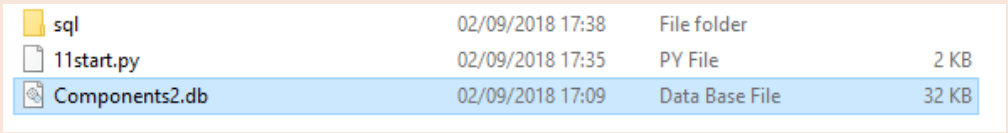
Reason	So the program can use the database
	`What component would you like to select?`
	`1. CPU`
	`2. Motherboard`
	`3. Memory`
	`4. Storage`
	`5. Video Card`
	`6. Power Supply`
	`7. Case`
	`8`
	`***Please enter a valid number***`
	
	`What component would you like to select?`

Test Data	1 (CPU)
Expected Result	Program continues

Code being tested	`cursor.execute('''SELECT * FROM {}'''.format(ComponentList[ComponentInput-1]))`
	`for row in cursor:`
	`  print(row)`

Actual Result	`cursor.execute('''SELECT * FROM {}'''.format(ComponentList[ComponentInput-1]))` `sqlite3.OperationalError: no such table: CPU`
Other Notes	Valid data type, -failed

Test No.	**1.3.2**
Test	Can the program access the database
Reason	So the program can use the database
Test Data	1 (CPU)
Expected Result	Program continues

Code being tested	`cursor.execute('''SELECT * FROM {}'''.format(ComponentList[ComponentInput-1]))`
	`for row in cursor:`
	`  print(row)`
	 

Actual Result	`(1, 'AMD Ryzen 2600', '6', 'AM4')`
	`(2, 'Intel Core i5 6600k', '4', 'LGA1151')`
	`(3, 'Intel Core i5 8600k', '6', 'LGA1151')`
	`(4, 'Intel Core i7 8700k', '6', 'LGA1151')`
	`(5, 'AMD Ryzen 2700X', '8', 'AM4')`

Other Notes	Fixed by putting the database in the same directory. This issue was caused by the database not being in the same directory. I will keep this in mind when installing to Arka's hardware. I could also specify the directory, that can only be modified by admin, to prevent hackers and increase security. -pass

Test No.	**1.3.3**
Test	Does the database output look good
Reason	So the program is aesthetically pleasing
Test Data	1 (CPU)
Expected Result	Looks good with all information

Code being tested	`cursor.execute('''SELECT * FROM {}'''.format(ComponentList[ComponentInput-1]))`
	`for row in cursor:`
	`  print(row)`

Actual Result	`(1, 'AMD Ryzen 2600', '6', 'AM4')`
	`(2, 'Intel Core i5 6600k', '4', 'LGA1151')`
	`(3, 'Intel Core i5 8600k', '6', 'LGA1151')`
	`(4, 'Intel Core i7 8700k', '6', 'LGA1151')`
	`(5, 'AMD Ryzen 2700X', '8', 'AM4')`

Other Notes	No column headings, -failed

Test No.	**1.3.4**
Test	Does the database output look good
Reason	So the program is aesthetically pleasing
Test Data	1 (CPU)
Expected Result	Looks good with all information

Code being tested	`x=[]`
	`for column in cursor.execute("SELECT * FROM Tower").description:`
	`  x.append(column[0])`

Actual Result	`CPU`
	`['id', 'Name', 'Cores', 'Socket']`
	`(1, 'AMD Ryzen 2600', '6', 'AM4')`
	`(2, 'Intel Core i5 6600k', '4', 'LGA1151')`
	`(3, 'Intel Core i5 8600k', '6', 'LGA1151')`
	`(4, 'Intel Core i7 8700k', '6', 'LGA1151')`
	`(5, 'AMD Ryzen 2700X', '8', 'AM4')`

Other Notes	Better but not good enough, -failed

Test No.	**1.3.5**
Test	Does the database output look good
Reason	So the program is aesthetically pleasing
Test Data	1 (CPU)
Expected Result	Looks good with all information
Code being tested	`import pandas` `print (pandas.read_sql_query("SELECT * FROM CPU", Database))`
Actual Result	id Name Cores Socket 0 1 AMD Ryzen 2600 6 AM4 1 2 Intel Core i5 6600k 4 LGA1151 2 3 Intel Core i5 8600k 6 LGA1151 3 4 Intel Core i7 8700k 6 LGA1151 4 5 AMD Ryzen 2700X 8 AM4
Other Notes	Better but includes unneeded an index, -failed

Test No.	**1.3.5**
Test	Does the database output look good
Reason	So the program is aesthetically pleasing
Test Data	1 (CPU)
Expected Result	Looks good with all information
Code being tested	`print ("\n", pandas.read_sql_query("SELECT * FROM {}".format(ComponentList[ComponentInput-1]), Database, "id"))`
Actual Result	Name Cores Socket id 1 AMD Ryzen 2600 6 AM4 2 Intel Core i5 6600k 4 LGA1151 3 Intel Core i5 8600k 6 LGA1151 4 Intel Core i7 8700k 6 LGA1151 5 AMD Ryzen 2700X 8 AM4
Other Notes	Looks better, fixed by using the id column as the index, -pass

Test No.	**1.3.6**
Test	Can the user select the components required correctly
Reason	So the user can use the program correctly
Test Data	2 (No)
Expected Result	Stops asking for more components

Code being tested	`AnotherComponent = input("""\nDo you want to select another component?`
	`1. Yes`
	`2. No`
	`""")`
	`if AnotherComponent == 1:`
	`pass`
	`elif AnotherComponent == 2:`
	`break`

Actual Result	Do you want to select another component? 1. Yes 2. No 2 What component would you like to select? 1. CPU 2. Motherboard 3. Memory 4. Storage 5. Video Card 6. Power Supply 7. Case 8. Other
Other Notes	Program does not break the loop, -failed

Test No.	**1.3.7**
Test	Can the user select the components required correctly
Reason	So the user can use the program correctly
Test Data	2 (No)
Expected Result	Stops asking for more components

Code being tested	`AnotherComponent = int(input("""\nDo you want to select another component?`
	`1. Yes`
	`2. No`
	`"""))`
	`if AnotherComponent == 1:`
	`pass`
	`elif AnotherComponent == 2:`
	`break`

Actual Result	Do you want to select another component? 1. Yes 2. No 2
Other Notes	The program was comparing a string against integer, so it was not working correctly, -pass

FINAL CODE FOR MODULE 1.3 ITEMSELECTION()

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def ItemSelection():
    global ComponentList
    while True:
        ComponentSelection()

        Database = sqlite3.connect("Components2.db")        #The object Database represents the Components database
        cursor = Database.cursor()                          #The object cursor allows you to execute commands
        ComponentList = ["CPU","Motherboard","Memory","Storage","VideoCard","PowerSupply","Tower"]
                                                            #Array storing table names in database Compoonents
        print("\n", pandas.read_sql_query("SELECT * FROM {}".format(ComponentList[ComponentInput-1]), Database, "id"))
                                                            #Uses pandas to print the results of the search, indexed by id
        while True:
            ItemChoice = input("\nWhat item would you like? \n")    #Allows the user to enter the id of the component to add
            cursor.execute('''SELECT id FROM '''+ ComponentList[int(ComponentInput)-1] + ''' WHERE id = ''' + ItemChoice)
            Check = cursor.fetchall()                               #Checks if the chosen id is in the database
            print("----------------------------------------------------------------------------------------------------")
            if Check == []:
                print("***Please enter a valid id***")              #If the database returns no results, try again
            else:
                ComponentChoice[ComponentInput-1] = ItemChoice      #Saves the choice for exporting
                break

        AnotherComponent = int(input("""\nDo you want to select another component?
        1. Yes
        2. No
"""))
        if AnotherComponent == 1:                                   #If the user want to add another product...
            pass                                                    #...Does not break loop, so it repreats
        elif AnotherComponent == 2:
            break
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Test No.	**1.4.1**
Test	Can the user enter a file name correctly
Reason	So that their choices are saved and exported
Test Data	Kilesh Nundlall
Expected Result	Program continues
Code being tested	`except sqlite3.OperationalError:` `    UserName = input("A file with this name already exists. Please enter another name.\n")`
Actual Result	A file with this name already exists. Please enter another name. Kilesh Nundlall A file with this name already exists. Please enter another name.
Other Notes	Extreme valid test data, This file does not already exist, -failed

Test No.	**1.4.2**
Test	Can the user enter a file name correctly
Reason	So that their choices are saved and exported
Test Data	Kilesh Nundlall
Expected Result	Program continues
Code being tested	`except sqlite3.OperationalError:` `    UserName = input("Your chosen file name is not suitable as it may contain special characters. Please enter another name.\n")`
Actual Result	Your chosen file name is not suitable as it may contain special characters. Please enter another name.
Other Notes	This new response should inform the user that a suitable name is required. The problem occurred due to the space in the name. This was incompatible with SQL table names and threw an error. I provided a standard response to all errors (using Try Except) -pass

Test No.	**1.4.3**
Test	Can the item be fetched from the components database
Reason	So the items can be exported
Test Data	Using the program
Expected Result	Program continues

Code being tested	`cursor.execute('SELECT Name F`
	`for row in cursor:`
	`  ItemText = row[0]`

Actual Result	`sqlite3.OperationalError: near " ": syntax error`
Other Notes	Error if a user did not fill all of the components, -failed

Test No.	**1.4.4**
Test	Can the item be fetched from the components database
Reason	So the items can be exported
Test Data	Using the program
Expected Result	Program continues

Code being tested	`try:`
	`  cursor.execute('SELECT Name FR`
	`  for row in cursor:`
	`  ItemText = row[0]`
	`except sqlite3.OperationalError:`
	`  ItemText = 0`

Actual Result	Program continues
Other Notes	Fixed by using try except to make value 0 if no component is selected, -pass

Test No.	**1.4.5**
Test	Do the user's components save to the database
Reason	So the user can access their data later on
Test Data	Bhupinder
Expected Result	Saved
Code being tested	`cursor.execute("""CREATE TABLE {}(id INTEGER PRIMARY KEY, Component TEXT, Name TEXT) """.format(UserName))`
Actual Result	`UnboundLocalError: local variable 'UserName' referenced before assignment`
Other Notes	variable scope error, -failed

Test No.	**1.4.6**
Test	Do the user's components save to the database
Reason	So the user can access their data later on
Test Data	Bhupinder
Expected Result	Saved
Code being tested	`global UserName` `cursor.execute("""CREATE TABLE {}(id INTEGER PRIMARY KEY, Component TEXT, Name TEXT) """.format(UserName))`
Actual Result	`sqlite3.OperationalError: table Bhupinder already exists`
Other Notes	Valid test data, score error fixed, program has an error and stops, -failed

Test No.	**1.4.7**
Test	Do the user's components save to the database
Reason	So the user can access their data later on
Test Data	Bhupinder
Expected Result	Saved

Code being tested	`while True:`
	`  try:`
	`  cursor.execute (""" CREATE TABLE {}(id INTEGER PRIMARY KEY, Component TEXT, Name TEXT)""".format(`
	`  break`
	`  except sqlite3.OperationalError:`
	`  UserName = input("A file with this name already exists. Please enter another name.\n")`

Actual Result	A file with this name already exists. Please enter another name. Bhupinder Saved
Other Notes	Asks the user for another name, -pass

Test No.	**1.4.8**
Test	Does the data save correctly
Reason	So that the program works as intended
Test Data	Using the program
Expected Result	Data displayed correctly
Code being tested	`UserChoices[loops][1] = ItemText`
Actual Result	id Component Name 1 CPU 0 2 Motherboard AMD Ryzen 2600 3 Memory 0 4 Storage 0 5 Video Card 0 6 Power Supply 0 7 Case 0
Other Notes	All data saving one row below, -failed

Test No.	**1.4.9**
Test	Does the data save correctly
Reason	So that the program works as intended
Test Data	Using the program
Expected Result	Data displayed correctly
Code being tested	`UserChoices[loops-1][1] = ItemText`
Actual Result	
Other Notes	Python indexes from 0, not 1, therefore saving incorrectly, -pass

Actual Result

id	Component	Name
1	CPU	AMD Ryzen 2600
2	Motherboard	0
3	Memory	0
4	Storage	0
5	Video Card	0
6	Power Supply	0
7	Case	0

Test No.	**1.4.10**
Test	Does the data save correctly
Reason	So that the program works as intended
Test Data	Using the program
Expected Result	Data saved correctly

Code being tested	`for position in ComponentChoice:`
	`  while loops < 7:`
	`  try:`
	`  cursor.execute('SELECT Name FROM ' +`
	`  for row in cursor:`
	`  ItemText = row[0]`
	`  except sqlite3.OperationalError:`
	`  ItemText = 0`
	`  UserChoices[loops-1][1] = ItemText`
	`  loops = loops + 1`

Actual Result	Name FormFactor id 1 NZXT S340 ATX 2 Corsair 200R ATX 3 Cooler Master Lite 5 ATX 4 Fractal Design Focus G ATX 5 Thermaltake Core V21 Micro ATX What item would you like? 1 Saves as: (7, 'Case', '0')
Other Notes	Valid test data, data not saving, -failed

Test No.	**1.4.11**
Test	Does the data save correctly
Reason	So that the program works as intended
Test Data	Using the program
Expected Result	Data saved correctly
Code being tested	`while loops <= 7:`
Actual Result	(7, 'Case', 'NZXT S340')
Other Notes	Would not loop 7 times, therefore passing the case, -pass

FINAL CODE FOR MODULE 1.4 USERCOMPONENTS()

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def UserComponents():
    global UserName
    Database = sqlite3.connect("Components2.db")
    cursor = Database.cursor()

    loops = 0                                       #loops variable for number of loops
    UserChoices = [["CPU",""],                      #2d arrays store component category and component for export
                   ["Motherboard",""],
                   ["Memory",""],
                   ["Storage",""],
                   ["Video Card",""],
                   ["Power Supply",""],
                   ["Case",""]
                   ]

    for position in ComponentChoice:                #For each component category
        while loops <= 7:                           #while loops is less than or equal to 7
            try:
                cursor.execute('SELECT Name FROM ' + ComponentList[loops-1] + ' WHERE id = {}'.format(ComponentChoice[loops-1]))
                for row in cursor:                  #Gets the name of each component to save in the builds database
                    ItemText = row[0]               #[0] saves the component name without brackets
            except sqlite3.OperationalError:        #If there is an error
                ItemText = 0                        #Set the component to 0, meaning not choosen
            UserChoices[loops-1][1] = ItemText      #Set this component to the corresponding index in the 2d array
            loops = loops + 1                       #Manually increment loop

    db = sqlite3.connect("Customers.db")            #Connects to customers builds database
    cursor = db.cursor()
    while True:
        try:
            cursor.execute("""CREATE TABLE {}(id INTEGER PRIMARY KEY, Component TEXT, Name TEXT)""".format(UserName))
            break                                   #Only break loop if it can create a new table
        except sqlite3.OperationalError:            #Or get a new file name and try again
            UserName = input("A file with this name may already exists or this may contain special characters. Please enter another name.\n")
    cursor.executemany("""INSERT INTO """+UserName+""" Values (NULL,?,?)""",UserChoices)
    Database.commit()
    db.commit()                                     #Save the users components into their table
##    cursor.execute('''SELECT * FROM ''' +UserName)    #Use this method to view all the build without pandas
##    for row in cursor:
##        print(row)
    print("Saved")
    print(pandas.read_sql_query("SELECT * FROM "+UserName, db, "id")) #Print table using pandas
    Database.close()                                #Close the database to prevent data corruption
    db.close()
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Test No.	**1.5.1**
Test	Does the compatibility checking algorithm work
Reason	So that the program can inform the user when parts are incompatible
Test Data	CPU 2 (LGA 1151) and Mobo 2 (LGA 1151)
Expected Result	Parts are compatible
Code being tested	In final code
Actual Result	pass
Other Notes	Valid test data, -pass

Test No.	**1.5.2**
Test	Does the compatibility checking algorithm work
Reason	So that the program can inform the user when parts are incompatible
Test Data	CPU 1 (AM4) and Mobo 4(LGA 1151)
Expected Result	Parts are compatible
Code being tested	In final code
Actual Result	Incompatible, try again
Other Notes	Invalid test data, -pass

Test No.	**1.5.3**
Test	Does the compatibility checking algorithm work
Reason	So that the program can inform the user when parts are incompatible
Test Data	PSU 4 (ATX) and Case 2 (ATX)
Expected Result	Parts are compatible
Code being tested	In final code
Actual Result	pass
Other Notes	Valid test data, -pass

FINAL CODE FOR ALGORITHM 1.5 BASICMODE()

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def BasicMode():
    Database = sqlite3.connect("Components2.db")
    cursor = Database.cursor()

    while True:
        while True:
            #Socket compatiblity
            if ComponentChoice[0] and ComponentChoice[1] != "":                     #If a CPU and Motherboard have been selected...
                print("Checking socket compatibilty.")
                CPUCheck = cursor.execute('''SELECT Socket FROM CPU WHERE id = ''' + ComponentChoice[0]).fetchall()                #Save CPU socket
                MotherboardCheck = cursor.execute('''SELECT Socket FROM Motherboard WHERE id = ''' + ComponentChoice[1]).fetchall()     #Save Mobo socket
                if CPUCheck == MotherboardCheck:                                     #If the socket is the same, continue
                    print("Socket is Compatible.")
                    break
                else:
                    print("The socket is not compatible. Please choose another CPU or Motherboard.")
                    ItemSelection()                                                  #Run the module ItemSelection to allow them to choose another component
            else:
                print("Socket Compatibility check not required.")
                break
        while True:
            #FormFactor compatibility
            if ComponentChoice[1] and ComponentChoice[5] != "":                     #Motherboard and Power Supply
                print("Checking form factor compatibilty.")
                MotherboardCheck = cursor.execute('''SELECT FormFactor FROM Motherboard WHERE id = ''' + ComponentChoice[1]).fetchall()
                PowerSupplyCheck = cursor.execute('''SELECT FormFactor FROM PowerSupply WHERE id = ''' + ComponentChoice[5]).fetchall()
                if MotherboardCheck == PowerSupplyCheck:
                    print("Form Factor is Compatible-1")
                    break
                else:
                    print("The Form Factor is not compatible. Please choose another Power Supply or Motherboard.")
                    ItemSelection()


            else:
                print("Form Factor Compatibility check not required-1")
                break
        while True:
            if ComponentChoice[5] and ComponentChoice[6]!= "":                      #Power Supply and Case
                print("Checking form factor compatibilty.")
                PowerSupplyCheck = cursor.execute('''SELECT FormFactor FROM PowerSupply WHERE id = ''' + ComponentChoice[5]).fetchall()
                TowerCheck = cursor.execute('''SELECT FormFactor FROM Tower WHERE id = ''' + ComponentChoice[6]).fetchall()
                if PowerSupplyCheck == TowerCheck:
                    print("Form Factor is Compatible-2")
                    break
                else:
                    print("The Form Factor is not compatible. Please choose another Power Supply or Case.")
                    ItemSelection()
            else:
                print("Form Factor Compatibility check not required-2")
                break
        while True:
            if ComponentChoice[1] and ComponentChoice[6]!= "":                      #Motherboard and Case
                print("Checking form factor compatibilty.")
                MotherboardCheck = cursor.execute('''SELECT FormFactor FROM Motherboard WHERE id = ''' + ComponentChoice[1]).fetchall()
                TowerCheck = cursor.execute('''SELECT FormFactor FROM Tower WHERE id = ''' + ComponentChoice[6]).fetchall()
                if MotherboardCheck == TowerCheck:
                    print("Form Factor is Compatible-3")
                    break
                else:
                    print("The Form Factor is not compatible. Please choose another Case or Motherboard.")
                    ItemSelection()
            else:
                print("Form Factor Compatibility check not required-3")
                break
        break
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Test No.	**1.6.1**
Test	Does the compatibility checking algorithm work
Reason	To ensure that Basic mode works
Test Data	1
Expected Result	Compatibility check running
Code being tested	`if ModeInput == "1":` `    BasicMode()`
Actual Result	Program skips compatibility check
Other Notes	Compatibility check does not run, -failed

Test No.	**1.6.2**
Test	Does the compatibility checking algorithm work
Reason	To ensure that Basic mode works
Test Data	1
Expected Result	Compatibility check running
Code being tested	`if ModeInput == 1:` `    BasicMode()`
Actual Result	`Checking socket compatibilty.` `The socket is not compatible. Please choose another CPU or Motherboard.`
Other Notes	Compatibility check runs as expected, -pass

FINAL CODE FOR MAIN PROGRAM

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ComponentChoice = ["","","","","","",""]
Mode()
ItemSelection()
if ModeInput == 1:
    BasicMode()
UserComponents()
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

EXAMPLE OF USE

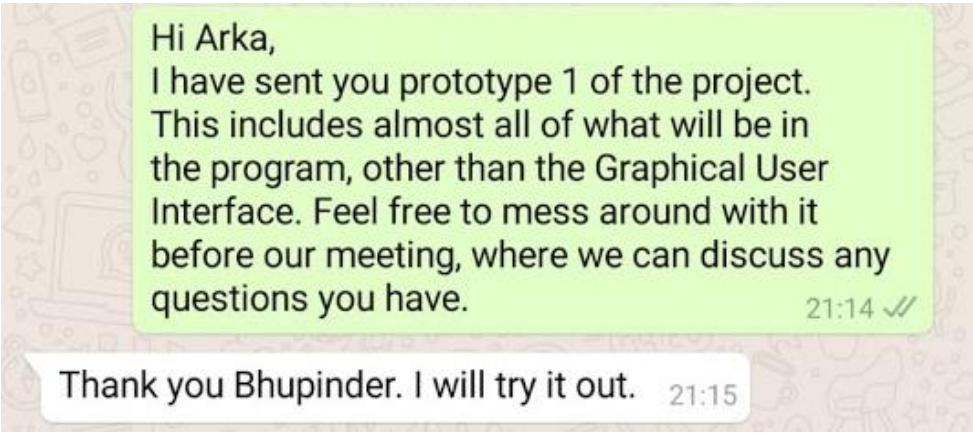
```
 ______ _______    ______        _ _     _ _                _______          _ _     _
(_____ (_______)  (____  \      (_) |   | (_)              (_______)        | || |   (_)  _
 _____) )         ____)  )_   _ _| | __| |_ ____   ____       _  ___   ___ | || |  _ _ _| |_
|  ____/ |       |  __  (| | | | |/ _  | |  _ \ / _  |     | |/ _ \ / _ \| || |_/ ) (_   _)
| |    | |_____  | |__)  ) |_| | | ( (_| | | | | ( (_| |     | | |_| | |_| | ||  _ (| | | |_
|_|    \_______) |______/|____/|_|\_)____|_|_| |_|\___ |     |_|\___/ \___/ \_)_| \_)_|  \__)
                                                  (_____|
What is your name (Will be used as the file name)?
BhupinderDhoofer

Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
3
In Basic mode, compatibility checks will be enabled throughout the program, meaning that the components that you
select will work with each other. Advanced mode is only for professionals as they will not have any compatibility
checks.

Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
5
***Please enter a valid number***


Would you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
1
What component would you like to select?
    1. CPU
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
1

                   Name Cores   Socket
id
1        AMD Ryzen 2600     6      AM4
2   Intel Core i5 6600k     4  LGA1151
3   Intel Core i5 8600k     6  LGA1151
4   Intel Core i7 8700k     6  LGA1151
5       AMD Ryzen 2700X     8      AM4

What item would you like?
1
-----------------------------------------------------------------------------------------------------
```

```
Do you want to select another component?
        1. Yes
        2. No
1

What component would you like to select?
    1. CPU
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
2
                    Name FormFactor   Socket
id
1         MSI Z370 A PRO        ATX  LGA1151
2         Gigabyte B360M  Micro ATX  LGA1151
3      Asus STRIX B350-F        ATX      AM4
4   MSI Z270 Gaming Plus        ATX  LGA1151
5     ASRock AB350M Pro4  Micro ATX      AM4

What item would you like?
1
-----------------------------------------------------------------------------------------------------

Do you want to select another component?
        1. Yes
        2. No
1

What component would you like to select?
    1. CPU
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
3

                        Name  Type Modules  Size
id
1      Corsair Vengeance LPX  DDR4   1x8GB   8GB
2        Team Vulcan T-Force  DDR4   2x8GB  16GB
3    Patriot Viper White LED  DDR4   2x8GB  16GB
4   G.Skill Ripjaws V Series  DDR4   2x4GB   8GB
5       Kingston HyperX Fury  DDR4  2x16GB  32GB

What item would you like?
2
-----------------------------------------------------------------------------------------------------
```

```
Do you want to select another component?
        1. Yes
        2. No
1

What component would you like to select?
    1. CPU
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
7

                      Name FormFactor
id
1                NZXT S340        ATX
2             Corsair 200R        ATX
3     Cooler Master Lite 5        ATX
4   Fractal Design Focus G        ATX
5     Thermaltake Core V21  Micro ATX

What item would you like?
1
-------------------------------------------------------------------------------------------------

Do you want to select another component?
        1. Yes
        2. No
2
Checking socket compatibilty.
The socket is not compatible. Please choose another CPU or Motherboard.
What component would you like to select?
    1. CPU
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
2

                     Name FormFactor   Socket
id
1         MSI Z370 A PRO        ATX  LGA1151
2         Gigabyte B360M  Micro ATX  LGA1151
3      Asus STRIX B350-F        ATX      AM4
4   MSI Z270 Gaming Plus        ATX  LGA1151
5     ASRock AB350M Pro4  Micro ATX      AM4

What item would you like?
3
-------------------------------------------------------------------------------------------------
```

```
Do you want to select another component?
        1. Yes
        2. No
2
Checking socket compatibilty.
Socket is Compatible.
Form Factor Compatibility check not required-1
Form Factor Compatibility check not required-2
Checking form factor compatibilty.
Form Factor is Compatible-3
Saved
       Component                Name
id
1            CPU       AMD Ryzen 2600
2    Motherboard    Asus STRIX B350-F
3         Memory  Team Vulcan T-Force
4        Storage                    0
5     Video Card                    0
6   Power Supply                    0
7           Case            NZXT S340
```

STAKEHOLDER REVIEW: PROTOTYPE 1



During the meeting, we discussed many aspects of the program. Arka said that everything was as he wanted it, however, one major aspect that had to be included was mentioned. A new requirement was that the user's component choices must be exported to a file on request, rather than just saved to a database. This could be a text file or PDF. I will include this in the next prototype.

