Centre: 51201, Candidate number: 1036

Database and Algorithm based bottleneck detection application

Corkscrew

A level NEA

Sebastian Dixon

Contents

[Analysis 2](#_Toc35770196)

[Preface 2](#_Toc35770197)

[Research to the problem 2](#_Toc35770198)

[Current system 3](#_Toc35770199)

[Proposed solution details 5](#_Toc35770200)

[Identification of end-user/ supervisor 6](#_Toc35770201)

[Prospective user(s) 6](#_Toc35770202)

[8](#_Toc35770203)

[Set of requirements 10](#_Toc35770204)

[Modelling of the problem 12](#_Toc35770205)

[Acceptable limitations (agreed with end-user/supervisor) 13](#_Toc35770206)

[Log/diary of research collection 15](#_Toc35770207)

[Documented design 17](#_Toc35770208)

[High-level overview (structure/hierarchy chart; system flowchart; object/class diagrams) 17](#_Toc35770209)

[Description of algorithms 20](#_Toc35770210)

[Description of data structures 20](#_Toc35770211)

[File structure and organisation 21](#_Toc35770212)

[Database design 21](#_Toc35770213)

[Queries (for database projects) 23](#_Toc35770214)

[Design of user interface (do not focus solely on the interface; pay more attention to the design of algorithms and documenting those) 23](#_Toc35770215)

[Technical solution 26](#_Toc35770216)

[Testing 43](#_Toc35770217)

[Evaluation 48](#_Toc35770218)

[How were the objectives met and achieved 48](#_Toc35770219)

[How can your proposed solution be improved? 48](#_Toc35770220)

[Analysis on feedback from the end user/supervisor 48](#_Toc35770221)

# Analysis

## Preface

The challenges associated with building a computer extend beyond the building process itself. The compatibility and synergy of the components is a large factor that plays into whether the computer will perform to its true potential. Often without the knowledge on how to configure the components of a computer correctly, the result might incur specific pieces of hardware of which are more powerful than others.

Forums which focus around PC hardware such as Pc Part Picker have many people who discuss the optimisation of computer systems. Here people post and debate different ideas and personal builds for all things related to computer hardware, accomplishments shared between users promoting a high degree of craftsmanship. This is the main source for all the information required in determining if the components within any system is a good choice or not.

Other sources of information came from YouTube channels such as Paul’s Hardware, Linus Tech Tips and Tech Quickie. Therefore, learning more on this topic and be able to personally have an opinion on which components are most or least compatible.

Bottlenecking is widely known as a term used to describe the underperforming component, preventing the whole system from performing well. Therefore, to help the user identify why their computer is not performing to the standard advertised, producing a piece of software to be run to administer the hardware an evaluate which component is ultimately keeping back the computer’s performance.

This information developed over a period simply being exposed to more and more examples of bottlenecked systems. This knowledge was the instilled on the algorithms and database within the software. Given all this information a piece of pc hardware limit detection and recommendation software. The software uses benchmarking utilities that stress the CPU and GPU independently.

## A close up of electronics Description automatically generated Research to the problem

The reason for prioritising the CPU and GPU is that they are the two most important and expensive components of the computer, rendering them the highest priority and contain the ability to throw off the balance of power between them very quickly. This inherently makes it very easy to choose a bottlenecked system from these two alone, but if not, then other factors must be considered as to be mentioned later.

It is important to understand that the state of a system being bottlenecked is not Boolean, but a gradient. Therefore, every single system has a degree of this property, large or small. This is an important factor to include when assessing the bottleneck of a system. Therefore, my project calculates the specific bottleneck degree between components, then checking if that value found correlates to a boundary for A to F. An A grade showing there is a very small bottleneck within the system, and F grade showing that there is a large bottleneck. Both grades will produce a list of recommendations for the user, although a lower grade will give a larger range of upgrades as the bottlenecked component is clearly less powerful.

A close up of electronics

Description automatically generatedAs the performance of the PC is not wholly reliant on the CPU and GPU, the amount of memory in the user’s pc will also be considered. If the quantity of RAM is too low in comparison to the other components, only then will the RAM potentially be identified as the bottleneck, as well as the other possible components holding the system back.

A close up of a device

Description automatically generatedIf the grade delta (difference in grade) is too small, no upgrade recommendation will be made. When in this case the performance is lower than is expected, the bottleneck will be resided on the memory capacity of the system.

It is important to understand that whilst the CPU and GPU are both very influential in the diagnosing of the system for a bottleneck. The raw RAM quantity will very quickly change the ceiling of performance for any system. The CPU and GPU could be perfectly suited to each other and yet with insufficient RAM, the system won’t perform to three quarters the speed of which is expected.

To make things more complicated the RAM inside a system has two main properties: speed and quantity. The speed commonly measured in MHz and quantity in GB, now approaching TB in extreme cases. It is difficult to assess if it is the speed of the RAM bottlenecking the system as different platforms from z390 to x299 all use and benefit differently from high speed RAM, therefore it could simply not be a factor. Another reason for which the speed of the RAM won’t be measured or accounted in this assessment of the system is that most modern motherboards are equipped with a function called XMP. This is an automatic overclocking profile which is applied to the system memory, making it run at its intended speeds out of the box, no to little user input required.

The raw size of the system memory is the more important factor at play, given that it can’t be changed in software, there is no such thing as downloading more RAM. However, it is possible to allocate a fixed portion of a fast system drive to act as back up for RAM which overflows, to put it simply.

Given that the quantity of the RAM is the most important factor, any recommendation including a RAM bottleneck will not suggest a brand or speed specific RAM upgrade, as the CPU or GPU recommendation would, but instead give a suggestion on a size requirement.

## Current system

A similar system called ‘The Bottlenecker’ is a piece of software which identifies the possible bottleneck between common hardware. This is accessed on a web browser, therefore requires an internet connection to run. This is a downside as it can’t be used in an offline environment, although not a common problem, isn’t something of which my program is dependent on. One upside of this system is that the list of components can easily be updated in the background, requiring no input from the user. This is a feature of my program of which can also be done, given the database is cloud based so can also be updated live with new compatibilities between components.

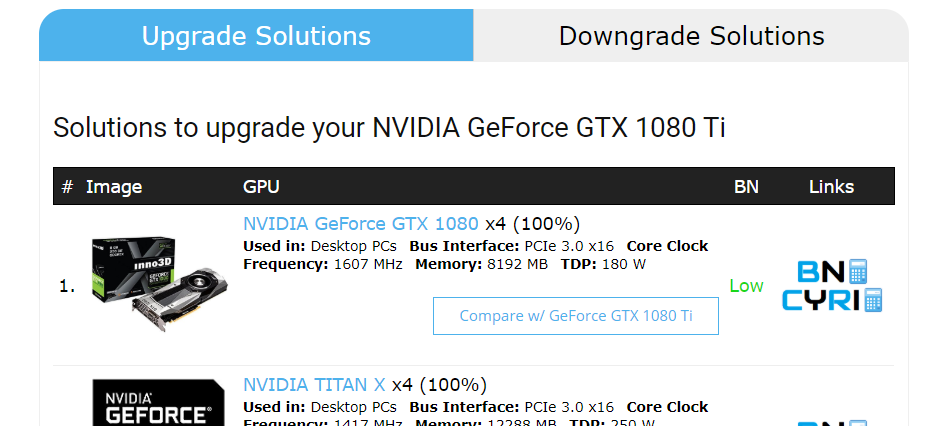
The program starts with a user navigating drop down menus for you to pick your system specification from. This includes picking your CPU, GPU, RAM. This is an intuitive design and is responsive in operation. There is a wide selection of both CPUs and GPUs to choose from, different major brands and components of up to 6 years prior.

Certain limitations of this program have been found whilst testing. The first major issue found was that when trying to input the latest pieces of hardware, they are not included. However, the nature of the software being based online is that the database of the parts can easily be updated in the background. Another issue is that the user might not know what hardware is in their machine. This issue would not occur within my program as the computer’s hardware is automatically identified, therefore it doesn’t need to be known or input by the user.

Another issue with the program is in the overclocking region. Overclocking is the function of increasing the power or clock rate of a component as to increase the overall performance of the component. The option for the CPU and GPU clock can only be increased to 130% the stock frequency of the component. This capping of the overclock is too limiting as many components have the capability to increase their core clock frequency far higher than 130%. This is done using voltage percentile modification in either the BIOS or overclocking software. This increased overclock can sometimes decrease the bottleneck between components, therefore limiting the overclock a component can be applied in a drop-down menu would potentially give an inaccurate representation of the user’s system on a whole.

One other issue with this overclocking system is that many components now come pre-overclocked, therefore the user might not be aware of the fact they have a component which is overclocked or not, this would inherently introduce either an increase or decrease in bottleneck. The exact percentage of the factory overclock is unknown to the user, therefore the drop-down menu would not be a useful metric of performance. This issue wouldn’t happen in my solution given that the performance on the component is measured in a real-world test, removing the necessity to let the user indicate how powerful their specific CPU or GPU is.

The clock speeds percentile drops down menu is purely manipulating the frequency of the chip itself, when on the graphics card the memory, power draw and several other items can be changed, each variable affecting performance enough that the bottleneck could increase or decrease. My method avoids this problem as the overclocked properties of the part produces a unique benchmark result, which indicates the performance of the part. My method removes the necessity for the user to know what multiplier is applied to the core and different parts of the system, as it is all recorded automatically during the benchmarking process. Therefore, this website’s method is less adaptable and accurate compared to my benchmark method when overclocking is involved.

