

**The University of the West Indies**

# **Department of Computing and Information Technology**

INFO 3604 – Undergraduate Final Project

**Final Project Report**

Project Name: Budget Builder

Group Name: The PC BoyZ

Aaron Meighoo - 816000303

Kyle Fonrose - 816002546

Jaleel Mohammed - 816001154

Table of Contents

[**Department of Computing and Information Technology** 1](#_Toc40671245)

[Abstract 3](#_Toc40671246)

[**Introduction** 4](#_Toc40671247)

[**Problem Description** 4](#_Toc40671248)

[**Risk Assessment and Management** 5](#_Toc40671249)

[**Positioning** 7](#_Toc40671250)

[**Stakeholder Descriptions** 7](#_Toc40671251)

[**Requirements Specification** 9](#_Toc40671252)

[**System Diagrams** 19](#_Toc40671253)

[**Technical Constraints** 23](#_Toc40671254)

[**Operating Requirements** 23](#_Toc40671255)

[**Safety** 41](#_Toc40671256)

[**User friendliness** 41](#_Toc40671257)

[**Performance** 41](#_Toc40671258)

[**Convenience** 41](#_Toc40671259)

[**Legal issues** 41](#_Toc40671260)

[**Reliability** 41](#_Toc40671261)

[**Availability** 41](#_Toc40671262)

[**Maintainability** 42](#_Toc40671263)

[**Technology** 42](#_Toc40671264)

[**Design Methods** 42](#_Toc40671265)

[Individual contributions 43](#_Toc40671266)

[Survery 45](#_Toc40671267)

[Responses 53](#_Toc40671268)

**Conclusions, Lessons learned and Recommendations…………………………………………………66**

[BudgetBuilder Final Presentation Video Link 69](#_Toc40671269)

[Hosting Via Heroku Link 69](#_Toc40671270)

# Abstract

Budget Builder is a web application which is targeted to those who are interested in purchasing or building a computer system. When deciding on a new system the choices are limitless and can become very confusing with the varying parts and specifications.

Budget Builder is an optimization app which helps eliminate all the guesswork and doubts by providing users with the best possible options based on a user specified budget and purpose of the computer system. The app will provide the user with all the links to purchase the recommended products. Budget Builder would be affiliated with a few parts suppliers in the future, which we will earn a commission from when the application when user make a purchase from the provided links.

The overall objective of the application is to save users time by providing them with all the necessary information in one place and ensuring them that the parts selected are the best combination for the given budget.

This project report highlights the development of the application over a 3-month period. An overall breakdown with Methodologies taken, Designs and Functions implemented, Business Aspects, Data collected and Analysis Performed, Project Costing and Future Projections are documented in this report.

# **Introduction**

# **Problem Description**

When building a custom computer system there are many varying parts and specifications to choose from and without knowledge of these components, there may be compatibility issues or poor choices made when putting together a custom computer system.

This application is an optimizer which requires the user to enter some information which it then processes and produces the best results based on the user’s input. The system ensures that there are no compatibility issues and that the best prices from reputable retailers are provided for the recommended parts.

The application is also expected to generate revenue via multiple streams such as premium subscriptions, affiliations with suppliers and from paid advertisements.

# **Risk Assessment and Management**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk** | **Risk Type** | **Probability** | **Effects** | **Affects** | **Strategies** |
| Staff Turnover (Illness, Firing, Relocation) | Project | Moderate | Serious | Project | ***Contingency*** –  Ensure each staff member is trained and all work documented so that in the absence of a team member, another or new member can pick up from where left off. |
| The time required to develop the software is underestimated. | Project | Moderate | Tolerable | Project | ***Avoidance*** - Properly define the scope of the problems. Schedule tasks and goals to be completed and due dates. |
| System failures during the development process | Project and Product | Moderate | Serious | Project and Product | ***Minimization*** - Ensure backups are made periodically to minimize loss in the event of system failures |
| Software Unavailability | Project and Product | Moderate | Catastrophic | Project and Product | ***Contingency –***  In case of a particular software not being available, Alternate solutions should be developed and implemented as seen fit. |
| The database used is unable to process transactions as rapidly as is needed | Product | Moderate | Serious | Product and Business | ***Avoidance*** – Investigate possibility of buying a higher performance database |

**Table 1 showing:** Table of Possible Risk and Strategies

# **Positioning**

Our target audience range from the average person who is interested in the acquisition of a personal computer, tech enthusiast, IT Staff and persons who are looking for a purpose-oriented computer system. This application will help save persons time and money by removing the guesswork and making all the required products available on one platform. There are a very few numbers of applications that perform similar tasks that Budget Builder offers. What makes Budget Builder different is that the system suggests all the components needed for a particular computer system given the customer's budget and ensures that they are compatible with each other and also provides the link for the purchasing of the components. As opposed to our rivals that have you choose the components and hope that it falls within your budget and also that they work together, these are just some of the advantages that our application has over the others.

# **Stakeholder Descriptions**

**Application User**

Any individual who is interested in the purchasing or building of a new computer system. This can also include someone who is a PC enthusiast or IT professional who may find interest in new computer technology and current products on the market. Basic users are not required to pay any fees to register to the system or do not necessarily have to make a purchase via the application.

**Product Overview**

The application would be made available on the web and would be supported by most modern internet browsers. Users would be required to register and login to their account in order to use the application and have their data stored for later retrieval. Users will then be presented with a screen which they will be required to fill out the necessary information such as the purpose of the system and the budget. After submitting the information the system will generate either or both a list of components for those who wish to purchase the parts individually and assemble the computer system themselves, or they will be given a list of pre-built systems with the recommended components within the specified budget and ensuring it will be able to perform that assumed tasks of the purpose selected.

# **Requirements Specification**

**Analysis of Requirements**

Requirements for our system were identified based on our personal experience as computer enthusiasts facing problems when we were building our personal computer systems. Research and observations were done on the problems and similar solutions were examined. Combining our knowledge gained at the UWI in our respective fields of study and collection of data through surveys and interviews we developed the system requirements, specifications and designs.

**System Requirements**

1. The application should be supported by any web browser.
2. The application should allow users to register an account.
3. The application should allow users to login to an account.
4. The application should allow users to input required information.
5. The application should allow users to browse items.
6. The application should allow users to save their build list.
7. The application should allow users to edit their information.
8. The application should allow users to edit saved build lists**.**

**Functional Requirements**

1. The system should ask the user to sign up for an account.
2. The system must ask the user to log in if they already have an account.
3. The system should allow the user to search for products.
4. The system should allow the user to start a build list.
5. The system should recommend to the customer builds that are similar to the user’s list.
6. The recommended lists should be both less than the total and more than the total cost of the user’s list.
7. The system should allow users to compare products.
8. The system should provide the user with ample information for them to make a decision.
9. The system should allow the user to have multiple build lists.
10. The system should have a layout such that the user can follow along to it easily.
11. The user should have an account to use the system
12. The system should allow the user to choose parts that are compatible with other parts.

**Non-functional Requirements**

**Usability:**

The system will be user friendly with simple and self-explanatory interfaces

**Reliability:**

The system should be available and accessible to all users 24/7 provided a stable internet connection is established.

**Performance:**

The system will be used by multiple users, simultaneously and will be able to handle all the traffic without reducing its performance and functionality. Credentials validations, the loading of the system interfaces and data processing shall be done in a suitable time frame.

**Security:**

The system allows users to log in to their account using their email and password. The data collected and stored by the system will be encrypted in order to maintain security and confidentiality and to ensure consumer information is protected during online payments.

**Authentication:**

The system shall allow only authorized users to access their accounts.

**User Stories**

1. Alice wants to build her own personal computer. She heard about the web application called Budget Builder. She decides to make an account and try the application. She makes the account and successfully logs in. Alice is greeted with a homepage that asks her to either start a new build, continue a build or browse the products. She decides to browse the product options. By clicking the “browse products” tab, she is presented with a dropdown menu that shows her different categories of PC parts that she can choose from e.g “memory”, “CPU”. By clicking “Memory” she is then presented with a page that shows a list of all the memory brands, speeds and the capacities of the memory. She can then either “add” the specified memories to a list of her products or simply view how the product looks by clicking the picture of the memory.
2. Bob works for a company that needs a PC built that will maximize performance and minimize the cost of building the PC. He decides to use the budget builder app to make the pc build in software first. He logs into the app and chooses “New Build”, he is then met with a layout page that specifies all the necessary computer parts with the selections empty, just to be filled in. He clicks on “add” on the motherboard section and he adds a motherboard to the list. He then clicks on “CPU” and then chooses the appropriate CPU that is compatible with his motherboard. Bob then completes his build and the system gives him an accurate total of all the parts’ cost so he can know and would be able to show his company proof of the total of the build option he chooses.
3. Candice built a PC using the application but she does not like the total cost she received. She wants to reduce the cost of the build that she has. She decides to choose “alter build” in the build section. Her saved build is shown on the screen, she does not want to do a complete build. There is an option that says “Similar builds” where she can click and the site will take her to a page that shows her a list of builds similar to hers, but the price range is either less than her build or more than her build. The site recommends to her a build that  is very similar but the “GPU” is less expensive than the choice she had originally chosen. She decides to choose this build instead and clicks on the “save spec list” option for that specific build.
4. Darren wants to compare two CPU’s since he is not familiar with the architecture of both. He opens the app and chooses “compare parts”, he then chooses the option to choose which category of components he wants to compare. He selects the two CPUs and clicks compare. This gives him a list of details showing the comparison between the two CPUs, showing details such as pricing, TDP, cores etc.
5. Riad is unfamiliar with computer components and unsure of the parts he may need to make a build. He navigates to the “Generate Build” section and then chooses what he wants his computer to do. He is then presented with a list of different builds at different price ranges and from there he can select something to suit his price range. If he so chooses he can select any build that he shows interest towards and get a listing of the parts required with the cost of each part listed.
6. Paul has a list of parts that he would like to obtain however he is unable to locate any in his general area so he opts to purchase them online. He compiles his build on the builder and is then given a few credible sources that he can then choose from. He can click a link to take him to his desired part, such as “CPU” on the desired website eg. “Amazon”. After making a purchase he can then return to the builder and get sources for his other parts to complete his build.
7. Tom is a developer and maintains the website and its components. When an issue arises he must make every effort to resolve it and because of this he is able to log into the system as an admin. The system will check his credentials and once confirmed he is granted access to the system with admin privileges.

**Use Cases**

**ID:** UC1

**Name:** Login

**Actors:** User, Administrator.

**Preconditions:** Must have an existing account.

**Description:** An Actor is able to log into their accounts where they can find saved information and access certain features of the website.

**Postconditions:** Actor is logged on.

**ID:** UC2

**Name:** Create New Account

**Actors:** Websurfers

**Preconditions:** Must not have an existing account.

**Description:** A new/potential visitor to the system can choose to create an account to access more features of the system.

**Postconditions:** Visitor now has a n account on the system.

**ID:** UC3

**Name:** Start New Build

**Actors:** Users

**Preconditions:** Must have an existing account.

**Description:** A user is able to start building a new PC system by selecting this function after logging in.

**Postconditions:** Persons with access to accounts can now start a new build for their pc using

the system. People with accounts can save their build and resume or review it at a later time.

**ID:** UC4.0

**Name:** Save Build (extended)

**Actors:** Users

**Preconditions:** Must have an existing account.

**Description:** A user is able to save a build that they have compiled or is compiling.

**Postconditions:** After selecting “save build” the user’s account is cleared of all the parts that they previously compiled.

**ID:** UC4.1

**Name:** Edit Build (extended)

**Actors:** Users

**Preconditions:** Must have an existing account, Must have an existing build.

**Description:** A user is able to edit a build that they were previously creating. Once they log onto the website they can continue a build that they were doing at an earlier time once it was saved.

**Postconditions:** After selecting this option an existing user will be presented with the compilation of parts they had previously.

**ID:** UC4.2 (extended)

**Name:** Delete Build

**Actors:** Users

**Preconditions:** Must have an existing account.

**Description:** A user is able to delete a build that they were compiling previously after reviewing it.

**Postconditions:** After selecting “delete build” the user’s account is cleared of all the parts that they previously compiled.

**ID:** UC5

**Name:** Search Item

**Actor:** User

**Description:** The user wants to find a product on the website. They click the search button and type in the product that they desire.

**Precondition:** The user wants to search for an item. the site contains such item.

**Postcondition:** the site is used to search the product

User clicks on the product

User reads the description of the product

User adds the item to their list

Product is found and used appropriately.

**ID:** UC6.0

**Name:** BrowseCategories.

**Actors:** Users, Visitors

**Preconditions:** Actors must be on the website.

**Description:** If Actor is just curious about the components(parts) of a PC (desktop) they can browse through the categories of parts like “Memory” or “CPU”.