PROTOTYPE 2

My main focus for prototype 2 is creating the GUI following all of Arka's requirements. I will not be considering the aesthetics, but rather the layout of the GUI elements.

Test No.	**2.1.1**
Test	Can the user enter a file name
Reason	So that their choices are saved and exported
Test Data	Arka
Expected Result	Program continues
Code being tested	`self.P1E1 = Entry(self.Page1)` `self.P1B1.configure(command = lambda : [self.AllTabs.select(self.Page2), UserName(self.P1E1.get())])`
Actual Result	Program continues

Other Notes	-pass, Lambda allows for the following code to be run, to store the contents of the username entry field. It also allows the button to go to the next page.
	`def UserName(UserName):`
	`  global UserNameV`
	`  UserNameV=UserName`

Test No.	**2.1.2**
Test	Can the user select the mode correctly
Reason	So the program works as the user wants
Test Data	Click on advanced mode
Expected Result	Program continues

Code being tested	`self.P1B2 = ttk.Button(self.Page1)`
	`self.P1B2.place(relx=0.01, rely=0.72, height=85, width=1876)`
	`self.P1B2.configure(text='''Build a PC - Advanced Mode''')`
	`self.P1B2.configure(command = lambda : [self.AllTabs.select(self.Page2), UserName(self.P1E1.get())])`

Actual Result	Continues to next page
Other Notes	-pass

FINAL CODE FOR MODULE 2.1 MODE SELECTION

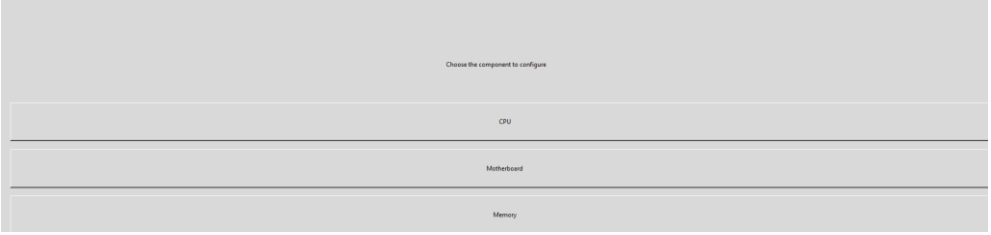
```
        self.P1L1 = Label(self.Page1)
        self.P1L1.place(relx=0.01, rely=0.02, height=211, width=1874)
        self.P1L1.configure(background="#0B0C10")
        self.P1L1.configure(foreground="#c5c6c7")
        self.P1L1.configure(text='''PC Building Toolkit''')

        self.P1B1 = Button(self.Page1)
        self.P1B1.place(relx=0.01, rely=0.35, height=315, width=1876)
        self.P1B1.configure(text='''Build a PC!''')
        self.P1B1.configure(command = lambda : [self.AllTabs.select(self.Page2), UserName(self.P1E1.get())])

        self.P1B2 = Button(self.Page1)
        self.P1B2.place(relx=0.01, rely=0.72, height=85, width=1876)
        self.P1B2.configure(text='''Build a PC - Advanced Mode''')
        self.P1B2.configure(command = lambda : [self.AllTabs.select(self.Page2), UserName(self.P1E1.get())])

        self.P1L2 = ttk.Label(self.Page1)
        self.P1L2.place(relx=0.01, rely=0.83, height=139, width=1866)
        self.P1L2.configure(background="#0B0C10")
        self.P1L2.configure(foreground="#c5c6c7")
        self.P1L2.configure(relief=FLAT)
        self.P1L2.configure(text='''What is the difference?

When using "Build a PC!", we will run a compatibility check to ensure that all of your parts will work together.
By using "Advanced Mode", we will not run the compatibility checker.''')

        self.P1E1 = Entry(self.Page1)
        self.P1E1.place(relx=0.32, rely=0.24, relheight=0.08, relwidth=0.67)
        self.P1E1.configure(background="#0B0C10")
        self.P1E1.configure(font=font10)
        self.P1E1.configure(foreground="#c5c6c7")
        self.P1E1.configure(insertbackground="black")

        self.P1L3 = Label(self.Page1)
        self.P1L3.place(relx=0.03, rely=0.24, height=71, width=544)
        self.P1L3.configure(background="#0B0C10")
        self.P1L3.configure(foreground="#c5c6c7")
        self.P1L3.configure(text='''What is your name?
This will be used as the file name''')
        self.P1L3.configure(justify=RIGHT)
```

```
def UserName(UserName):                     #Passing by reference
    global UserNameV
    UserNameV=UserName
    while True:
        for i in range(6):                  #If the file name is taken
            if UserChoices[i][1] != "":     #and a build has been created
                break
        self.AllTabs.select(self.Page5)     #Show final build page
        break
```

This is the code for page 1, where the mode selection takes place. I have used a very simple variable naming scheme, with the prefix P and a number standing for the relevant page and 3rd character standing for the object it is representing and its position in the page. For example, P1B1 is the first button on page 1. P1L1 is the first label on page one. P1E1 is the first entry box on page one. P2BB is the back button on page 2. P2FB is the forward button/skip on page 2.

Test No.	**2.2.1**
Test	Can the user select a component type
Reason	So a user can select the category correctly
Test Data	Click Case
Expected Result	Continues to next page with the relevant products

Code being tested	`self.P2B2 = Button(self.Page2)`
	`self.P2B2.place(relx=0.01, rely=0.29, height=74, width=1907)`
	`self.P2B2.configure(text='''Motherboard''')`
	`self.P2B2.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(2)])`
	
	`self.P2B3 = Button(self.Page2)`
	`self.P2B3.place(relx=0.01, rely=0.38, height=74, width=1907)`
	`self.P2B3.configure(text='''Memory''')`
	`self.P2B3.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(3)])`
	
	`self.P2B3 = Button(self.Page2)`
	`self.P2B3.place(relx=0.01, rely=0.38, height=74, width=1907)`
	`self.P2B3.configure(text='''Memory''')`
	`self.P2B3.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(3)])`

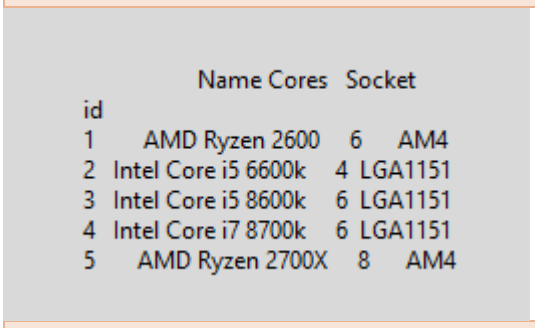
Actual Result	
Other Notes	-pass

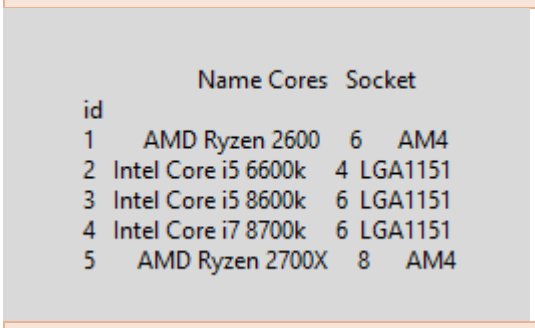
FINAL CODE FOR MODULE 2.2 COMPONENT SELECTION

```
self.P2L1 = Label(self.Page2)
self.P2L1.place(relx=0.01, rely=0.02, height=211, width=1874)
self.P2L1.configure(activebackground="#f9f9f9")
self.P2L1.configure(activeforeground="black")
self.P2L1.configure(background="#0B0C10")
self.P2L1.configure(foreground="#c5c6c7")
self.P2L1.configure(text='''Choose the component to configure''')

self.P2BB = Button(self.Page2)
self.P2BB.place(relx=-0.01, rely=0.93, height=75, width=966)
self.P2BB.configure(text='''Back''')
self.P2BB.configure(command = lambda : self.AllTabs.select(self.Page1))


self.P2B1 = Button(self.Page2)
self.P2B1.place(relx=0.01, rely=0.2, height=74, width=1907)
self.P2B1.configure(pady="0")
self.P2B1.configure(text='''CPU''')
self.P2B1.configure(width=1907)
self.P2B1.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(1)])

self.P2B2 = Button(self.Page2)
self.P2B2.place(relx=0.01, rely=0.29, height=74, width=1907)
self.P2B2.configure(pady="0")
self.P2B2.configure(text='''Motherboard''')
self.P2B2.configure(width=1907)
self.P2B2.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(2)])

self.P2B3 = Button(self.Page2)
self.P2B3.place(relx=0.01, rely=0.38, height=74, width=1907)
self.P2B3.configure(pady="0")
self.P2B3.configure(text='''Memory''')
self.P2B3.configure(width=1907)
self.P2B3.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(3)])

self.P2B4 = Button(self.Page2)
self.P2B4.place(relx=0.01, rely=0.47, height=74, width=1907)
self.P2B4.configure(pady="0")
self.P2B4.configure(text='''Storage''')
self.P2B4.configure(width=1907)
self.P2B4.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(4)])
```

```
self.P2B5 = Button(self.Page2)
self.P2B5.place(relx=0.01, rely=0.56, height=74, width=1907)
self.P2B5.configure(pady="0")
self.P2B5.configure(text='''Video Card''')
self.P2B5.configure(width=1907)
self.P2B5.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(5)])

self.P2B6 = Button(self.Page2)
self.P2B6.place(relx=0.01, rely=0.65, height=74, width=1907)
self.P2B6.configure(pady="0")
self.P2B6.configure(text='''Power Supply''')
self.P2B6.configure(width=1907)
self.P2B6.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(6)])

self.P2B7 = Button(self.Page2)
self.P2B7.place(relx=0.01, rely=0.74, height=74, width=1907)
self.P2B7.configure(pady="0")
self.P2B7.configure(text='''Case''')
self.P2B7.configure(width=1907)
self.P2B7.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(7)])

self.P2FB = Button(self.Page2)
self.P2FB.place(relx=0.51, rely=0.93, height=75, width=956)
self.P2FB.configure(text='''Skip''')
self.P2FB.configure(command = lambda : self.AllTabs.select(self.Page3))
```

I feel that the layout of the buttons is very good and easy to use. However, the module ItemSelection could be worked on to make the output bigger.

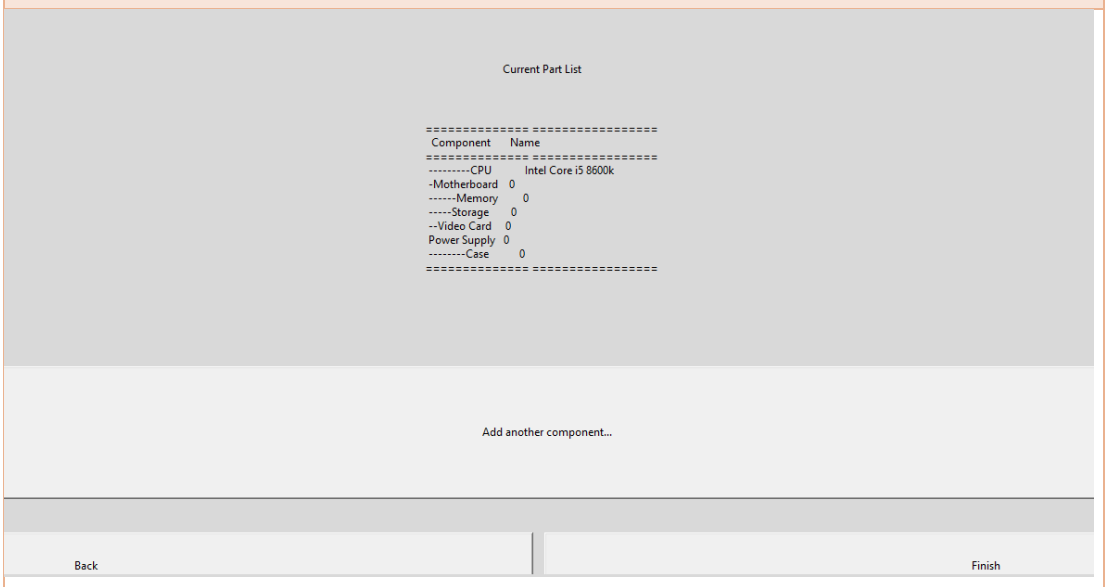
Test No.	**2.3.1**
Test	Can the program access the database
Reason	So the program can use the database
Test Data	CPU
Expected Result	Program displays all CPUs
Code being tested	`self.P2B1.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(1)]` `self.P3M1.configure(text=str(pandas.read_sql_query("SELECT * FROM {}".format(ComponentList[ComponentInput-1]), Database, "id")))`
Actual Result	
Other Notes	-pass, function ItemSelection() used from prototype 1

Test No.	**2.3.2**
Test	Does the database output look good
Reason	So the program is aesthetically pleasing
Test Data	CPU
Expected Result	Looks good with all information
Code being tested	`self.P2B1.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(1)]` `self.P3M1.configure(text=str(pandas.read_sql_query("SELECT * FROM {}".format(ComponentList[ComponentInput-1]), Database, "id")))`
Actual Result	
Other Notes	-pass, but slightly misaligned and could be bigger.

Test No.	**2.3.3**
Test	Can the user select the components required correctly
Reason	So the user can use the program correctly
Test Data	2 (No)
Expected Result	Stops asking for more components
Code being tested	

```
self.P4L1 = Label(self.Page4)
self.P4L1.place(relx=0.01, rely=0.02, height=211, width=1874)
self.P4L1.configure(text='''Current Part List''')

self.P4BB = ttk.Button(self.Page4)
self.P4BB.place(relx=-0.01, rely=0.93, height=75, width=966)
self.P4BB.configure(text='''Back''')
self.P4BB.configure(command = lambda : self.AllTabs.select(self.Page3))

self.P4L1 = Label(self.Page4)
self.P4L1.place(relx=0.01, rely=0.23, height=211, width=1874)
self.P4L1.configure(text='''Your chosen parts will show here''')

self.P4B1 = ttk.Button(self.Page4)
self.P4B1.place(relx=0.01, rely=0.66, height=145, width=1886)
self.P4B1.configure(text='''Add another component...''')
self.P4B1.configure(command = lambda : self.AllTabs.select(self.Page2))

self.P4B2 = ttk.Button(self.Page4)
self.P4B2.place(relx=0.5, rely=0.93, height=75, width=956)
self.P4B2.configure(text='''Finish''')
self.P4B2.configure(command = lambda : self.AllTabs.select(self.Page5))
```

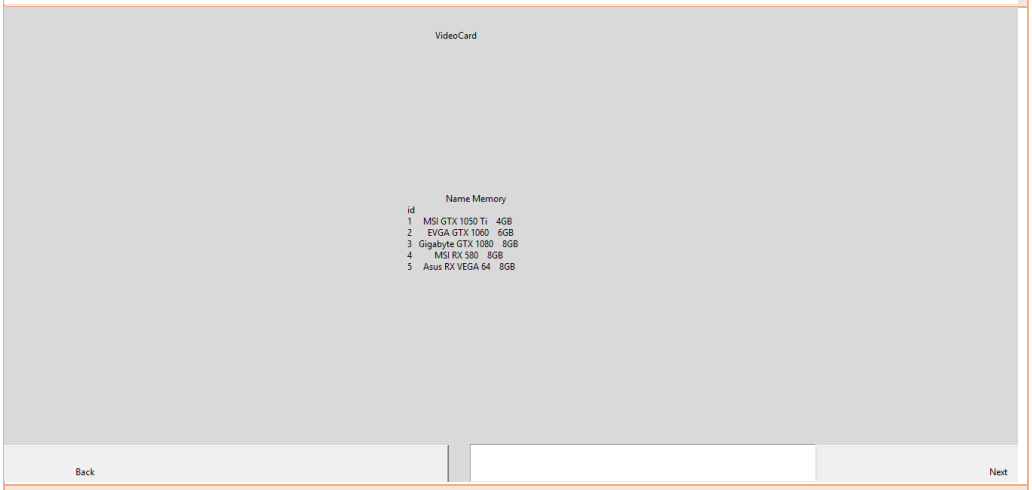
Actual Result



Other Notes	-pass

Test No.	**2.3.4**
Test	Can the user select the item with ease
Reason	So the program is easy to use
Test Data	Choose an item
Expected Result	Program continues

Code being tested	`self.P3E1 = Entry(self.Page3)`
	`self.P3E1.place(relx=0.51, rely=0.93, relheight=0.07, relwidth=0.24)`
	
	`self.P3BF = ttk.Button(self.Page3)`
	`self.P3BF.place(relx=0.75, rely=0.93, height=75, width=486)`
	`self.P3BF.configure(text='''Next''')`
	`self.P3BF.configure(command = lambda : [self.AllTabs.select(self.Page4),ItemSave(self.P3E1.get()),UserComponents()])`

Actual Result	
Other Notes	-pass, entry field used

FINAL CODE FOR MODULE 2.3 ITEMSELECTION()

```
ComponentChoice = ["","","","","","","",""]
ComponentList = ["CPU","Motherboard","Memory","Storage","VideoCard","PowerSupply","Tower"]
def ItemSelection(ComponentInput):
    global ComponentInputV
    ComponentInputV = ComponentInput
    Database = sqlite3.connect("Components2.db")
    cursor = Database.cursor()
    self.P3L1.configure(text=str(pandas.read_sql_query("SELECT * FROM {}".format(ComponentList[ComponentInput-1]), Database, "id")))
    #Prints pandas table to label with the id as the index
```

This module is now smaller than in prototype 1 as a lot of the loops and validations are not needed. For example, if using a Command Line Interface, we would need to ask the user for a component category, and if they entered an incorrect one, error handling code would be required. Now with the GUI, the user is unable to make a mistake, as the options are clickable buttons. If they do click the wrong one, they can click the back button.

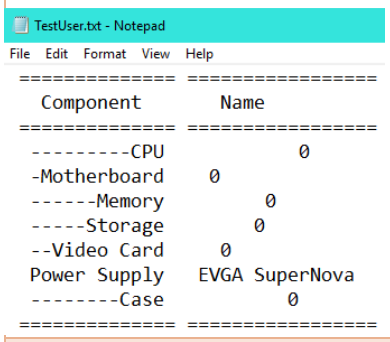
Test No.	**2.4.1**
Test	Do the user's components save to a text file or PDF
Reason	So the user can export their choice
Test Data	TestUser, Power Supply, 1
Expected Result	Saved to file TestUser.txt

Code being tested

```
            table = """

 ============== =================
   Component       Name
 ============== =================
  ---------CPU            {}
  -Motherboard    {}
  ------Memory         {}
  -----Storage        {}
  --Video Card     {}
  Power Supply   {}
  --------Case           {}
 ============== =================

""".format(UserChoices[0][1],UserChoices[1][1],UserChoices[2][1],
           UserChoices[3][1],UserChoices[4][1],UserChoices[5][1],
           UserChoices[6][1])
            self.P4L1.configure(text=table,justify = LEFT)
            self.P5L2.configure(text=table,justify = LEFT)
```

```
file = open(UserNameV+".txt","w+")
file.write(table)
file.close()
```

Actual Result



Other Notes	-pass

FINAL CODE FOR EXPORTING THE BUILD

```
        def UserComponents():
            global table
            global UserChoices
            Database = sqlite3.connect("components2.db")
            cursor = Database.cursor()

            loops = 0                                                   #loops variable for number of loops
            UserChoices = [["CPU",""],                                  #2d arrays store component category and component for export
                           ["Motherboard",""],
                           ["Memory",""],
                           ["Storage",""],
                           ["Video Card",""],
                           ["Power Supply",""],
                           ["Case",""]
                           ]

            for position in ComponentChoice:                #For each component category
                while loops <= 7:                           #while loops is less than or equal to 7
                    try:
                        cursor.execute('SELECT Name FROM ' + ComponentList[loops-1] + ' WHERE id = {}'.format(ComponentChoice[loops-1]))
                        for row in cursor:                  #Gets the name of each component to save in the builds database
                            ItemText = row[0]               #[0] saves the component name without brackets
                    except sqlite3.OperationalError:        #If there is an error
                        ItemText = 0                        #Set the component to 0, meaning not choosen
                    UserChoices[loops-1][1] = ItemText      #Set this component to the corresponding index in the 2d array
                    loops = loops + 1                       #Manually increment loop

            table = """

 ============== =================
   Component    \tName
 ============== =================
  ---------CPU  \t{}
  -Motherboard  \t{}
  ------Memory  \t{}
  -----Storage  \t{}
  --Video Card  \t{}
  Power Supply  \t{}
  --------Case  \t{}
 ============== =================

""".format(UserChoices[0][1],UserChoices[1][1],UserChoices[2][1],UserChoices[3][1],UserChoices[4][1],UserChoices[5][1],UserChoices[6][1])
            self.P4L1.configure(text=table,justify = LEFT)
            self.P5L2.configure(text=table,justify = LEFT)              #Displays table

        def Export():
            global UserNameV
            global UserChoices
            file = open(UserNameV+".txt","w+")                  #Will create or ammend the file
            file.write(table)                                   #Writes the contents of variable table to the file
            file.close()                                        #Closes file to prevent data corruption

            db = sqlite3.connect("Customers.db")                #Connects to customers builds database
            cursor = db.cursor()
            try:
                cursor.execute("""CREATE TABLE {} (id INTEGER PRIMARY KEY, Component TEXT, Name TEXT)""".format(UserNameV))    #Creates table
                cursor.executemany("""INSERT INTO """+UserNameV+""" Values (NULL,?,?)""",UserChoices)                          #Saves build to database
                db.commit()
                cursor.execute('''SELECT * FROM ''' +UserNameV)
                print("Saved")
                print(pandas.read_sql_query("SELECT * FROM "+UserNameV, db, "id"))
                self.P4L1.configure(text=pandas.read_sql_query("SELECT * FROM "+UserNameV, db, "id"))                           #Displays current build
                db.close()
            except(sqlite3.OperationalError, UnboundLocalError):                                                                #If an error occurs.
                tkMessageBox.showerror("File Name Error", "A file with this name may already exists or this may contain special characters. Please enter another name and try again.")
                self.AllTabs.select(self.Page1)                                                                                 #Asks the user to try another file name and try again
```

As per the stakeholder requirements, I can now export the user's build to a text file, if they want. I have also cleaned up the output so that it looks nicer.

Test No.	**2.5.1**
Test	Do the forward, back and other buttons work
Reason	So that the program can be navigated
Test Data	Using the program
Expected Result	Program goes to the relevant page
Code being tested	`self.P3BB.configure(command = lambda : self.AllTabs.select(self.Page2))`
Actual Result	Goes to the previous page
Other Notes	-pass

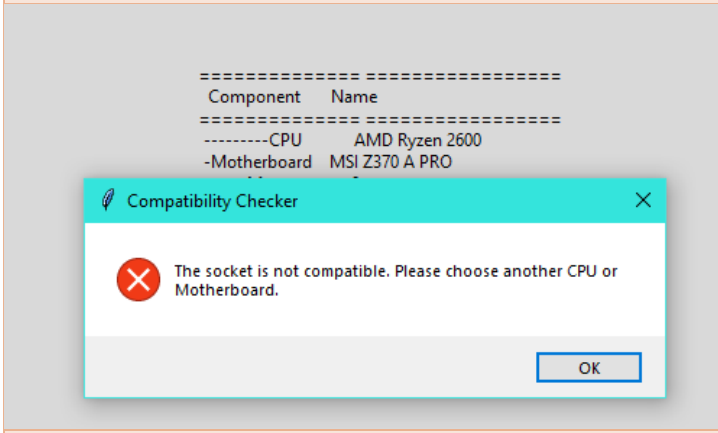
Test No.	**2.5.2**
Test	Does the compatibility checking algorithm work
Reason	To ensure that Basic mode works
Test Data	CPU = 1, Motherboard = 1
Expected Result	Compatibility check running, error shown
Code being tested	BasicMode()
Actual Result	No error show
Other Notes	-failed

Test No.	**2.5.3**
Test	Does the compatibility checking algorithm work
Reason	To ensure that Basic mode works
Test Data	CPU = 1, Motherboard = 1
Expected Result	Compatibility check running, error shown

Code being tested	`self.P4B2 = ttk.Button(self.Page4)`
	`self.P4B2.place(relx=0.5, rely=0.93, height=75, width=956)`
	`self.P4B2.configure(text='''Finish''')`
	`self.P4B2.configure(command = lambda : [self.AllTabs.select(self.Page5), BasicMode()])`

Actual Result	Works, but error only shown in CLI
Other Notes	-failed, must create a (popup) error

Test No. **2.5.4**

Test	Does the compatibility checking algorithm work
Reason	To ensure that Basic mode works
Test Data	CPU = 1, Motherboard = 1
Expected Result	Compatibility check running, error shown
Code being tested	`tkMessageBox.showerror("Compatibility Checker", "The socket is not compatible. Please` `self.AllTabs.select(self.Page2)`
Actual Result	
Other Notes	-pass

STAKEHOLDER REVIEW: PROTOTYPE 2

After completing this prototype, I set up a meeting with Arka. Here, he would be able to use the program on his hardware and we could discuss what changes and improvements had to be made. Below is a summary of our final decisions.