After inputting your hardware in the drop-down menus, a report is produced which indicates the possible bottleneck between components. A recommendation to downgrade the GPU is neither necessary or helpful as the performance of the system would then decrease and the user is spending money on a component which is completely useless. Better to buy a component which improves system performance and reduces the bottleneck.

The detail offered to the user to inform them of the specification behind their own components and potential upgrades is extensive and offers possibilities for the user to compare each of them. A link is provided along with recommendations to popular shopping sites, one limitation of the website’s implementation is that the price and site region is fixed to North America, therefore this information and functionality is less useful to anyone outside that region.

After inputting a variety of hardware specification, an abnormal function occurs, a suggestion of upgrades is only made when the bottleneck between components is greater than a calculated score of 10%, therefore a relatively bottlenecked system is not suggested any help, even given the fact a user would reach to use the program given that they believe the performance of their system is lower than expected.

The recommendation of how to improve performance of the system with RAM doesn’t consider nor incorporate the users inputted quantity for flash memory. The suggestion made is entirely based on the quality of the CPU, and how much memory they believe is suitable for the system. This isn’t an accurate method of recommendation given quantity of RAM is specific to that of the use case.

The program does produce an accurate assessment of the bottleneck between the components. This appears to have been calculated via a ranking system of independent hardware devices. Although each component has been given a score and total ranking, it isn’t clear how this is calculated. This could have been made in an environment which is very different to the user’s specific use case for their system. Therefore, although this testing environment would have been made as to most accurately model the average use case and scenario, it’s not clear to the user, with nothing said about it.

It is a benefit of this system to include such a large range of components to assess and test, however, this isn’t beneficial to the user when both incompatible and unavailable components are suggested along with the list of recommendations. A common example of this is where a GPU bottleneck is detected, resulting in the output of a list of more powerful GPU’s. This list ranges from components which are just more powerful to the most powerful and expensive components. Some of these including some of the most powerful and expensive GPUs, these aren’t in fact available to regular consumers for purchase, e.g. Titan V CEO edition.

One fault of the scoring component scoring system is that there is no check made to ensure that components suggested are compatible with the CPU or GPU the user intends to carry on using. In one example having input a selection of desktop hardware, the system recommended me a mobile series GPU of which wouldn’t be readily compatible with the system is use. This shows that the recommendation is simply score based as that would be a basic omission of compatibility algorithms.

## Proposed solution details

A rating of common hardware would be stored in a data base so that a new piece of hardware could be suggested to the user to resolve the bottleneck, an amazon link also for ease of use.

An inbuilt leader board feature will allow the user to compare their components to the users who have run the program too, giving them an idea of the quality and compatibility of their parts off the bat.

So that the system is not entirely reliant on a benchmarking utility for the analysis of the system, another method will be used to asses for a bottleneck. This includes recording the percentage utilisation of the GPU and CPU during the benchmarking process to produce a graph. This graph will be analysed, a lower than normal percentage utilisation of either component will indicate that it is more powerful and has more headroom than the other. This method is complimentary towards the previous system and the result will be a combination of the two methods described.

The database of results for the benchmarks were sourced from a set of Forums who have collected the data from reliable sourced over a long period of time. This is useful since it includes data from older and very modern hardware. This makes the grading system far more accurate since most hardware that will be used during the test will have been tested already.

## Identification of end-user/ supervisor

Client:

The Client was to the be the IT department of my school, as they would be associated with building computers around the school and bottlenecking of hardware, is something they would have encountered. Therefore, it made sense to approach them professionally.

Dialogue with Client:

Hi,

I am writing a program for my computer science A-level non-examination assessment which requires a client, of whom can't be a teacher of mine. Therefore, I thought the IT department would be the next best thing, especially since your department is more familiar with the direction of the project I am undergoing. The clientele needs to know what they need out of the program, which will then influence the direction I take the features of the software. I am planning to produce a piece of software which will detect and try to assist you in eliminating the bottlenecks in your pc. The benchmarking utilities will produce data of which is readable by the program, once scraped, based on this data the scores are compared to a leader board, which produce a grade of the components. Using the grade delta, I can determine if a bottleneck is present or not.

Since this project is still in production there would be more to ask of the client in the future such as the features they wanted or the things the project might not have grasped which they wanted.

I hope you consider this

Regards,

Sebastian Dixon

## Prospective user(s)

Four pupils in a computer science class are appropriate end users, as they have previously worked with computers which can be upgraded. They are familiar with the challenges that occur when building your own computer. They have all successfully built their own computers and could use the software effectively. Their names are Stefan, William, Aaron and Sam.

**Stefan**:

Have you previously searched for software of a similar feature?

* No.

How do you think you could benefit from this software?

* To know which part of my pc needs to be upgraded first.

Would you refer this program to friends in similar situations?

* Yes, most definitely.

What specific features would you like to add to the software?

* A price to performance ratio grade on the components and information on motherboard type.

Do you know how to detect a bottleneck without this software?

* No, I don’t know how to detect a bottleneck without the software.

**William**:

Have you previously searched for software of a similar feature?

* Yes.

How do you think you could benefit from this software?

* It would allow me to save money improving my pc in the most efficient way.

Would you refer this program to friends in similar situations?

* Yes.

What specific features would you like to add to the software?

* Maybe a recommendation for the kind of upgrade you should get.

Do you know how to detect a bottleneck without this software?

* No.

**Aaron**:

Have you previously searched for software of a similar feature?

* Yes.

How do you think you could benefit from this software?

* When upgrading my computer, it would be useful to ensure that I’m not wasting my money, therefore getting the most out of my new hardware

Would you refer this program to friends in similar situations?

* Probably not, because most my friends use laptops which can’t be upgraded

What specific features would you like to add to the software?

* A way to assess the specs of the monitor being used and see if it is too high a resolution or low a refresh rate for the hardware.

Do you know how to detect a bottleneck without this software?

* No.

**Sam**:

Have you previously searched for software of a similar feature?

* No.

How do you think you could benefit from this software?

* I would use the leader board feature to check new and upcoming hardware in comparison to my own.

Would you refer this program to friends in similar situations?

* Yes.

What specific features would you like to add to the software?

* Maybe a recommendation for the kind of upgrade you should get.

Do you know how to detect a bottleneck without this software?

* Yes, not very accurately all and with limited hardware.

## 

From these results we can see that even within the small portion of interviews that there is a demand for this type of program with half of clients all interested in this product. Of the four people who are suitable for this program 75 percent of which would then refer the program to others, showing that they believe that the program is effective on all hardware bases and that there are even more people who could benefit from this information. Another piece of interesting information to take from this interview would be that the clients were, on a whole, unfamiliar with how to detect if a system is being bottleneck or not. Thereby this program being in demand and the solution to a problem.

## Set of requirements

Functionality:

* Request login information
  + The login username and password are requested from the user at the login window.
* Register or login user
  + The user can either login with existing details or register a new account which is pushed to the login database
* Login password hashed and checked against database value
  + The password is securely hashed to increase the security of the program, so no plaintext passwords are stored on the login database
* test CPU utilisation
  + the library pyadl and cpuinfo are used for logging the percentage utilisation of the CPU during the benchmark process
* test GPU utilisation
  + the library GPUtil is used for logging the percentage utilisation of the NVIDIA GPU during benchmarking
  + the library pyadl is used to monitor and log the percentage utilisation of an AMD GPU during benchmark process
* test ram utilisation
  + the system library can be utilised to monitor the percentage utilisation of the RAM in the system during the benchmark process
* start benchmark software
  + the software Unigine heaven benchmark increases both processor and graphical processer utilisation as a large number of calculations are required to be made for the program to run with acceptable performance. This increases the load placed on the CPU and GPU which can be analysed for calculating the bottleneck.
* parse benchmark result HTML file for results
  + the unigine benchmark outputs a results file which has html values. These can be parsed and used for the automatic writing to the users database for the score and fps values of the hardware configuration
* train buffer for deduction of cut and dry bottlenecks
  + a buffer between CPU and GPU bottlenecks is necessary to provide an accurate analysis of the system, so there is not just one percentage difference between deducing a CPU or GPU bottleneck
* analyse utilisation arrays for bottleneck
  + comparing the averages of the utilisation of the CPU and GPU over the same period of time gives an estimate of the type of relative performance between the two components.
* offer double system bottleneck if the ram is insufficient
  + in the situation where there is no CPU or GPU bottleneck the ram utilisation array is analysed to suggest if the performance imbalance is due to the lack of ram for the score given
* assign bottleneck to CPU/GPU
  + the bottleneck function uses the buffer and the higher average percentage utilisation from the benchmark to determine which of the two or three components renders a bottleneck
* find said CPU/GPU name
  + the item name which is being bottlenecked is then searched using either cpuinfo for the CPU or GPUtil for the GPU. The name of the specific ram in use isn’t important, only the capacity and speed.
* search hardware database for more powerful component
  + an SQL query is made with the parameters of finding the opposite pair value to the more powerful component in the users system. The output being a list of more powerful compatible components
* sort hardware database for compatible components
  + sorting the values upon request from more powerful / expensive to less powerful / expensive for the users understanding
* output list of amazon URL for recommended solutions
  + an amazon URL is output along with the list of recommended upgrades. This makes the purchase of the upgrade easier and faster for the user. Also giving the ability to gauge an idea for how expensive the upgrade may be.
* push recommended items to user database
  + for the user to keep track of the upgrades available to them, the recommended upgrades are pushed to their accounts user database for checking in later
* issue video recommendation of upgrading bottleneck type.
  + If the upgrade is either a CPU, GPU or RAM a video URL can be supplied in the help window if the user isn’t familiar with the process required to carry out the upgrade