**Postconditions:** Actors will most times than not find parts to their liking in a selected category.

**ID:** UC6.1

**Name:** Compare Items.

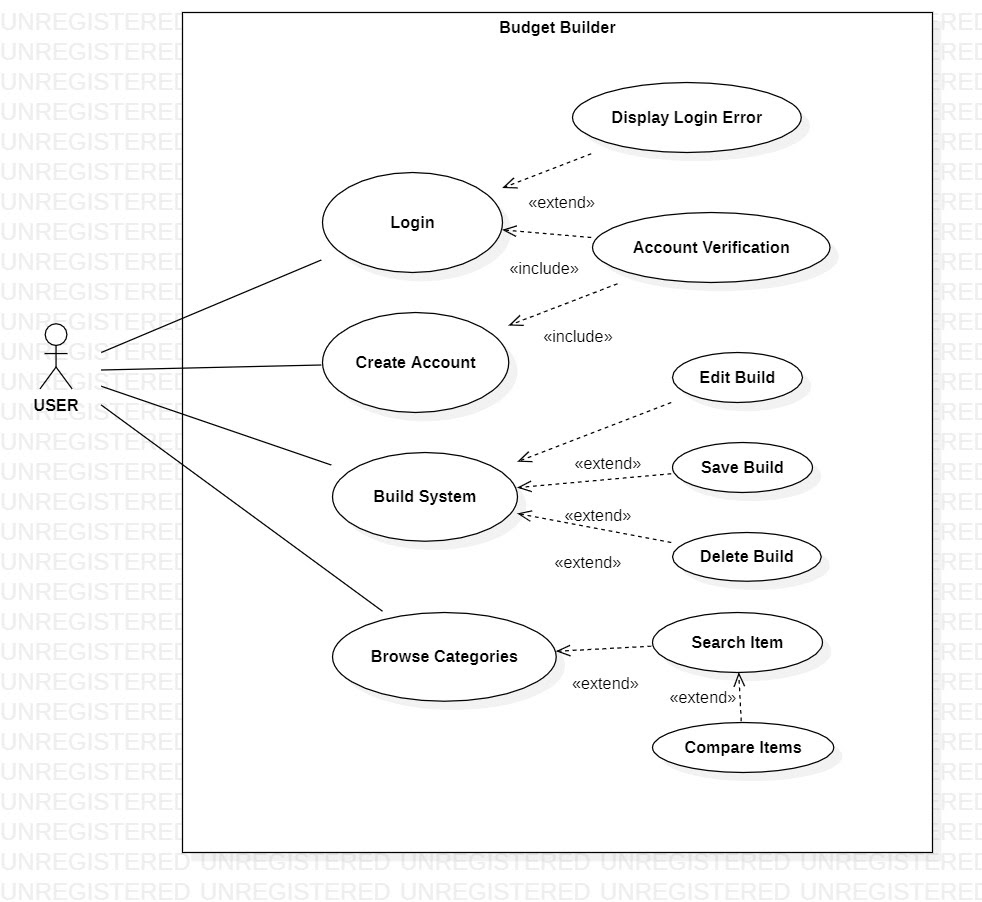
**Actors:** Users, Visitors

**Preconditions:** Actors must be on the website.

**Description:** If an Actor wants to compare components(parts) of a PC (desktop) they can browse through the categories of parts and compare different brands and specifications.

**Postconditions:** Actors will compare the price and specifications of different parts when making a choice.

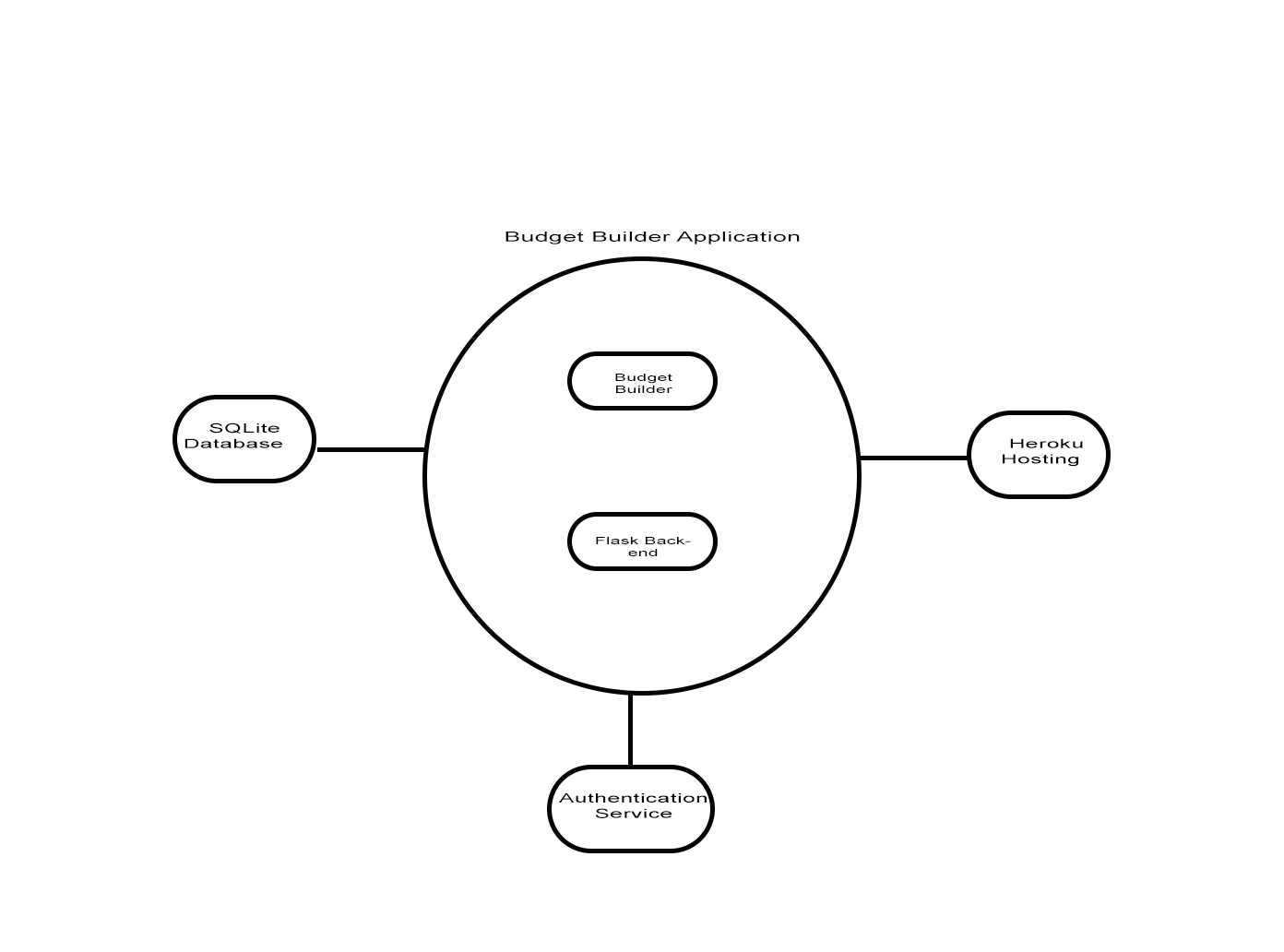
**USE CASE DIAGRAM**



**Figure 4:** Use Case Diagram for Budget Builder

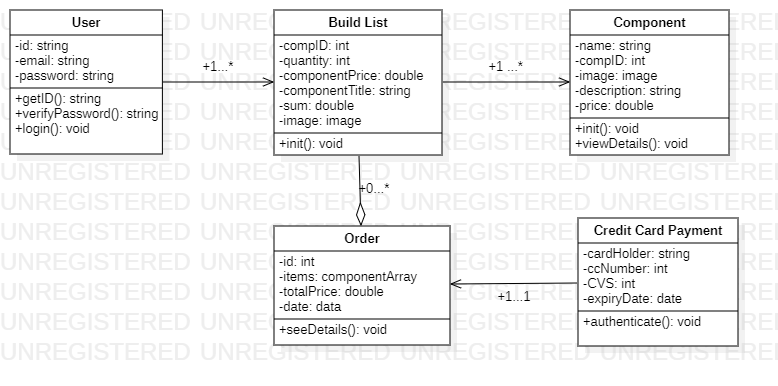
# **System Diagrams**

System Diagram



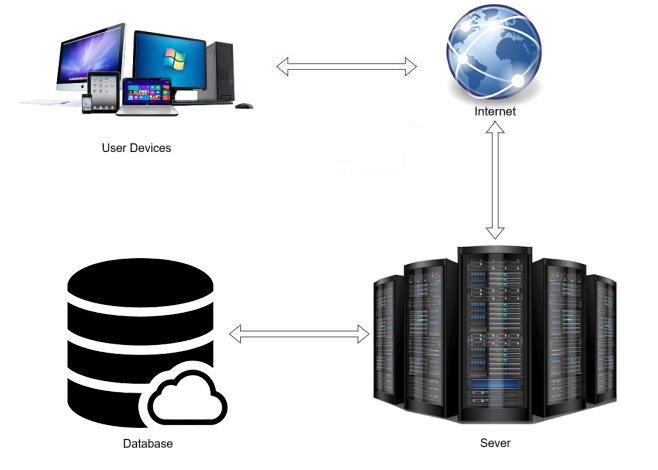
**Figure 2:** System Diagram for Budget Builder

**System Class Diagram**

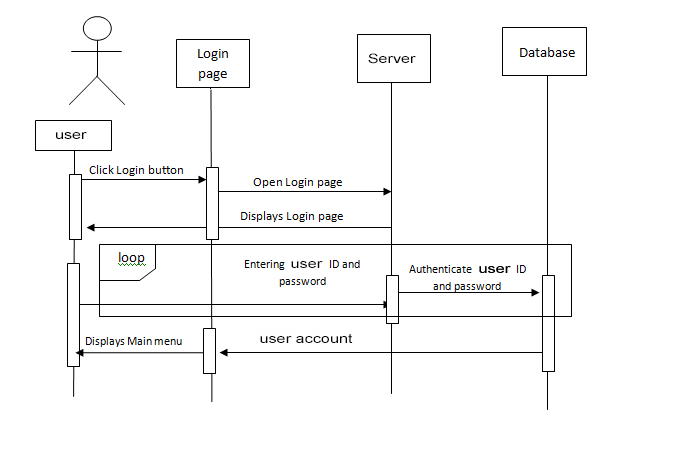


**Figure 5:** System Class Diagram for Budget Builder

**Logic Diagram**



**Authentication Sequence Diagrams**

****

# **Technical Constraints**

For Budget Builder, technical constraints include all aspects that affected the overall development and time management of the project. Most of the constraints was due to lack of experience and lack of data of the project topic.

Programming Language

Python

Framework

Flask

The framework used for the application was Flask, which is a mico web framework written in Python It is classified as a microframework because it does not require particular tools or libraries. ... However, **Flask** supports extensions that can add application features as if they were implemented in **Flask** itself.

# **Operating Requirements**

In order for BudgetBuilder to work as seamlessly as possible, several factors need to be adhered to by its users:

* The user must own or have access to a smartphone/PC/Laptop
* The user’s smartphone must have WiFi or Internet access
* The user must have approximately 1GB of storage space on their mobile device

**Discussion of Alternative Designs**

There were no other major alternative design plans as we think that we have achieved a very simple and straight forward navigation throughout the app interfaces. However, we considered adding another interface geared toward enthusiasts, which will give them more options to choose from and to specify when doing custom computer builds. Those users generally have a better understanding of component specifications and may be familiar with certain brands, thereby giving them options to narrow down their choices via the generation of build list.

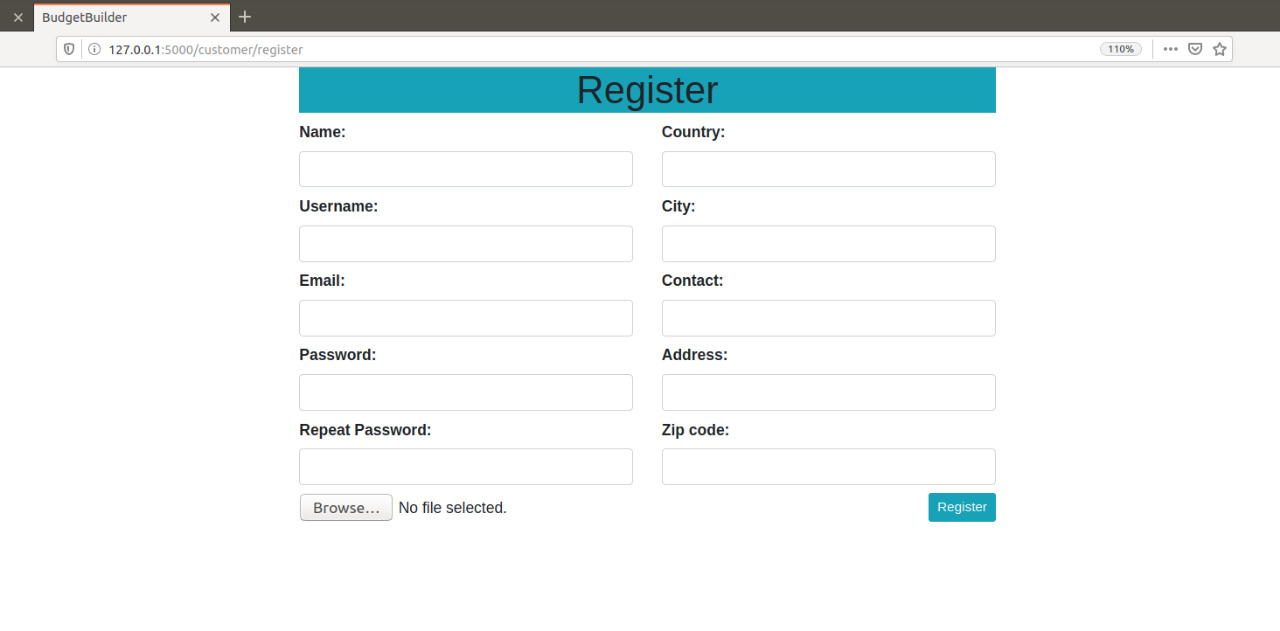
However, one option that was a possibility for the webapp was the use of a dropdown menu in the navbar. This was the default view that was created to be used by the users by the developer Jaleel Mohammed, but as it turned out, the bootstrap model that was bring used was such that there was a bug in the bootstrap code itself and it could not be fixed.