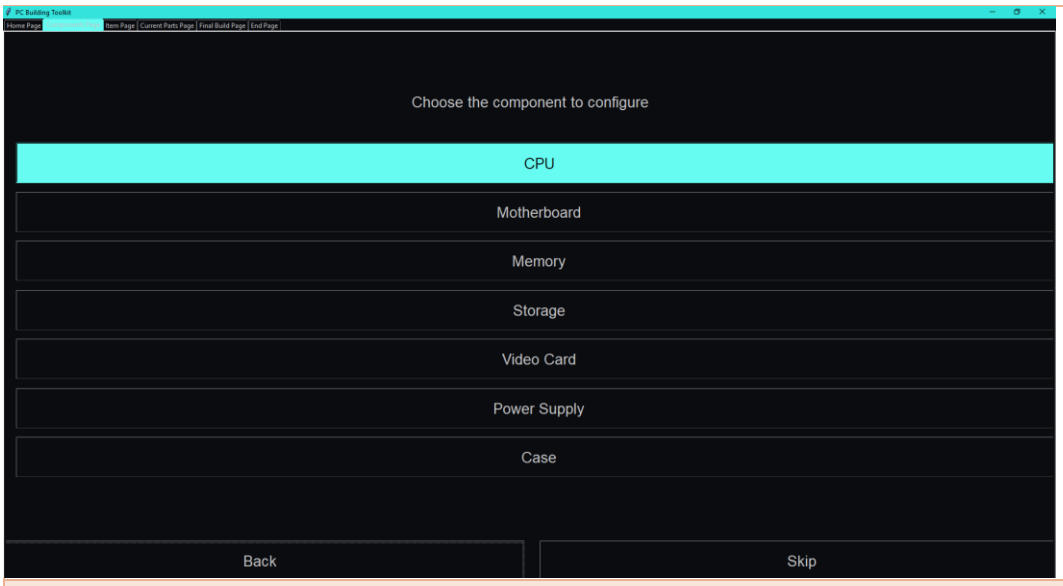
- The colour scheme must be implemented. Arka would first like to try a black background, cyan accent colour and white text.
- The database must be represented in a better way. Currently, it is not aligned.
- He likes the exported text file for the user and that they are also saved to an SQL database for his business.
- Add a feedback form on the final page.

PROTOTYPE 3

The key focus for prototype 3 is to make the program aesthetically pleasing and to add a few more features.

Test No.	**3.1.1**
Test	Is the colour scheme suitable
Reason	So the program suites the stakeholder's needs
Test Data	Black, blue and white
Expected Result	Aesthetically pleasing

Code being tested	`def theme(widget):`
	`  widget.configure(activebackground="#66FCF1")  #Cyan, Active Background colour`
	`  widget.configure(activeforeground="#0B0C10")  #Black, Active Foreground colour`
	`  widget.configure(background="#0B0C10")  #Black, Background colour`
	`  widget.configure(foreground="#c5c6c7")  #Gray, Foreground colour`

Actual Result	
Other Notes	-pass, all colours correctly implemented

Test No.	**3.1.2**
Test	Database is aligned correctly
Reason	So the program suites the stakeholder's needs
Test Data	Selected items in the database
Expected Result	Printed correctly

Code being tested	`def ItemSelection(ComponentInput):`
	`  global ComponentInputV`
	`  ComponentInputV = ComponentInput`
	`  Database = sqlite3.connect("Components2.db")`
	`  cursor = Database.cursor()`
	`  tabless = []`
	`  colname = []`
	`  for column in cursor.execute("SELECT * FROM CPU").description:`
	`  colname.append(column[0])`
	`  cursor.execute('''SELECT * FROM ''' + ComponentList[ComponentInput-1])`
	`  rowss = []`
	`  for row in cursor:`
	`  rowss.append(row)`
	`  print('\t '.join(rowss))`
	`  self.P3L1.configure(text=ComponentList[ComponentInput-1])`

Actual Result	`  print('\t '.join(rowss))` `TypeError: sequence item 0: expected str instance, tuple found`
Other Notes	-fail, this method cannot be used due to the dataframe

Test No.	**3.1.3**
Test	Database is aligned correctly
Reason	So the program suites the stakeholder's needs
Test Data	Selected items in the database
Expected Result	Printed correctly

Code being tested	`cursor.execute('''SELECT * FROM ''' + ComponentList[ComponentInput-1])`
	`rowss = []`
	`for row in cursor:`
	`  rowss[0].extend((row[0],row[1],row[2],row[3]))`
	`print(rowss)`
	`tabless = colname + rowss`
	`texttj = ('\t '.join(map(str,tabless)))`
	`self.P3L2.configure(wraplength = 1000,text=texttj)`

Actual Result	`  rowss[0].extend((row[0],row[1],row[2],row[3]))` `IndexError: list index out of range`
Other Notes	-fail, cannot easily change data type by putting data into 2D arrays

Test No.	**3.1.4**
Test	Database is aligned correctly
Reason	So the program suites the stakeholder's needs
Test Data	Selected items in the database
Expected Result	Printed correctly

Code being tested	`colname = []`
	`for column in cursor.execute("SELECT * FROM {}".format(ComponentList[ComponentInput-1])).description:`
	`  colname.append(column[0])`
	`cursor.execute('''SELECT * FROM ''' + ComponentList[ComponentInput-1])`
	`rowss = []`
	`for row in cursor:`
	`  rowss.extend(row)`
	`tabless = colname + rowss`
	`texttj = ('\t '.join(map(str,tabless)))`
	`self.P3L2.configure(wraplength = 1000,text=texttj)`

Actual Result	id Name Cores Socket 1 AMD Ryzen 2600 6 AM4 2 Intel Core i5 6600k 4 LGA1151 3 Intel Core i5 8600k 6 LGA1151 4 Intel Core i7 8700k 6 LGA1151 5 AMD Ryzen 2700X 8 AM4
Other Notes	-fail, no new lines but equally spaced content

Test No.	**3.1.5**
Test	Database is aligned correctly
Reason	So the program suites the stakeholder's needs
Test Data	Selected items in the database
Expected Result	Printed correctly

Code being tested	`cursor = Database.cursor()`
	`self.P3L2.configure(text=str(pandas.read_sql_query("SELECT * FROM {}".format(ComponentList[ComponentInput-1]), Database, "id")))`
	`self.P3L1.configure(text=ComponentList[ComponentInput-1])`

Actual Result	Name Cores Socket id 1 AMD Ryzen 2600 6 AM4 2 Intel Core i5 6600k 4 LGA1151 3 Intel Core i5 8600k 6 LGA1151 4 Intel Core i7 8700k 6 LGA1151 5 AMD Ryzen 2700X 8 AM4
Other Notes	-fail, not alighted and condensed

Test No.	**3.1.6**
Test	Database is aligned correctly
Reason	So the program suites the stakeholder's needs
Test Data	Selected items in the database
Expected Result	Printed correctly
Code being tested	`self.P3L2 = Button(self.Page3)`
Actual Result	No change in alignment
Other Notes	-fail, same as 22.3

Test No.	**3.1.7**
Test	Database is aligned correctly
Reason	So the program suites the stakeholder's needs
Test Data	Selected items in the database
Expected Result	Printed correctly

Code being tested	`x = []`
	`for column in cursor.execute("SELECT * FROM {}".format(ComponentList[ComponentInput-1])).description:`
	`  x.append(column[0])`
	`for row in cursor:`
	`  x.extend(row)`
	`print(x)`
	`for i in x:`
	`  if isinstance(i,int) == True:`
	`  print("")`
	`  print(i, end = "|\t|")`
	`print("\n")`
	`self.P3L2.configure(text=x)`

Actual Result	id| |Name| |Cores| |Socket| | 1| | AMD Ryzen 2600| |6| |AM4 | | 2| |Intel Core i5 6600k| |4| |LGA1151| | 3| |Intel Core i5 8600k| |6| |LGA1151| | 4| |Intel Core i7 8700k| |6| |LGA1151| | 5| | AMD Ryzen 2700X| |8| |AM4 | |
Other Notes	-improvement, new format in CLI

Test No.	**3.1.8**
Test	Database is aligned correctly
Reason	So the program suites the stakeholder's needs
Test Data	Selected items in the database
Expected Result	Printed correctly

Code being tested	`cursor.execute('''SELECT * FROM ''' + ComponentList[ComponentInput-1])`
	`x = []`
	`y=""`
	`for column in cursor.execute("SELECT * FROM {}".format(ComponentList[ComponentInput-1])).description:`
	`    x.append(column[0])`
	`for row in cursor:`
	`    x.extend(row)`
	`for i in x:`
	`    if isinstance(i,int) == True:`
	`        print("")`
	`        y=y+"\n"`
	`    print(i, end = "|\t|")`
	`    y=y+str(i)+"|\t|"`
	`print("\n")`
	`y=y+"\n"`
	`print(y)`
	`self.P3L2.configure(text=y)`

Actual Result	id| |Name| |Cores| |Socket| | 1| |AMD Ryzen 2600| |6| |AM4| | 2| |Intel Core i5 6600k| |4| |LGA1151| | 3| |Intel Core i5 8600k| |6| |LGA1151| | 4| |Intel Core i7 8700k| |6| |LGA1151| | 5| |AMD Ryzen 2700X| |8| |AM4| |
Other Notes	-improvement, string concatenation used to output to GUI

Test No.	**3.1.9**
Test	Database is aligned correctly
Reason	So the program suites the stakeholder's needs
Test Data	Selected items in the database
Expected Result	Printed correctly
Code being tested	

```
def ItemSelection(ComponentInput):
    global ComponentInputV
    ComponentInputV = ComponentInput
    self.P3L1.configure(text=ComponentList[ComponentInput-1])
    Database = sqlite3.connect("Components2.db")
    cursor = Database.cursor()
    cursor.execute('''SELECT * FROM ''' + ComponentList[ComponentInput-1])
    x = []
    y=""
    for column in cursor.execute("SELECT * FROM {}".format(ComponentList[ComponentInput-1])).description:
        x.append(column[0])
    for row in cursor:
        x.extend(row)
    for i in x:
        if isinstance(i,int) == True:
            print("")
            y=y+"\n\n"
        print("{:<25s}".format(str(i)),end="")
        y=y+"{:<40s}".format(str(i))

    print("\n")
    y=y+"\n"
    print(y)
    self.P3L2.configure(text=y)
```

Actual Result

id	Name	Cores	Socket
1	AMD Ryzen 2600	6	AM4
2	Intel Core i5 6600k	4	LGA1151
3	Intel Core i5 8600k	6	LGA1151
4	Intel Core i7 8700k	6	LGA1151
5	AMD Ryzen 2700X	8	AM4

Other Notes -pass, a lot more spread out and aligned

FINAL CODE FOR MODULE 3.1 DATABASE LAYOUT

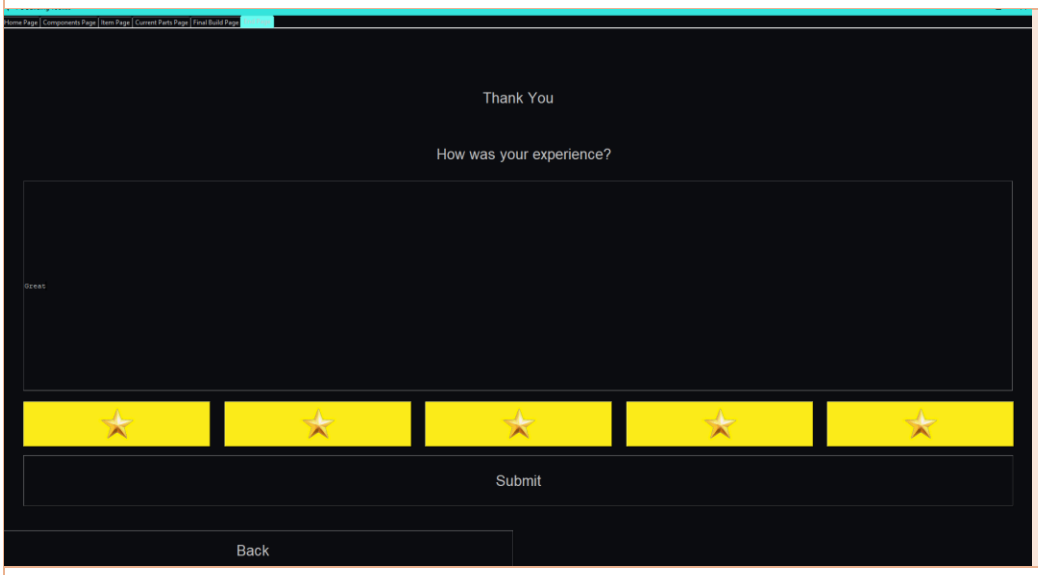
```
def ItemSelection(ComponentInput):
    global ComponentInputV
    ComponentInputV = ComponentInput
    self.P3L1.configure(text=ComponentList[ComponentInput-1])              #To meet stakeholder requirements,
                                                                           #Shows category name as title
    Database = sqlite3.connect("Components2.db")
    cursor = Database.cursor()
    cursor.execute('''SELECT * FROM ''' + ComponentList[ComponentInput-1])  #Fetches all components from category
    x = []                                                                 #empty array
    y=""                                                                   #empty string

    for column in cursor.execute("SELECT * FROM {}".format(ComponentList[ComponentInput-1])).description:
                                                                           #Gets table headings...
        x.append(column[0])                                                #Stores them in array x
    for row in cursor:
        x.extend(row)                                                      #Adds all components to array
    for i in x:                                                            #This is to format the text correctly
        if isinstance(i,int) == True:                                      #When the index is the id as int,
            print("")                                                     #Prints a new row, for diagnostics
            y=y+"\n\n"                                                     #Adds a new row to the string
        print("{:<25s}".format(str(i)),end="")                             #The sting will be 25 characters
                                                                           #Left aligned, not printing a new line
        y=y+"{:<40s}".format(str(i))                                       #Adds to string, left aligned 40 char
    print("\n")                                                           #New line
    y=y+"\n"                                                           #New Line
    print(y)                                                           #For diagnostics/monitoring
    self.P3L2.configure(text=y)                                        #Prints on label
```

Test No.	**3.2.1**
Test	Feedback form displays
Reason	So that Arka knows how to improve the service
Test Data	Great program, 5 stars
Expected Result	"Great program, 5 stars"

Code being tested	`self.P6L2 = Label(self.Page6)`
	`self.P6L2.place(relx=0.01, rely=0.21, height=41, width=1894)`
	`self.P6L2.configure(background="#0B0C10")`
	`self.P6L2.configure(foreground="#c5c6c7")`
	`self.P6L2.configure(text='''How was your experience?''')`
	`self.P6L2.configure(width=1894)`
	
	`self.P6E1 = Entry(self.Page6)`
	`self.P6E1.place(relx=0.02, rely=0.28, relheight=0.39, relwidth=0.96)`
	`self.P6E1.configure(background="#0B0C10")`
	`self.P6E1.configure(font=font10)`
	`self.P6E1.configure(foreground="#c5c6c7")`
	`self.P6E1.configure(insertbackground="black")`
	`self.P6E1.configure(width=1834)`
	
	`self.star = PhotoImage(file="star.gif")`
	
	`self.P6B1 = Button(self.Page6)`
	`self.P6B1.place(relx=0.02, rely=0.69, height=84, width=347)`
	`self.P6B1.configure(image=self.star, command = lambda : ChooseStar(1))`
	...

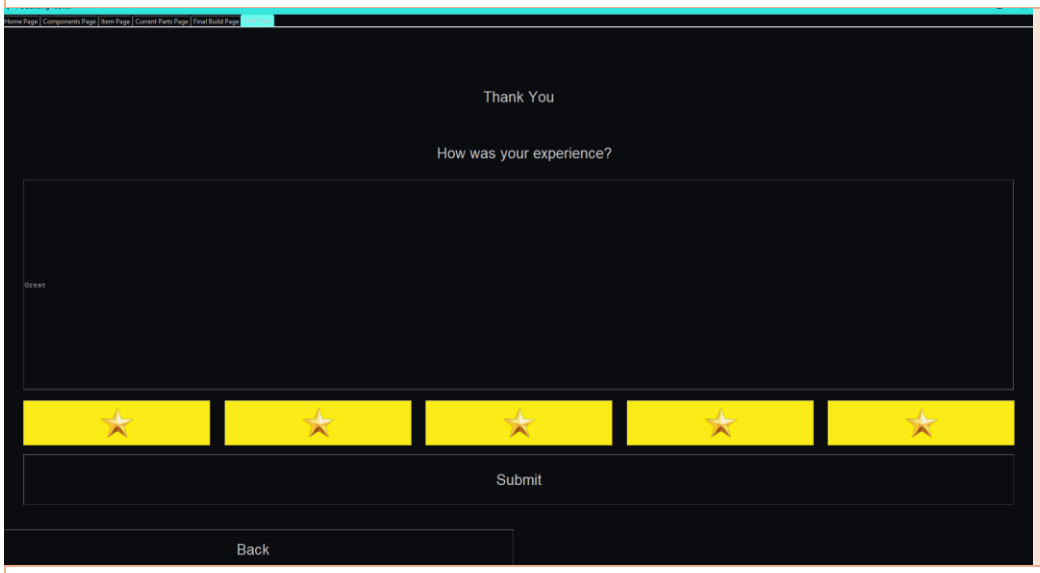
Actual Result	Displays correctly
Other Notes	Pass

```
self.P6B5 = Button(self.Page6)
self.P6B5.place(relx=0.8, rely=0.69, height=84, width=347)
self.P6B5.configure(image=self.star, command = lambda : ChooseStar(5))

self.P6FB = Button(self.Page6)
self.P6FB.place(relx=0.02, rely=0.79, height=95, width=1836)
self.P6FB.configure(text='''Submit''')
self.P6FB.configure(command = lambda : [self.P6FB.configure(text="Submitted"), Feedback(star)])
star = 0
def ChooseStar(x):
    global star
    star = x
    stararray = [self.P6B1,self.P6B2,self.P6B3,self.P6B4,self.P6B5]
    for i in range(x):
        stararray[i].configure(background="#FBEB19")               #Yellow
    i+=1
    while True:
        if i != 5:
            stararray[i].configure(background="#0B0C10")
            i+=1
        else:
            break
```

Test No.	**3.2.2**
Test	Feedback form saves
Reason	So that Arka knows how to improve the service
Test Data	
Expected Result	"Great program, 5 stars" saved to SQL database

Code being tested	`def Feedback(star):`
	`  Comment = self.P6E1.get()`
	`  db = sqlite3.connect("Customers.db")`
	`  cursor = db.cursor()`
	`  cursor.execute("""CREATE TABLE {}Feedback (id INTEGER PRIMARY KEY, Stars TEXT, Comment TEXT)""".format(UserNameV))`
	`  cursor.executemany("""INSERT INTO """+UserNameV+"""Feedback Values (NULL,?,?)""",[star,Comment])`
	`  db.commit()`
	`  db.close()`
	`  print(pandas.read_sql_query("SELECT * FROM "+UserNameV+"Feedback", db, "id"))`

Actual Result	`ValueError: parameters are of unsupported type`
Other Notes	-fail, SQL error

Test No.	**3.2.2**
Test	Feedback form saves
Reason	So that Arka knows how to improve the service
Test Data	
Expected Result	"Great, 5 stars" saved to SQL database

Code being tested	`def Feedback():`
	`  Comment = self.P6E1.get()`
	`  db = sqlite3.connect("Customers.db")`
	`  cursor = db.cursor()`
	`  try:`
	`  cursor.execute("""CREATE TABLE {}Feedback (id INTEGER PRIMARY KEY, Stars TEXT, Comment TEXT)""".format(UserNameV))`
	`  continueExport = 1`
	`  except (sqlite3.OperationalError, UnboundLocalError):`
	`  tkMessageBox.showerror("File Name Error", "A file with this name may already exists or this may contain special cha`
	`  self.AllTabs.select(self.Page1)`
	`  if continueExport ==1:`
	`  cursor.execute("""INSERT INTO """+UserNameV+"""Feedback (Stars, Comment) Values (?,?)""",(star,Comment))`
	`  db.commit()`
	`  print(pandas.read_sql_query("SELECT * FROM "+UserNameV+"Feedback", db, "id"))`
	`  db.close()`

Actual Result	`  id Stars Comment` `0  1  5  Great`
Other Notes	-pass, saved correctly with error handling

FINAL CODE FOR MODULE 3.2 FEEDBACK

```
def ChooseStar(x):
    global star
    star = x                                                    #Stores the number of stars provided from button input
    stararray = [self.P6B1,self.P6B2,self.P6B3,self.P6B4,self.P6B5]
    for i in range(star):
        stararray[i].configure(background="#FBEB19")            #Makes all buttons Yellow
    i+=1
    while True:
        if i != 5:                                              #If 5(i) stars is not given
            stararray[i].configure(background="#0B0C10")        #Makes the last star black
            i+=1
        else:
            breakpoint                                          #Continues till only the clicked stars and previous are yellow

def Feedback():
    Comment = self.P6E1.get()                                   #Fetches the feedback from the entry box
    db = sqlite3.connect("Customers.db")
    cursor = db.cursor()
    try:
        cursor.execute("""CREATE TABLE {}Feedback (id INTEGER PRIMARY KEY, Stars TEXT, Comment TEXT)""".format(UserNameV))
                                                                #Inserts the comment and number of stars to the database
        cursor.execute("""INSERT INTO """+UserNameV+"""Feedback (Stars, Comment) Values (?,?)""",(star,Comment))
        db.commit()
        print(pandas.read_sql_query("SELECT * FROM "+UserNameV+"Feedback", db))
        db.close()
    except (sqlite3.OperationalError, UnboundLocalError):       #If an error occurs..
        tkMessageBox.showerror("File Name Error", "A file with this name may already exists or this may contain special characters. Please enter another name and try again.")
        self.AllTabs.select(self.Page1)                         #Asks the user to try another name
```

Test No.	**3.3,1**
Test	Program is fluid
Reason	Reduce the number of clicks needed if an error occurs
Test Data	No file name added
Expected Result	The program must go to the final page after adding a file name after an error

Code being tested	`def UserName(UserName):`
	`  global UserNameV`
	`  UserNameV=UserName`
	`  while True:`
	`  for i in range(6):`
	`  print(UserChoices[i][1])`
	`  if UserChoices[i][1] != "":`
	`  break`
	`  self.AllTabs.select(self.Page5)`
	`  break`

Actual Result	Returns to the final page if the program has been used
Other Notes	-pass

Test No.	**3.3,2**
Test	Program is consistent
Reason	To make the program clean and aesthetic
Test Data	
Expected Result	All buttons are black

Code being tested	`def theme(widget):`
	`  widget.configure(activebackground="#66FCF1")  #Cyan, Active Background colour`
	`  widget.configure(activeforeground="#0B0C10")  #Black, Active Foreground colour`
	`  widget.configure(background="#0B0C10")  #Black, Background colour`
	`  widget.configure(foreground="#c5c6c7")  #Gray, Foreground colour`
	
	`theme(self.P1B1)`
	`theme(self.P1B2)`
	`theme(self.P2BB)`
	`theme(self.P2B1)`
	`theme(self.P2B2)`
	`theme(self.P2B3)`
	`theme(self.P2B4)`
	`theme(self.P2B5)`
	`theme(self.P2B6)`
	`theme(self.P2B7)`
	`theme(self.P2FB)`
	`theme(self.P3BB)`
	`theme(self.P3FB)`
	`theme(self.P4BB)`
	`theme(self.P4B1)`
	`theme(self.P4B2)`
	`theme(self.P5BB)`
	`theme(self.P5B1)`
	`theme(self.P5B2)`
	`theme(self.P5BB)`
	`theme(self.P6BB)`
	`theme(self.P6B1)`
	`theme(self.P6B2)`
	`theme(self.P6B3)`
	`theme(self.P6B4)`
	`theme(self.P6B5)`
	`theme(self.P6FB)`

Actual Result	All buttons were black
Other Notes	-pass, but this was a waste of code and storage space

Test No. **3.3,3**

Test	Program is consistent and efficient
Reason	To make the program clean, aesthetic and efficient on resources
Test Data	
Expected Result	All buttons are black

Code being tested	`def theme(widget):`
	`  widget.configure(activebackground="#66FCF1")  #Cyan, Active Background colour`
	`  widget.configure(activeforeground="#0B0C10")  #Black, Active Foreground colour`
	`  widget.configure(background="#0B0C10")  #Black, Background colour`
	`  widget.configure(foreground="#c5c6c7")  #Gray, Foreground colour`
	
	`ButtonsToTheme = [self.P1B1,self.P1B2,self.P2BB,self.P2B1,self.P2B2,self.P2B3,self.P2B4,`
	`  self.P2B5,self.P2B6,self.P2B7,self.P2FB,self.P3BB,self.P3FB,self.P4BB,`
	`  self.P4B1,self.P4B2,self.P5BB,self.P5B1,self.P5B2,self.P5BB,self.P6BB,`
	`  self.P6B1,self.P6B2,self.P6B3,self.P6B4,self.P6B5,self.P6FB]`
	
