Security Assessment Report

Latest Version of:

April 17, 2023

Table of Contents

1. Summary ....................................................................................................................................3 1. Assessment Scope .............................................................................................................3

2. Summary of Findings ........................................................................................................3

3. Summary of Recommendations ........................................................................................4

2. Goals, Findings, and Recommendations ....................................................................................4 1. Assessment Goals .............................................................................................................4

2. Detailed Findings ..............................................................................................................5

3. Recommendations .............................................................................................................5

3. Methodology for the Security Control Assessment ...................................................................5 4. Figures and Code .......................................................................................................................7 4.1.1 Process flow of System (this one just describes the process for requesting) ........7

4.1.2 Other figure of code ..............................................................................................7

5. Works Cited ...............................................................................................................................7

1. Summary

In essence, the goal of this project was to recognize the issues within the Menu Randomizer Project. To find these problems the code was heavily reviewed and put into practice with black box methods much akin to QA testers. According to the findings, there were many corrections to be made within the program and security concerns to be addressed. Issues such as data storage, unhandled exceptions, syntactical errors and buffer overflow were all featured in the original program version. In order to repair these issues many alterations would have to be made. New variables would need to be declared, alternative storage methods set up, and conditions and statements made. For example, the majority of the stored information was stored in arrays and these arrays would change size quite often. Making a switch to something with more dynamic memory storage such as a vector would be more prudent. A change like this is merely an ease of life, however, other alterations would need to be implemented to actually protect the program in different ways. Amending the required conditions of a username or password or adding security policy within Github itself.

1. Assessment Scope

The primary coding software used was Microsoft Visual Studios and everything was run on Windows 11. Brave browser was used to search most solutions for everything else Google was utilized. Github helped secure the code and share it when necessary. Websites such as GeekforGeeks, Opengenus, Stackoverflow and learn.Microsoft were very helpful links to finding solutions and learning what proper code I should use.

2. Summary of Findings

Of the findings discovered during our assessment, 5 were considered High risks, 4 Moderate risks, 3 Low, and 2 Informational risks. The SWOT used for planning the assessment are broken down as shown in Figure 2.