This bug was known to the developers of the bootstrap and multiple sources online stated the same error without fix.

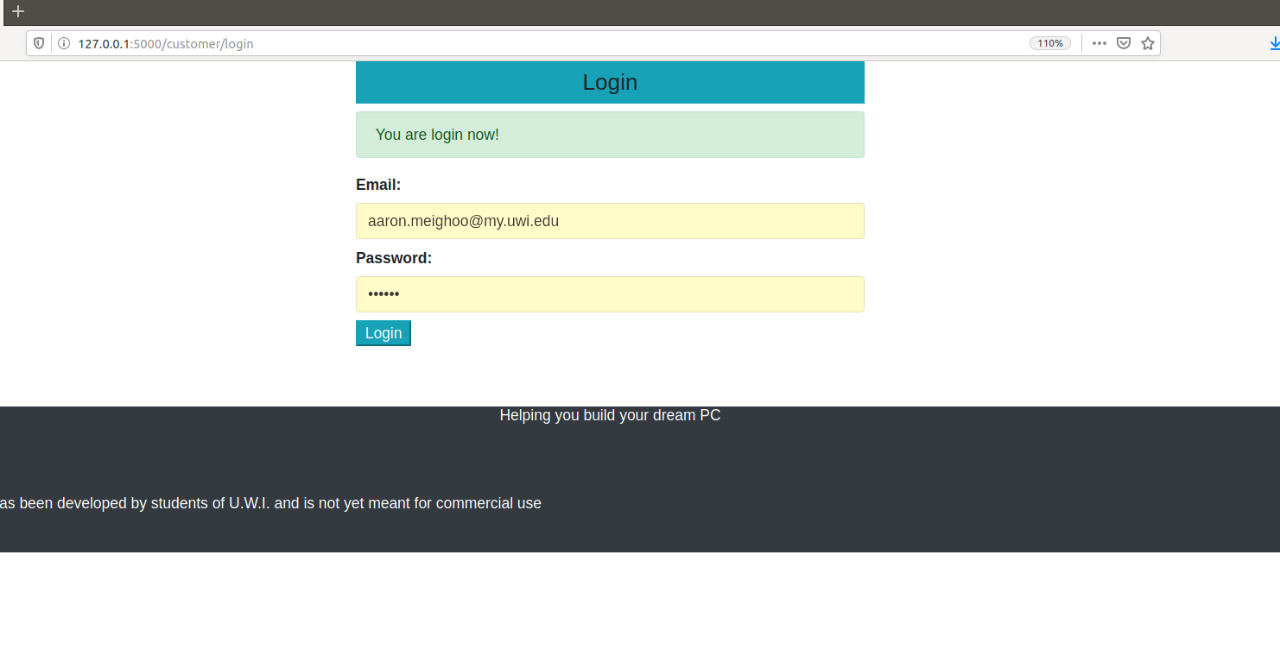
Another alternative design that was taken into consideration was the use of a budget in order to generate a PC based on this price that was entered, this was not achieved as mentioned in our presentation and is designated for future work.