	`for i in ButtonsToTheme:  #For all of the buttons in the array`
	`  theme(i)  #Theme them`

Actual Result	All buttons were black
Other Notes	Using a loop reduces the amount of code but has the same outcome-pass

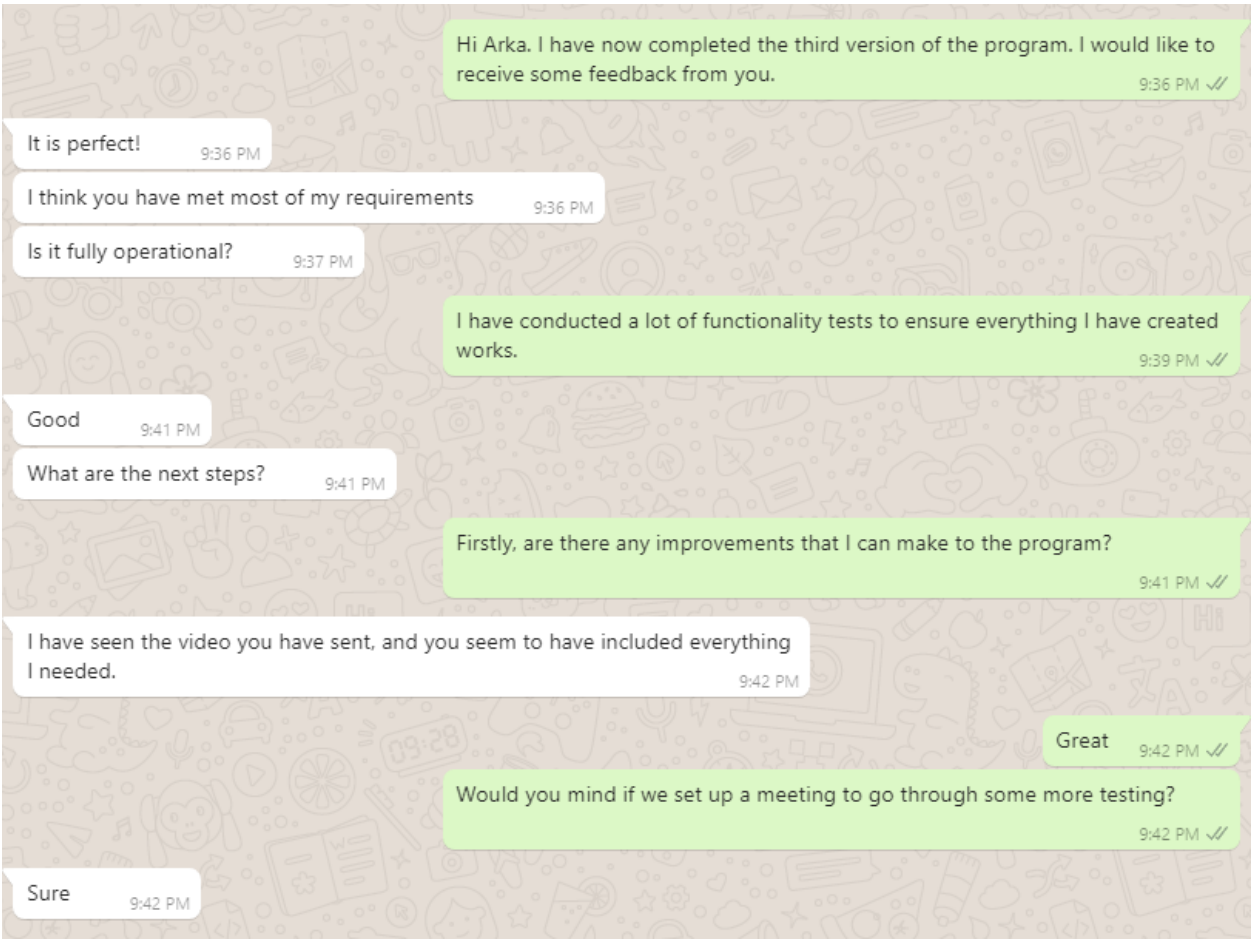
FINAL CODE FOR MODULE 3.3 THEME

```
def theme(widget):
    widget.configure(activebackground="#66FCF1")        #Cyan, Active Background colour
    widget.configure(activeforeground="#0B0C10")        #Black, Active Foreground colour
    widget.configure(background="#0B0C10")              #Black, Background colour
    widget.configure(foreground="#c5c6c7")              #Gray, Foreground colour

ButtonsToTheme = [self.P1B1,self.P1B2,self.P2BB,self.P2B1,self.P2B2,self.P2B3,self.P2B4,
                  self.P2B5,self.P2B6,self.P2B7,self.P2FB,self.P3BB,self.P3FB,self.P4BB,
                  self.P4B1,self.P4B2,self.P5BB,self.P5B1,self.P5B2,self.P5BB,self.P6BB,
                  self.P6B1,self.P6B2,self.P6B3,self.P6B4,self.P6B5,self.P6FB]

for i in ButtonsToTheme:                    #For all of the buttons in the array
    theme(i)                                #Theme them
```

EXAMPLE OF USE

In Video.

STAKEHOLDER REVIEW: PROTOTYPE 3

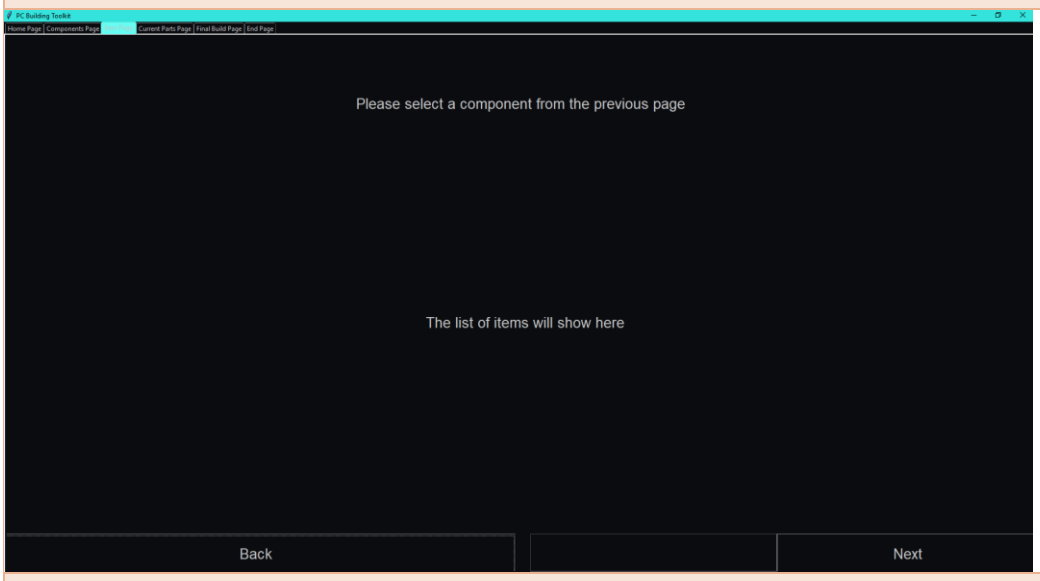


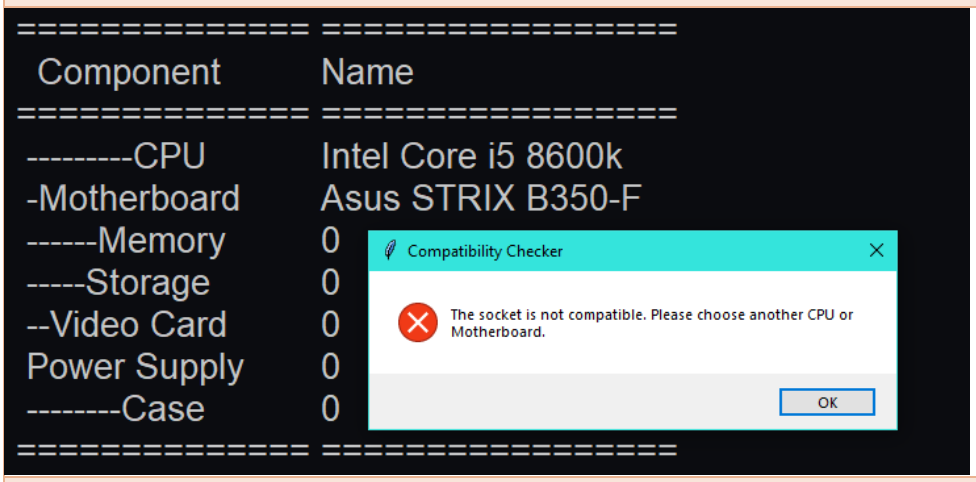
From my discussion with Arka, we have decided that this will be the final version of the program. We have arranged a meeting where we will go through the evaluative testing. This will help with robustness or usability issues.

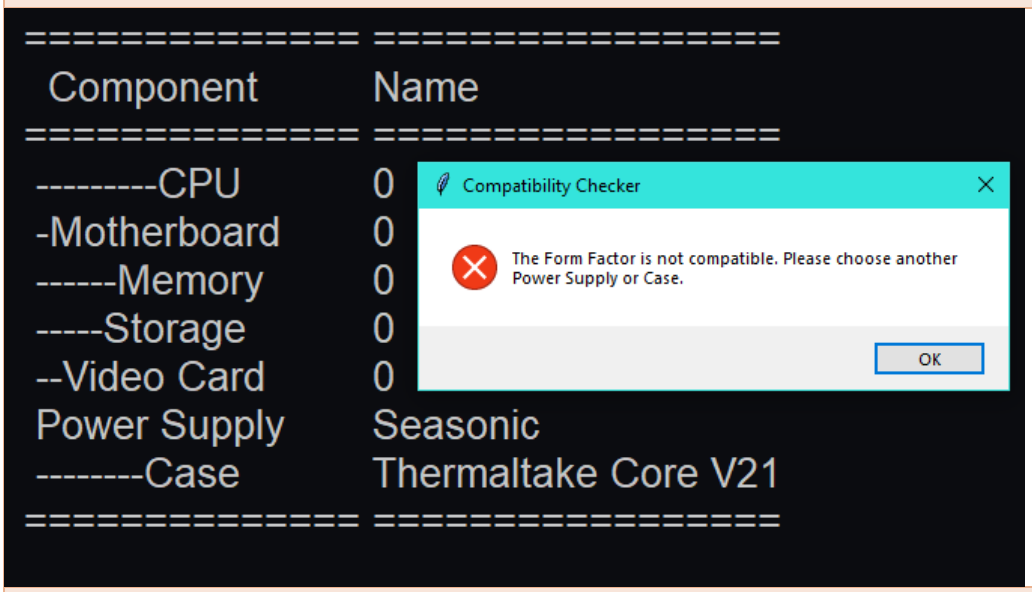
EVALUATION

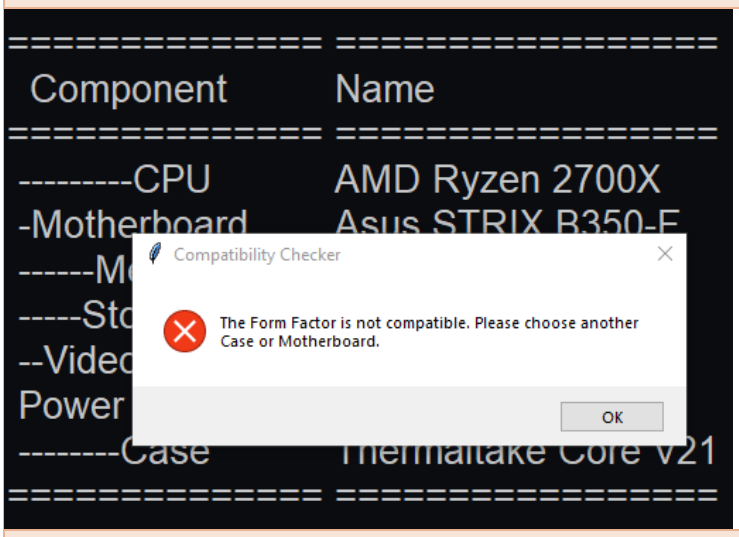
FINAL TESTING

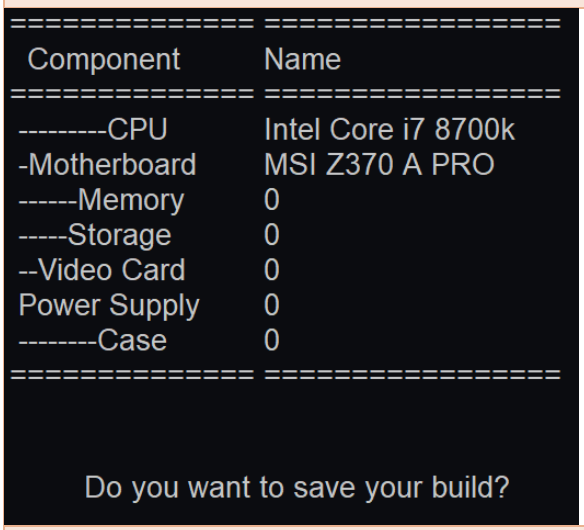
For this section, Arka and I decided to invite a few friends each to try out the program. We gave them examples of things to test (from the post development test data) and documented their results. This would allow us to check the usability of the program and the program's robustness. This will be conducted as black box testing, and the tester will choose the expected results and test data. They will be use the program, however, they would like, and I will amend the code (if required) without them being able to view it.

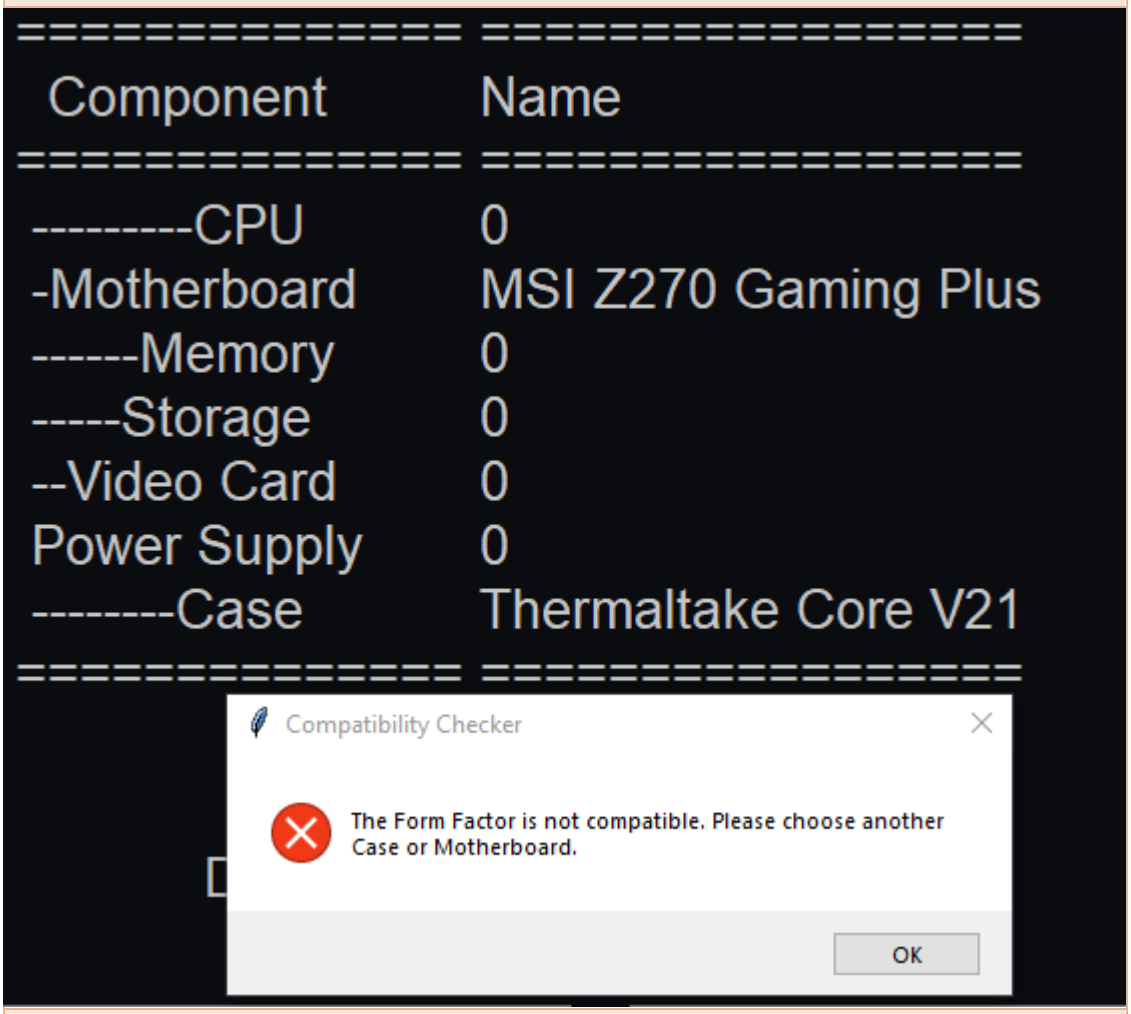
Test No.	**1**
Test	Do all the buttons work
Test Data	Skip (extreme) with no component input
Expected Result	Please select a component
Actual Result	
Other Notes	Default text on label shown telling the user to add a component-pass

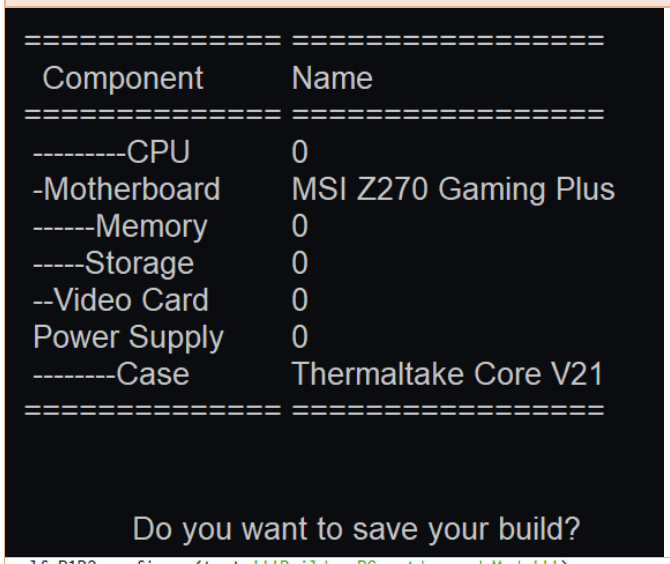
Test No.	**2**
Test	Does the compatibility checking algorithm work
Test Data	1151 CPU and AM4 Motherboard (Invalid)
Expected Result	Pop-up error
Actual Result	
Other Notes	Pop-up as expected.-pass

Test No.	**3**
Test	Does the compatibility checking algorithm work
Test Data	Small Case and large PSU (invalid)
Expected Result	Pop-up error
Actual Result	
Other Notes	Pop-up as expected.-pass

Test No.	**4**
Test	Does the compatibility checking algorithm work
Test Data	AM4 CPU and ATX motherboard, but micro ATX case (invalid)
Expected Result	Pop-up error
Actual Result	
Other Notes	Pop-up as expected.-pass

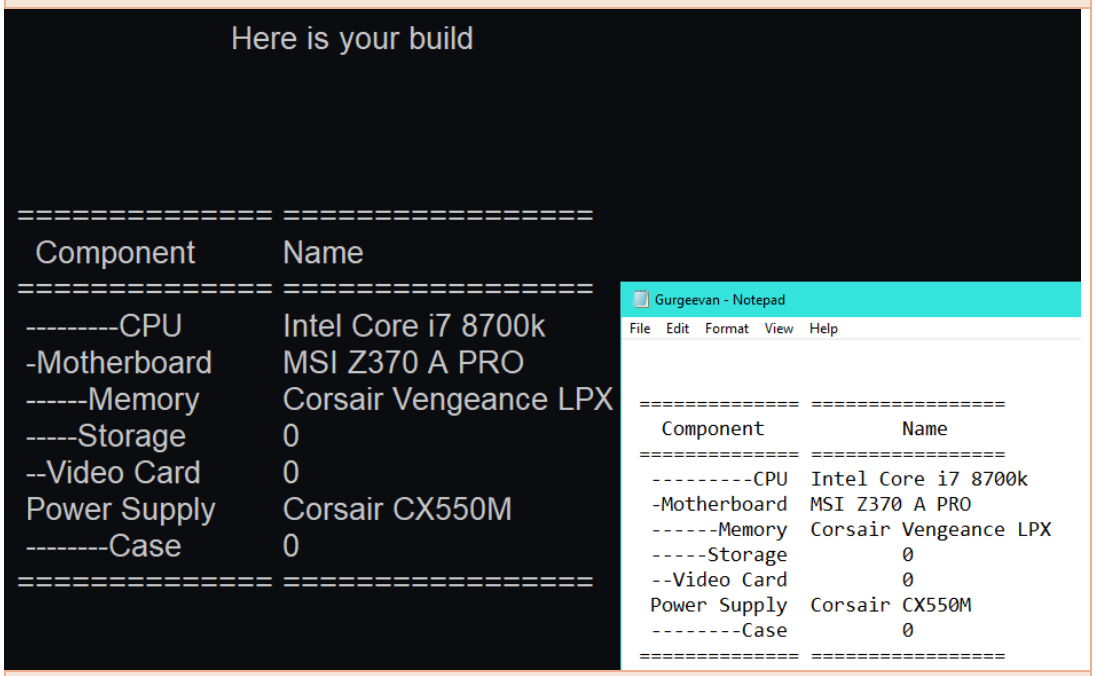
Test No.	**5**
Test	Does the compatibility checking algorithm work
Test Data	1151 CPU and 1151 Motherboard (valid)
Expected Result	Continues
Actual Result	
Other Notes	Continues to save screen as expected, -pass

Test No.	**6**
Test	Does the advanced mode work
Test Data	Large motherboard and small case
Expected Result	Continues
Actual Result	
Other Notes	Compatibility check enabled. This is a large issue and should be fixed immediately - failed

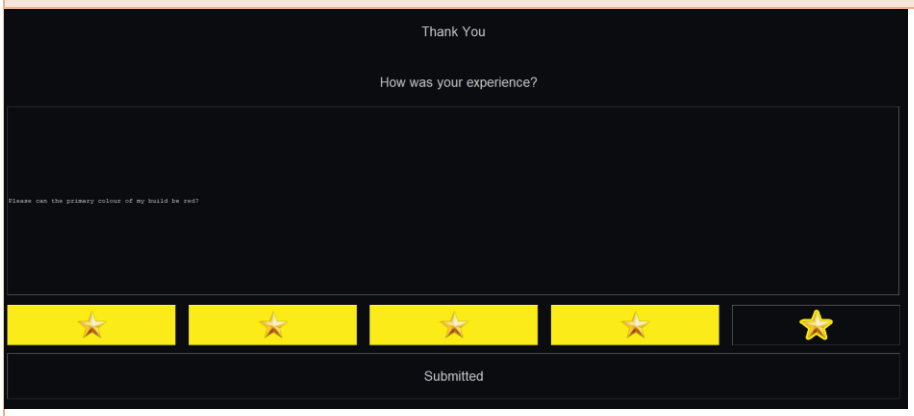
Test No.	**7**
Test	Does the advanced mode work
Test Data	Large motherboard and small case
Expected Result	Continues
Actual Result	

Changes to code	`self.P1B2.configure(text='''Build a PC - Advanced Mode''')`
	`self.P1B2.configure(command = lambda : [self.AllTabs.select(self.Page2), UserName(self.P1E1.get()), AdvancedMode()])`
	
	`def AdvancedMode():`
	`  AdvancedMode = True  #Turns boolean True`
	
	`self.P4B2.configure(text='''Finish''')`
	`self.P4B2.configure(command = lambda : [self.AllTabs.select(self.Page5), ModeSelection()])`
	
	`def ModeSelection():`
	`  global AdvancedMode`
	`  if AdvancedMode == True:  #Checks boolean status`
	`  pass  #Ignores the check if True`
	`  else:`
	`  BasicMode()  #Else it will run it`

Other Notes	Added a function that would turn a Boolean true. Then, before enabling the compatibility check, it would check if the Boolean was true, -pass

Test No.	**8**
Test	Can a user access their build later
Test Data	Partial build (extreme)
Expected Result	Exports
Actual Result	
Other Notes	Perfect, and works with partially built systems, -pass

Test No.	**9**
Test	Can Gurgeevan add feedback
Test Data	“red build please”, 4 stars
Expected Result	Data saved
Actual Result	Program crashes when adding star feedback
Other Notes	This worked in previous testing, so it should be easy to fix using backtracking (a computational thinking method)- failed

Test No.	**10**
Test	Can Gurgeevan add feedback
Test Data	"red build please", 4 stars
Expected Result	Data saved
Actual Result	