GUI:

* output component names in readable GUI
  + taking the array generated of upgrades and outputting them to the user in a interface.
* give assistance / help window on how to use program
  + a help window for easy identification of what settings to be used in the benchmark for calibrating the performance and score / fps with the other components in the hardware database.
* bring up easy to use finder window for selection of html file
  + a familiar user friendly file explorer window generated for the user to indicate the file path for the html file containing the score and fps values for the system in benchmarking
* show specific component name of bottleneck
  + upon calculating the bottleneck a pop up window familiarises the user with the specific component type and its name which is preventing full performance

## Modelling of the problem

A close up of a map

Description automatically generatedData Flow Diagram:

## Acceptable limitations (agreed with end-user/supervisor)

One limitation of the program lies within the analysis of the components within the motherboard buses. The speed of the buses which control the flow of data within the processor and motherboard can affect the overall performance of the computer however the difference it makes overall isn’t measurable in terms of score or fps, metrics which my program uses for performance. The difficulty taken with which to measure the values in the buses is difficult to measure. Given the combination of small functional difference and high difficulty of measuring would make this an acceptable limitation and therefore not necessary to be addressed in the program.

In the line of computer hardware there are two main categories, one being consumer hardware which is easily accessible to general users due to its wide availability and approachable price. The other being industry level hardware, often referred to as server hardware, this is often designed for a different purpose or has greater brute performance in sacrifice for aesthetics or noise / heat etc. These components are therefore not ones used by the audience of the program and therefore can’t be as effectively monitored or assessed by the methods and algorithms used in this program. The processors are often optimised for different workloads with a high number of cores, but lower frequency, this would appear to the program as a weak processor for the workload being targeted at improving. However, in effect this architecture is simply designed for another purpose. Therefore, including these in the recommendation process would not only be confusing and off-putting to the user, but wouldn’t be compatible with operating systems and or many other components in their system.

The intended platform is limited to personal computers as they are the easiest for someone to upgrade themselves. For this reason, the program wasn’t extended for use in mobile situations such as laptop or phone hardware, as the componentry of these system is far more complicated and differs a lot between units. The ability to suggest upgrades to the system would be very difficult as there would not only be no or a limited number of components to upgrade to. Were a recommendation to be made, the physical upgrade would often have to be carried out by a professional due to the high risk of damaging the rest of the system in replacing a part. The operating systems and BIOS of these devices also commonly rejects different hardware to what was installed in its production, making the upgrade pointless.

Each processor and motherboard have different constraints on memory capacity. Motherboards may only support a few physical slots for RAM modules and knowledge of how these are compromised in the system isn’t possible for a program of this level. The different configurations of memory come from being able to change the number of channels the motherboard is using for the quantity of memory.

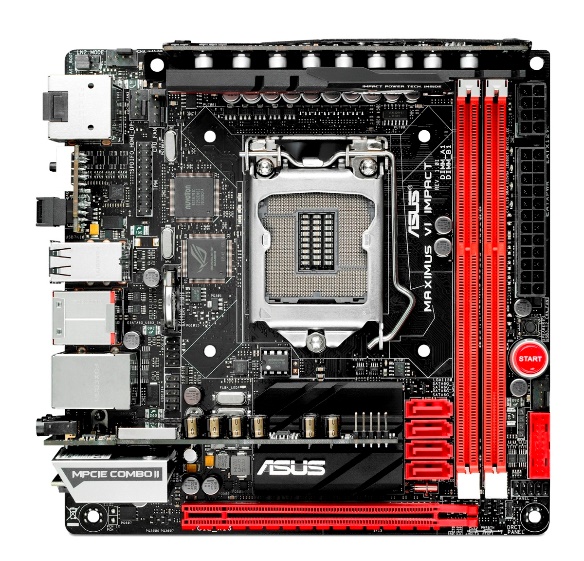
As an example, it would be possible for a specific motherboard to have only 2 available memory slots or DIMM slots (Figure 1). The registered quantity of memory would read as 16GB in the program. It is therefore unknown if the user has a configuration of one 16GB module or two 8GB modules. The later situation would prevent them from expanding their memory capacity without getting rid of the current components.

Figure 1



Figure 2

In another example it would be possible for another motherboard to have 8 available DIMM slots (Figure 2) and a registered memory capacity of 64GB with two 32GB individual RAM modules being used. In this situation the user would have space to upgrade and therefore a recommendation would be possible if the system required it.

The CPU and OS type both have a maximum logical memory allocation, it isn’t possible to identify this limit for each processor or operating system being tested or recommended. Therefore, given the uncertainty of the user’s personal configuration of memory components, it doesn’t appear to be a worthwhile feature to implement into the program.

Graphics cards have the capability to run with multiple other versions of itself, in a multi GPU array (Figure 3). This is often used as it is the only other method of increasing the raw GPU power of a single system. However, this feature is specific to the GPU model, CPU model and lastly motherboard specifics such as physical space and available GPU modules, known as PCI-E slots.

For the GPU the model manufacturer must have enabled this multi GPU feature on the chip, known as either SLI or nvlink for NVIDIA and Crossfire technology for AMD cards.

This feature must be first possible as each GPU requires a PCI-E slot. In Figure 1 the motherboard is shown to only have one PCI-E slot, therefore any multi GPU recommendation made would be incorrect. For Figure 2 there are multiple available bays, therefore there is physical space and it would be possible.

To further confuse the notion of motherboard compatibility with this feature, some motherboard models may well have the capacity and ability to accommodate more GPU’s, however, the technology of SLI or Crossfire etc. may not be supported by the chipset.

Each PCI-E lane is directly connected to the CPU. Therefore, the CPU must have enough bandwidth to handle the higher number of channels at once. Without enough bandwidth on the CPU, the possibility of multi GPU processing is nul.

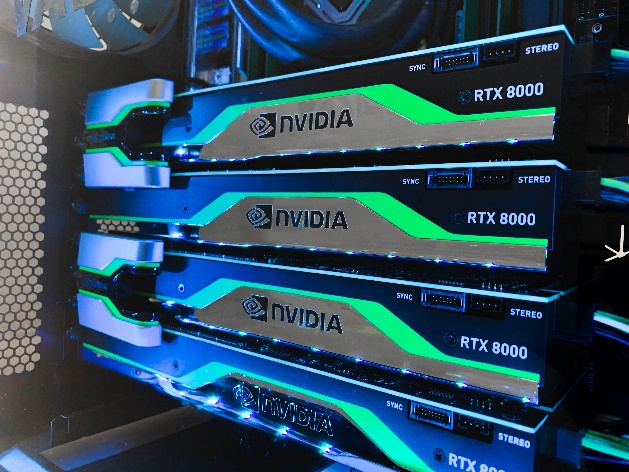
Making a recommendation to upgrade with another GPU in a multi GPU structure would have the potential at removing a GPU bound bottleneck as you are increasing the performance on a whole in a slightly different way. However, identifying every single reliant property of the user’s motherboard isn’t a feasible task and therefore has become an acceptable limitation for the program.

Figure 3

## Log/diary of research collection

<http://zetcode.com/gui/pyqt5/firstprograms/>

* documentation into the details of producing first pyqt5 program, initialisation of classes, framework for running and showing windows

<https://github.com/anderskm/gputil>

* documentation for python library into analysing the NVIDIA GPU

<https://pythonprogramminglanguage.com/pyqtgraph-plot/>

* documentation for python library for producing an interactable graph for the user to view based off array data structures

<https://pythonprogramminglanguage.com/pyqt/>

* help with GUI library pyqt5

<https://github.com/nicolargo/pyadl>

* documentation for python library pyadl for AMD GPU

<https://doc.qt.io/qt-5/qcombobox.html#details>

* documentation for python library pyqt5 and one of its widgets a Qcombobox, its different parameters and functionality

<https://riverbankcomputing.com/pipermail/pyqt/2007-December/018036.html>

* documentation for python library pyqt5 and one of its widgets a Qtextbrowser, used for learning how to integrate into GUI

<https://pythonspot.com/pyqt5-file-dialog/>

* documentation for python library pyqt5 for the GUI of file explorer used in the benchmarking process of the program

<https://pythonspot.com/pyqt5-input-dialog/>

* documentation for python library pyqt5 for inputting data from the user into the program

<https://www.overclock.net/forum/21-benchmarking-software-discussion/1235557-official-top-30-heaven-benchmark-4-0-scores.html>

* a forum post used to document users hardware performance metrics over a with a CPU and GPU pair

<https://stackoverflow.com/questions/11551996/why-do-we-need-the-finally-clause-in-python>

* a forum post used to learn how to correctly implement a finally clause in python

<https://docs.google.com/spreadsheets/d/1d5ZpydfFuJ77ZZPtofuiwl-eJkbM60oGyfSEDMPlVPQ/edit#gid=1>

* the spreadsheet from the forum post for all the users hardware pairings and their fps and score in the heaven Unigine benchmark.