**User Interface Design**

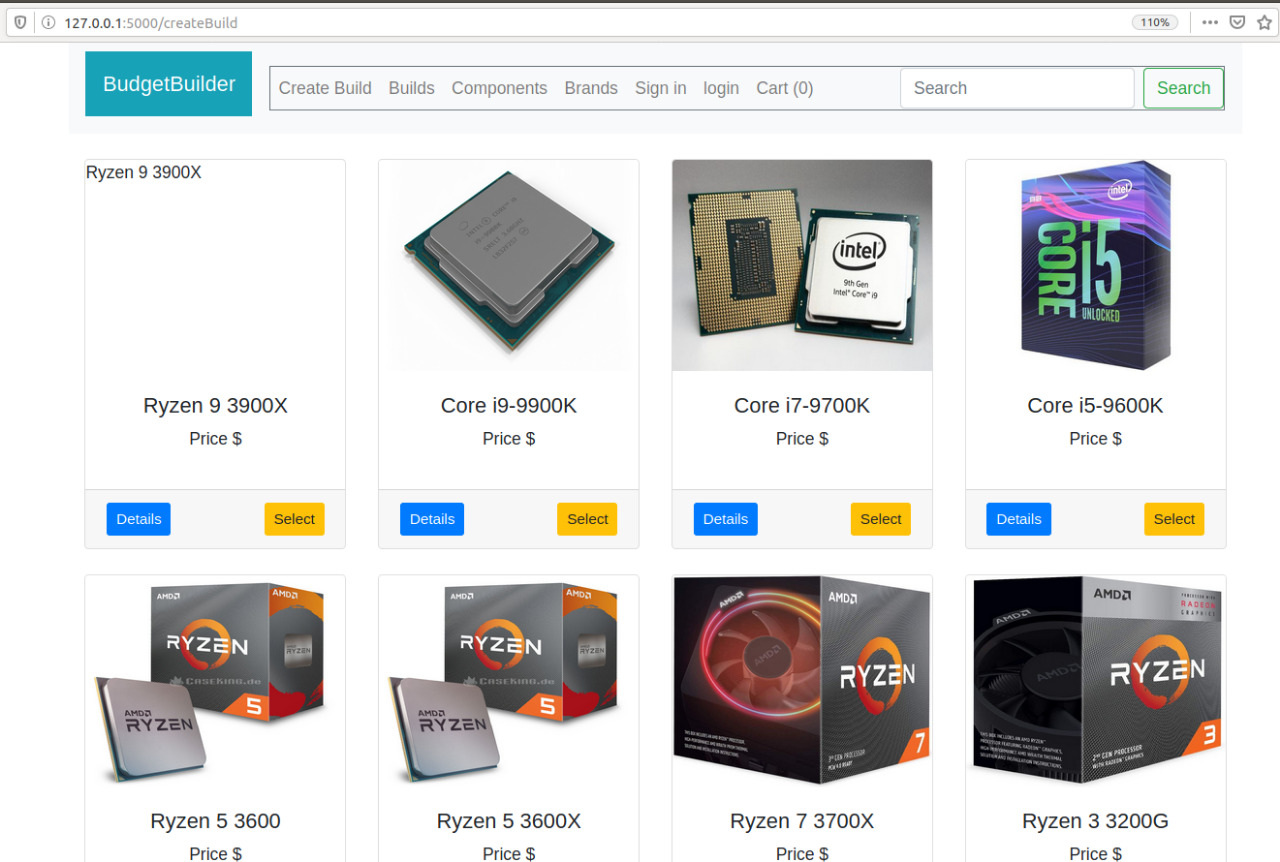
**Registration**

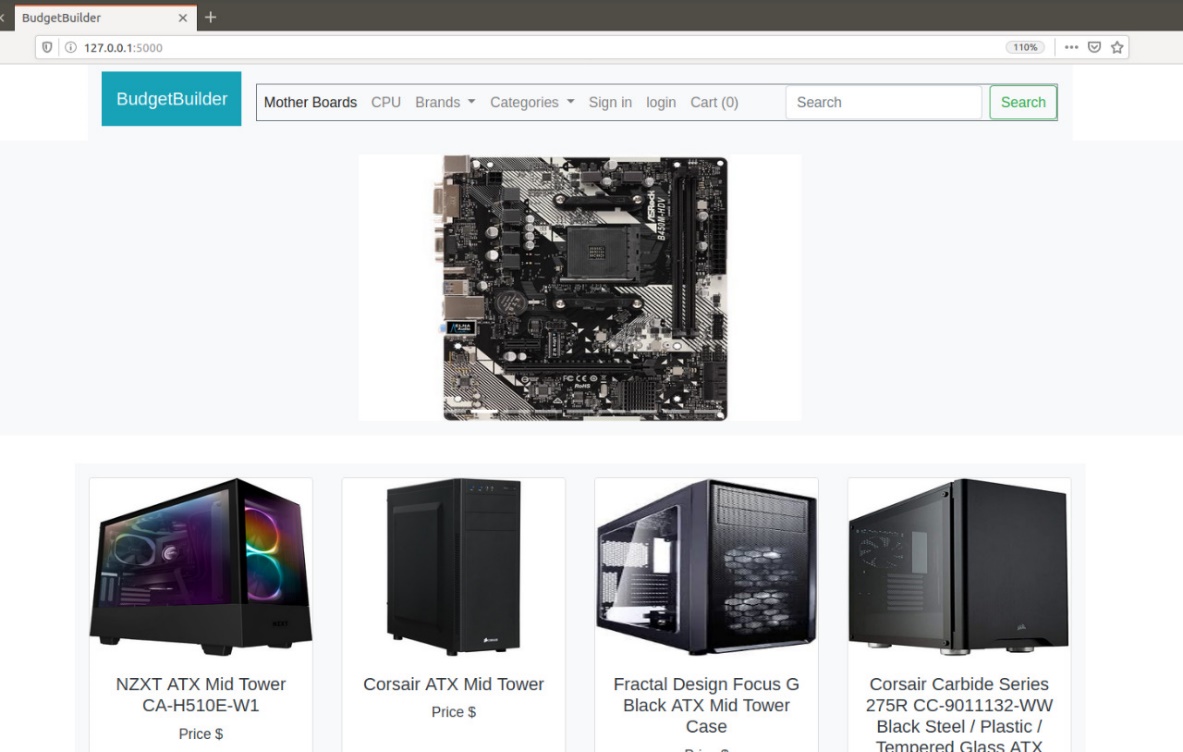
****

**Login**

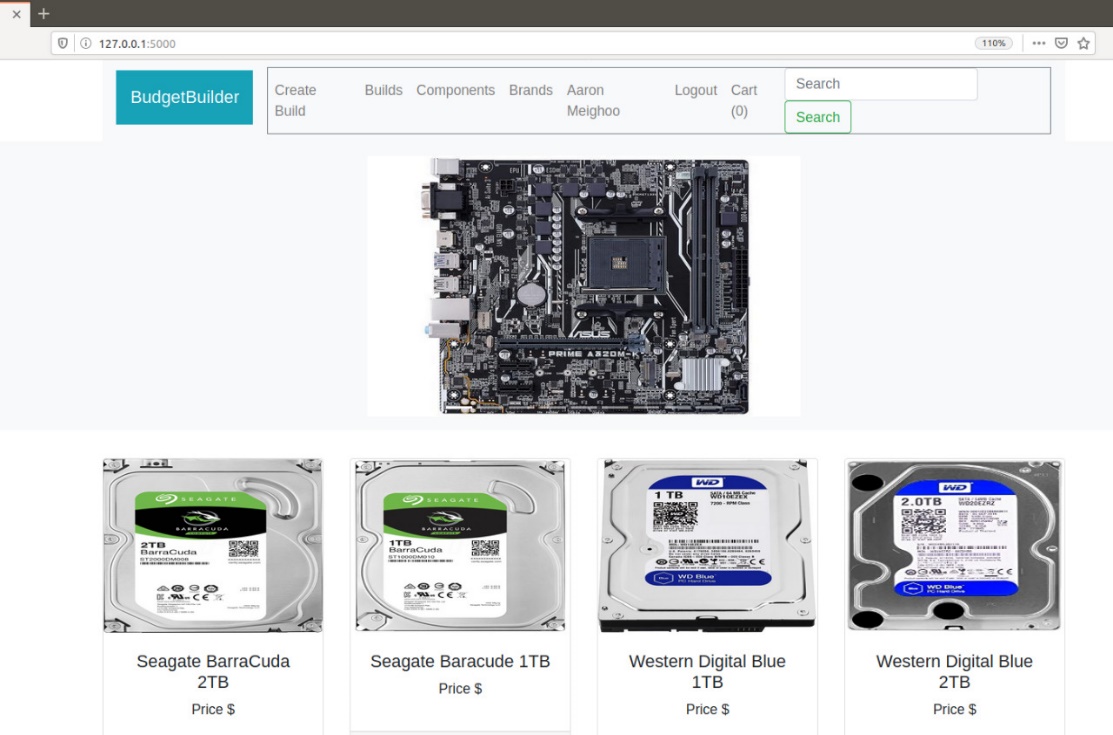
****

**Home Interface**

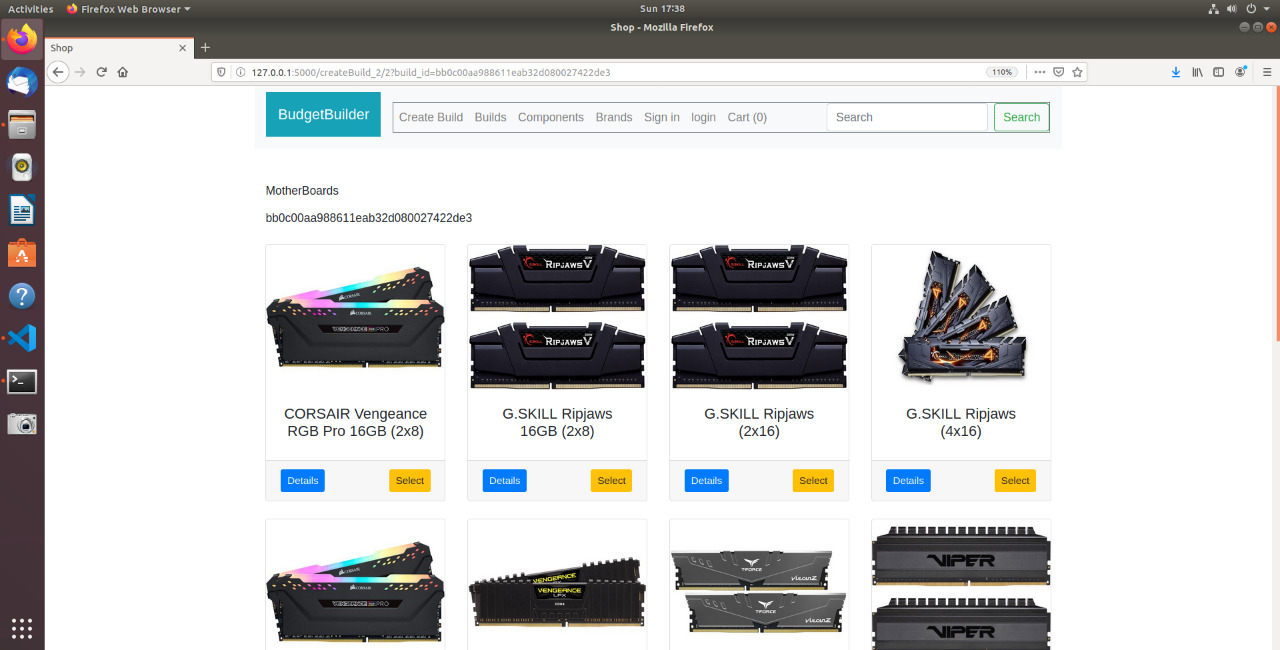


**Motherboard tab**

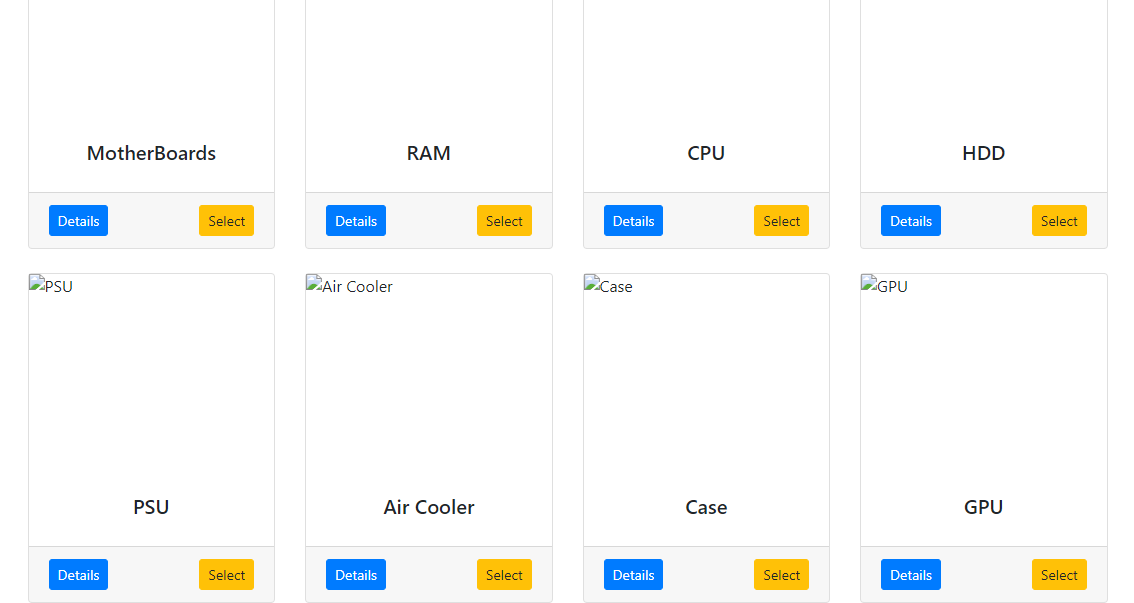
**HDD Tab**

****

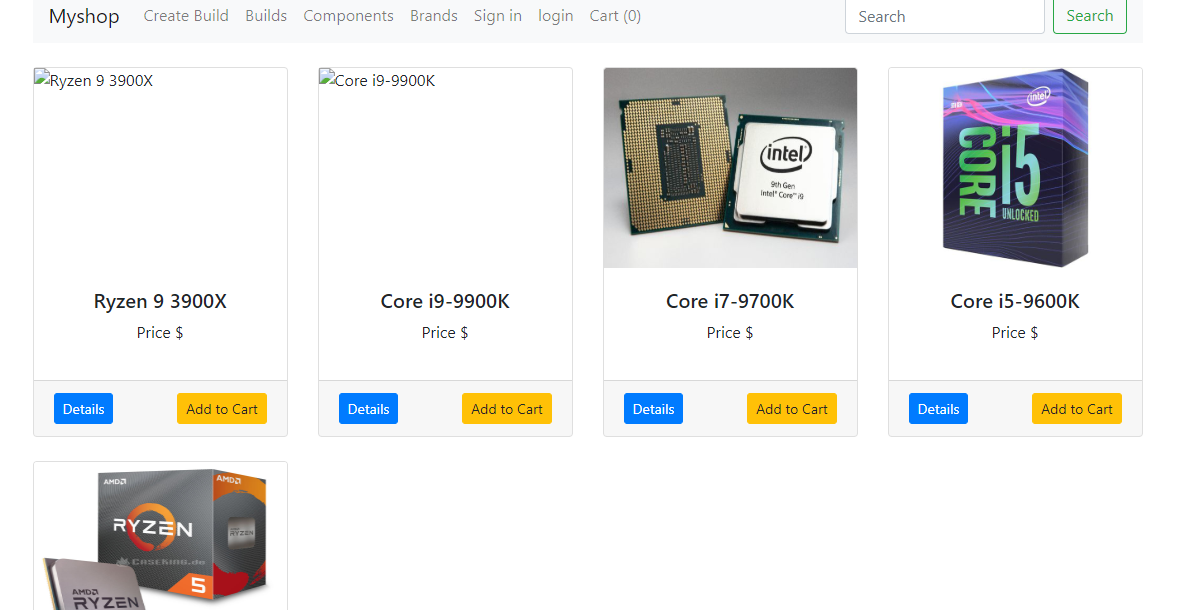
**R.A.M Tab**

****

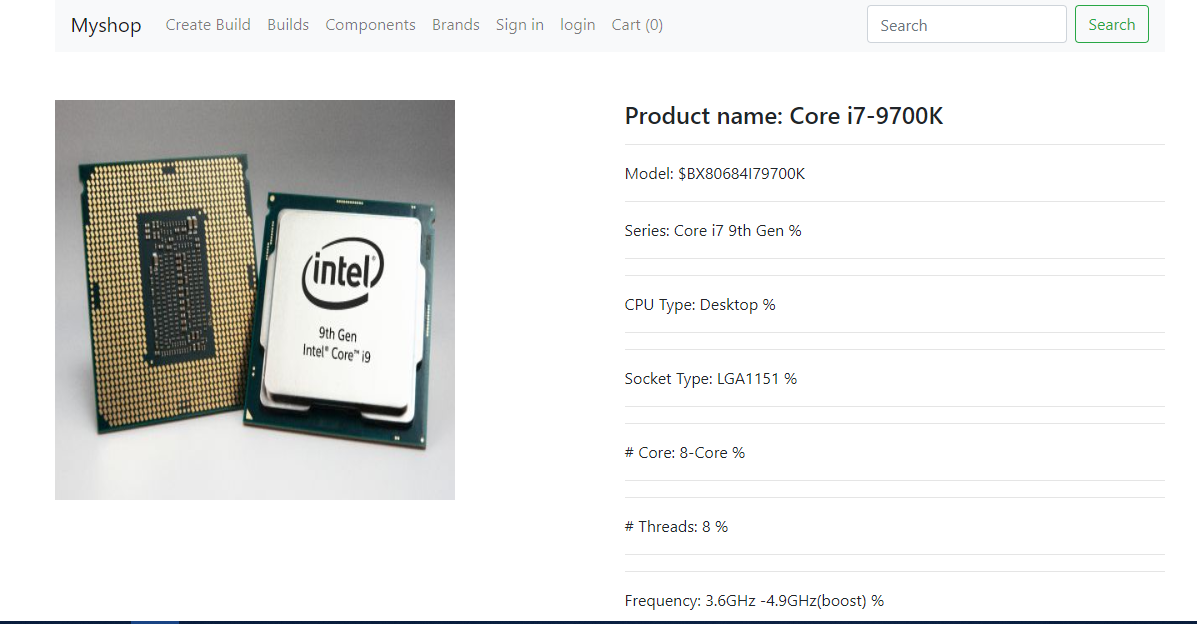
**Components**

****

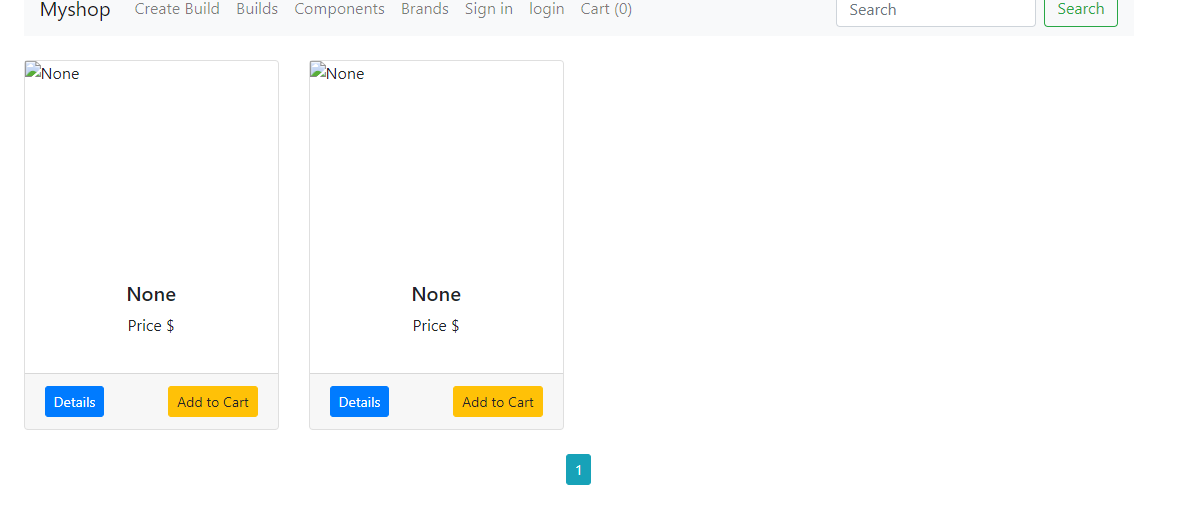
**CPU Page in components (1 of many)**

****

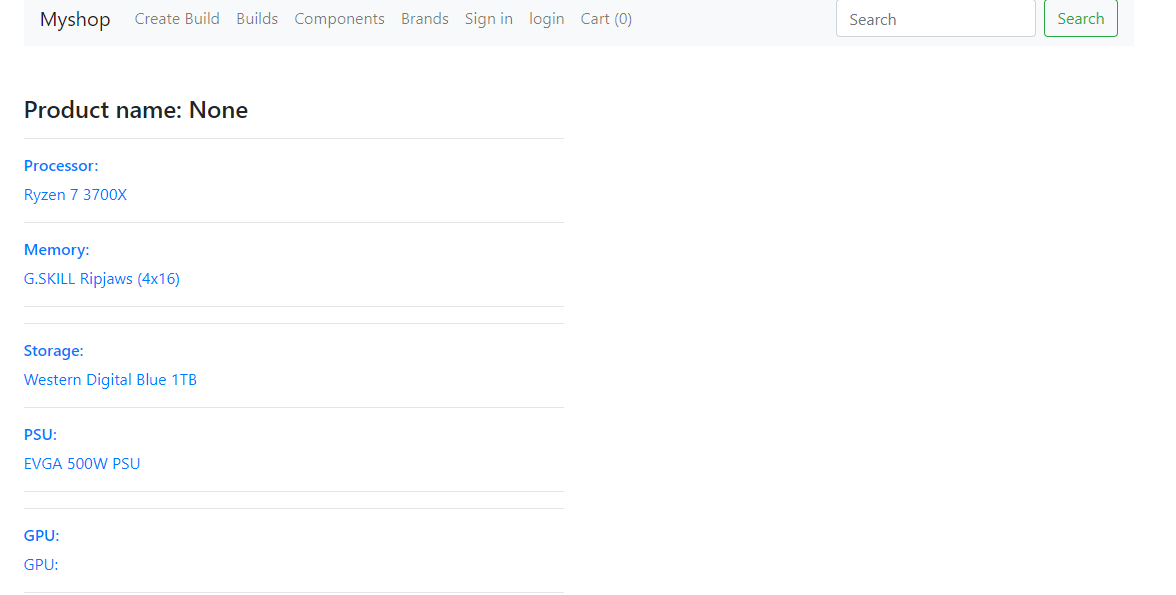
**Details page of singular CPU product**

****

**Selection of Pre-Built systems**

****

**Details of Pre-Built Systems: each item is clickable**

****

**Implementation**

**Methodology, Technologies, Languages and Libraries**

**Agile Methodology**

The agile methodology was used for the project development and management of the project. This means that continuous iterations of development and testing was done concurrently throughout the project development lifecycle. Each group member worked together and switched roles throughout the development where our different strengths were utilized.

**Meetings**

The group met at least once weekly to have face to face discussions, while all other communications were done over the web via WhatsApp and resources were shared using Google Docs and Google Drive. Weekly status reports were also documented to track progress and to schedule tasks for each member of the group.