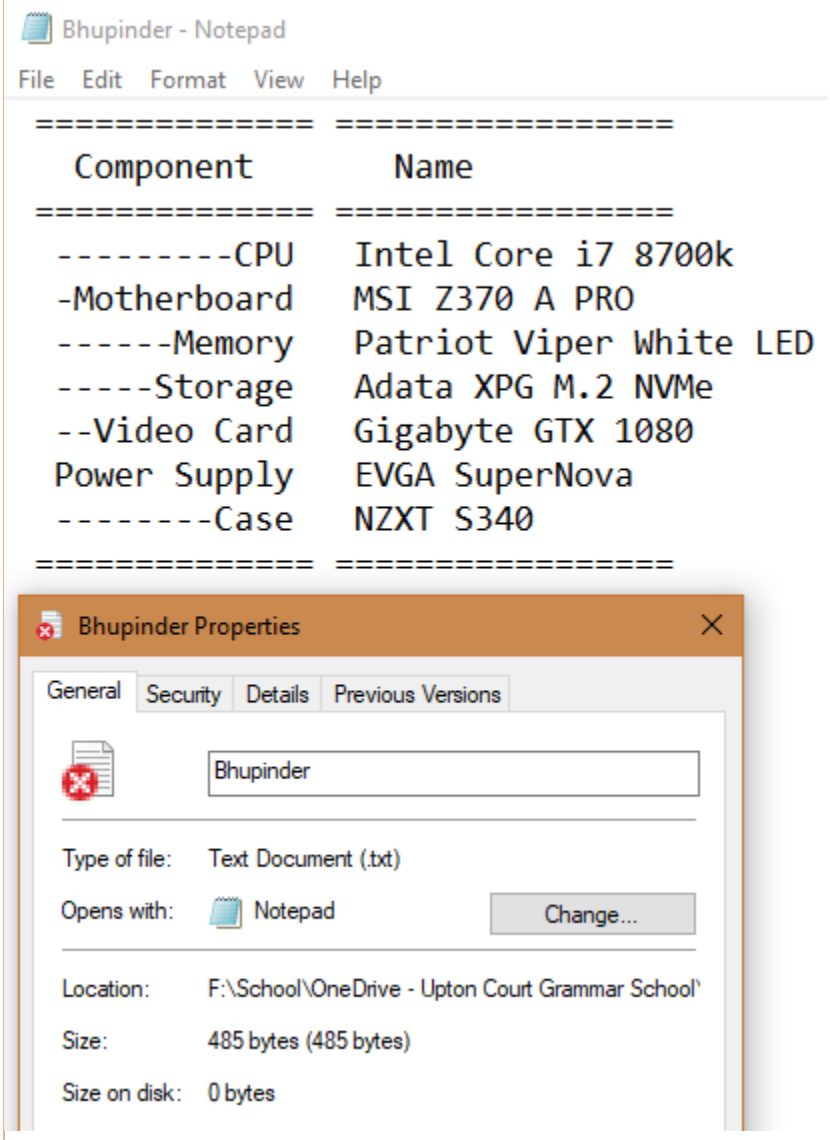
```
41          Gurgeevan
42  GurgeevanFeedback

Enter user name: GurgeevanFeedback
   Stars                                          Comment
id
1      4  Please can the primary colour of my build be red?
```

```
 else:
     breakpoint
```

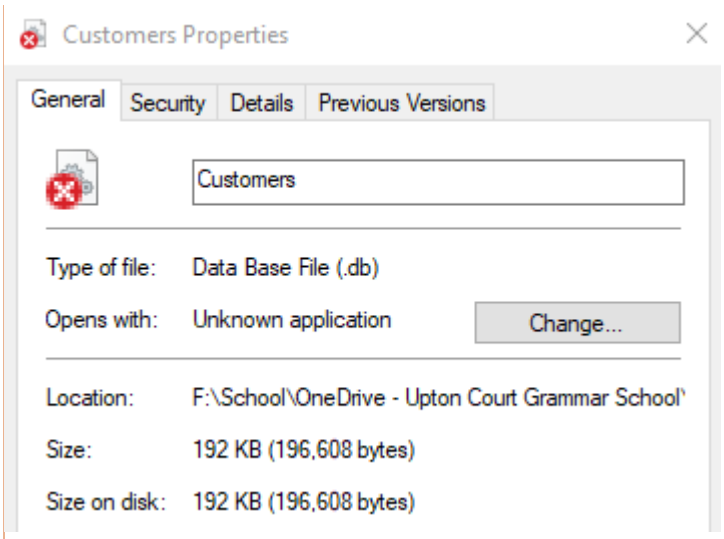
Other Notes	Accidental typo caused by pressing tab with the cursor on 'break'. Should have been 'break'. -pass

Test No.	**11**
Test	Can the stakeholder view Gurgeevan's build and feedback
Test Data	Gurgeevan and GurgeevanFeedback
Expected Result	Data can be accessed
Actual Result	41 Gurgeevan 42 GurgeevanFeedback Enter user name: Gurgeevan Component Name id 1 CPU Intel Core i7 8700k 2 Motherboard MSI Z370 A PRO 3 Memory Corsair Vengeance LPX 4 Storage 0 5 Video Card 0 6 Power Supply Corsair CX550M 7 Case 0 Enter user name: GurgeevanFeedback Stars Comment id 1 4 Please can the primary colour of my build be red?
Other Notes	I have created a small program that allows Arka to view the builds that have been created using his program and saved to the database. -pass This is the code for the database. I have added the code to create the databases in the Project Code section.

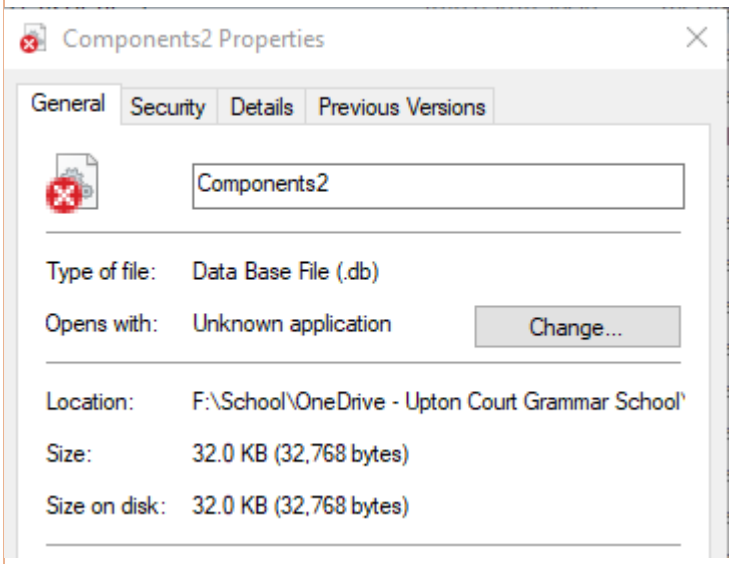
```
Database = sqlite3.connect("Customers.db")
cursor = Database.cursor()

print(pandas.read_sql_query("SELECT name FROM sqlite_master WHERE type='table'", Database))     #Prints everything

x=0
while x != "":                                  #Will ask the stakeholder for name till they press enter with no name
    x = input("\nEnter user name: ")
    print(pandas.read_sql_query("Select * from {} ".format(x), Database,"id"))
```

Test No.	**12**
Test	Are system resources used efficiently
Test Data	Full build
Expected Result	Program is light and does not require a lot of resources (CPU, RAM and storage)
Actual Result	

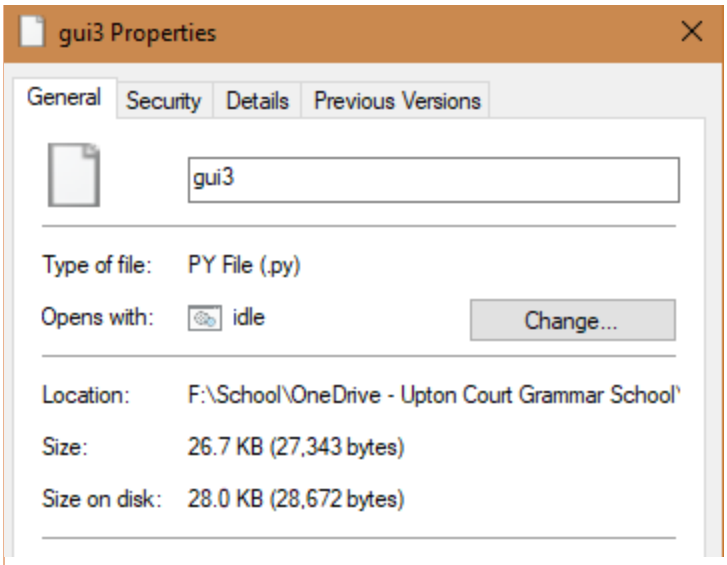
This is the exported file for a user. It is extremely efficient as it is a simple text file that does not need to take up much space.



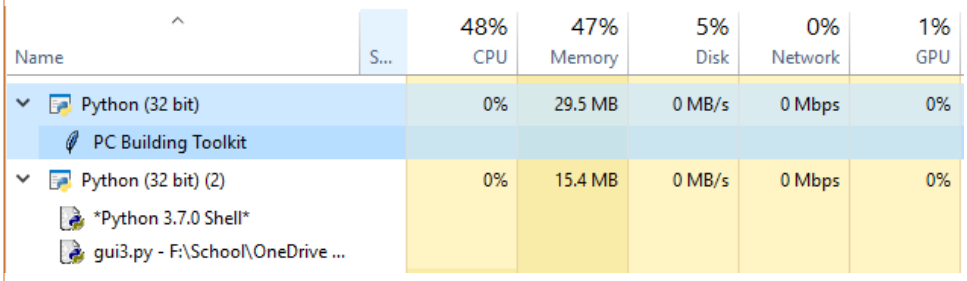
This is the “Customers” database where builds and feedback are stored. Almost 200KB are used to store the builds and feedback from 15 users. According to this data, around 75,000 users can use this application before a Gigabyte of storage is used (1,000,000,000B/(200,000B /15 users) = 75000 users). If 10 people used this application per hour during working hours, it would take over 2 and a half years to fill the database with a Gigabytes worth of data (75000/(8 working hours*10 people per hour) = 938 days including weekends).



The components database is extremely light, but this is mainly due to the limited data set. As more are added, this will increase in size, but due to the capacity of storage mediums, it will be relatively insignificant.



This is the GUI application. This is also really light and will not take a lot of resources.

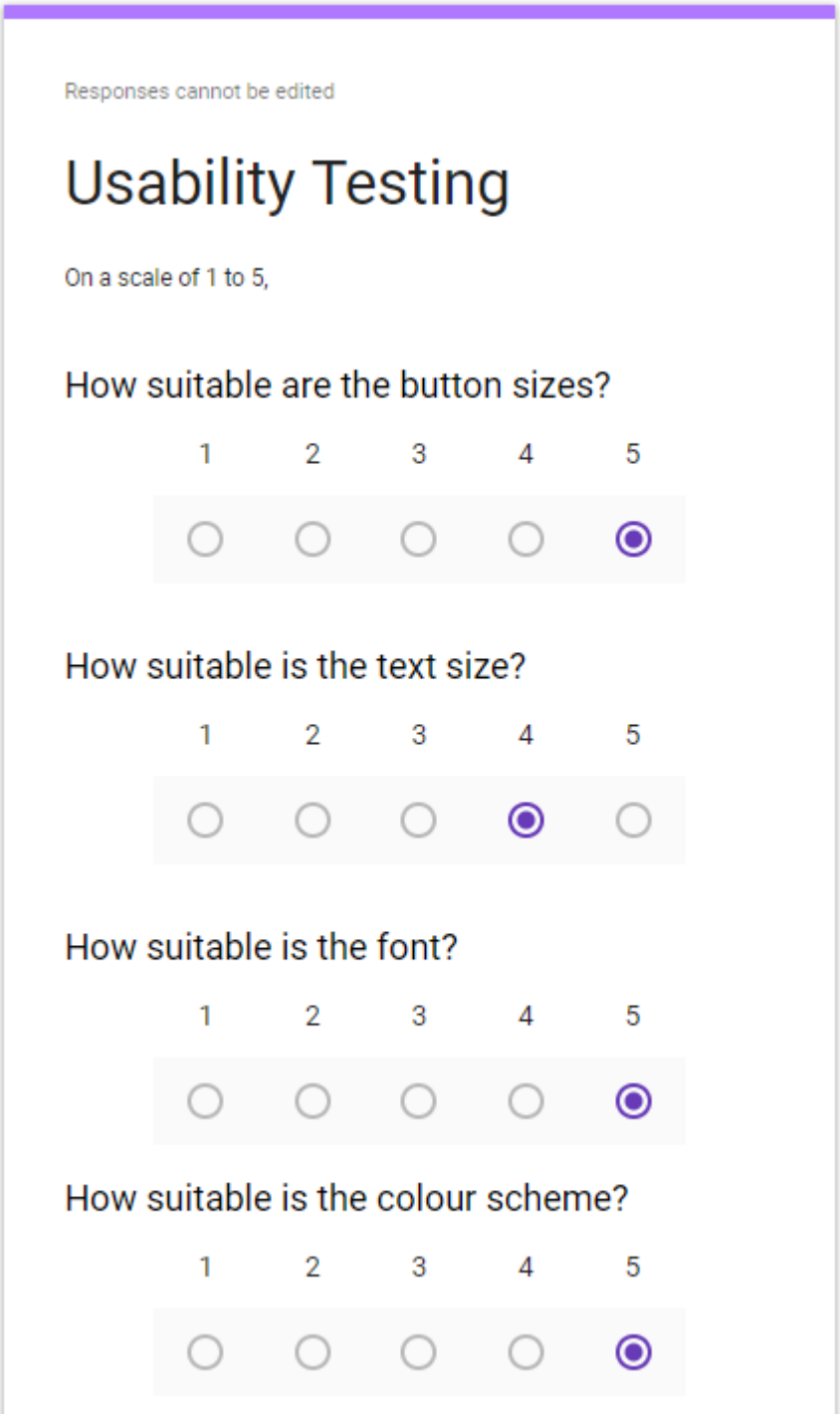


In total, the python application is using 45MB of ram. This is insignificant to a modern day computer, with even smartphones have 6GB of RAM. Also, the relative CPU performance is not measurable to 1 decimal place.
From all of this data, it is safe to conclude that my application is using system resources efficiently and will easily run on old or low powered hardware, given that they meet the operating system requirements.

Overall, I feel that the final testing has been very successful. A lot of the tests were similar to my development testing, however, some of the tests were conducted with new test data or tested areas of the program that I did not (test no. 6,7 and 9). These tested the robustness and usability of my program and I was able to fix the issues.

USABILITY TESTING

After all of this final testing, I sent Arka a questionnaire in order to understand how functional the program is. Below are his ratings (with a 5 being the highest). I chose to use Google Forms as it is easy to send the questionnaire to the recipient, who can complete it on various devices at their convenience. Below are the results.



Responses cannot be edited

Usability Testing

On a scale of 1 to 5,

How suitable are the button sizes?

1 2 3 4 5

How suitable is the text size?

1 2 3 4 5

How suitable is the font?

1 2 3 4 5

How suitable is the colour scheme?

1 2 3 4 5

How suitable is the navigation method?

1 2 3 4 5

○ ○ ○ ○ ◉

How consistent is the user interface?

1 2 3 4 5

○ ○ ○ ○ ◉

How suitable is the mode selection method?

1 2 3 4 5

○ ○ ○ ○ ◉

How suitable is the advice on the program?

1 2 3 4 5

○ ○ ○ ○ ◉

How easy is it to add another component to the build?

1 2 3 4 5

○ ○ ○ ○ ◉

How easy will it be to update the components database?

1 2 3 4 5

○ ○ ◉ ○ ○

How suitable is the export method?

1 2 3 4 5

○ ○ ○ ○ ◉

How convenient is the export method?

1 2 3 4 5

○ ○ ◉ ○ ○

How convenient is the feedback collection method?

1 2 3 4 5

○ ○ ○ ○ ◉

How easy do you think it would be to add more features to this application?

1 2 3 4 5

○ ○ ○ ◉ ○

How easy would it be for another developer to work on the program?

1 2 3 4 5

○ ○ ○ ○ ◉

How efficiently does this program use the CPU?

1 2 3 4 5

○ ○ ○ ○ ◉

How efficiently does this program use memory?

1 2 3 4 5

How efficiently does this program use storage?

1 2 3 4 5

How suitable are the validations used?

1 2 3 4 5

How easy is it to view the builds and feedback?

1 2 3 4 5

SUCCESS CRITERIA EVALUATION

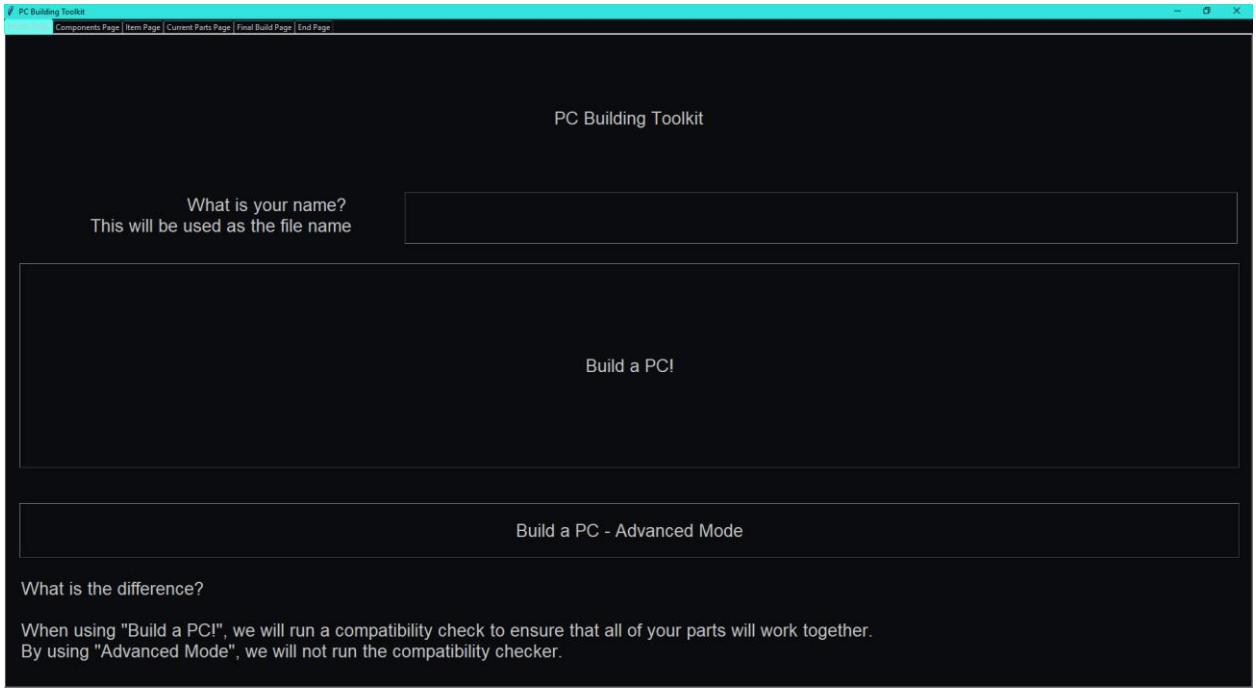
How the requirement was met		Evidence
Design and output requirements		
Display a Graphical User Interface to operate the program-from user research	I have successfully created a GUI that is easy to operate and is created around the stakeholder's hardware and design requirements.	Prototype 2 and 3 development, final testing No. 1.
Large buttons to support a touchscreen input-from stakeholder IDI	As shown in the questionnaire, the buttons are perfectly sized for the application.	Questionnaire, No 1.
Easy to read text-from stakeholder IDI	The text is very easy to see as the colour contrasts with the background and is also large enough for the application.	Questionnaire, No 2 and 3.
Suitable advice and recommendations displayed-from user research	All advice is simple and easy to read. They are also relevant, and the stakeholder agrees.	Questionnaire, No 8.
All features of the GUI must be labelled and easy to use-from stakeholder IDI	All features of the GUI are logically placed and labelled. No instructions are needed for a user to use the application.	Questionnaire, No 6.
A suitable colour scheme must be used-from stakeholder IDI	As per stakeholder requirements, the background is black, with white text and a blue accent (which is shown when elements of the GUI are interacted with).	Questionnaire, No 4.
Allow users to filter through components-from user research	The user is able to choose the component category and then the component. This breaks down the building process making it easier to choose and view.	Development test 1.3.X
Allow users to toggle advanced mode-from user research	On the first page, a user has the option to select basic mode or advanced mode. A description has been added at the bottom of the page so the user knows the difference.	Questionnaire, No 7.

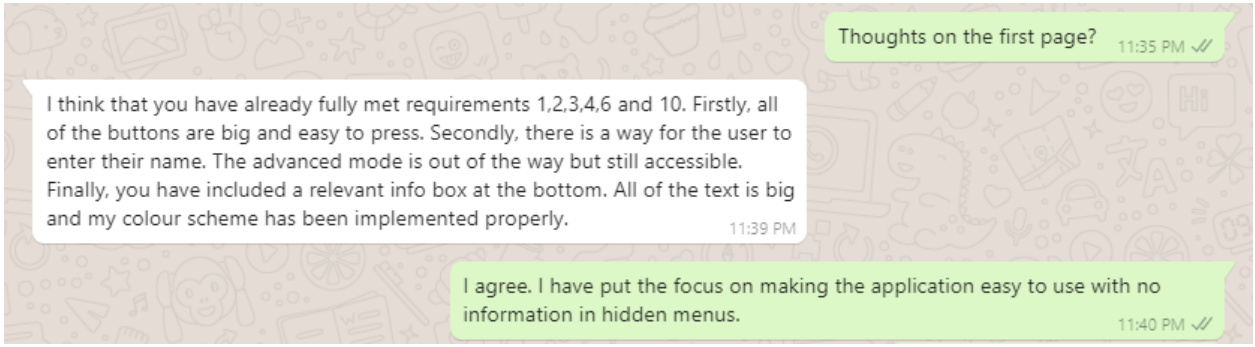
It must save the build for later access by the same user-from user research	All builds are saved to an SQL database but, due to a limitation in time, a user is unable to edit the saved data. This help with security, as a malicious user cannot access someone else's build. This is not a problem as the user can export the build.	Development test 2.4.1
The user can choose to save their build for personal use. - from user research	The user is able to export the build to a text file on request. This is a very flexible format which does not take a lot of storage space.	Questionnaire, No 11 and 12.
All builds created must be saved for stakeholder analysis-from stakeholder IDI	Once a build is created, it is automatically added to the database for the stakeholder to use. Feedback is also stored here. I have created a simple CLI application for this, but Arka can use another (3rd party application) for this if needed.	Questionnaire, No 20.
Input requirements		
Must support touchscreen inputs and touch-from stakeholder IDI	All buttons are large, making it easy to press on a touchscreen. This was designed in sketches 3 and the programming was designed around this.	Final testing 1 and development testing.
Must be able to amend a database-from stakeholder IDI	The program creates a table in the Customers database with the new build and feedback. The application for the stakeholder can also amend the components database.	Development test 1.4.X and 3.2.X
Text must be entered as a string	Python's input method automatically takes data as a string unless formatted afterwards.	Development tests
Any data must be stored in a variable	All data inputs have validations to ensure data is collected and stored correctly.	Development tests
Each user must enter their name and a method of contact-from stakeholder IDI	A user's name is taken at the start of the program and, if there is an error, you can create a unique ID to store the database. Due to a change in requirements, we do not want to store a method of contact due to the DPA and the program being used locally, in store.	Development test 1.4.X
Each user must be assigned a new CustomerNumber-from stakeholder IDI	This is best practice for a database and ensures data is not overwritten or deleted accidentally. For each build save, a new table is created with a new and unique ID.	Development test 1.4.X

Allow users to 'make a note'/leave feedback-from stakeholder IDI	This is on the last page of the program and is also saved in the database. A user can add a comment, leave a rating out of 5, or both. This makes it very quick and convenient to use.	Development test 3.2.X
Process requirements		
Able to create a useable system for a user and their preferences-from stakeholder IDI	The process was created in prototype 1 and tested in the final testing. The user can choose components from categories and add them to or amend their build.	Development Prototype 1
Use an SQL database, which holds the properties of components-from stakeholder IDI	This was the best way to store the components as it would be easy to add more components. Various applications use applications and will allow this program to be easy to update in the future.	Development test 1.3.X and 'Code for admin'
Give advice to the user on what things are in a minimalistic way-from user research	This was only needed on page 1, with the basic and advanced mode. This was added on the bottom of the page, so it was only read if the user needed help and not in the way of an experienced user.	Sketches 3 and Development Prototype 2.1
Ensure parts are compatible-from user research	This was achieved by creating the components database in a logical way. The algorithm will check the same fields in components and if they are not compatible, a pop-up error is displayed.	Development test 1.5.X and final test no. 2,3,4 and 5
Uses system resources efficiently-from stakeholder IDI	This is explained in the results of Final Test No. 12 and concluded with the fact that my program uses system resources efficiently and could easily run on a wide range of hardware that supports the required operating system and software requirements.	Final test no. 12
Gives relevant recommendations -from stakeholder IDI	Due to a lack of time, I was unable to do this. However, it would not have saved a lot of time for most users, so the time required to create this algorithm would not have been worth it.	Essential features - limitations

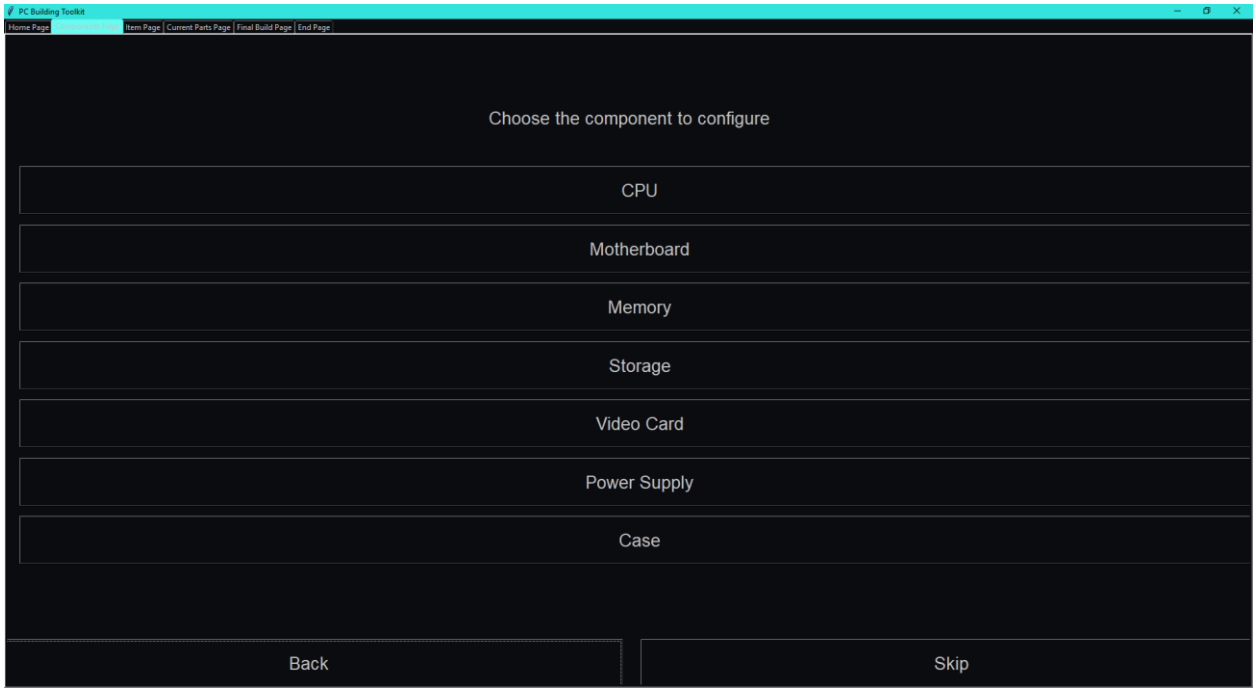
USABILITY FEATURES EVALUATION

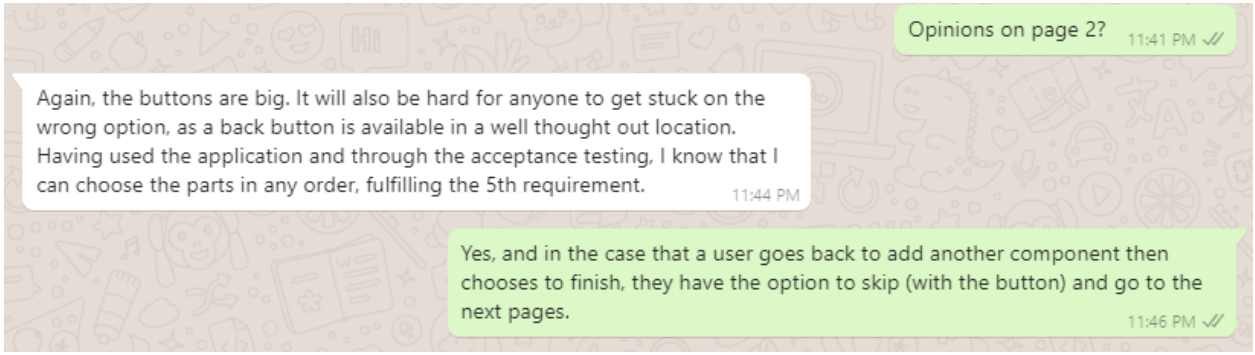
To test the usability features against the success criteria, I met with Arka and discussed his general thoughts on the program. I use the data from the questionnaire to aid with the discussion.