<http://zetcode.com/python/pymysql/>

* documentation for python library pymysql which was used for learning how to connect to a MySQL database

<https://dev.mysql.com/doc/connector-python/en/connector-python-example-cursor-select.html>

* documentation for python library pymysql which was used for learning how to connect to a MySQL database

<https://pc-builds.com/calculator/>

* the other system for identifying bottlenecks within a system. I used this to gauge an aspect for what users commonly find useful when learning more about their system, I evaluated the pros and cons of this software and designed my program around improving on it and finding different methods of accomplishing the same goal.

<https://stackoverflow.com/questions/42672047/how-to-hide-password-in-qlineedit>

* used for blocking out the characters in the password login widget

<https://nitratine.net/blog/post/how-to-hash-passwords-in-python/>

* different hashing algorithms compared for password hashing algorithm

<https://stackoverflow.com/questions/9594125/salt-and-hash-a-password-in-python>

* implementing the salt library usage in python

<https://stackoverflow.com/questions/12790876/python-mysql-how-to-properly-use-s-for-counting-rows>

* different methods of counting the total number of rows for the library mysql

<https://stackoverflow.com/questions/5191503/how-to-select-the-last-record-of-a-table-in-sql>

* SQL dialect for identification of last item in a column

# Documented design

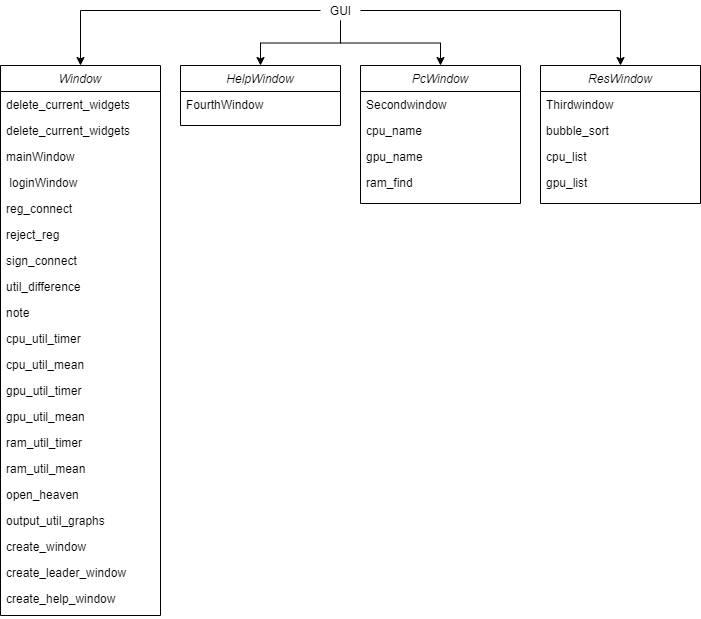
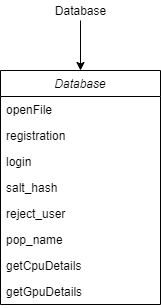
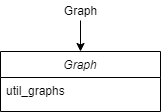
The program is to be produced using the programming languages Python and SQL. Python is ideal for this project as it includes both object orientated and functional programming functionality. The object orientated structure of the project will allow for easier navigation and breaking down of the task into smaller parts. The functional property will allow for objects to be individually utilised and categorised all working together as small building blocks for a final product. Libraries in python make importing of key information and functions useful for quick development. For example, the Unittest library is one which can be setup to test the logical functionality of the program. Python is a high-level language and is therefore easier for people to understand and write.

SQL was chosen as the language for database management. This is the most popular and powerful query language and so is chosen for the reading and writing of data into the tables.

## High-level overview (structure/hierarchy chart; system flowchart; object/class diagrams)

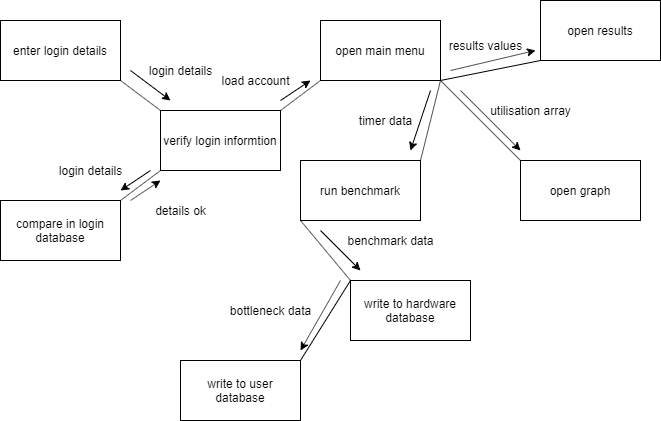
Class diagram

The GUI class is made up of four different classes which all hold independent windows to the user.



The Database class includes only one class and is related to the GUI class. The Graph script contains only one function and is related to the GUI class.

Structure chart



A close up of text on a whiteboard

Description automatically generatedSystem flowchart

## Description of algorithms

check\_creds

* checks the integrity of the credential provided to the database, if the password isn’t given in a hashed form, the password is hashed and then given to the database in the correct form.

util\_difference

* uses the mean average values of the utilisations for the CPU and GPU and then compare the values, having integrated the buffer in, determining the bottleneck in the two values.

cpu\_util\_mean

* the algorithm uses the time taken to record the values and the percentage utilisation to find the mean utilisation over time. It purposefully doesn’t run at the same time as other utilisation checkers to prevent overlap

gpu\_util\_mean

* the algorithm uses the time taken to record the values and the percentage utilisation to find the mean utilisation over time.

ram\_util\_mean

* the algorithm uses the time taken to finish the benchmark and the percentage utilisation to find the mean utilisation over time.

bubble\_sort

* uses a bubble sort for arranging the order of the recommended components to the user in length order.

salt\_hash

* implements a hashing function which takes two inputs of both salt and password plaintext. The salt is a bytes object and password a string data type. The salt is random upon computation. The password hash is output.

## Description of data structures

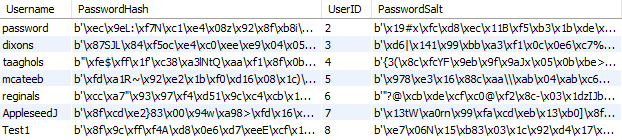
|  |  |
| --- | --- |
| Name of data structure | Description |
| widgets | The array contains the widgets of the GUI which are currently on display. The j |
| cpuinfo.get\_cpu\_info()['brand'] | A dictionary containing the information about the cpu, the brand key being chosen |
| GPUtil.getGPUs(); | A dictionary containing the gpu current information |
| results | An array storing the parsed html information such as score and fps |
| recommend\_cpu | An array for the recommended cpu model names |
| recommend\_gpu | An array for the recommended gpu model names |
| connection | A tuple storing the sql database connection variables |
| rows | A dictionary with the pulled database rows and columns |
| temp | First temporary array for storing the components pulled from the database for recommendation |
| temp2 | Second temporary array for length of component names |
| rows3 | Dictionary for the item from the database select statement |
| cpu\_y | An array for cpu utilisation |
| gpu\_y | An array for gpu utilisation |
| ram\_y | An array for ram utilisation |
| time\_x | An array for time taken for utilisation testing |
| recommend\_cpu\_url | An array for the recommended cpu url |
| recommend\_gpu\_url | An array for the recommended gpu url |

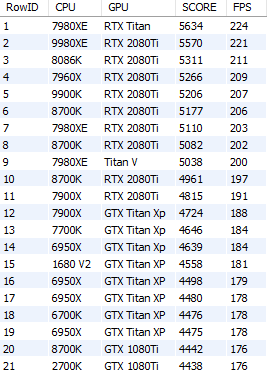
## File structure and organisation

The project is separated into three main script files and a README file for informing users on the origins of the program and what it was designed for. All of which are organised under the Corkscrew file.

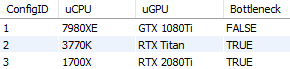
## Database design

One database connection called Hardware holds one schema called hardware. This schema holds three different tables which have independent purpose. the three different tables are login, parts and user. The database is in normalised 3NF form. Each table has a primary key which is auto generated in the database class. Therefore, each row has a unique identifier which is important for the parts table as each combination of different components must be identified in the database. There is inheritance between the table login and user in order to track the username of the login. There is no transitive dependency in the database.

The login table has columns Username, PasswordHash, UserID and PasswordSalt. The UserID is the primary key and is represented by an integer value next to each account. The Username is stored as plaintext in the database in order to compare it to new registered users for the user system. The password is not stored as plaintext, but as a hash of the original value, therefore it is safer, meaning the owner of the database doesn’t have access to the users account directly due to the password being encrypted. The hashing produced is designed to be a one-way hash in order to prevent the plaintext from being calculated if the hashing algorithm used is found by a malicious SQL injection. The storage of the password is even more secure having used a salt. This is a random generated string which is also input to the hash algorithm. Given the random nature of the salt, in order to check if the users password is correct or not, the generated salt must be stored along with the users account. Upon logging back into the users account, the salt is pulled from the username input and then input to the hashing algorithm along with the plaintext password. If the output generated from the function is the same value as the password hash stored against the same UserID, then the password is correct, and the account is accessed.