Programming Languages:

* Python
* HTML
* JavaScript
* CSS

Technologies:

* Flask Framework
* GitHub
* SQLite
* Visual Studio Code
* Heroku (hosting)

Databases:

* SQLite

**Libraries:**

* alembic==1.4.2
* appdirs==1.4.3
* astroid==2.3.3
* bcrypt==3.1.7
* certifi==2019.11.28
* cffi==1.14.0
* Click==7.0
* distlib==0.3.0
* filelock==3.0.12
* Flask==1.1.1
* Flask-Bcrypt==0.7.1
* Flask-Login==0.5.0
* Flask-Migrate==2.5.3
* flask-msearch==0.2.5
* Flask-SQLAlchemy==2.4.1
* Flask-Uploads==0.2.1
* Flask-WTF==0.14.3
* isort==4.3.21
* itsdangerous==1.1.0
* Jinja2==2.11.1
* lazy-object-proxy==1.4.3
* Mako==1.1.2
* MarkupSafe==1.1.1
* mccabe==0.6.1
* pdfkit==0.6.1
* pipenv==2018.11.26
* pycparser==2.19
* pylint==2.4.4
* python-dateutil==2.8.1
* python-editor==1.0.4
* six==1.14.0
* SQLAlchemy==1.3.13
* virtualenv==20.0.4
* virtualenv-clone==0.5.3
* Werkzeug==0.15.6
* wrapt==1.11.2
* WTForms==2.2.1

App Navigation

• **Registration:**

- signUp

- Build System

**• Login:**

- login

• **Build System:**

- Component Details

- Add Component

- Remove Component

- Navigation Drawer

• **Navigation Menu:**

- Create Build

- Builds

- Components

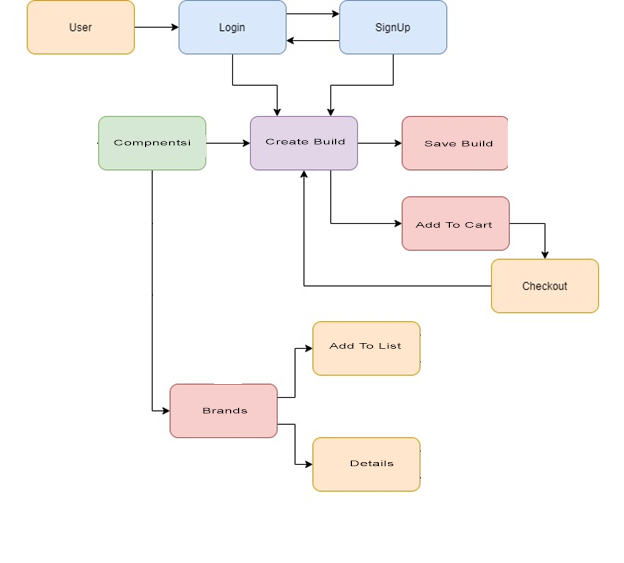
- Brands

- Login

- Cart

- Search

Navigation Map



Testing

**Unit Testing Plan**

We decided on performing unit testing, where individual components and test cases would be tested to validate that they are working as expected.

<http://softwaretestingfundamentals.com/unit-testing/>

*The Table Below outlines the cases to be tested, the test steps and the expected results.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Test Steps** | **Expected Results** |
| **1** | **User Profile Creation (Signup)** | Step 1: Enter incorrect data into fields and click submit  Step 2: Ensure appropriate field error messages displayed  Step 3: Enter valid data into fields  Step 4: Click submit  Step 5: Navigate to database and search for appropriate user by “email address” | The system acknowledges the errors and highlights the mistakes in a red field and prompts the user to reenter the information |
| **2** | **Login** | Step 1: Enter incorrect data into fields and click submit  Step 2: Ensure appropriate field error messages displayed  Step 3: Enter valid data into fields  Step 4: Click submit  Step 5: Navigate to database and search for appropriate user by “email address”  Step 6: Verify information displayed corresponds with what was entered upon creation | The system acknowledges the errors and highlights the mistakes in a red field and prompts the user to reenter the information |
| **3** | **Testing Products** | Step 1: Navigate to catalogue screen  Step 2: Select item on the screen.  Step 3: Verify that details of the product are displayed including description and image.  Step 4: ensure the carousel is displaying proper images on home page | Displays a blank space with relevant text if the images or products is not there |
| **4** | **Testing Layouts** | Step 1: Navigate to catalogue screen  Step 2: Select item on the screen.  Step 3: Verify that details of the product are displayed including description and image.  Step 4: ensure each page is outputting exactly what is needs to output | **B**lank image with descriptive text if not outputting correct information |
| **5** | **Testing builds saved database** | Step 1: add a build list from the create build option  Step 2: ensure that the build list exists in the build section  Step 3): must be logged into an account for database to load the files from the account | Correctly loads the saved build.  Else shows an empty list |

**Business Aspects**

Many people have difficulty when it comes to choosing a computer system or computer components. They often wonder if they are making the right choice or if they are getting their money's worth when making these choices. This is where the Budget Builder application shines as it takes away all the guesswork by providing the user with the best options given the information they would have supplied to the application.

IT professionals may also find our application to be of value to them if they are to provide a computer system(s) for a customer, business or organization. The key features of the application are having the option to choose the purpose for the system and the budget of the system. This will help cut the time taken to design a system and ensure compatibility, while all together staying within a budget. Having all the options together with purchase links and the ability to save the build list and components all in one place is very convenient to any user.

Registration to the application as well as the use of its tools and generators are free to all users. The business model of our application generates revenue from the established affiliations with well-known marketplaces and computer suppliers, where we receive a commission for each time a user purchases items or a complete system using the links produced by the application. Revenue will also be generated from advertisements on the web platform.

### **Safety**

The app offers user validation and verification mechanisms at all user entry points, this is done to safeguard the app’s users and its backend databases from threats and unauthorised personnel gaining access to and potentially modifying the data within the app. The safety of users accounts are also priority.

### **User friendliness**

BudgetBuilder proudly presents a simplistic, minimalistic design where the user does not have to worry about getting lost in a menu of choices and options that they know nothing about. The application is that of one overall goal and that goal is met quickly and efficiently in order to save time and be convenient.

### **Performance**

BudgetBuilder was made to operate efficiently and quickly. Button taps and page redirections should take within one to two seconds.

### **Convenience**

BudgetBuilder shines in this aspect because it lays out in a solid linear, easy to follow format, all the necessary information for a user to make their decisions quickly and effectively.

### **Legal issues**

BudgetBuilder does not pay any regard to corporations that manufacture products, it is simply an education type application.

### **Reliability**

BudgetBuilder is reliable in the fact that it is non profit and will not mislead the user.

### **Availability**

BudgetBuilder shall be available at all times.

### **Maintainability**

BudgetBuilder is a webapp, which means that there is a database that must be kept up to date in order to provide accurate information. It is hosted via Heroku and as such can be maintained using the python language and its mySQL database.

### **Technology**

BudgetBuilder utilised the following technologies throughout its development life cycle:

* Visual Studio Code - utilised for coding, compiling, testing and debugging of the application’s features and interfaces
* MySQL/MangoDB - used for the storage of student, course, document and grade records
* Github - for version control and accessibility to the project’s files
* Ubuntu
* virtualenvironments

### **Design Methods**

BudgetBuilder was constructed by implementing a cusp between Agile and Scrum methodologies. Our team was very informal. With this methodology merge, the team also tested several segments of the application’s code as it was in the process of being developed.

# Individual contributions

Jaleel Mohammed : - Coding, most of the code for the project webapp.

(coding lead) All the testing, (user/system)

Creating the Database

Debugging code

Software Testing

Database population

Research (Linear Programming)