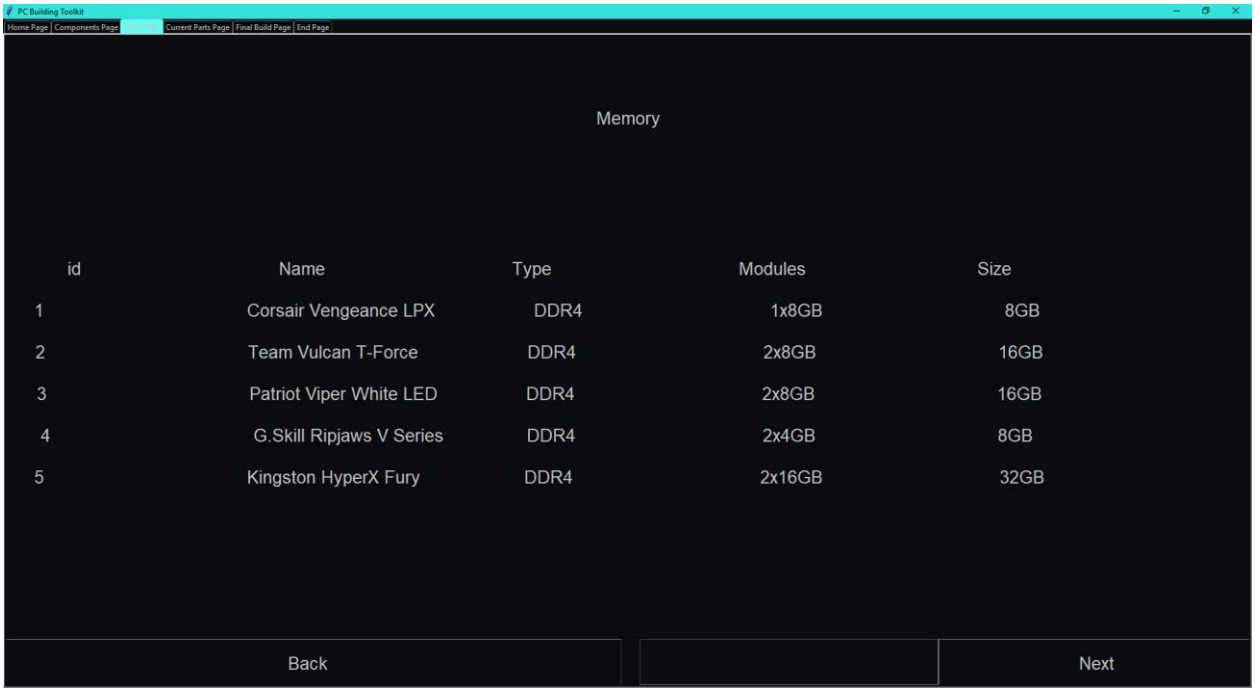


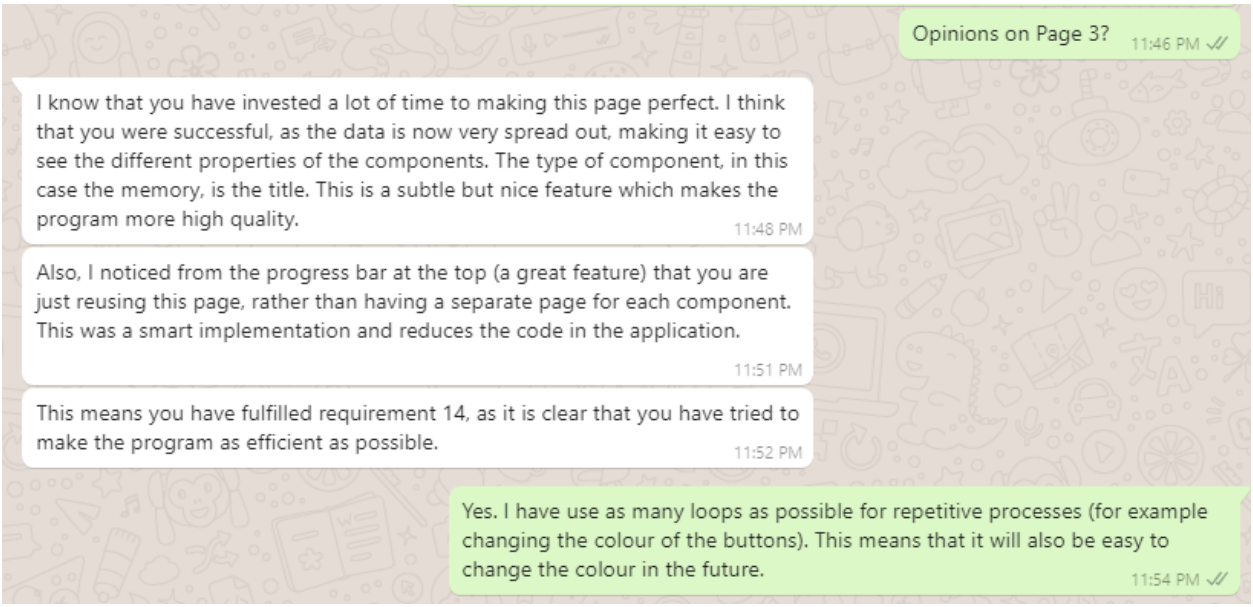
The reason question 2 received a 4 out of 5 was because he would have preferred the title text to be slightly larger. This would be fairly easy to implement and could be implemented in a similar way to the theme() function.



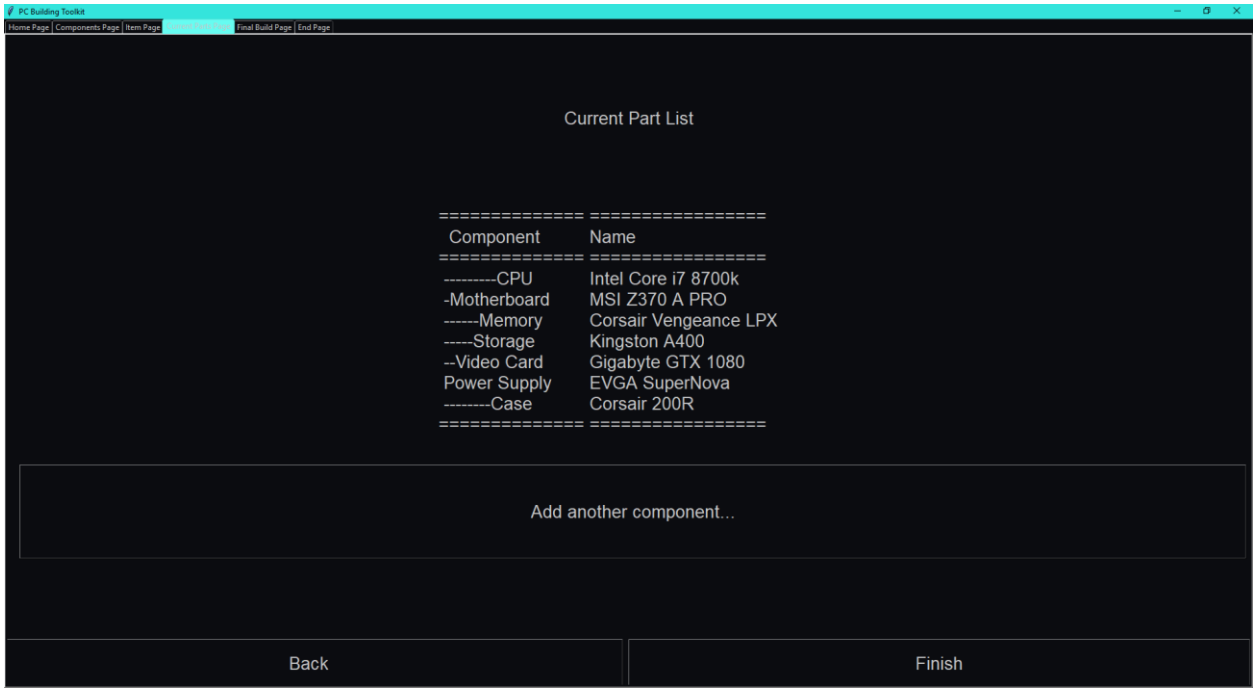


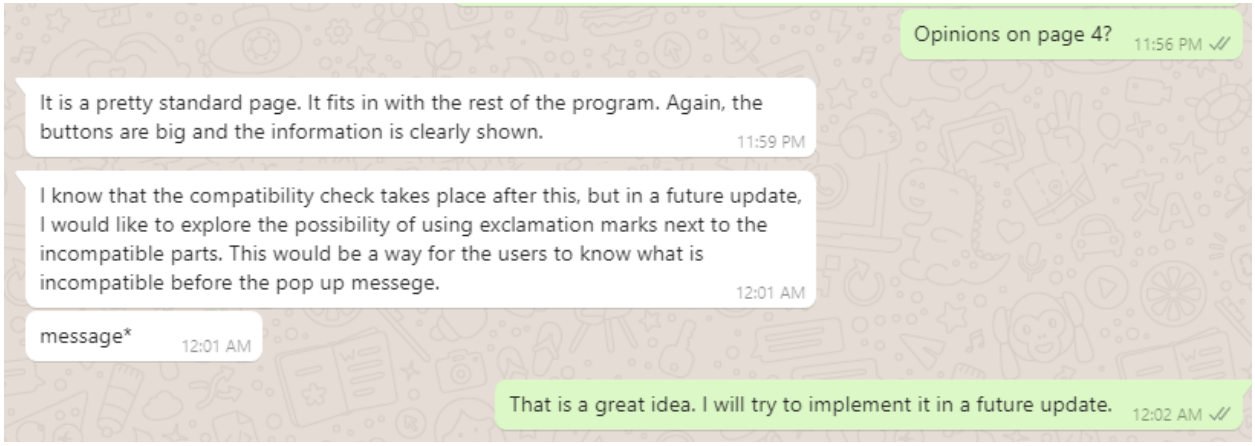
Something that Arka and I discuss in person was the possibility of adding more component categories in the future. For example, we could add CPU Coolers (such as All In One or customer water cooling), PCIe Cards (e.g. sound cards or network cards) and other accessories (such as bluetooth adaptors, monitors and speakers). Otherwise, this page is perfect and enough to build a working PC.



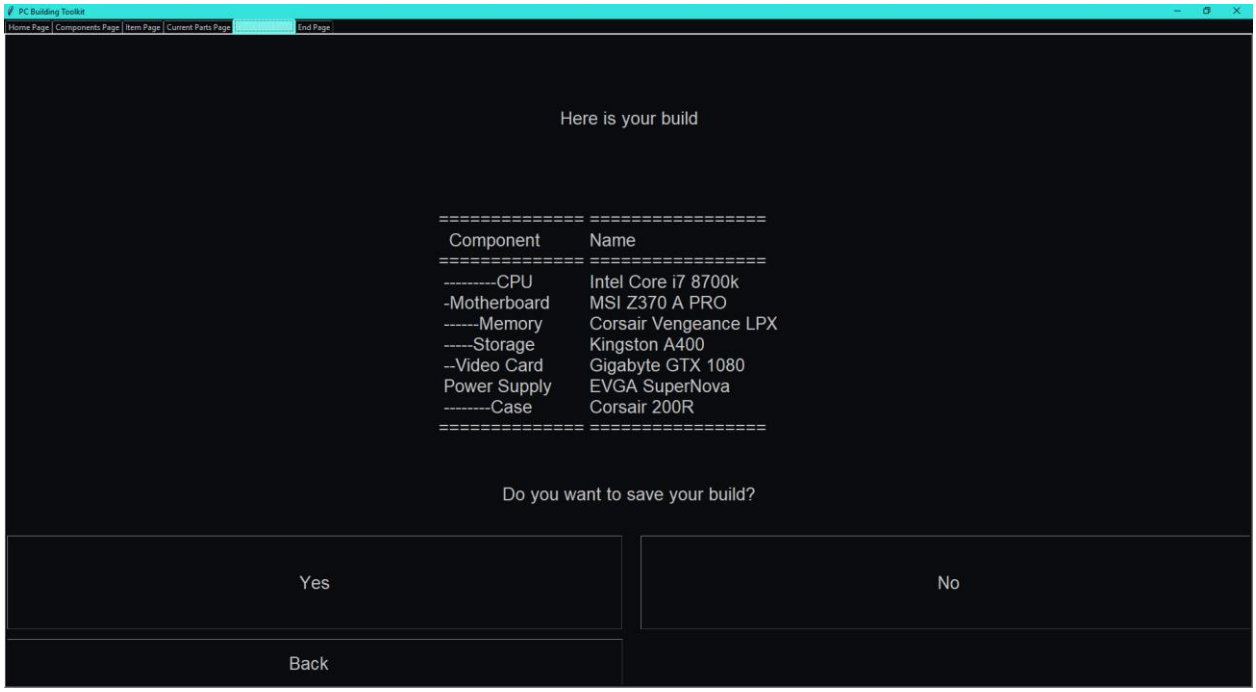


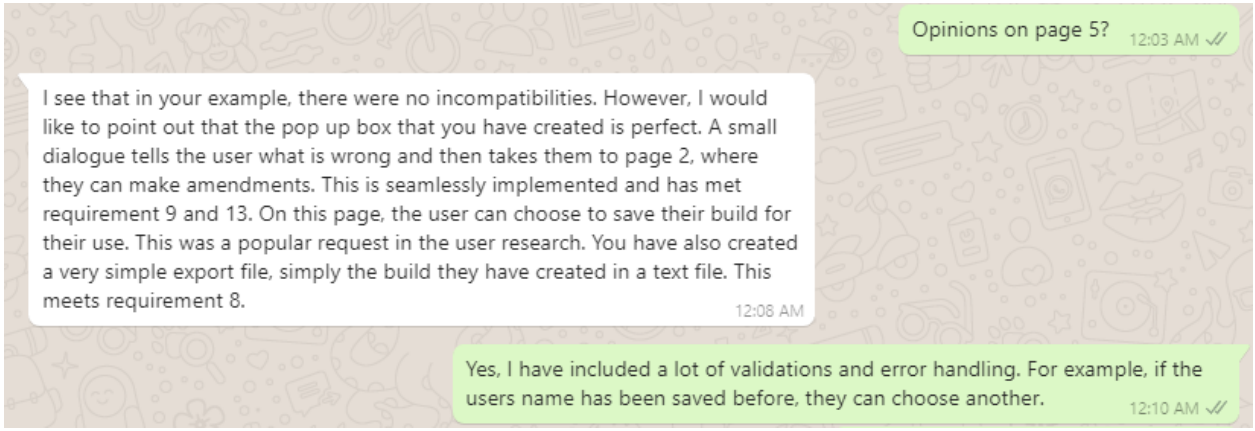
The reason question 10 in the questionnaire was given a 3 out of 5 was because of the way in which the database could be updated. I included a basic Command Line Interface for the stakeholder and any future developers to understand how the database was created, how to add more components and how to view the current data. I felt that this was the best way as it will allow any future developer to use their preferred DBMS.



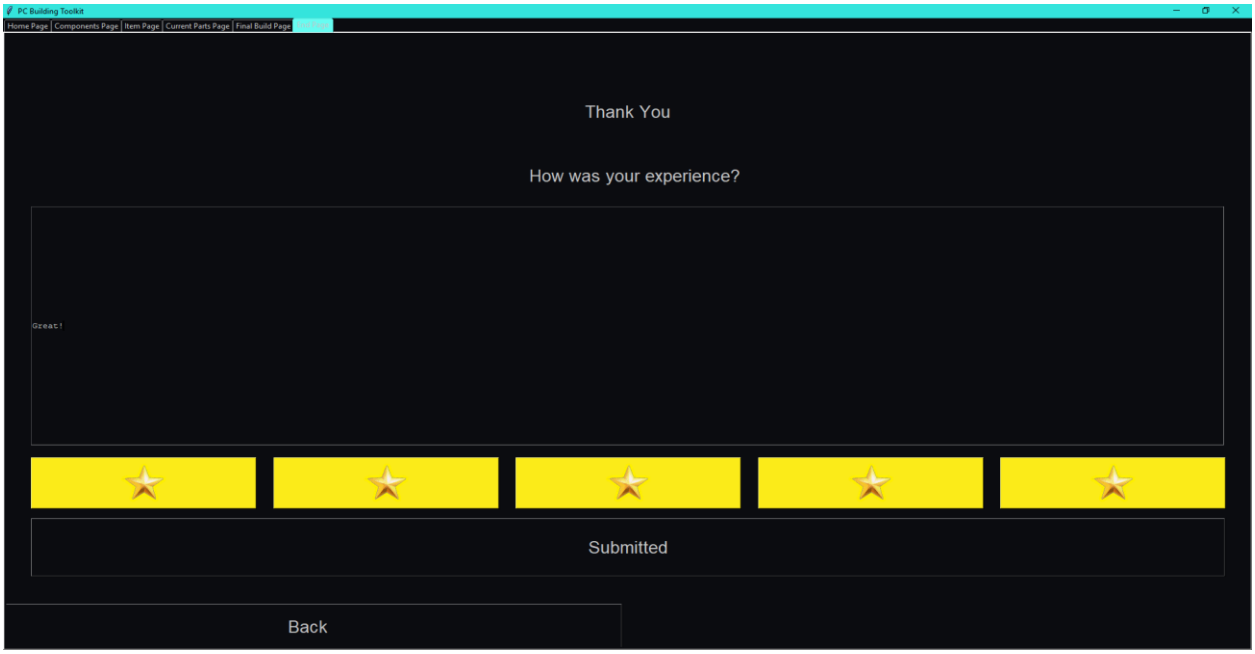


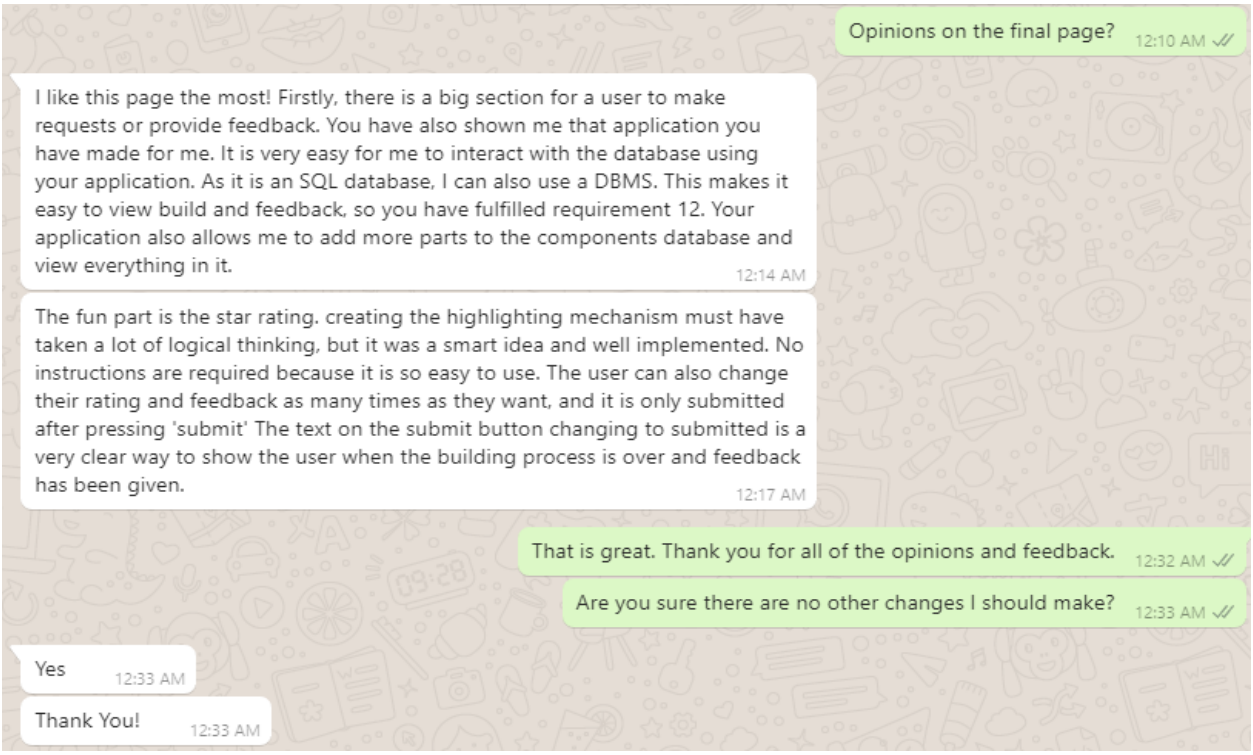
This idea would be useful on this page as it will prevent the user from seeing the pop-up error once they press Finish (which would enable the compatibility check). This is something that we could add in a future update, and it could be implemented in multiple ways. A really simple method would be to change the text colour of any incompatible components and adding concatenating a string with an exclamation mark to the component name.





The reason that question 12 in the questionnaire was given a 3/5 was due to the process of saving the text file. The user would need the use the PC to email them self the file and delete it, or bring their own USB. This could be difficult for some people to do. One way to overcome this would be to email the results to the user. The only issue with this is the related security risks. The python module would require the username and password to the email address it would send from. It would also need the user's email address, which would need to be stored. If this data is not stored correctly, user data could be illegally accessed and leaked, stopping people from wanting to use the application.





The reason that this received a 3/5 was also due to the CLI program I have included by default. However, as Arka can use a DBMS of his choice, this will not be an issue.

From all my conversations with Arka shown above, it is clear that I have successfully achieved the success criteria. All solutions requirements have been met or a suitable alternative has been discussed with the Stakeholder. I have also managed to impress Arka with the rating system on the final page and overall, I believe that this project has been successfully completed.

LIMITATIONS

The largest limitation is with the programming language used to create the program. I have used Python 3 and although it was suitable for this problem, it does not leave any room for future services. For example, this program cannot be easily modified to work as an app or webpage. It would need to be remade completely. This also means that this program can only work on devices that run python and have the required modules. This would include Windows and Linux, but not mobile devices. The operating system would need pandas installed. It is easy to install modules (for Windows, you could type pip install windows), however, this is still an extra step.