The Parts table is arranged to store the information relating to the different configurations of the CPU and GPU. This is the main table accessed in the assessment of a systems bottleneck. This gives the recommendations to the user based on the already confirmed results of the pairs in the listing. Each pairing has an identifier which is the primary key of the table RowID and a SCORE and FPS value. The result of which indicate the overall performance of the pairing. The necessity for both metrics of performance is due to multiple rows having the same FPS value, as it is less of a distinguishing factor, although still important in the comparison of the devices. The SCORE is a metric of the raw performance of the components combined power. The FPS is a metric of the effective performance of the components. Having a higher FPS to SCORE ratio shows that the pairing is more compatible together as you have a higher effective performance compared to the raw calculated performance of the parts used.

The user table is designed to store the status of a bottleneck of hardware configurations which shouldn’t be stored in the parts table. The values shouldn’t be stored in the parts table in order to prevent a false recommendation which may not resolve the bottleneck. The columns uCPU and uGPU get the data from the users very own CPU and GPU hardware and the bottleneck Boolean value after the benchmarking analysis. The ConfigID column is the primary key for the row in order to categorise the hardware combinations. There is a chance of duplicate data being input, therefore if the combination of both CPU and GPU is attempted to input, it is rejected, as the Bottleneck status is highly likely to be the same.



## Queries (for database projects)

## Design of user interface (do not focus solely on the interface; pay more attention to the design of algorithms and documenting those)

The GUI module contains the algorithms and manages the data which the user directly interacts with in the application. the use of object orientated programming meant that a class was made for each main section of the interface. this was broken down into the login window, main window, help window and results window.

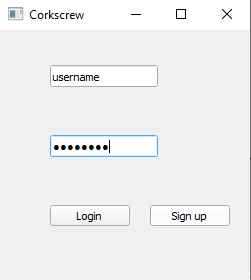
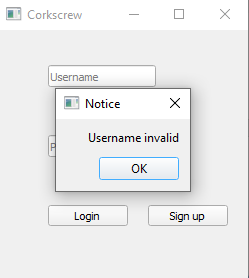
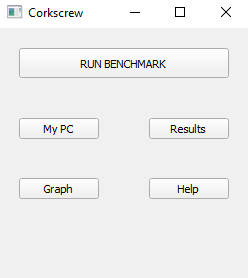
The first prompt the user is given is to login, or to make an account for the program. The username is visible, and the password is put in a dot form for privacy. From here, if the user signs up having input a username already used, the form clears, and the user is given an error message. *Figure 1.*

Figure 4

the main window displays a grid of 5 buttons for the user to navigate the program. Each opens a new window beside the main menu as to open multiple tabs at once. From here the user can find the devices internal hardware, the results from their most recent benchmark, a graph comparing the utilisations of the users CPU/GPU and a text document helping them efficiently use the program.

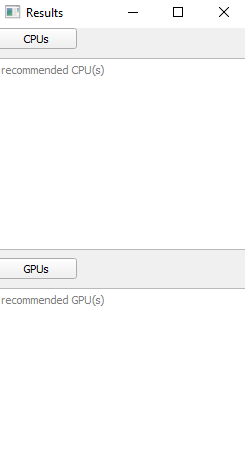
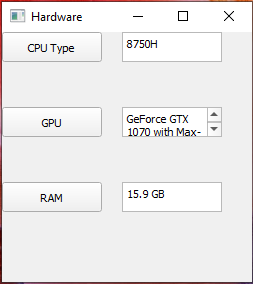


4

2

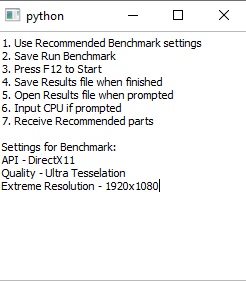
3

1



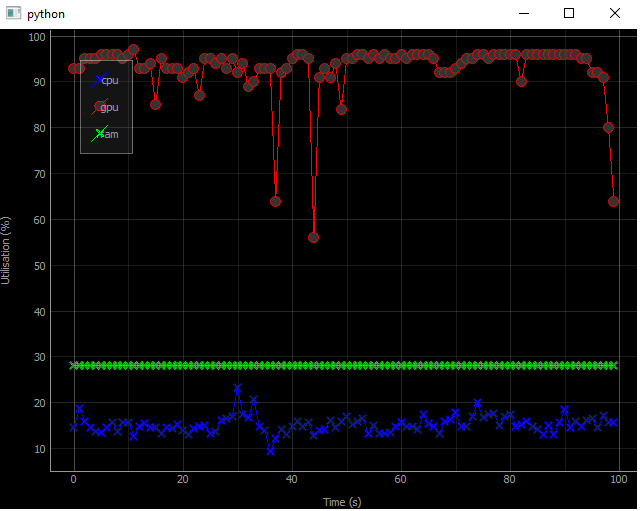
**1.** The My PC tab allows the user to find the names of the individual components in their system. This was a widely requested feature, as many users were not familiar with how to find this out themselves. The specific reasons for which it was requested was due to being able to compare the internals with others and understand their differences in performance.

2. The result of the benchmark analysis will output a list of recommended parts and URL’s shown to the user here.



4. The Help tab indicates to the user how the program is designed to be used. Also specifying parameters, the benchmark should use in order to produce results which can be compared to those in the Parts database.

2. The result of the benchmark analysis will output a list of recommended parts shown to the user here.



3. The graph is essential to the user fully understanding the power relationship between the CPU and GPU, and where the performance deficit is coming from. The data is taken from the cpu\_y and gpu\_y data arrays in the graph module.

# Technical solution

GUI Script:

import Graph

import Database

import pyadl

import cpuinfo

import os

import psutil

import subprocess

import GPUtil

from psutil import virtual\_memory

import sys

from PyQt5.QtWidgets import \*

from PyQt5.Qt import QApplication

import time

bottle = ""

class Window(QWidget and QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.widgets = []

self.loginWindow()

def delete\_current\_widgets(self):

"""

removes all current widgets in the window

"""

try:

for \_ in range(len(self.widgets)):

if type(self.widgets[0]) is not str:

self.widgets[0].deleteLater()

self.widgets.remove(self.widgets[0])

except RuntimeError:

print("no1")

def show\_widgets(self):

"""

displays all the new widgets for the new window

"""

try:

for widget in self.widgets:

widget.show()

except RuntimeError:

print("no2")

def mainWindow(self):

"""

the main menu for navigating Corkscrew

assigns the order of operations for the benchmarking process.

"""

self.delete\_current\_widgets()

self.resize(250, 200)

self.setWindowTitle('Corkscrew')

self.runBtn = QPushButton('RUN BENCHMARK', self)

self.runBtn.setGeometry(20, 20, 210, 30)

self.runBtn.clicked.connect(self.open\_heaven)

self.runBtn.clicked.connect(self.ram\_util\_timer)

self.runBtn.clicked.connect(self.gpu\_util\_timer)

self.runBtn.clicked.connect(self.cpu\_util\_timer)

self.runBtn.clicked.connect(self.cpu\_util\_mean)

self.runBtn.clicked.connect(self.ram\_util\_mean)

self.runBtn.clicked.connect(self.gpu\_util\_mean)

self.runBtn.clicked.connect(self.util\_difference)

self.graph = QPushButton('Graph', self)

self.graph.resize(self.graph.sizeHint())

self.graph.move(20, 150)

self.graph.clicked.connect(self.output\_util\_graphs())

self.pcBtn = QPushButton('My PC', self)

self.pcBtn.resize(self.pcBtn.sizeHint())

self.pcBtn.move(20, 90)

self.pcBtn.clicked.connect(self.create\_window)

self.leadBtn = QPushButton('Results', self)

self.leadBtn.resize(self.leadBtn.sizeHint())

self.leadBtn.move(150, 90)

self.leadBtn.clicked.connect(self.create\_leader\_window)

self.helpBtn = QPushButton('Help', self)

self.helpBtn.resize(self.helpBtn.sizeHint())

self.helpBtn.move(150, 150)

self.helpBtn.clicked.connect(self.create\_help\_window)

self.delete\_current\_widgets()

self.widgets = [self.runBtn, self.graph, self.pcBtn, self.leadBtn, self.helpBtn]

self.show\_widgets()

self.show()

# account info

def loginWindow(self):

"""

allows a user to login or sign up with an account for the application

the first window of the program

blocks out the password plaintext for security

starts the database account system

"""

self.delete\_current\_widgets()

self.resize(250, 250)

self.setWindowTitle('Corkscrew')

self.username = QLineEdit(self)

self.username.move(50, 50)

self.username.resize(self.username.sizeHint())

self.username.setPlaceholderText("Username")

self.password = QLineEdit(self)

self.password.move(50, 120)

self.password.resize(self.password.sizeHint())

self.password.setEchoMode(QLineEdit.Password)

self.password.setPlaceholderText("Password")

self.log\_in = QPushButton("Login", self)

self.log\_in.move(150, 200)

self.log\_in.resize(self.log\_in.sizeHint())

self.sign\_up = QPushButton("Sign up", self)

self.sign\_up.move(50, 200)

self.sign\_up.resize(self.sign\_up.sizeHint())

self.sign\_up.clicked.connect(lambda: self.reg\_connect(self.username.text(), self.password.text()))

self.log\_in.clicked.connect(lambda: self.sign\_connect(self.username.text(), self.password.text()))

self.widgets = [self.username, self.password, self.sign\_up]

self.show\_widgets()

self.show()

def reg\_connect(self, username, password):