Hosted the webapp

Aaron Meighoo: API

(project lead) : Some of the coding

:Video Editing and rendering

Project Website

Project Github

Software testing

Database population

Research for frontend

Minor Backend

Kyle Fonrose: Documentation and Reports

(Document Lead): Record meeting discussions

Weekly Status Reports

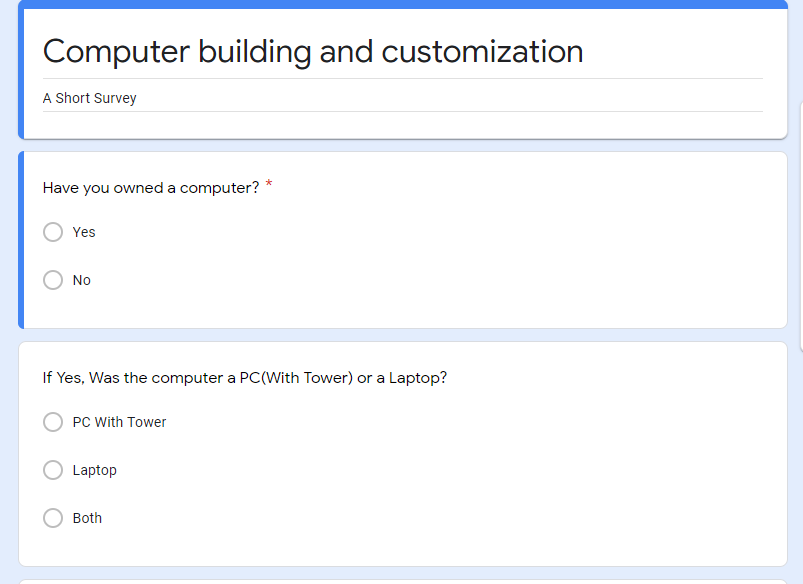
Project Documentation

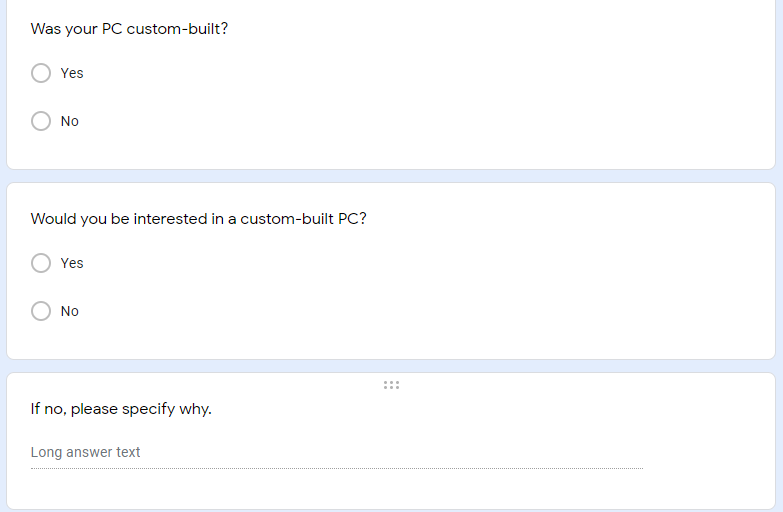
Data Collection

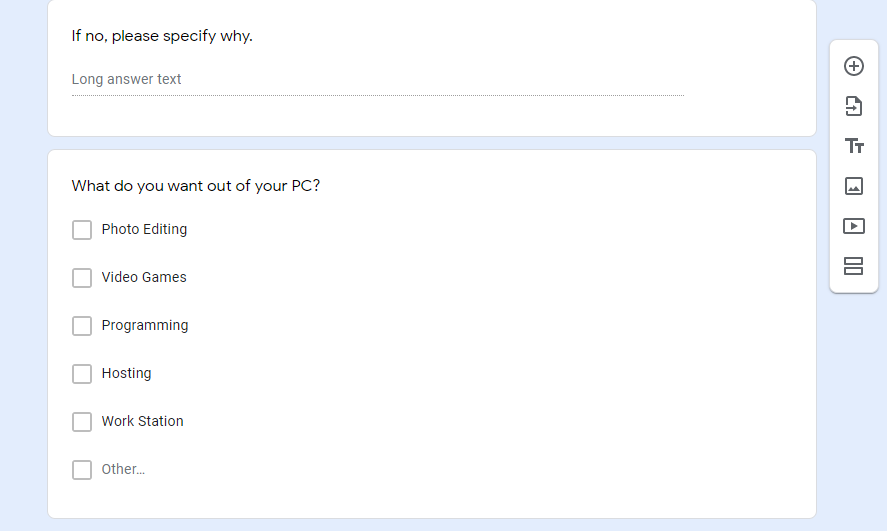
Data Analysis

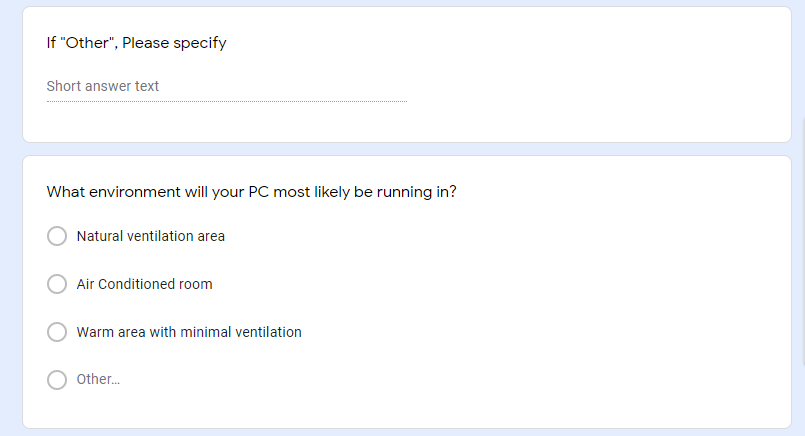
Implementation Testing

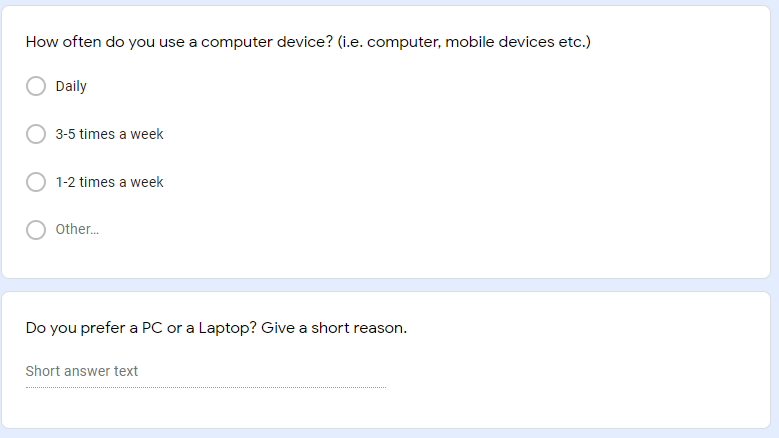
# Survery

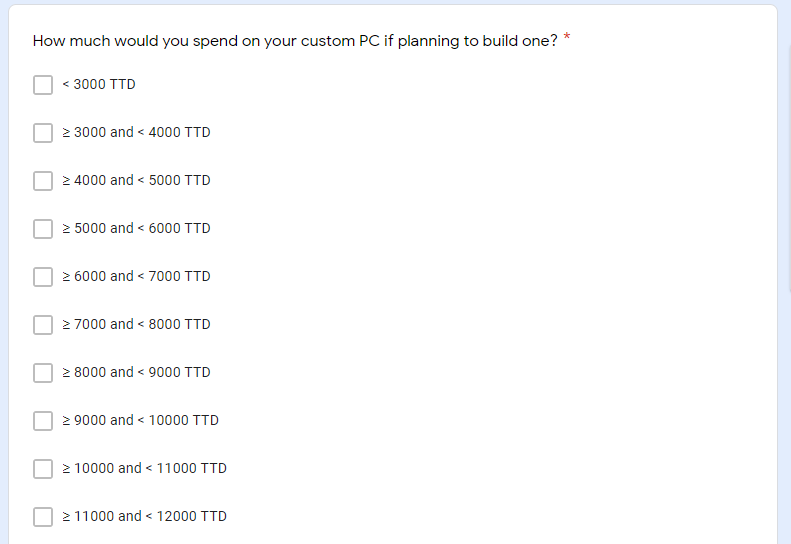
****

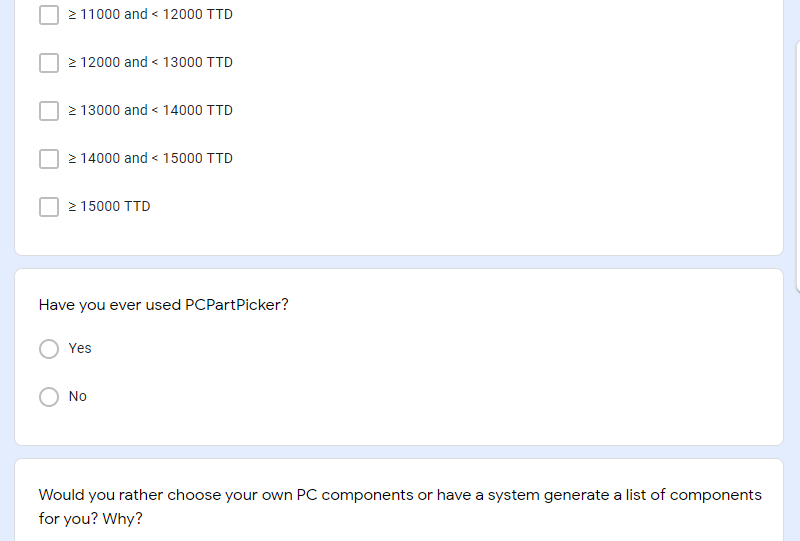
****

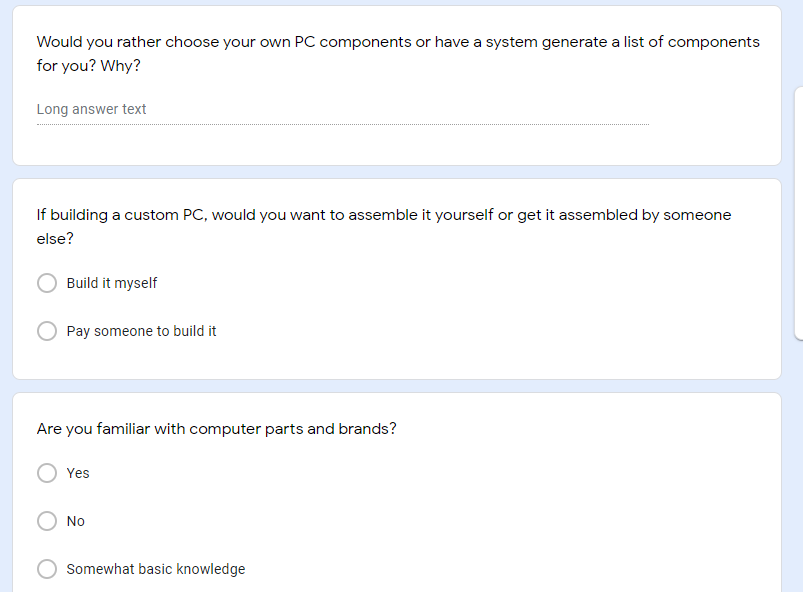
****

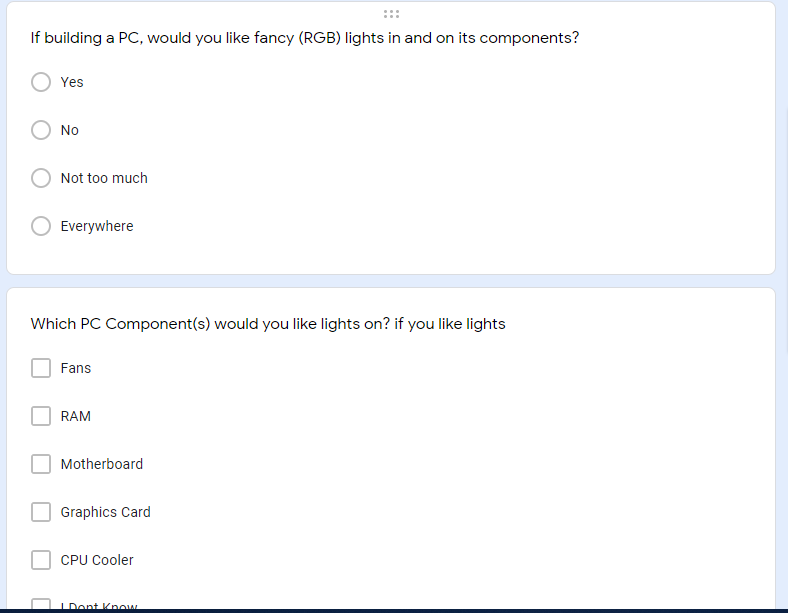
****

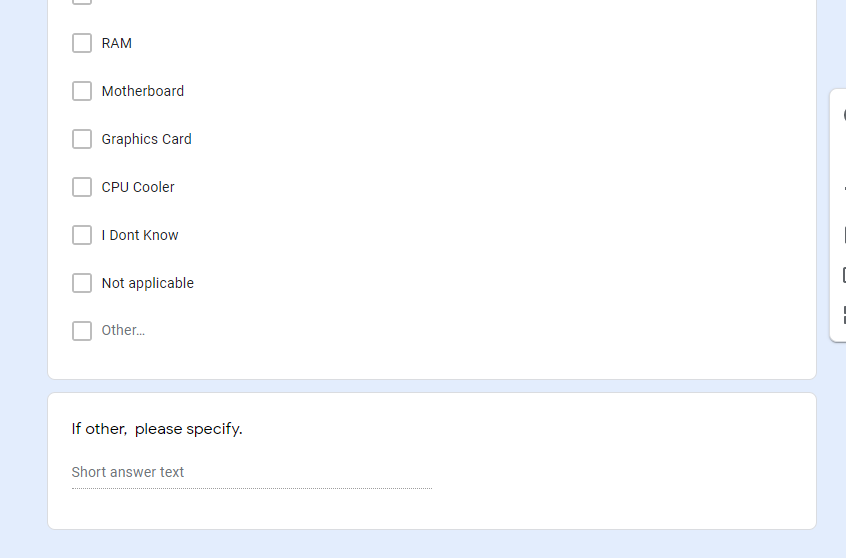
****

****

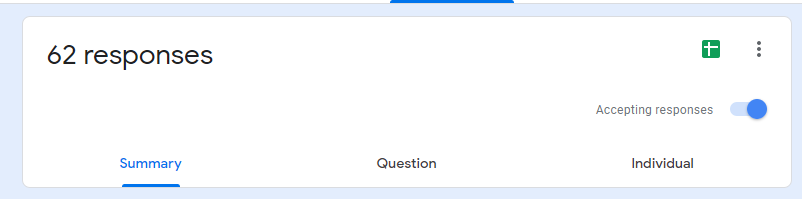
****

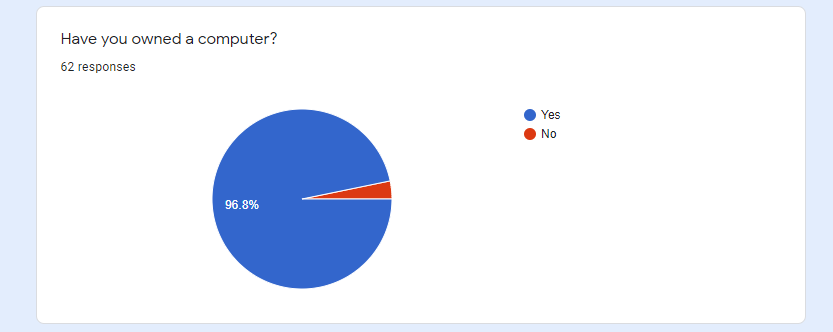
****

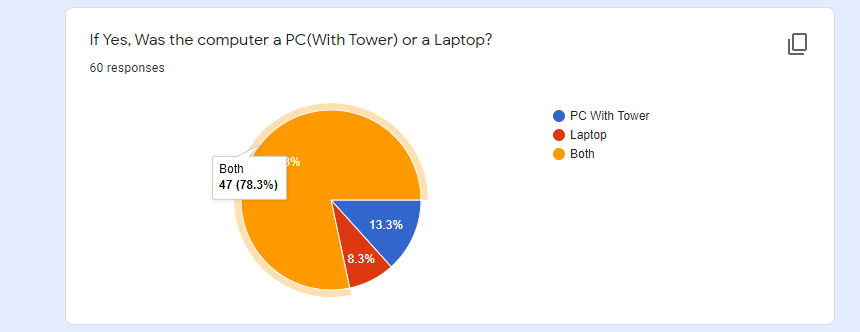
****

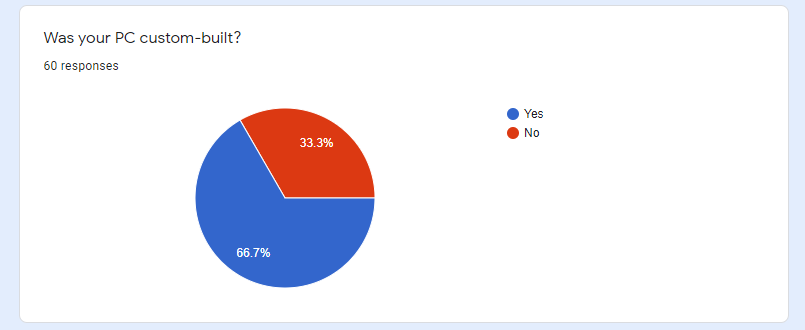
****

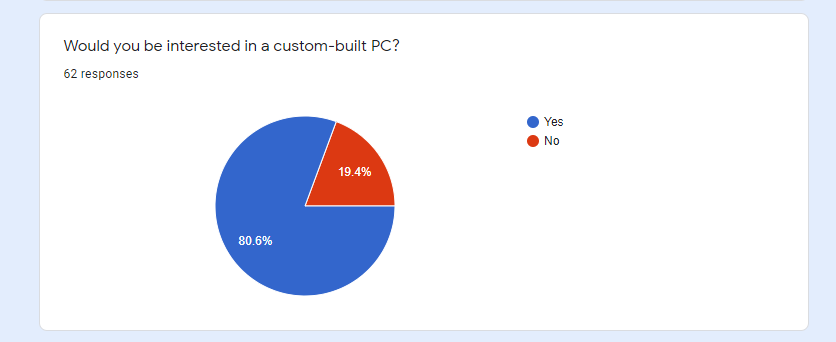
# Responses

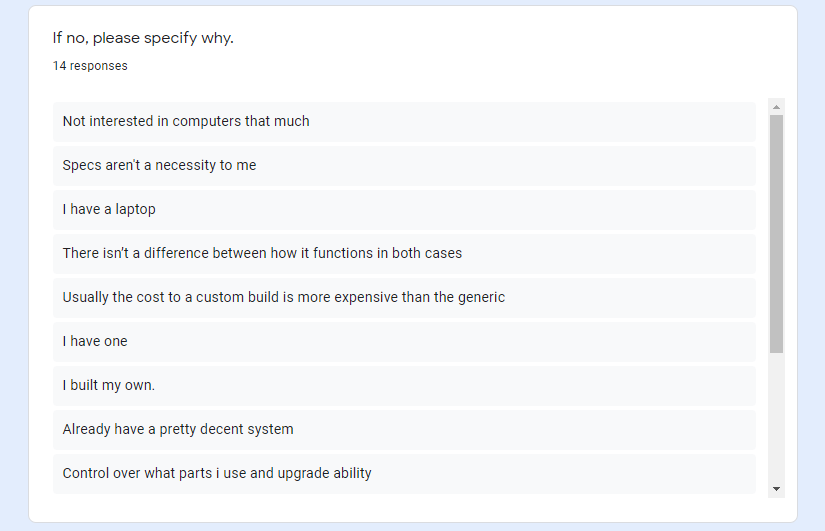
****

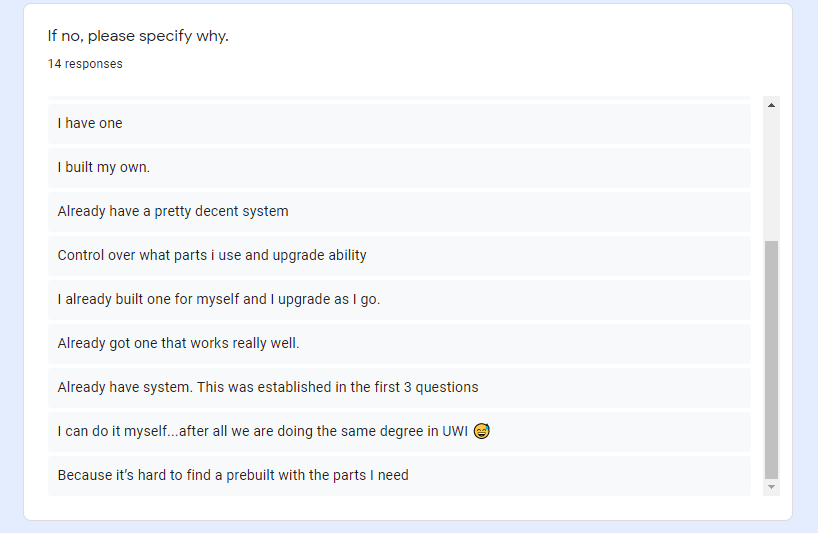
****

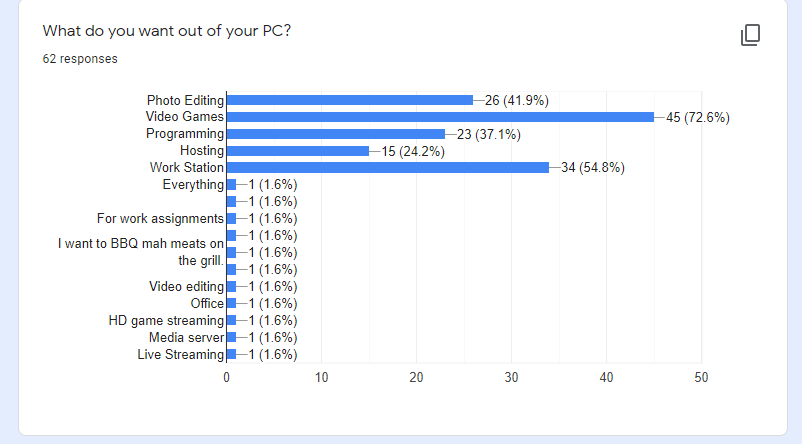