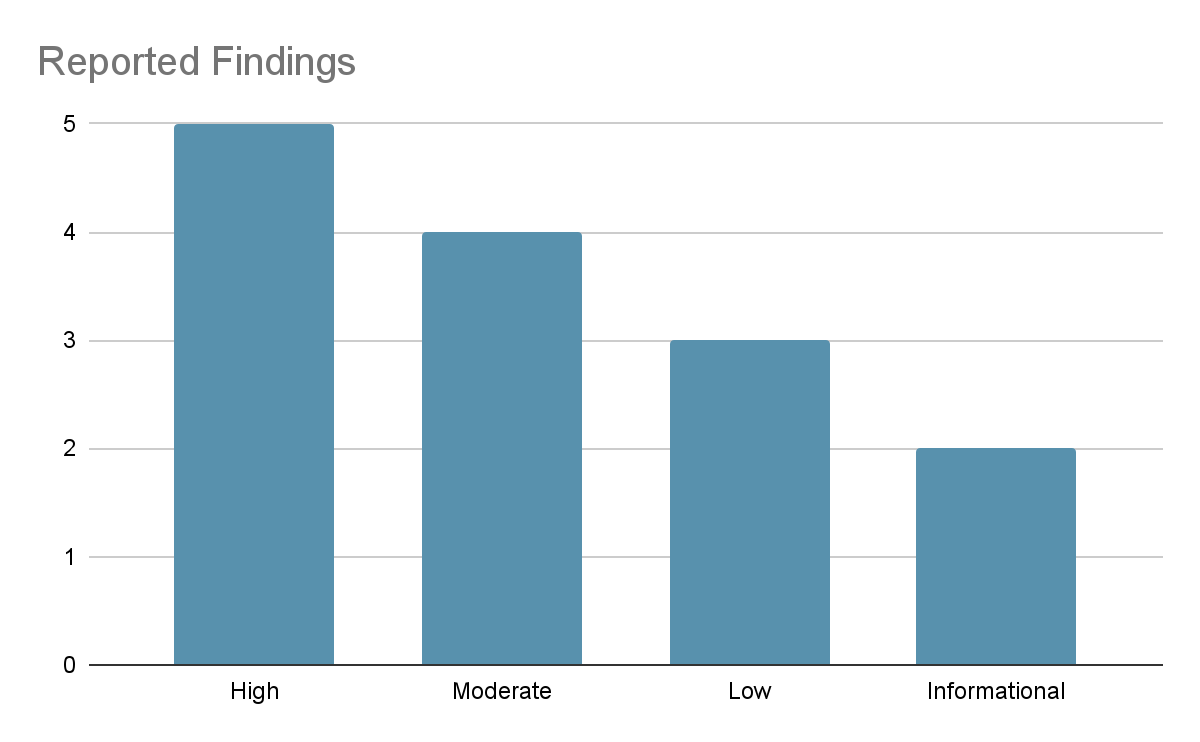


Figure 1. Findings by Risk Level

The largest number of risks were found to be high and were all heavily related to the front-end security and these vulnerabilities could lead to back-end issues. For example, one error which was present checked for the negative responses of whether the user had an account already but not the positive responses. This permitted the user to enter anything besides no and receive a continuation. An issue such as this could lead to breaking the program in such a way that usernames and passwords could be accessed. So, the solution was to look for the positive answers and place limitations on what constitutes a username or password. Aside from this, the more moderate or low risks were more storage or syntax based. Altering arrays to vectors or rewriting of some lines were necessary to repair some of the problems. The few informational risks were not necessarily fixable but could be acknowledged or managed adequately. These include an error with the Visual Studio terminal and the utilization of Github’s security capabilities such as security policies or insights.