"""

initialises the connection to the database class passing the username and password from the login window

the registration object in the class is specified

"""

db = Database.Database()

return db.registration(username, password)

def reject\_reg(self):

"""

returns the user back to the login window if the information is rejected by the database class

"""

QMessageBox.about(self, "Notice", "Username invalid")

self.show()

def sign\_connect(self, username, password):

"""

initialises the connection to the database class passing the username and password from the login window

the sign up object in the class is specified

"""

db = Database.Database()

return db.login(username, password)

# bottleneck calculator

def util\_difference(self):

"""

determines the bottleneck using the utilisation tests during the benchmark

comparing the mean average utilisation of the components a bottleneck is found

a file system GUI is opened in order for the user to specific the directory of the benchmark results file

the note object is called after determining the bottleneck

"""

global bottle

c\_length = len(Graph.cpu\_y)

g\_length = len(Graph.gpu\_y)

cpu\_total = 0

gpu\_total = 0

for i in range(c\_length):

cpu\_total += Graph.cpu\_y[i]

for x in range(g\_length):

gpu\_total += Graph.gpu\_y[x]

cpu\_average = cpu\_total / len(Graph.cpu\_y)

gpu\_average = gpu\_total / len(Graph.gpu\_y)

Database.Database.openFile(self)

if cpu\_average > gpu\_average:

bottle = 'CPU'

else:

bottle = 'GPU'

self.note()

self.result\_box = QTextEdit(self)

self.result\_box.move(120, 0)

self.result\_box.setPlaceholderText('Component')

def note(self):

"""

Gives a window notification to the user detailing which component is bottlenecking the system

calls the specific component object in the database class

"""

db = Database.Database()

if bottle == 'CPU':

QMessageBox.about(self, "Notice", "Bottleneck = CPU")

db.getGpuDetails()

else:

QMessageBox.about(self, "Notice", "Bottleneck = GPU")

db.getCpuDetails()

self.show()

# cpu

def cpu\_util\_timer(self):

"""

the percentage utilisation of the cpu is found and append to the cpu specific array in the graph script

100 seconds for each components testing to fill the 300 second benchmark time

"""

for n in range(100):

Graph.cpu\_y.append(psutil.cpu\_percent())

Graph.time\_x.append(n)

time.sleep(1)

print(Graph.cpu\_y)

def cpu\_util\_mean(self):

"""

finds the mean average of the cpu utilisaiton in the array of values

"""

length = len(Graph.cpu\_y)

product = 0

for x in range(length):

product += Graph.cpu\_y[x]

mean = product / length

rounded\_mean = round(mean, 3)

print('Average CPU utilisation =', rounded\_mean, '%')

# gpu

def gpu\_util\_timer(self):

"""

the percentage utilisation of the gpu is found and append to the gpu specific array in the graph script

the brand of gpu is found and the utilisaiton is found using either the NVIDIA or AMD based gpu library

100 seconds for each components testing to fill the 300 second benchmark time

"""

for n in range(100):

try:

GPUs = GPUtil.getGPUs()

gpu\_load = GPUs[0].load \*100

Graph.gpu\_y.append(gpu\_load)

except:

Graph.gpu\_y.append(pyadl.ADLDevice.getCurrentUsage)

Graph.time\_x.append(n)

time.sleep(1)

print(Graph.gpu\_y)

def gpu\_util\_mean(self):

"""

finds the mean average of the gpu utilisaiton in the array of values

"""

length = len(Graph.gpu\_y)

product = 0

for x in range(0, length):

product += Graph.gpu\_y[x]

mean = product / length

rounded\_mean = round(mean, 3)

print('Average GPU utilisation =', rounded\_mean, '%')

# ram

def ram\_util\_timer(self):

"""

the percentage utilisation of the ram is found and append to the ram specific array in the graph script

100 seconds for each components testing to fill the 300 second benchmark time

"""

mem = virtual\_memory()

for \_ in range(100):

Graph.ram\_y.append(mem.percent)

time.sleep(1)

print(Graph.ram\_y)

def ram\_util\_mean(self):

"""

finds the mean average of the ram utilisaiton in the array of values

"""

length = len(Graph.ram\_y)

product = 0

for x in range(0, length):

product += Graph.ram\_y[x]

mean = product / length

rounded\_mean = round(mean, 3)

print('Average RAM utilisation =', rounded\_mean, '%')

# benchmark

def open\_heaven(self):

"""

opens the directory for the Uningine heaven benchmark software on the users device

the function supports both windows and macOS operating systems

"""

try:

os.startfile(

'C:/ProgramData/Microsoft/Windows/Start Menu/'

'Programs/Unigine/Heaven Benchmark 4.0/Heaven Benchmark 4.0.lnk')

except:

subprocess.call(

["/usr/bin/open", "-W", "-n", "-a", "/Applications/Heaven.app"])

# window

def output\_util\_graphs(self):

return Graph.util\_graphs

def create\_window(self):

self.next = PcWindow()

def create\_leader\_window(self):

self.next = ResWindow()

def create\_help\_window(self):

self.next = HelpWindow()

class HelpWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.FourthWindow()

def FourthWindow(self):

"""

displays a list of help items for how to manage the program

shows what settings to use for the benchmark

"""

self.resize(250, 250)

self.setWindowTitle('Help')

self.b = QPlainTextEdit(self)

self.b.insertPlainText('1. Use Recommended Benchmark settings\n2. Save Run Benchmark\n'

'3. Press F12 to Start\n'

'4. Save Results file when finished\n5. Open Results file when prompted\n'

'6. Input CPU if prompted\n7. Receive Recommended parts\n\n'

'Settings for Benchmark:\nAPI - DirectX11\nQuality - Ultra Tesselation\n'

'Extreme Resolution - 1920x1080')

self.b.move(0, 0)

self.b.resize(250, 250)

self.show()

class PcWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.Secondwindow()

def Secondwindow(self):

"""

three button and text box window for the user to find out the components inside their system

"""

self.resize(250, 250)

self.setWindowTitle('Hardware')

# cpu

self.cpu1btn = QPushButton('CPU Type', self)

self.cpu1btn.sizeHint()

self.cpu1btn.move(0,0)

self.cpu1btn.clicked.connect(self.cpu\_name)

self.cpu\_box = QTextEdit(self)

self.cpu\_box.move(120,0)

self.cpu\_box.setPlaceholderText('CPU')

# gpu

self.gpu1btn = QPushButton('GPU', self)

self.gpu1btn.sizeHint()

self.gpu1btn.move(0, 75)

self.gpu1btn.clicked.connect(self.gpu\_name)

self.gpu\_box = QTextEdit(self)

self.gpu\_box.setPlaceholderText('GPU')

self.gpu\_box.move(120, 75)

# ram

self.rambtn = QPushButton('RAM', self)

self.rambtn.sizeHint()

self.rambtn.move(0, 150)

self.rambtn.clicked.connect(self.ram\_find)

self.ram\_box = QTextEdit(self)

self.ram\_box.move(120, 150)

self.ram\_box.setPlaceholderText('RAM')

self.show()

# functions

def cpu\_name(self):

"""

uses the library cpuinfo for the model part of the users current CPU

"""

before = cpuinfo.get\_cpu\_info()['brand']

after = before.split(' ')

model = after[2].split('-')[1]

self.cpu\_box.insertPlainText(model)

return model

def gpu\_name(self):

"""

finds the model part of the users current GPU(s)

Uses either the GPUtil or pyadl library for NVIDIA or AMD support

"""

try:

try:

GPUs = GPUtil.getGPUs()

model = GPUs[0].name

self.gpu\_box.insertPlainText(model)

except:

model = str(pyadl.ADLManager.getInstance().getDevices()[0].adapterName)

self.gpu\_box.insertPlainText(model)

except:

self.gpu\_box.insertPlainText('no gpu found')

def ram\_find(self):

"""

uses the os library for finding the quantity of ram in the system

"""

mem = virtual\_memory()

lower = mem.total / 1073741824

gigByte = round(lower, 1)

output = str(gigByte)

units = ' GB'

self.ram\_box.insertPlainText(output)

self.ram\_box.insertPlainText(units)

class ResWindow(QMainWindow and QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.Thirdwindow()

def Thirdwindow(self):

"""

a GUI display of the result of the benchmarking and analysis process

two buttons and text boxes for the recommended items on both parts, either CPU or GPU recommendations

the URL for each upgrade is also given here

"""

self.resize(250, 450)

self.setWindowTitle('Results')

# cpu

self.cpubtn = QPushButton('GPUs', self)

self.cpubtn.sizeHint()

self.cpubtn.move(0, 0)

self.cpu\_rec = QTextEdit(self)

self.cpu\_rec.move(0, 30)

self.cpu\_rec.setPlaceholderText('recommended CPU(s)')

self.cpubtn.clicked.connect(self.cpu\_list)

# gpu

self.gpubtn = QPushButton('CPUs', self)

self.gpubtn.sizeHint()

self.gpubtn.move(0, 230)

self.gpu\_rec = QTextEdit(self)

self.gpu\_rec.move(0, 260)

self.gpu\_rec.setPlaceholderText('recommended GPU(s)')

self.gpubtn.clicked.connect(self.gpu\_list)

self.show()

def bubble\_sort(self, arr):

"""

a bubble sort function for sorting the lengths of the recommended components

"""

length = len(arr)

for i in range(length):

for n in range(0, length-i-1):

if arr[n] > arr[n+1] :

arr[n], arr[n+1] = arr[n+1], arr[n]

return arr

def cpu\_list(self):