Another limitation is the GUI. I have used Tkinter as it was fit for purpose, but if Arka decides to buy hardware of a different resolution, such as 720p or 2k, the application may not function as it should. This objects on the frame would be spread out or squashed, and it would take a lot more time to make the GUI able to scale to the resolution of the display.

The final limitation is the database. Unfortunately, I was unable to find a suitable API to use to get the components. To overcome this, I created my own SQL database with a fixed dataset. Arka will be able to add more components to this, but this will take up his time. A separate application has been made for this purpose.

MAINTENANCE

I feel that it will be very easy for any developer to maintain and add features to my code. I have included a lot of detailed comments to allow another developer to understand what each line does. Any technical terms have been explained in the comments, meaning that they should be understandable by a non-programmer.

The code is also very modular, through the use of functions. I have also created documentation (this file), which explains how the functions were made and how any issues were resolved. I have also included validations where possible (using try except) meaning that any common errors are explained in runtime, and a solution is implemented. An example is in the SQL table name. If it cannot be added, an error is given but the user is asked to use another file name.

In my third prototype, I have minimised my use of global variables, meaning that it will be a lot easier to diagnose any issues that occur during runtime. I have also reduced the code where possible so there is less to maintain. For example, I created a function to theme all of the buttons (which are stored in an array). If more buttons are created in a future update, a developer would only need to add the variables storing the button to the array.

POTENTIAL FUTURE IMPROVEMENTS

There are various features/improvements that can be added to this program. For example, if Arka does not want to add data to the database manually, a 'web scraper' could be created. This would search the web for new components and automatically add them to a database. However, this could cause legal issues if done incorrectly. It may also take a lot of computational power.

I could also find a module (such as smtplib) that would allow the saved exported file to be emailed to a user. This would require the program to connect to an SMTP server and take the user's email address. This would also need to be handled correctly, to comply with the Data Protection Act 2018. I could use the following code to create this feature:

I could also add a timer feature to the program, so the data is saved for 10 minutes before being reset if no changes are made. This would stop people from spending too long at the machine and protect others from viewing their build or details if a person leaves without clearing their data. As mentioned by Arka, I could add an exclamation mark system that highlights incompatible items before the pop-up error which is activated on pressing Finish.

Another potential improvement could be to the program for the admin. Although this does not need to be used (as they can use a DBMS), I could develop this code further by possibly creating a graphical user interface. This could require the use of a login and password and would be a lot easier for an inexperienced user to use (fortunately Arka has an understanding of python and is able to easily use my command line program). Finally, I would make the title text size slightly bigger, to make it clear that it is a title. All of these improvements would hopefully get a 5/5 in the Usability Testing Questionnaire if there was a future version with no limitations.

BIBLIOGRAPHY

DATE	WEBSITE
22/05/18	https://en.wikibooks.org/wiki/A-level_Computing/AQA/The_Computing_Practical_Project/Picking_a_project
29/05/19	https://swcarpentry.github.io/sql-novice-survey/10-prog/
11/06/18	https://pythonschool.net/category/databases.html
26/06/18	https://www.freelancinggig.com/blog/2017/09/22/best-programming-languages-windows-application/
28/06/18	http://istqbexamcertification.com/what-is-agile-methodology-examples-when-to-use-it-advantages-and-disadvantages/
03/07/18	https://www.reddit.com/r/pcmasterrace/comments/3dobwu/a_new_alternative_to_pcpartpicker_is_here/
26/08/18	https://pchound.com/
01/09/18	https://uk.pcpartpicker.com/
04/09/18	https://www.google.co.uk/forms/about/
15/09/18	https://software.intel.com/en-us/distribution-for-python/system-requirements
15/09/18	https://www.pythoncentral.io/introduction-to-sqlite-in-python/
15/09/18	https://kite.com/python/docs/sqlite3.Cursor.description
19/10/18	https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_sql_query.html
02/11/18	https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting
03/11/18	https://www.w3schools.com/python/ref_func_isinstance.asp
08/02/19	https://www.pythonforbeginners.com/concatenation/string-concatenation-and-formatting-in-python
08/02/19	https://stackoverflow.com/questions/34734572/tabs-in-print-are-not-consistent-python
15/02/19	https://support.microsoft.com/en-gb/help/10737/windows-7-system-requirements
18/02/19	https://pcpartpicker.com/forums/topic/6397-api

PROJECT CODE

CODE FOR PROTOTYPE 1

```
import sqlite3
import pandas
print("""
 ______ _______    ______        _ _     _ _                _______          _   _     _
(_____ (_______)  (_____ \      (_) |   | (_)              (_______)        | | | |   (_)  _
 _____) )          _____) )_   _ _| | __| |_ ____   ____      _  ____   ____ | | | |  _ _ _| |_
|  ____/ |        |  __  (| | | | | |/ _  | |  _ \ / _  |    | |/ _  \ / _  \| | | |_/ ) (_   _)
| |      | |_____ | |__)  ) |_| | | ( (_| | | | | | ( (_| |    | | |_| | |_| | | |  _ (| | | |_
|_|       \______)|______/|____/|_|\_)____|_|_| |_|\___  |    |_|\___/ \___/ \_)_| \_)_|  \__)
                                                   (_____|                                  """)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def Mode():
    global UserName, ModeInput                                                #Declares these variables as global
    UserName = input("What is your name (Will be used as the file name)? \n")   #Takes an input from a user, \n is a new line
    while True:                                                               #Stars a loop till made false
        try:                                                                  #Tests the next block of code for errors
            ModeInput = int(input("""\nWould you like to use:
    1. Basic Mode
    2. Advanced Mode
    3. More information
"""))                                                                         #Takes an input from the user as an integer
            if ModeInput == 3:                                                #If the user asks for more info
                print("In Basic mode, compatibility checks will be enabled throughout the program, meaning that the components that you select will work with each other. Advanced mode is only for professionals as they will not have any compatibility checks.")
            elif ModeInput == 1 or ModeInput == 2:                            #If a valid input is received, break loop
                break
            else:                                                             #If a number out of range is provided
                print("***Please enter a valid number***\n")                  #Tells the user to give a valid number
        except ValueError:                                                    #If the input from the user is not an integer
            print("***Please enter the number***\n")                          #Tell the user to enter an integer
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def ComponentSelection():
    global ComponentInput
    while True:
        try:
            ComponentInput = int(input("""\nWhat component would you like to select?
    1. CPU
```

```
    2. Motherboard
    3. Memory
    4. Storage
    5. Video Card
    6. Power Supply
    7. Case
"""))
            if ComponentInput <1 or ComponentInput >7:      #If the number is out of the range 1-7
                print("***Please enter a valid number***")
            else:
                break                                       #Stops when a valid number has been given
        except ValueError:
            print("***Please enter the number***\n")
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def ItemSelection():
    global ComponentList
    while True:
        ComponentSelection()

        Database = sqlite3.connect("Components2.db")        #The object Database represents the Components database
        cursor = Database.cursor()                          #The object cursor allows you to execute commands
        ComponentList = ["CPU","Motherboard","Memory","Storage","VideoCard","PowerSupply","Tower"]
                                                            #Array storing table names in database Components
        print("\n", pandas.read_sql_query("SELECT * FROM {}".format(ComponentList[ComponentInput-1]), Database, "id"))
                                                            #Uses pandas to print the results of the search, indexed by id
        while True:
            ItemChoice = input("\nWhat item would you like? \n")    #Allows the user to enter the id of the component to add
            cursor.execute('''SELECT id FROM '''+ ComponentList[int(ComponentInput)-1] + ''' WHERE id = ''' + ItemChoice)
            Check = cursor.fetchall()                               #Checks if the chosen id is in the database
            print("---------------------------------------------------------------------------------------------------")
            if Check == []:
                print("***Please enter a valid id***")              #If the database returns no results, try again
            else:
                ComponentChoice[ComponentInput-1] = ItemChoice      #Saves the choice for exporting
                break

        AnotherComponent = int(input("""\nDo you want to select another component?
        1. Yes
        2. No
"""))
        if AnotherComponent == 1:                                   #If the user wants to add another product...
            pass                                                    #...Does not break loop, so it repeats
        elif AnotherComponent == 2:
            break
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
83. def UserComponents():
84.     global UserName
85.     Database = sqlite3.connect("Components2.db")
86.     cursor = Database.cursor()
87.
88.     loops = 0                                       #loops variable for number of loops
89.     UserChoices = [["CPU",""],                      #2d arrays store component category and component for export
90.                    ["Motherboard",""],
91.                    ["Memory",""],
92.                    ["Storage",""],
93.                    ["Video Card",""],
94.                    ["Power Supply",""],
95.                    ["Case",""]
96.                    ]
97.
98.     for position in ComponentChoice:                #For each component category
99.         while loops <= 7:                           #while loops is less than or equal to 7
100.                try:
101.                    cursor.execute('SELECT Name FROM ' + ComponentList[loops-1] + ' WHERE id = {}'.format(ComponentChoice[loops-1]))
102.                    for row in cursor:                  #Gets the name of each component to save in the builds database
103.                        ItemText = row[0]               #[0] saves the component name without brackets
104.                except sqlite3.OperationalError:        #If there is an error
105.                    ItemText = 0                        #Set the component to 0, meaning not chosen
106.                UserChoices[loops-1][1] = ItemText      #Set this component to the corresponding index in the 2d array
107.                loops = loops + 1                       #Manually increment loop
108.
109.         db = sqlite3.connect("Customers.db")            #Connects to customers builds database
110.         cursor = db.cursor()
111.         while True:
112.             try:
113.                 cursor.execute("""CREATE TABLE {}(id INTEGER PRIMARY KEY, Component TEXT, Name TEXT)""".format(UserName))
114.                 break                                   #Only break loop if it can create a new table
115.             except sqlite3.OperationalError:            #Or get a new file name and try again
116.                 UserName = input("A file with this name may already exists or this may contain special characters. Please enter another
    name.\n")
117.         cursor.executemany("""INSERT INTO """+UserName+""" Values (NULL,?,?)""",UserChoices)
118.         Database.commit()
119.         db.commit()                                     #Save the user's components into their table
120.     ##    cursor.execute('''SELECT * FROM ''' +UserName)    #Use this method to view all the build without pandas
121.     ##    for row in cursor:
122.     ##        print(row)
123.         print("Saved")
124.         print(pandas.read_sql_query("SELECT * FROM "+UserName, db, "id")) #Print table using pandas
125.         Database.close()                                #Close the database to prevent data corruption
126.         db.close()
```

```
    #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    def BasicMode():
        Database = sqlite3.connect("Components2.db")
        cursor = Database.cursor()

        while True:
            while True:
                #Socket compatiblity
                if ComponentChoice[0] and ComponentChoice[1] != "":                          #If a CPU and Motherboard have been selected...
                    print("Checking socket compatibilty.")
                    CPUCheck = cursor.execute('''SELECT Socket FROM CPU WHERE id = ''' + ComponentChoice[0]).fetchall()     #Save CPU socket
                    MotherboardCheck = cursor.execute('''SELECT Socket FROM Motherboard WHERE id = ''' + ComponentChoice[1]).fetchall()     #Save Mobo socket
                    if CPUCheck == MotherboardCheck:                                           #If the socket is the same, continue
                        print("Socket is Compatible.")
                        break
                    else:
                        print("The socket is not compatible. Please choose another CPU or Motherboard.")
                        ItemSelection()                                                        #Run the module ItemSelection to allow them to choose another component
                else:
                    print("Socket Compatibility check not required.")
                    break
            while True:
                #FormFactor compatibility
                if ComponentChoice[1] and ComponentChoice[5] != "":                          #Motherboard and Power Supply
                    print("Checking form factor compatibilty.")
                    MotherboardCheck = cursor.execute('''SELECT FormFactor FROM Motherboard WHERE id = ''' + ComponentChoice[1]).fetchall()
                    PowerSupplyCheck = cursor.execute('''SELECT FormFactor FROM PowerSupply WHERE id = ''' + ComponentChoice[5]).fetchall()
                    if MotherboardCheck == PowerSupplyCheck:
                        print("Form Factor is Compatible-1")
                        break
                    else:
                        print("The Form Factor is not compatible. Please choose another Power Supply or Motherboard.")
                        ItemSelection()
                else:
                    print("Form Factor Compatibility check not required-1")
                    break
            while True:
                if ComponentChoice[5] and ComponentChoice[6]!= "":                           #Power Supply and Case
                    print("Checking form factor compatibilty.")
```

```
                PowerSupplyCheck = cursor.execute('''SELECT FormFactor FROM PowerSupply WHERE id = ''' + ComponentChoice[5]).fetchall()
                TowerCheck = cursor.execute('''SELECT FormFactor FROM Tower WHERE id = ''' + ComponentChoice[6]).fetchall()
                if PowerSupplyCheck == TowerCheck:
                    print("Form Factor is Compatible-2")
                    break
                else:
                    print("The Form Factor is not compatible. Please choose another Power Supply or Case.")
                    ItemSelection()
            else:
                print("Form Factor Compatibility check not required-2")
                break
        while True:
            if ComponentChoice[1] and ComponentChoice[6]!= "":                      #Motherboard and Case
                print("Checking form factor compatibilty.")
                MotherboardCheck = cursor.execute('''SELECT FormFactor FROM Motherboard WHERE id = ''' + ComponentChoice[1]).fetchall()
                TowerCheck = cursor.execute('''SELECT FormFactor FROM Tower WHERE id = ''' + ComponentChoice[6]).fetchall()
                if MotherboardCheck == TowerCheck:
                    print("Form Factor is Compatible-3")
                    break
                else:
                    print("The Form Factor is not compatible. Please choose another Case or Motherboard.")
                    ItemSelection()
            else:
                print("Form Factor Compatibility check not required-3")
                break
        break
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ComponentChoice = ["","","","","","",""]
Mode()
ItemSelection()
if ModeInput == 1:                          #If basic mode is selected
    BasicMode()
UserComponents()
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

CODE FOR PROTOTYPE 3/FINAL

```
#Bhupinder Dhoofer GUI

import sys
import sqlite3
import pandas
from tkinter import messagebox as tkMessageBox
from tkinter import *
import tkinter.ttk as ttk

 def start_gui():
     global root
     root = Tk()
     top = TopWindow (root)
     root.mainloop()

 class TopWindow:
     def __init__(self, top=None):
         _bgcolor = '#0B0C10'   # Background colour for the widget
         _fgcolor = '#c5c6c7'   # Foreground color for the widget (tab text)
         _compcolor = '#66FCF1' # Current Tab Background colour
         _ana2color = '#45A29E' # Active Background colour for tabs
         font10 = "-family {Courier New} -size 10 -weight normal -slant"  \
             " roman -underline 0 -overstrike 0"
         self.style = ttk.Style()
         if sys.platform == "win32":
             self.style.theme_use('winnative')
         self.style.configure('.',background=_bgcolor)
         self.style.configure('.',foreground=_fgcolor)
         self.style.configure('.',font="TkDefaultFont")
         self.style.map('.',background=
             [('selected', _compcolor), ('active',_ana2color)])

         top.geometry("1920x1027+-1+-19")
         top.title("PC Building Toolkit")
         top.configure(background="#0B0C10")

         root.option_add("*Font", "helvetica 20")#Large

```

```
39.         self.style.configure('TNotebook.Tab', background=_bgcolor)
40.         self.style.configure('TNotebook.Tab', foreground=_fgcolor)
41.         self.style.map('TNotebook.Tab', background=
42.             [('selected', _compcolor), ('active',_ana2color)])
43.         self.AllTabs = ttk.Notebook(top)
44.         self.AllTabs.place(relx=0.0, rely=0.0, relheight=1.0, relwidth=1.0)
45.         self.AllTabs.configure(width=1894)
46.         self.Page1 = ttk.Frame(self.AllTabs)
47.         self.AllTabs.add(self.Page1, padding=3)
48.         self.AllTabs.tab(0, text="Home Page",underline="-1",)
49.         self.Page2 = ttk.Frame(self.AllTabs)
50.         self.AllTabs.add(self.Page2, padding=3)
51.         self.AllTabs.tab(1, text="Components Page",underline="-1",)
52.         self.Page3 = ttk.Frame(self.AllTabs)
53.         self.AllTabs.add(self.Page3, padding=3)
54.         self.AllTabs.tab(2, text="Item Page",underline="-1",)
55.         self.Page4 = ttk.Frame(self.AllTabs)
56.         self.AllTabs.add(self.Page4, padding=3)
57.         self.AllTabs.tab(3, text="Current Parts Page",underline="-1",)
58.         self.Page5 = ttk.Frame(self.AllTabs)
59.         self.AllTabs.add(self.Page5, padding=3)
60.         self.AllTabs.tab(4, text="Final Build Page",underline="-1",)
61.         self.Page6 = ttk.Frame(self.AllTabs)
62.         self.AllTabs.add(self.Page6, padding=3)
63.         self.AllTabs.tab(5, text="End Page",underline="-1",)
64.
65.         def theme(widget):
66.             widget.configure(activebackground="#66FCF1")        #Cyan, Active Background colour
67.             widget.configure(activeforeground="#0B0C10")        #Black, Active Foreground colour
68.             widget.configure(background="#0B0C10")              #Black, Background colour
69.             widget.configure(foreground="#c5c6c7")              #Grey, Foreground colour
70.
71.         self.P1L1 = Label(self.Page1)
72.         self.P1L1.place(relx=0.01, rely=0.02, height=211, width=1874)
73.         self.P1L1.configure(background="#0B0C10")
74.         self.P1L1.configure(foreground="#c5c6c7")
75.         self.P1L1.configure(text='''PC Building Toolkit''')
76.
77.         self.P1B1 = Button(self.Page1)
78.         self.P1B1.place(relx=0.01, rely=0.35, height=315, width=1876)
```

```
79.          self.P1B1.configure(text='''Build a PC!''')
80.          self.P1B1.configure(command = lambda : [self.AllTabs.select(self.Page2), UserName(self.P1E1.get())])
81.
82.          self.P1B2 = Button(self.Page1)
83.          self.P1B2.place(relx=0.01, rely=0.72, height=85, width=1876)
84.          self.P1B2.configure(text='''Build a PC - Advanced Mode''')
85.          self.P1B2.configure(command = lambda : [self.AllTabs.select(self.Page2), UserName(self.P1E1.get()),
    AdvancedMode()])
86.
87.          def AdvancedMode():
88.              global AdvancedMode
89.              AdvancedMode = True     #Turns boolean True
90.
91.          self.P1L2 = ttk.Label(self.Page1)
92.          self.P1L2.place(relx=0.01, rely=0.83, height=139, width=1866)
93.          self.P1L2.configure(background="#0B0C10")
94.          self.P1L2.configure(foreground="#c5c6c7")
95.          self.P1L2.configure(relief=FLAT)
96.          self.P1L2.configure(text='''What is the difference?
97.
98.  When using "Build a PC!", we will run a compatibility check to ensure that all of your parts will work together.
99.  By using "Advanced Mode", we will not run the compatibility checker.''')
100.
101.         self.P1E1 = Entry(self.Page1)
102.         self.P1E1.place(relx=0.32, rely=0.24, relheight=0.08, relwidth=0.67)
103.         self.P1E1.configure(background="#0B0C10")
104.         self.P1E1.configure(font=font10)
105.         self.P1E1.configure(foreground="#c5c6c7")
106.         self.P1E1.configure(insertbackground="black")
107.
108.         self.P1L3 = Label(self.Page1)
109.         self.P1L3.place(relx=0.03, rely=0.24, height=71, width=544)
110.         self.P1L3.configure(background="#0B0C10")
111.         self.P1L3.configure(foreground="#c5c6c7")
112.         self.P1L3.configure(text='''What is your name?
113. This will be used as the file name''')
114.         self.P1L3.configure(justify=RIGHT)
115.
116.         self.P2L1 = Label(self.Page2)
117.         self.P2L1.place(relx=0.01, rely=0.02, height=211, width=1874)
```

```
118.         self.P2L1.configure(activebackground="#f9f9f9")
119.         self.P2L1.configure(activeforeground="black")
120.         self.P2L1.configure(background="#0B0C10")
121.         self.P2L1.configure(foreground="#c5c6c7")
122.         self.P2L1.configure(text='''Choose the component to configure''')
123.
124.         self.P2BB = Button(self.Page2)
125.         self.P2BB.place(relx=-0.01, rely=0.93, height=75, width=966)
126.         self.P2BB.configure(text='''Back''')
127.         self.P2BB.configure(command = lambda : self.AllTabs.select(self.Page1))
128.
129.
130.         self.P2B1 = Button(self.Page2)
131.         self.P2B1.place(relx=0.01, rely=0.2, height=74, width=1907)
132.         self.P2B1.configure(pady="0")
133.         self.P2B1.configure(text='''CPU''')
134.         self.P2B1.configure(width=1907)
135.         self.P2B1.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(1)])
136.
137.         self.P2B2 = Button(self.Page2)
138.         self.P2B2.place(relx=0.01, rely=0.29, height=74, width=1907)
139.         self.P2B2.configure(pady="0")
140.         self.P2B2.configure(text='''Motherboard''')
141.         self.P2B2.configure(width=1907)
142.         self.P2B2.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(2)])
143.
144.         self.P2B3 = Button(self.Page2)
145.         self.P2B3.place(relx=0.01, rely=0.38, height=74, width=1907)
146.         self.P2B3.configure(pady="0")
147.         self.P2B3.configure(text='''Memory''')
148.         self.P2B3.configure(width=1907)
149.         self.P2B3.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(3)])
150.
151.         self.P2B4 = Button(self.Page2)
152.         self.P2B4.place(relx=0.01, rely=0.47, height=74, width=1907)
153.         self.P2B4.configure(pady="0")
154.         self.P2B4.configure(text='''Storage''')
155.         self.P2B4.configure(width=1907)
156.         self.P2B4.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(4)])
157.
```

```
158.         self.P2B5 = Button(self.Page2)
159.         self.P2B5.place(relx=0.01, rely=0.56, height=74, width=1907)
160.         self.P2B5.configure(pady="0")
161.         self.P2B5.configure(text='''Video Card''')
162.         self.P2B5.configure(width=1907)
163.         self.P2B5.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(5)])
164.
165.         self.P2B6 = Button(self.Page2)
166.         self.P2B6.place(relx=0.01, rely=0.65, height=74, width=1907)
167.         self.P2B6.configure(pady="0")
168.         self.P2B6.configure(text='''Power Supply''')
169.         self.P2B6.configure(width=1907)
170.         self.P2B6.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(6)])
171.
172.         self.P2B7 = Button(self.Page2)
173.         self.P2B7.place(relx=0.01, rely=0.74, height=74, width=1907)
174.         self.P2B7.configure(pady="0")
175.         self.P2B7.configure(text='''Case''')
176.         self.P2B7.configure(width=1907)
177.         self.P2B7.configure(command = lambda : [self.AllTabs.select(self.Page3), ItemSelection(7)])
178.
179.         self.P2FB = Button(self.Page2)
180.         self.P2FB.place(relx=0.51, rely=0.93, height=75, width=956)
181.         self.P2FB.configure(text='''Skip''')
182.         self.P2FB.configure(command = lambda : self.AllTabs.select(self.Page3))
183.
184.         self.P3L1 = Label(self.Page3)
185.         self.P3L1.place(relx=0.01, rely=0.02, height=211, width=1874)
186.         self.P3L1.configure(activebackground="#f9f9f9")
187.         self.P3L1.configure(activeforeground="black")
188.         self.P3L1.configure(background="#0B0C10")
189.         self.P3L1.configure(foreground="#c5c6c7")
190.         self.P3L1.configure(text='''Please select a component from the previous page''')
191.
192.         self.P3BB = Button(self.Page3)
193.         self.P3BB.place(relx=-0.01, rely=0.93, height=75, width=966)
194.         self.P3BB.configure(text='''Back''')
195.         self.P3BB.configure(command = lambda : self.AllTabs.select(self.Page2))
196.
197.         self.P3L2 = Label(self.Page3)
```

```
198.         self.P3L2.place(relx=0.01, rely=0.18, relheight=0.71, relwidth=0.99)
199.         self.P3L2.configure(background="#0B0C10")
200.         self.P3L2.configure(foreground="#c5c6c7")
201.         self.P3L2.configure(text='''The list of items will show here''')
202.         self.P3L2.configure(width=1890)
203.
204.         self.P3E1 = Entry(self.Page3)
205.         self.P3E1.place(relx=0.51, rely=0.93, relheight=0.07, relwidth=0.24)
206.         self.P3E1.configure(background="#0B0C10")
207.         self.P3E1.configure(font=font10)
208.         self.P3E1.configure(foreground="#c5c6c7")
209.         self.P3E1.configure(insertbackground="black")
210.         self.P3E1.configure(selectbackground="#c4c4c4")
211.         self.P3E1.configure(selectforeground="black")
212.         self.P3E1.configure(width=464)
213.
214.         self.P3FB = Button(self.Page3)
215.         self.P3FB.place(relx=0.75, rely=0.93, height=75, width=486)
216.         self.P3FB.configure(text='''Next''')
217.         self.P3FB.configure(width=486)
218.         self.P3FB.configure(command = lambda :
    [self.AllTabs.select(self.Page4),ItemSave(self.P3E1.get()),UserComponents()])
219.
220.         self.P4L1 = Label(self.Page4)
221.         self.P4L1.place(relx=0.01, rely=0.02, height=211, width=1874)
222.         self.P4L1.configure(activebackground="#f9f9f9")
223.         self.P4L1.configure(activeforeground="black")
224.         self.P4L1.configure(background="#0B0C10")
225.         self.P4L1.configure(foreground="#c5c6c7")
226.         self.P4L1.configure(text='''Current Part List''')
227.
228.         self.P4BB = Button(self.Page4)
229.         self.P4BB.place(relx=-0.01, rely=0.93, height=75, width=966)
230.         self.P4BB.configure(text='''Back''')
231.         self.P4BB.configure(command = lambda : self.AllTabs.select(self.Page3))
232.
233.         self.P4L1 = Label(self.Page4)
234.         self.P4L1.place(relx=0.01, rely=0.23, height=411, width=1874)
235.         self.P4L1.configure(activebackground="#f9f9f9")
236.         self.P4L1.configure(activeforeground="black")
```