****

****

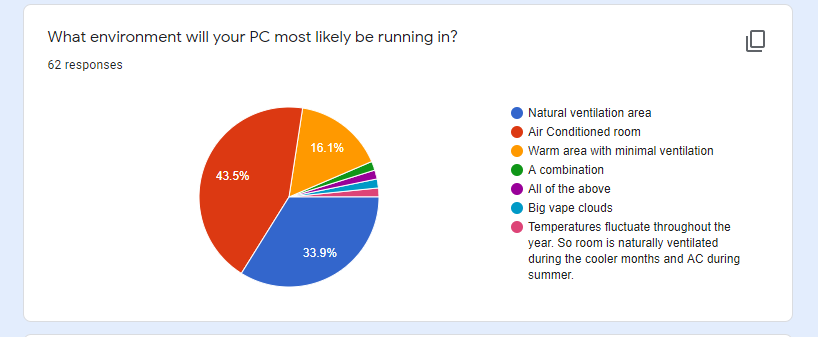
****

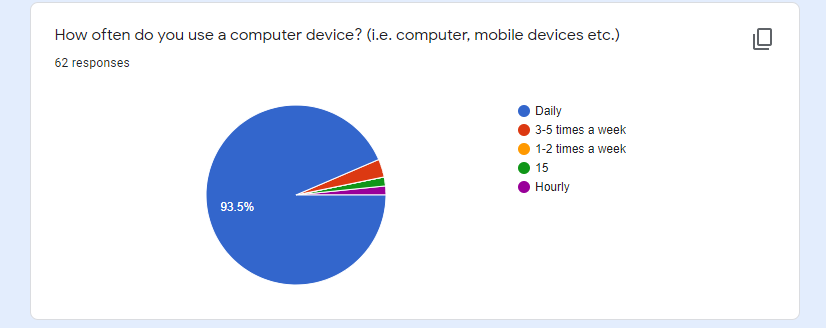
****

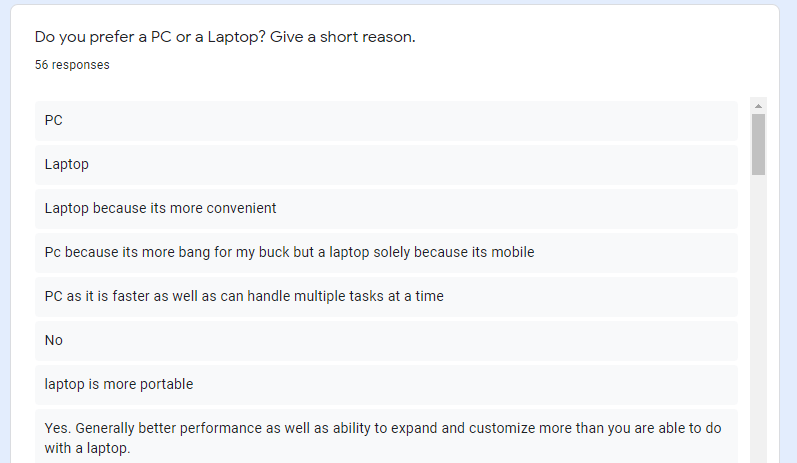
****

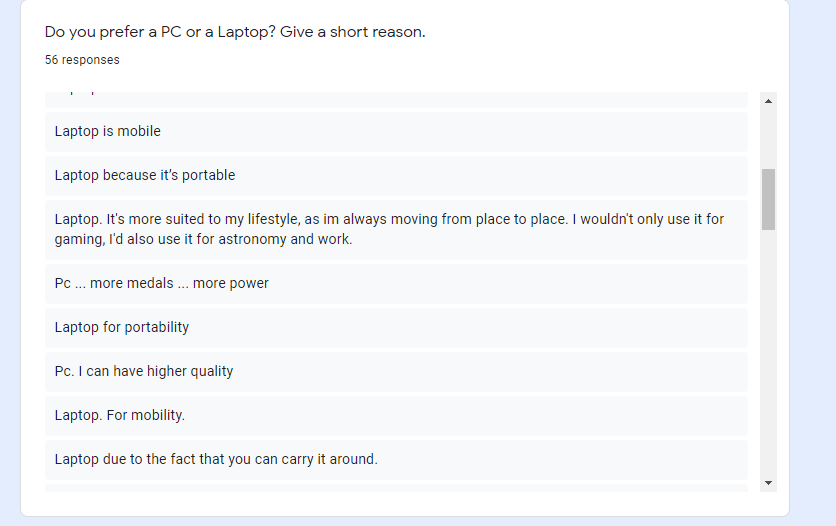
****

****

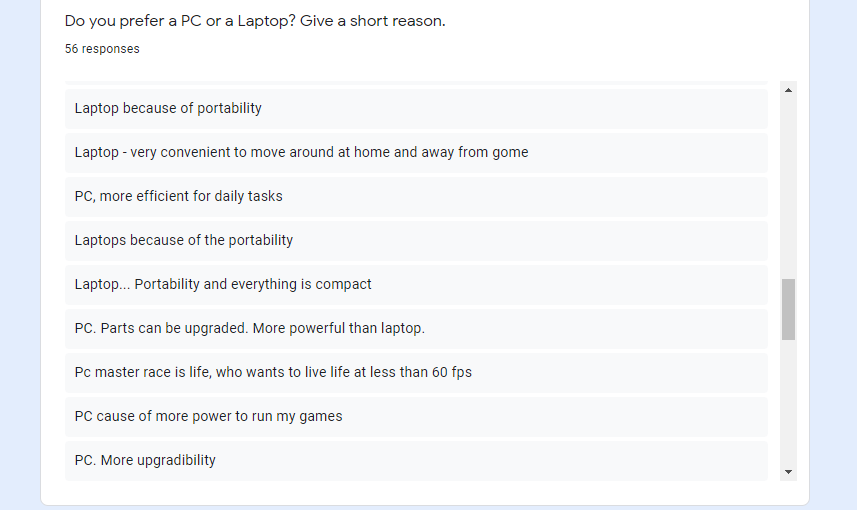
****

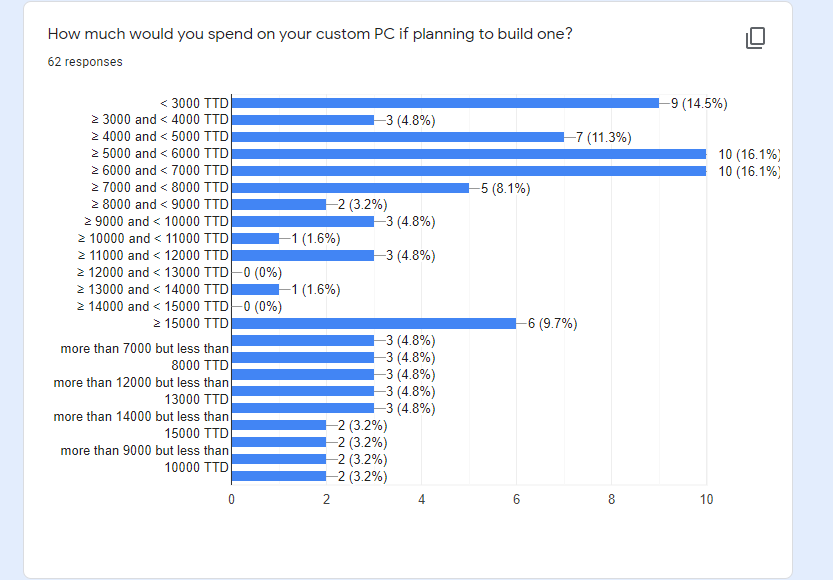
****

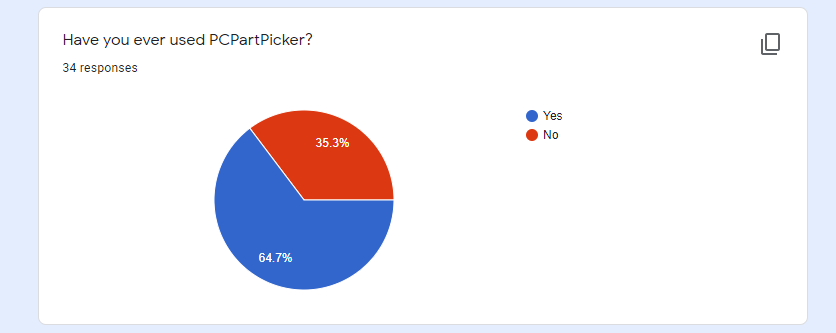
****

****

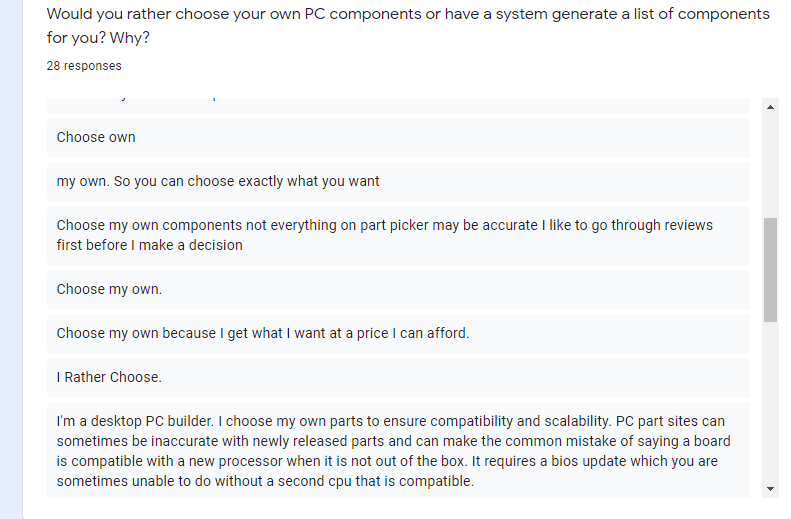
****

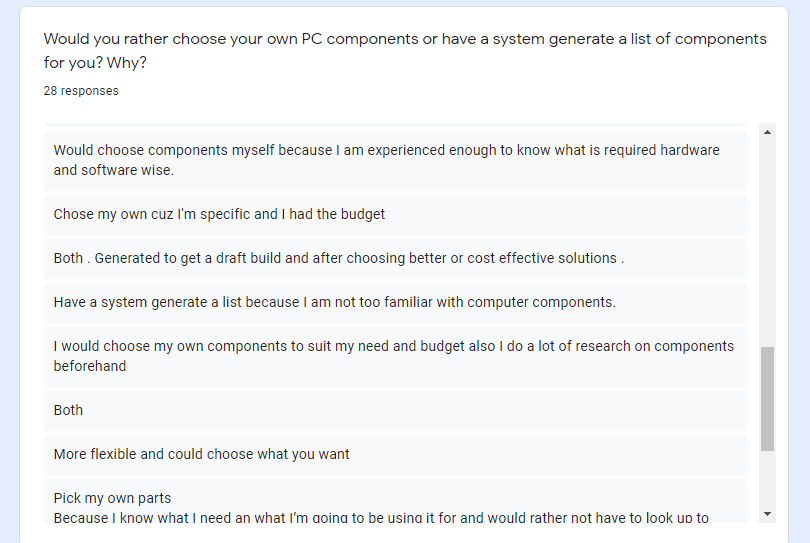
****

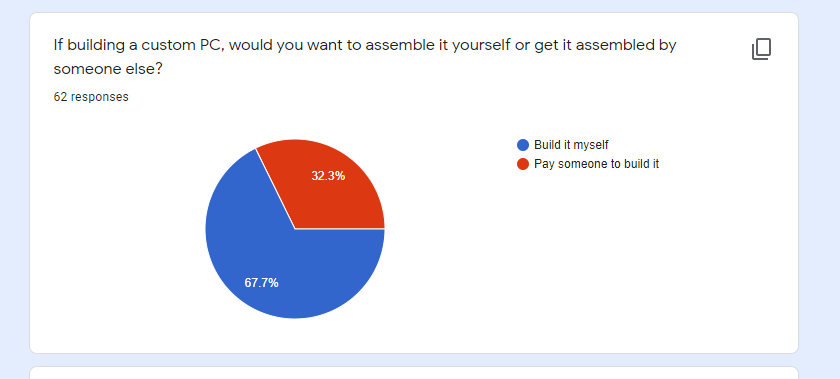
****

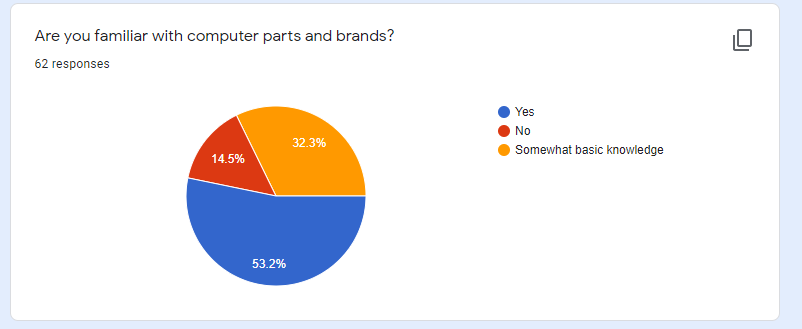
****

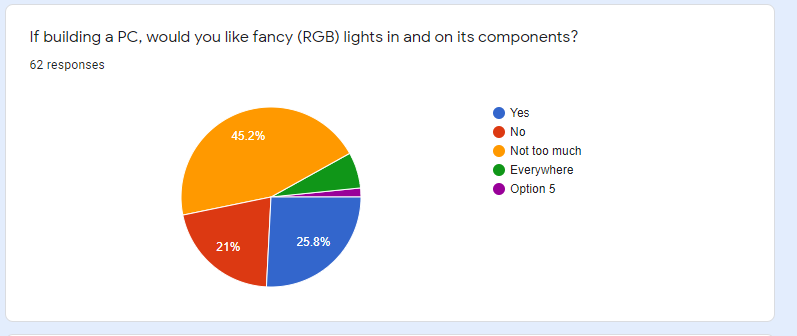
****

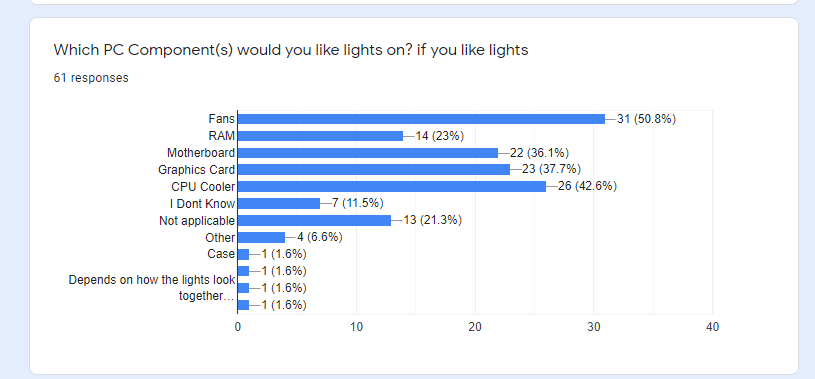
****

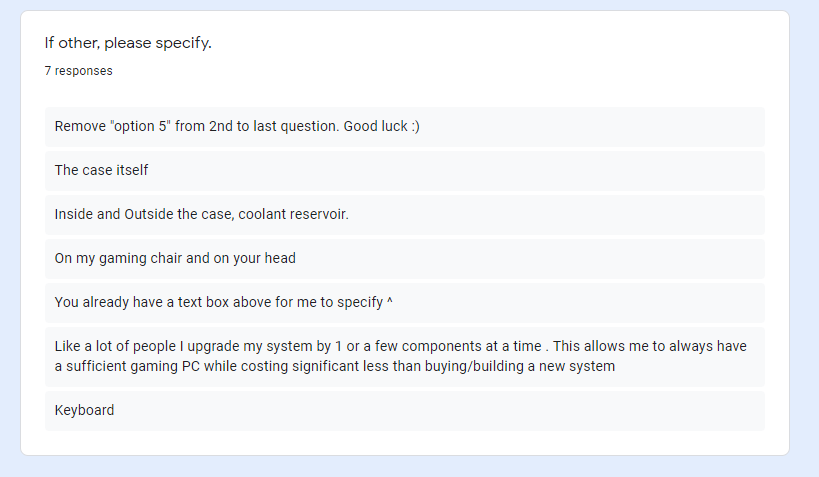
****

****

****

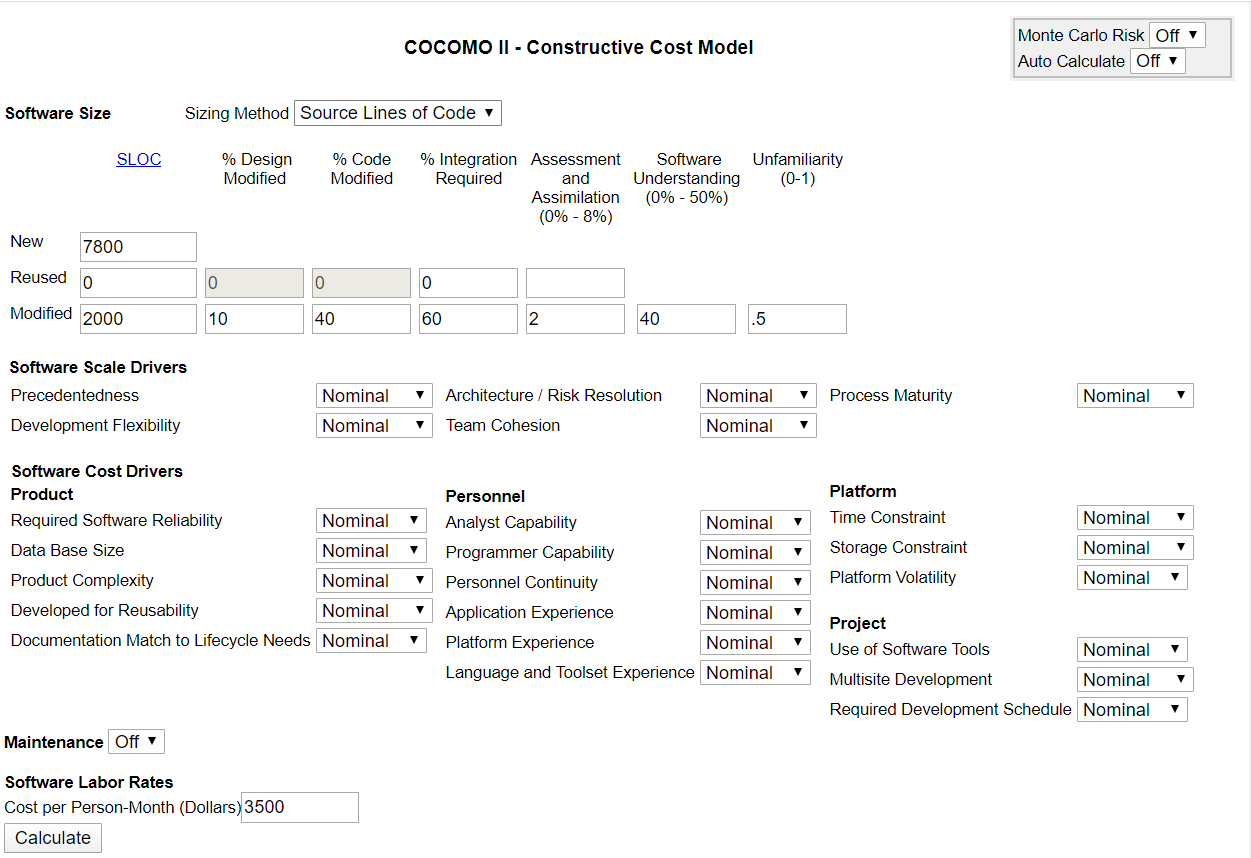
****

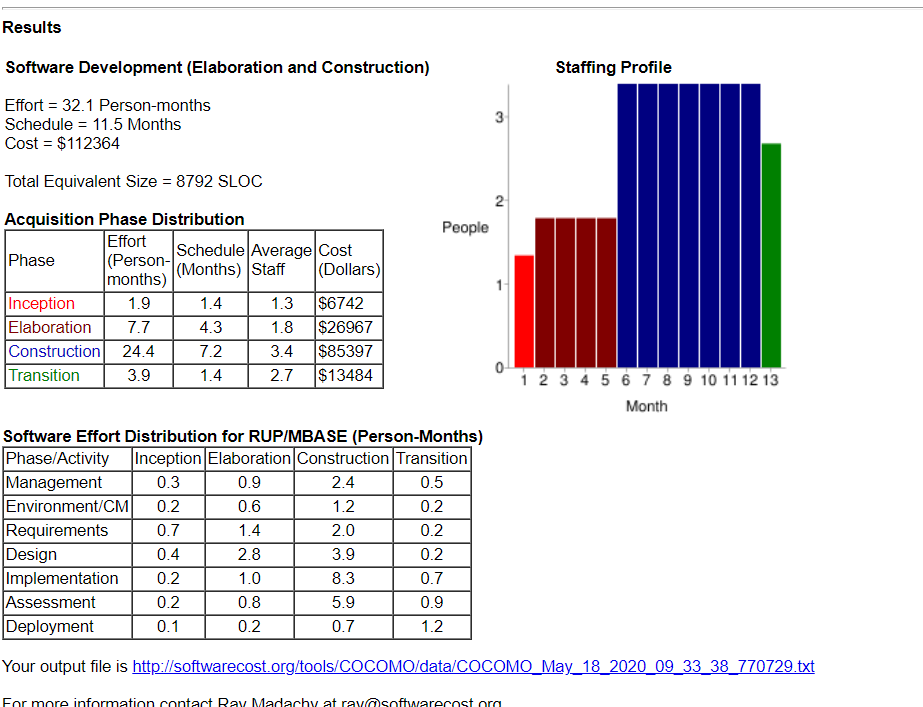
****

****

**Financial Considerations**

**Using The COCOMO Model**





**Conclusions, Lessons learned and**

**Recommendations**

State of Completion

At the end of our project timeframe it is with a bit of dissatisfaction that we were not able to complete the application to our desired goal. However, we were still able to implement a few key features which would hold hope for a developing platform in the future. Due to a lack of experience and unforeseen circumstances with the global pandemic, our members collectively faced some difficulties working and managing the overall project.

Summary of Feasibility

After discussions with some of the individuals of our focus group and analysis of the responses collected from our survey, it is no doubt that Budget Builder will be beneficial to people who faced similar dilemmas when deciding on a computer system. However, it may be a small group of persons who would be interested in use an application because

Future Work

There is a lot of future potential for the app “Budget Builder”, which will help enhance the user experience and generate more revenue.

Some of the possible future development:

* More user specific options geared toward enthusiast.
* Partnership with suppliers and affiliated programs
* Refreshed User Interface
* More backend/admin support

# BudgetBuilder Final Presentation Video Link

<https://1drv.ms/v/s!AnMpMAgVgXtOxl_vheyTYK6SJRlq?e=QEZEIh>

# Hosting Via Heroku Link

<https://gentle-stream-20277.herokuapp.com/>

**References**

<https://www.guru99.com/agile-scrum-extreme-testing.html>

<https://www.lucidchart.com/pages/>

<https://www.entrepreneur.com/encyclopedia/positioning>

<http://softwarecost.org/tools/COCOMO/>

<http://softwaretestingfundamentals.com/unit-testing/>