"""

a function for outputting the recommended cpu models to the results table

called when the button widget is toggled in the thirdwindow object

uses a set of temporary arrays for arranging unsorted and then sorted data

the bubble sort function is used for arranging the lengths of the components

uses string modulation and data structure parsing for outputting correct data

outputs data to the cpu\_rec widget object in thirdwindow

"""

db = Database

length = len(db.recommend\_cpu)

values = []

sort\_part = []

temp = db.recommend\_cpu[:]

for i in range(length):

values.append(len(db.recommend\_cpu[i]))

array = ResWindow.bubble\_sort(self, values)

arrlength = len(array)

for i in range(arrlength):

for n in range(length):

try:

if array[i] == len(temp[n]):

sort\_part.append(temp[n])

temp.remove(temp[n])

except:

print('out of range')

for i in range(len(db.recommend\_cpu\_url)):

sort\_part.append(db.recommend\_cpu\_url[i])

for i in range(length):

sort\_part.append(db.recommend\_cpu[i])

print(sort\_part)

part\_string = 'Recommendations are: '

for i in range(len(sort\_part)):

part\_string = part\_string + str(sort\_part[i]) + '\n'

print(part\_string)

self.cpu\_rec.insertPlainText(part\_string)

def gpu\_list(self):

"""

a function for outputting the recommended gpu models to the results table

called when the button widget is toggled in the thirdwindow object

uses a set of temporary arrays for arranging unsorted and then sorted data

the bubble sort function is used for arranging the lengths of the components

uses string modulation and data structure parsing for outputting correct data

outputs data to the gpu\_rec widget object in thirdwindow

"""

db = Database

length = len(db.recommend\_gpu)

values = []

sort\_part = []

temp = db.recommend\_gpu[:]

for i in range(length):

values.append(len(db.recommend\_gpu[i]))

array = ResWindow.bubble\_sort(self, values)

arrlength = len(array)

for i in range(arrlength):

for n in range(length):

try:

if array[i] == len(temp[n]):

sort\_part.append(temp[n])

temp.remove(temp[n])

except:

print('out of range')

for i in range(len(db.recommend\_gpu\_url)):

sort\_part.append(db.recommend\_gpu\_url[i])

part\_string = 'Recommendations are: '

for i in range(len(sort\_part)):

part\_string = part\_string + str(sort\_part[i]) + '\n'

print(part\_string)

self.gpu\_rec.insertPlainText(part\_string)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

app.setStyle('Fusion')

execute = Window()

sys.exit(app.exec\_())

Graph Script:

import pyqtgraph as pg

cpu\_y = []

gpu\_y = []

ram\_y = []

time\_x = []

def util\_graphs():

"""

a graph GUI is created to illustrate the utilisation differences between the core components of the system

the data is from the functions in the GUI script which appended data to the four arrays.

the library pyqtgraph is used to model the data

different colours for each line make identification of key components easier

"""

try:

plt = pg.plot()

plt.addLegend()

plt.showGrid(x=True,y=True)

plt.setLabel('left', 'Utilisation', units='%')

plt.setLabel('bottom', 'Time', units='s')

plt.setWindowTitle('Utilisation %')

plt.plot(time\_x, cpu\_y, pen='b', symbol='x', symbolPen='b', symbolBrush=0.2, name='cpu')

plt.plot(time\_x, gpu\_y, pen='r', symbol='o', symbolPen='r', symbolBrush=0.2, name='gpu')

plt.plot(time\_x, ram\_y, pen='g', symbol='x', symbolPen='g', symbolBrush=0.2, name='ram')

except:

print('data type error')

if \_\_name\_\_ == '\_\_main\_\_':

import sys

if sys.flags.interactive != 1 or not hasattr(pg.QtCore, 'PYQT\_VERSION'):

pg.QtGui.QApplication.exec\_()

Database script:

import pymysql.cursors

from PyQt5.QtWidgets import \*

from bs4 import BeautifulSoup

import os

import hashlib

import GUI

import cryptography

results = []

recommend\_cpu = []

recommend\_cpu\_url = []

recommend\_gpu = []

recommend\_gpu\_url = []

class Database():

def \_\_init\_\_(self):

self.connection = pymysql.connect(host='localhost',

user='root',

password='password',

db='Hardware',

charset='utf8mb4',

cursorclass= pymysql.cursors.DictCursor)

def openFile(self):

"""

uses the QFileDialog widget for a file explorer GUI

the html file input is parsed using beautiful soup

"""

options = QFileDialog.Options()

options |= QFileDialog.DontUseNativeDialog

fileName, \_ = QFileDialog.getOpenFileName(None, "QFileDialog.getOpenFileName()", "",

"All Files (\*);;Python Files (\*.py)", options=options)

if fileName:

file = open(fileName)

data = file.read()

soup = BeautifulSoup(data, features="html.parser")

for item in soup.find\_all('strong'):

results.append(float(item.text))

print('Score =', results[1])

print('Fps =', results[0])

def registration(self, user1, password):

"""

an account is created using the username and password input from the login window GUI

an mySQL table stores the information of the account under one row and four differe columns

the username is pushed as plaintext to the table, the password as a hash

the hash function takes input of both a random salt and password plaintext for more security

UPDATE, SELECT and INSERT SQL statements are used for registering a user

the main window from the GUI script is called at the end of the process given it is successful

"""

salt = os.urandom(32)

pass\_key = hashlib.pbkdf2\_hmac('sha256', password.encode('utf-8'), salt, 100000)

key\_string = str(pass\_key)

try:

with self.connection.cursor() as cursor:

last\_row = cursor.execute("SELECT `UserID` FROM Hardware.Login")

new\_row = last\_row+1

update\_sql1 = "UPDATE Hardware.Login SET `Username` = %s WHERE `UserID` = %s"

data1 = (user1, new\_row)

update\_sql2 = "UPDATE Hardware.Login SET `PasswordHash` = %s WHERE `UserID` = %s"

data2 = (key\_string, new\_row)

salt\_push = "UPDATE Hardware.Login SET `PasswordSalt` = %s WHERE `UserID` = %s"

data3 = (salt, new\_row)

cursor.execute("SELECT `Username` FROM Hardware.Login")

rows = cursor.fetchall()

for i in range(0, last\_row-1):

value = rows[i]['Username']

if value == user1:

print('username taken')

return self.reject\_user()

else:

print('not taken')

cursor.execute("INSERT INTO Hardware.Login(UserID) VALUES(%s)", new\_row)

cursor.execute(update\_sql1, data1)

cursor.execute(update\_sql2, data2)

cursor.execute(salt\_push, data3)

self.connection.commit()

except pymysql.err.IntegrityError:

print('Wrong')

gui = GUI.Window()

return gui.mainWindow()

def login(self, user2, password):

"""

the login process is initiated using the username and password from the login window

the user table stores the password salt and hash which are used in ensuring the password is correct

the calculated password hash and stored passwordhash are compared to check correct credentials against the username

SELECT statements are used for data input from the mySQL table

the main window from the GUI script is called at the end of the process given it is successful

"""

try:

with self.connection.cursor() as cursor:

cursor.execute("SELECT `PasswordSalt` FROM Hardware.Login WHERE `Username` = %s", user2)

row = cursor.fetchone()

pullsalt = row['PasswordSalt']

pass\_key = hashlib.pbkdf2\_hmac('sha256', password.encode('utf-8'), pullsalt, 100000)

key\_string = str(pass\_key)

cursor.execute("SELECT `PasswordHash` FROM Hardware.Login WHERE `Username` = %s", user2)

rows2 = cursor.fetchone()

val = rows2['PasswordHash']

new\_val = str(val)

if key\_string == new\_val:

print('correct password')

else:

print('wrong password')

return self.reject\_user()

self.connection.commit()

except pymysql.err.IntegrityError:

print('Wrong')

gui = GUI.Window()

return gui.mainWindow()

def salt\_hash(self, plain\_word):

"""

an input of the plaintext password is used with a random calculated salt to generate a password hash

the hash is generated in a byte data type

"""

salt = os.urandom(32)

pass\_key = hashlib.pbkdf2\_hmac('sha256', plain\_word.encode('utf-8'), salt, 100000)

print(pass\_key)

return pass\_key

def reject\_user(self):

"""

if the credentials of a login process are incorrect, this function returns the user back to the login window to try again

"""

gui = GUI.Window()

gui.reject\_reg()

return gui.loginWindow()

def pop\_name(self):

"""

this function opens a text box for the user to manually input the name of their component if the model cannot be specified

"""

text, okPressed = QInputDialog.getText(self, "Component Name", "name:", QLineEdit.Normal, "")

if okPressed and text != '':

return text

elif okPressed and text == '':

self.pop\_name()

def getCpuDetails(self):

"""

this function uses the parts table to find compatable components to output to the user as upgrades