```
237.         self.P4L1.configure(background="#0B0C10")
238.         self.P4L1.configure(foreground="#c5c6c7")
239.
240.         self.P4B1 = Button(self.Page4)
241.         self.P4B1.place(relx=0.01, rely=0.66, height=145, width=1886)
242.         self.P4B1.configure(text='''Add another component...''')
243.         self.P4B1.configure(command = lambda : self.AllTabs.select(self.Page2))
244.
245.         self.P4B2 = Button(self.Page4)
246.         self.P4B2.place(relx=0.5, rely=0.93, height=75, width=956)
247.         self.P4B2.configure(text='''Finish''')
248.         self.P4B2.configure(command = lambda : [self.AllTabs.select(self.Page5), ModeSelection()])
249.
250.         def ModeSelection():
251.             global AdvancedMode
252.             if AdvancedMode == True:        #Checks boolean status
253.                 pass                        #Ignores the check if True
254.             else:
255.                 BasicMode()                 #Else it will run it
256.
257.         self.P5L1 = Label(self.Page5)
258.         self.P5L1.place(relx=0.01, rely=0.02, height=211, width=1874)
259.         self.P5L1.configure(activebackground="#f9f9f9")
260.         self.P5L1.configure(activeforeground="black")
261.         self.P5L1.configure(background="#0B0C10")
262.         self.P5L1.configure(foreground="#c5c6c7")
263.         self.P5L1.configure(text='''Here is your build''')
264.
265.         self.P5BB = Button(self.Page5)
266.         self.P5BB.place(relx=-0.01, rely=0.93, height=75, width=966)
267.         self.P5BB.configure(text='''Back''')
268.         self.P5BB.configure(command = lambda : self.AllTabs.select(self.Page4))
269.
270.         self.P5L2 = Label(self.Page5)
271.         self.P5L2.place(relx=0.01, rely=0.21, height=461, width=1874)
272.         self.P5L2.configure(activebackground="#f9f9f9")
273.         self.P5L2.configure(activeforeground="black")
274.         self.P5L2.configure(background="#0B0C10")
275.         self.P5L2.configure(foreground="#c5c6c7")
276.
```

```
277.         self.P5B1 = Button(self.Page5)
278.         self.P5B1.place(relx=0.0, rely=0.77, height=145, width=946)
279.         self.P5B1.configure(text='''Yes''')
280.         self.P5B1.configure(command = lambda : [self.AllTabs.select(self.Page6),Export()])#save
281.
282.         self.P5B2 = Button(self.Page5)
283.         self.P5B2.place(relx=0.51, rely=0.77, height=145, width=946)
284.         self.P5B2.configure(text='''No''')
285.         self.P5B2.configure(command = lambda : self.AllTabs.select(self.Page6))
286.
287.         self.P6L1 = Label(self.Page5)
288.         self.P6L1.place(relx=0.01, rely=0.66, height=91, width=1874)
289.         self.P6L1.configure(activebackground="#f9f9f9")
290.         self.P6L1.configure(activeforeground="black")
291.         self.P6L1.configure(background="#0B0C10")
292.         self.P6L1.configure(foreground="#c5c6c7")
293.         self.P6L1.configure(text='''Do you want to save your build?''')
294.
295.         self.P6L1 = Label(self.Page6)
296.         self.P6L1.place(relx=0.01, rely=0.02, height=211, width=1874)
297.         self.P6L1.configure(activebackground="#f9f9f9")
298.         self.P6L1.configure(activeforeground="black")
299.         self.P6L1.configure(background="#0B0C10")
300.         self.P6L1.configure(foreground="#c5c6c7")
301.         self.P6L1.configure(text='''Thank You''')
302.
303.         self.P6BB = Button(self.Page6)
304.         self.P6BB.place(relx=-0.01, rely=0.93, height=75, width=966)
305.         self.P6BB.configure(text='''Back''')
306.         self.P6BB.configure(command = lambda : self.AllTabs.select(self.Page5))
307.
308.         self.P6L2 = Label(self.Page6)
309.         self.P6L2.place(relx=0.01, rely=0.21, height=41, width=1894)
310.         self.P6L2.configure(background="#0B0C10")
311.         self.P6L2.configure(foreground="#c5c6c7")
312.         self.P6L2.configure(text='''How was your experience?''')
313.         self.P6L2.configure(width=1894)
314.
315.         self.P6E1 = Entry(self.Page6)
316.         self.P6E1.place(relx=0.02, rely=0.28, relheight=0.39, relwidth=0.96)
```

```
317.         self.P6E1.configure(background="#0B0C10")
318.         self.P6E1.configure(font=font10)
319.         self.P6E1.configure(foreground="#c5c6c7")
320.         self.P6E1.configure(insertbackground="black")
321.         self.P6E1.configure(width=1834)
322.
323.         self.star = PhotoImage(file="star.gif")
324.
325.         self.P6B1 = Button(self.Page6)
326.         self.P6B1.place(relx=0.02, rely=0.69, height=84, width=347)
327.         self.P6B1.configure(image=self.star, command = lambda : ChooseStar(1))
328.
329.         self.P6B2 = Button(self.Page6)
330.         self.P6B2.place(relx=0.215, rely=0.69, height=84, width=347)
331.         self.P6B2.configure(image=self.star, command = lambda : ChooseStar(2))
332.
333.         self.P6B3 = Button(self.Page6)
334.         self.P6B3.place(relx=0.41, rely=0.69, height=84, width=347)
335.         self.P6B3.configure(image=self.star, command = lambda : ChooseStar(3))
336.
337.         self.P6B4 = Button(self.Page6)
338.         self.P6B4.place(relx=0.605, rely=0.69, height=84, width=347)
339.         self.P6B4.configure(image=self.star, command = lambda : ChooseStar(4))
340.
341.         self.P6B5 = Button(self.Page6)
342.         self.P6B5.place(relx=0.8, rely=0.69, height=84, width=347)
343.         self.P6B5.configure(image=self.star, command = lambda : ChooseStar(5))
344.
345.         self.P6FB = Button(self.Page6)
346.         self.P6FB.place(relx=0.02, rely=0.79, height=95, width=1836)
347.         self.P6FB.configure(text='''Submit''')
348.         self.P6FB.configure(command = lambda : [self.P6FB.configure(text="Submitted"), Feedback()])
349.
350.         def ChooseStar(x):
351.             global star
352.             star = x                                                    #Stores the number of stars provided
   from button input
353.             stararray = [self.P6B1,self.P6B2,self.P6B3,self.P6B4,self.P6B5]
354.             for i in range(star):
355.                 stararray[i].configure(background="#FBEB19")            #Makes all buttons Yellow
```

```
356.             i+=1
357.             while True:
358.                 if i != 5:                                                  #If 5(i) stars is not given
359.                     stararray[i].configure(background="#0B0C10")            #Makes the last star black
360.                     i+=1
361.                 else:
362.                     break                                                   #Continues till only the clicked
    stars and previous are yellow
363.
364.         def Feedback():
365.             Comment = self.P6E1.get()                                       #Fetches the feedback from the entry
    box
366.             db = sqlite3.connect("Customers.db")
367.             cursor = db.cursor()
368.             try:
369.                 cursor.execute("""CREATE TABLE {}Feedback (id INTEGER PRIMARY KEY, Stars TEXT, Comment
    TEXT)""".format(UserNameV))
370.                                                                             #Inserts the comment and number of
    stars to the database
371.                 cursor.execute("""INSERT INTO """+UserNameV+"""Feedback (Stars, Comment) Values (?,?)""",(star,Comment))
372.                 db.commit()
373.                 print(pandas.read_sql_query("SELECT * FROM "+UserNameV+"Feedback", db))
374.                 db.close()
375.             except (sqlite3.OperationalError, UnboundLocalError):           #If an error occurs..
376.                 tkMessageBox.showerror("File Name Error", "A file with this name may already exists or this may contain
    special characters. Please enter another name and try again.")
377.                 self.AllTabs.select(self.Page1)                             #Asks the user to try another name
378.
379.
380.
381.         ButtonsToTheme = [self.P1B1,self.P1B2,self.P2BB,self.P2B1,self.P2B2,self.P2B3,self.P2B4,
382.                           self.P2B5,self.P2B6,self.P2B7,self.P2FB,self.P3BB,self.P3FB,self.P4BB,
383.                           self.P4B1,self.P4B2,self.P5BB,self.P5B1,self.P5B2,self.P5BB,self.P6BB,
384.                           self.P6B1,self.P6B2,self.P6B3,self.P6B4,self.P6B5,self.P6FB]
385.
386.         for i in ButtonsToTheme:                        #For all of the buttons in the array
387.             theme(i)                                    #Theme them
388.
389.         def UserName(UserName):                         #Passing by reference
390.             global UserNameV
```

```
            UserNameV=UserName
            while True:
                for i in range(6):                  #If the file name is taken
                    if UserChoices[i][1] != "":     #and a build has been created
                        break
                self.AllTabs.select(self.Page5)     #Show final build page
                break



        ComponentChoice = ["","","","","","","",""]
        ComponentList = ["CPU","Motherboard","Memory","Storage","VideoCard","PowerSupply","Tower"]

        def ItemSelection(ComponentInput):
            global ComponentInputV
            ComponentInputV = ComponentInput
            self.P3L1.configure(text=ComponentList[ComponentInput-1])               #To meet stakeholder requirements,
                                                                                    #Shows category name as title
            Database = sqlite3.connect("Components2.db")
            cursor = Database.cursor()
            cursor.execute('''SELECT * FROM ''' + ComponentList[ComponentInput-1])  #Fetches all components from category
            x = []                                                                  #empty array
            y=""                                                                    #empty string

            for column in cursor.execute("SELECT * FROM {}".format(ComponentList[ComponentInput-1])).description:
                                                                                    #Gets table headings...
                x.append(column[0])                                                 #Stores them in array x
            for row in cursor:
                x.extend(row)                                                       #Adds all components to array
            for i in x:                                                             #This is to format the text correctly
                if isinstance(i,int) == True:                                       #When the index is the id as int,
                    print("")                                                       #Prints a new row, for diagnostics
                    y=y+"\n\n"                                                      #Adds a new row to the string
                print("{:<25s}".format(str(i)),end="")                              #The sting will be 25 characters
                                                                                    #Left aligned, not printing a new
   line
                y=y+"{:<40s}".format(str(i))                                        #Adds to string, left aligned 40 char
            print("\n")                                                            #New line
            y=y+"\n"                                                                #New Line
```

```
            print(y)                                                                    #For diagnostics/monitoring
            self.P3L2.configure(text=y)                                                 #Prints on label


        def ItemSave(ItemChoice):
            Database = sqlite3.connect("components2.db")
            cursor = Database.cursor()
            cursor.execute('''SELECT id FROM '''+ ComponentList[int(ComponentInputV)-1] + ''' WHERE id = ''' +
   ItemChoice)
            Check = cursor.fetchall()                                                   #Fetches chosen component from
   database
            print("-------------------------------------------------------------------------------------------")
            if Check == []:
                print("***Please enter a valid id***")                                  #If that id is not there (so
   incorrect input)
            else:
                ComponentChoice[ComponentInputV-1] = ItemChoice                         #Saves the component



        def UserComponents():
            global table
            global UserChoices
            Database = sqlite3.connect("components2.db")
            cursor = Database.cursor()

            loops = 0                                                       #loops variable for number of loops
            UserChoices = [["CPU",""],                                      #2d arrays store component category and
   component for export
                           ["Motherboard",""],
                           ["Memory",""],
                           ["Storage",""],
                           ["Video Card",""],
                           ["Power Supply",""],
                           ["Case",""]
                           ]

            for position in ComponentChoice:                #For each component category
                while loops <= 7:                           #while loops is less than or equal to 7
```

```
                    try:
                        cursor.execute('SELECT Name FROM ' + ComponentList[loops-1] + ' WHERE id =
{}'.format(ComponentChoice[loops-1]))
                        for row in cursor:                  #Gets the name of each component to save in the builds
database
                            ItemText = row[0]               #[0] saves the component name without brackets
                    except sqlite3.OperationalError:        #If there is an error
                        ItemText = 0                        #Set the component to 0, meaning not chosen
                    UserChoices[loops-1][1] = ItemText      #Set this component to the corresponding index in the 2d
array
                    loops = loops + 1                       #Manually increment loop

            table = """

 ============== =================
   Component    \tName
 ============== =================
  ---------CPU  \t{}
  -Motherboard  \t{}
  ------Memory  \t{}
  -----Storage  \t{}
  --Video Card  \t{}
  Power Supply  \t{}
  --------Case  \t{}
 ============== =================

""".format(UserChoices[0][1],UserChoices[1][1],UserChoices[2][1],UserChoices[3][1],UserChoices[4][1],UserChoices[5][1],Us
erChoices[6][1])
            self.P4L1.configure(text=table,justify = LEFT)
            self.P5L2.configure(text=table,justify = LEFT)              #Displays table

        def Export():
            global UserNameV
            global UserChoices
            file = open(UserNameV+".txt","w+")                      #Will create or ammend the file
            file.write(table)                                       #Writes the contents of variable table to the file
            file.close()                                            #Closes file to prevent data corruption

            db = sqlite3.connect("Customers.db")                    #Connects to customers builds database
            cursor = db.cursor()
```

```
502.             try:
503.                 cursor.execute("""CREATE TABLE {} (id INTEGER PRIMARY KEY, Component TEXT, Name
   TEXT)""".format(UserNameV))     #Creates table
504.                 cursor.executemany("""INSERT INTO """+UserNameV+""" Values (NULL,?,?)""",UserChoices)
   #Saves build to database
505.                 db.commit()
506.                 cursor.execute('''SELECT * FROM ''' +UserNameV)
507.                 print("Saved")
508.                 print(pandas.read_sql_query("SELECT * FROM "+UserNameV, db, "id"))
509.                 self.P4L1.configure(text=pandas.read_sql_query("SELECT * FROM "+UserNameV, db, "id"))
   #Displays current build
510.                 db.close()
511.             except(sqlite3.OperationalError, UnboundLocalError):
   #If an error occurs.
512.                 tkMessageBox.showerror("File Name Error", "A file with this name may already exists or this may contain
   special characters. Please enter another name and try again.")
513.                 self.AllTabs.select(self.Page1)
   #Asks the user to try another file name and try again
514.
515.
516.         def BasicMode():
517.             Database = sqlite3.connect("components2.db")
518.             cursor = Database.cursor()
519.             if ComponentChoice[0] and ComponentChoice[1] != "":
   #If a CPU and Motherboard have been selected...
520.                 print("Checking socket compatibilty.")
521.                 CPUCheck = cursor.execute('''SELECT Socket FROM CPU WHERE id = ''' + ComponentChoice[0]).fetchall()
   #Save CPU socket
522.                 MotherboardCheck = cursor.execute('''SELECT Socket FROM Motherboard WHERE id = ''' +
   ComponentChoice[1]).fetchall()              #Save Mobo socket
523.                 if CPUCheck == MotherboardCheck:
   #If the socket is the same, continue
524.                     print("Socket is Compatible.")
525.                 else:
526.                     print("The socket is not compatible. Please choose another CPU or Motherboard.")
527.                     tkMessageBox.showerror("Compatibility Checker", "The socket is not compatible. Please choose another
   CPU or Motherboard.")  #Pop-up box error
528.                     self.AllTabs.select(self.Page2)
   #Goes to page that needs to be fixed
529.             else:
```

```
530.                 print("Socket Compatibility check not required.")
531.             if ComponentChoice[1] and ComponentChoice[5] != "":
   #Motherboard and Power Supply
532.                 print("Checking form factor compatibilty.")
533.                 MotherboardCheck = cursor.execute('''SELECT FormFactor FROM Motherboard WHERE id = ''' +
   ComponentChoice[1]).fetchall()
534.                 PowerSupplyCheck = cursor.execute('''SELECT FormFactor FROM PowerSupply WHERE id = ''' +
   ComponentChoice[5]).fetchall()
535.                 if MotherboardCheck == PowerSupplyCheck:
536.                     print("Form Factor is Compatible-1")
537.                 else:
538.                     print("The Form Factor is not compatible. Please choose another Power Supply or Motherboard.")
539.                     tkMessageBox.showerror("Compatibility Checker", "The Form Factor is not compatible. Please choose
   another Power Supply or Motherboard.")
540.                     self.AllTabs.select(self.Page2)
541.             else:
542.                 print("Form Factor Compatibility check not required-1")
543.             if ComponentChoice[5] and ComponentChoice[6]!= "":
   #Power Supply and Case
544.                 print("Checking form factor compatibilty.")
545.                 PowerSupplyCheck = cursor.execute('''SELECT FormFactor FROM PowerSupply WHERE id = ''' +
   ComponentChoice[5]).fetchall()
546.                 TowerCheck = cursor.execute('''SELECT FormFactor FROM Tower WHERE id = ''' +
   ComponentChoice[6]).fetchall()
547.                 if PowerSupplyCheck == TowerCheck:
548.                     print("Form Factor is Compatible-2")
549.                 else:
550.                     print("The Form Factor is not compatible. Please choose another Power Supply or Case.")
551.                     tkMessageBox.showerror("Compatibility Checker", "The Form Factor is not compatible. Please choose
   another Power Supply or Case.")
552.                     self.AllTabs.select(self.Page2)
553.             else:
554.                 print("Form Factor Compatibility check not required-2")
555.             if ComponentChoice[1] and ComponentChoice[6]!= "":
   #Motherboard and Case
556.                 print("Checking form factor compatibilty.")
557.                 MotherboardCheck = cursor.execute('''SELECT FormFactor FROM Motherboard WHERE id = ''' +
   ComponentChoice[1]).fetchall()
558.                 TowerCheck = cursor.execute('''SELECT FormFactor FROM Tower WHERE id = ''' +
   ComponentChoice[6]).fetchall()
```

```
559.                 if MotherboardCheck == TowerCheck:
560.                     print("Form Factor is Compatible-3")
561.                 else:
562.                     print("The Form Factor is not compatible. Please choose another Case or Motherboard.")
563.                     tkMessageBox.showerror("Compatibility Checker", "The Form Factor is not compatible. Please choose
    another Case or Motherboard.")
564.                     self.AllTabs.select(self.Page2)
565.             else:
566.                 print("Form Factor Compatibility check not required-3")
567.
568. if __name__ == '__main__':
569.     start_gui()
570.
```

CODE FOR ADMIN

This is the code Arka can use to add components to the components database, as well as view the builds and feedback from users of the program.

```
import sqlite3

Database = sqlite3.connect("Components2.db")

#Uncomment the code below to fill the components database by selecting the whole code and pressing alt+4 to remove the comments

##
##cursor = Database.cursor()
##cursor.execute ("""  CREATE TABLE IF NOT EXISTS CPU(id INTEGER PRIMARY KEY, Name TEXT, Cores TEXT, Socket TEXT)""")
##
##Cpus = [
##    ["AMD Ryzen 2600","6","AM4"],
##    ["Intel Core i5 6600k","4","LGA1151"],
##    ["Intel Core i5 8600k","6","LGA1151"],
##    ["Intel Core i7 8700k","6","LGA1151"],
##    ["AMD Ryzen 2700X","8","AM4"],
##    ]
##
##
##cursor.executemany("""INSERT INTO CPU(Name, Cores, Socket) Values (?,?,?)""",Cpus)
##Database.commit()
##print("CPU Done")
##
### Print the users
####cursor.execute('''SELECT * FROM CPU''')
####for row in cursor:
####    print(row)
##
####~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##cursor.execute ("""  CREATE TABLE IF NOT EXISTS Tower(id INTEGER PRIMARY KEY, Name TEXT, FormFactor TEXT)""")
##
##Name = [
##    ("NZXT S340","ATX"),
##    ("Corsair 200R","ATX"),
##    ("Cooler Master Lite 5","ATX"),
```

```
##     ("Fractal Design Focus G","ATX"),
##     ("Thermaltake Core V21","Micro ATX"),
##     ]
##
##
##cursor.executemany("""INSERT INTO Tower(Name, FormFactor) Values (?,?)""",Name)
##Database.commit()
##print("Tower Done")
####~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##cursor.execute (""" CREATE TABLE IF NOT EXISTS PowerSupply(id INTEGER PRIMARY KEY, Name TEXT, FormFactor TEXT, Wattage TEXT, Modularity TEXT)""")
##
##Name = [
##     ("EVGA SuperNova","ATX","650W","Full"),
##     ("Corsair CX550M","ATX","550W","Semi"),
##     ("Seasonic","ATX","750W","Full"),
##     ("Thermaltake","ATX","650W","Full"),
##     ("Corsair SF600","Micro ATX","600W","Full"),
##     ]
##
##
##cursor.executemany("""INSERT INTO PowerSupply(Name, FormFactor, Wattage, Modularity) Values (?,?,?,?)""",Name)
##Database.commit()
##print("PowerSupply Done")
####~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##cursor.execute (""" CREATE TABLE IF NOT EXISTS Motherboard(id INTEGER PRIMARY KEY, Name TEXT, FormFactor TEXT, Socket TEXT)""")
##
##Motherboards = [
##     ("MSI Z370 A PRO","ATX","LGA1151"),
##     ("Gigabyte B360M","Micro ATX","LGA1151"),
##     ("Asus STRIX B350-F","ATX","AM4"),
##     ("MSI Z270 Gaming Plus","ATX","LGA1151"),
##     ("ASRock AB350M Pro4","Micro ATX","AM4"),
##     ]
##
##
##cursor.executemany("""INSERT INTO Motherboard(Name, FormFactor, Socket) Values (?,?,?)""",Motherboards)
##Database.commit()
##print("Motherboard Done")
```

```
####~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##cursor.execute (""" CREATE TABLE IF NOT EXISTS Memory(id INTEGER PRIMARY KEY, Name TEXT, Type TEXT, Modules TEXT, Size Text)""")
##
##Memorys = [
##    ("Corsair Vengeance LPX","DDR4","1x8GB","8GB"),
##    ("Team Vulcan T-Force","DDR4","2x8GB","16GB"),
##    ("Patriot Viper White LED","DDR4","2x8GB","16GB"),
##    ("G.Skill Ripjaws V Series","DDR4","2x4GB","8GB"),
##    ("Kingston HyperX Fury","DDR4","2x16GB","32GB"),
##    ]
##
##
##cursor.executemany("""INSERT INTO Memory(Name, Type, Modules, Size) Values (?,?,?,?)""",Memorys)
##Database.commit()
##print("Memory Done")
####~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##cursor.execute (""" CREATE TABLE IF NOT EXISTS VideoCard(id INTEGER PRIMARY KEY, Name TEXT, Memory TEXT)""")
##
##Name = [
##    ("MSI GTX 1050 Ti","4GB"),
##    ("EVGA GTX 1060","6GB"),
##    ("Gigabyte GTX 1080","8GB"),
##    ("MSI RX 580","8GB"),
##    ("Asus RX VEGA 64","8GB"),
##    ]
##
##
##cursor.executemany("""INSERT INTO VideoCard(Name, Memory) Values (?,?)""",Name)
##Database.commit()
##print("VideoCard Done")
####~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##cursor.execute (""" CREATE TABLE IF NOT EXISTS Storage(id INTEGER PRIMARY KEY, Name TEXT, Type TEXT, Capacity TEXT)""")
##
##Name = [
##    ("Western Digital Blue","7200RPM","1TB"),
##    ("Seagate Barracuda","7200RPM","2TB"),
##    ("Toshiba X300","SSD","5TB"),
##    ("Kingston A400","SSD","120GB"),
##    ("Adata XPG M.2 NVMe","SSD","480GB"),
```

```
##    ]
##
##
##cursor.executemany("""INSERT INTO Storage(Name, Type, Capacity) Values (?,?,?)""",Name)
##Database.commit()
##print("Storage Done")


import pandas
print(pandas.read_sql_query("SELECT * FROM CPU", Database))         #Prints all components in the CPU table and their attributes
print("")
print(pandas.read_sql_query("SELECT * FROM Tower", Database))
print("")
print(pandas.read_sql_query("SELECT * FROM PowerSupply", Database))
print("")
print(pandas.read_sql_query("SELECT * FROM Motherboard", Database))
print("")
print(pandas.read_sql_query("SELECT * FROM Memory", Database))
print("")
print(pandas.read_sql_query("SELECT * FROM VideoCard", Database))
print("")
print(pandas.read_sql_query("SELECT * FROM Storage", Database))
print("\n")

Database.close()

Database = sqlite3.connect("Customers.db")
cursor = Database.cursor()

print(pandas.read_sql_query("SELECT name FROM sqlite_master WHERE type='table'", Database))     #Prints everything, such as user builds and feedback

x=0
while x != "":                                  #Will ask the stakeholder for name till they press enter with no name
    name = input("\nEnter user name: ")            #Asks for the table name to show
    print(pandas.read_sql_query("Select * from {} ".format(name), Database,"id"))      #Prints the whole table

##cursor.execute("DROP TABLE bsd ")             #Example to remove a table from the database
```