SELECT SQL statements are used to find the GPU items

the url for each upgrade is created using string parsing

the results for which are output to the arrays for cpu recommendations

"""

temp = []

temp2 = []

try:

with self.connection.cursor() as cursor:

cursor.execute("SELECT `GPU` FROM Hardware.Parts WHERE CPU = 6700")

rows3 = cursor.fetchall()

for i in range(0, len(rows3)):

found = rows3[i]['GPU']

temp.append(found)

for n in range(len(temp)):

val = temp[n]

new\_val = val[0:-1]

temp2.append(new\_val)

self.connection.commit()

except pymysql.err.IntegrityError:

print('Wrong')

recommend\_cpu = list(dict.fromkeys(temp2))

print(recommend\_cpu)

part1 = 'https://www.amazon.co.uk/s?k='

part2 = '&ref=nb\_sb\_noss\_2'

for i in range(0, len(recommend\_cpu)):

try:

split1 = recommend\_cpu[i].split(' ')

joined = split1[0] + split1[1]

url = part1 + joined + part2

except:

url = part1 + recommend\_cpu[i] + part2

recommend\_cpu\_url.append(url)

print(recommend\_cpu\_url)

return recommend\_cpu

def getGpuDetails(self):

"""

this function uses the parts table to find compatable components to output to the user as upgrades

SELECT SQL statements are used to find the CPU items

the url for each upgrade is created using string parsing

the results for which are output to the arrays for gpu recommendations

"""

temp = []

temp2 = []

try:

with self.connection.cursor() as cursor:

cursor.execute("SELECT `CPU` FROM Hardware.Parts WHERE GPU = 1080")

rows = cursor.fetchall()

for i in range(0, len(rows)):

found = rows[i]['CPU']

temp.append(found)

for n in range(len(temp)):

val = temp[n]

new\_val = val[0:-1]

temp2.append(new\_val)

self.connection.commit()

except pymysql.err.IntegrityError:

print('Wrong')

recommend\_gpu = list(dict.fromkeys(temp2))

print(recommend\_gpu)

part1 = 'https://www.amazon.co.uk/s?k='

part2 = '&ref=nb\_sb\_noss\_2'

for i in range(0, len(recommend\_gpu)):

split1 = recommend\_gpu[i].split(' ')

joined = split1[0] + split1[1]

url = part1 + joined + part2

print(url)

return recommend\_gpu

# Testing

Different functional tests for features of the program

**Testing the registration feature**

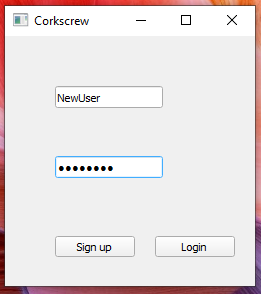
Expected results

1

* UserID incremented by 1 from previous row
* new username and hashed password pushed to the login table under respective columns
* Salt used pushed to column under same UserID
* Salt pushed in binary form

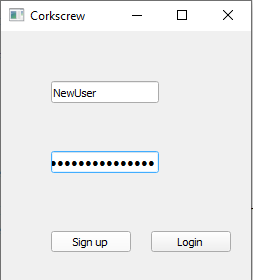
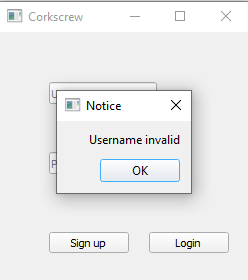
2

* If new user being registered has username already present in table, reject new user
* Bring up username invalid window
* Open a new login window

Result 1



**True**

Result 2

**True**

**Testing login feature**

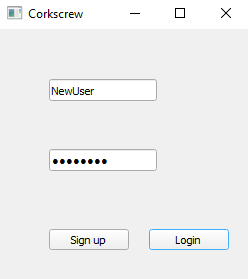
Expected results

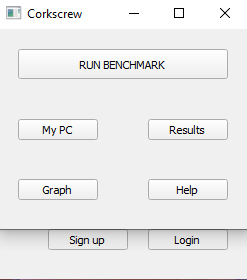
1

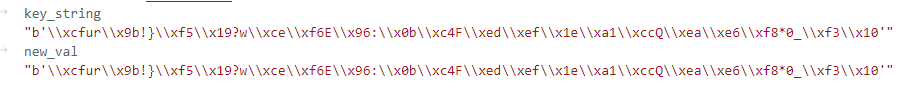
* If username and correct plaintext password are correct, open the main menu window
* The password hash value in the database table is the same as the new calculated hash

2

* Correct username and wrong password input, outputs error message
* Prompts new login
* Calculated hash and database hash are different

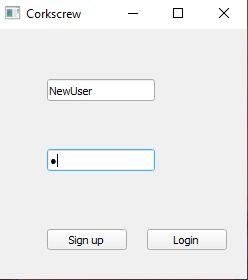
Result 1



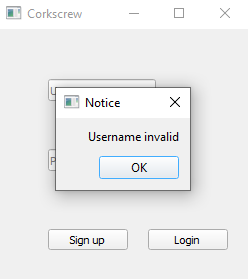


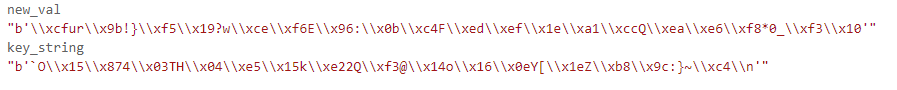


**True**



Result 2





**True**

**Testing Hardware feature**

Expected results

1

* CPU window outputs the model name of the users currently installed CPU

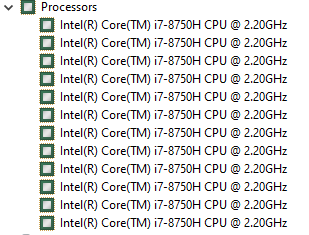
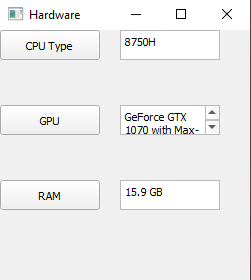
2

* GPU window outputs the model name of the users currently installed GPU

3

* RAM window outputs the quantity of the users currently installed RAM

Result 1



Result 2



Result 3



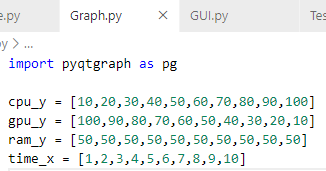
**True**

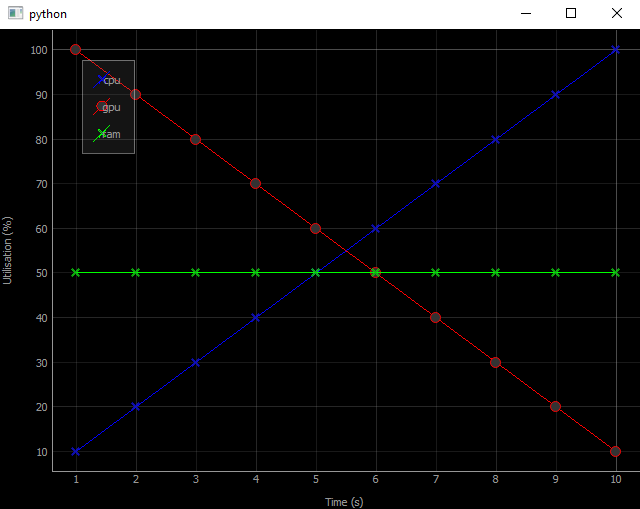
**Testing graph feature**

Expected results

* The values stored in each component array in the graph script will be output to the graph GUI

Result



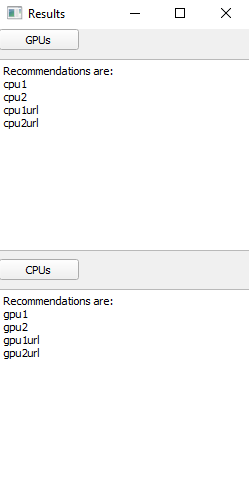


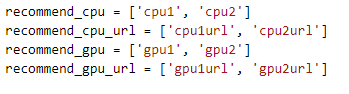
**True**

**Testing results feature**

Expected results

* The values stored in the recommended component array and URL array will be output to the respective text window

Result



**True**

# Evaluation

GitHub Code repository:

<https://github.com/SebastianDixon/Corkscrew>

A video of how the program operates:

<https://www.youtube.com/watch?v=MIeHNUJ6mYQ>

## How were the objectives met and achieved?

The majority of the objectives set out in the design have been fulfilled. One point was further analysed and then omitted from the final product. This was to omit the recommendation of a RAM upgrade, given the restrictions specified. One object wasn’t fulfilled in the final product, this was to include a link to the upgrade process for the component needing and upgrade. This was omitted since it seemed to be an unnecessary step to include, given each upgrade scenario is so different for each user, very different conditions for each user, would mean issuing one upgrade video would produce an incorrect recommendation to a user, and would introduce a reliance on an internet connection for the program to operate.

## How can your proposed solution be improved?

One section of program could have been further improved upon, although fully implemented. This was training the buffer for discerning a CPU or GPU bottleneck. A huge amount of data on bottlenecked and non-bottlenecked systems would be required for producing a completely accurate judgement.

The current system produces a moderately accurate assessment, however there are only a few hundred examples for the program to utilise explaining its current accuracy.

## Analysis on feedback from the end user/supervisor

Supervisor comments:

“Having just given your writeup a quick look over myself, I could suggest that your user interface is functional, power-user focused, and bordering on clunky. Your graph is stunning and delivers what could be expected but in an amazing way. That brings the project to life. You might want to look into some way of logging what happened each test so that the program could comment on any positive or negative spikes.”

End user comments:

“I really think the functionality is good but the user interface doesn’t make it easy for me to know how to start and which order I should be executing different parts of the program”