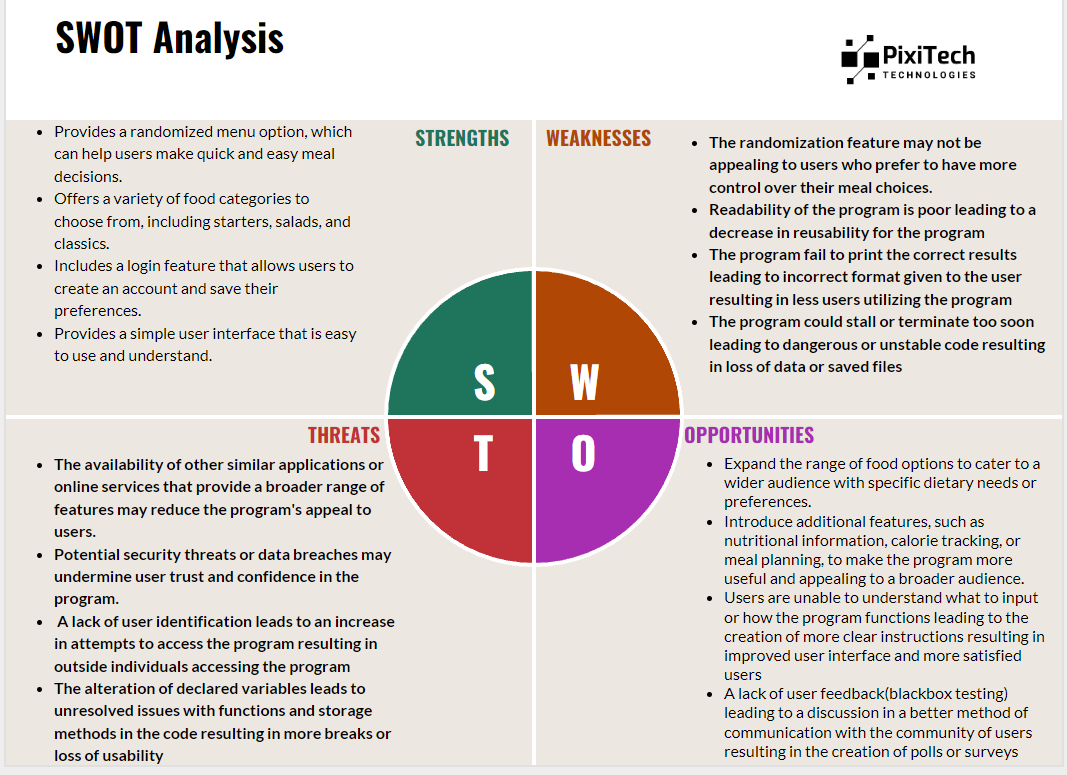


Figure 2. SWOT

The strengths of the program are addressed in this assessment heavily to ensure they are kept as strengths. For instance, without clear and stable storage for all of the items or perhaps the storage of user input which should be considered there is a clear danger here. User identification is also heavily addressed through creation of passwords and usernames as well as a secure method of storing the former. This is without mentioning that these vulnerabilities could erode the users trust in the program. Expanding upon the items stored or the altering of items stored is addressed and is quite clear to the user on how to do so.

3. Summary of Recommendations

As of this assessment, the majority of the issues have been solved aside from some high and low risks. A method to check whether the user has an account and to prompt the user for their username and password was implemented. With this change a hashing algorithm was added to encrypt the information provided. The use of namespace std was removed and fixed. Github security policy was added and the arbitrary lengths of arrays were removed through the implementation of vectors in their place. When all items are removed from an array(vector) there is an error message printed as well as a catch for any wrong input entered for the username and password functions. Some memory still needs to be moved to the heap and when the vectors are empty and a menu is produced memory locations are printed where items should be. Some exceptions when it comes to inputs are still present and the prices should be stored differently. The temp variables in the .h files are defined out of scope and the checking of a user’s username and password are not quite finished.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

· Ensure that the system was in compliance with regulations you had to deal with or any other requirements (to include the assignments themselves).

· Determine if the application was securely maintained.

· Etc.

2. Detailed Findings:

[Github Security](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/11)

The first vulnerability to the project was a lack of security in Github itself, so I set up what I could, given what I could do without paying. I made sure the security advisories were enabled to publish any vulnerabilities in the code. Then to ensure I found out about any errors or issues I was not aware of I made sure to include both a read me and a security policy. These detailed the overall purpose of the project both in function and for my educational benefit. They also explained what to do if a vulnerability is found and how to contact me. A branch protection rule was added to require approvals in order to protect against malicious code being pushed.

[Namespace Std Use](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/6)

Using the namespace std is largely known as bad practice in coding in most cases. I used it initially as I found it easier to write code without too many lengthy lines. But, after some research I found it was not such a hindrance on code length and actually a lot safer than using the namespace. Without namespace there is less of a chance of naming conflicts and it is also easier to locate lines that share the ‘std::’.

[Arbitrary Length of Arrays](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/10)

It was my intention to create the length of the arrays containing the menu items and their prices to a length that would be highly unlikely for any business to reach. However, it is true that this creates risk in the form of a method of breaking the program. This would also create space in memory potentially unnecessary for what information is being stored. A standard template library (STL) would be much more efficient and safe as it would limit the amount of space created. I solved this issue utilizing a vector as storage in place of an array.

[Password Protection](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/1)

Originally, the program ran without any sort of limitation on who could access it. Without such protections the program is open to just about anyone who wishes to use it. This would include someone who had malicious intentions when it came to the company or individual who is using the program.

[Hashing Algorithm](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/2)

A simple password implementation would not be sufficient as it would be much too easily broken. Especially if the usernames and passwords are stored in plain text. Therefore, a hashing algorithm would store all usernames and passwords as encrypted mixed characters. Although this would help with the risk, the hashing function should be called when inputting usernames or passwords and when retrieving.

[getUser/getPass Issue](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/4)

Because the usernames and passwords are stored in separate vectors when both are checked it does not matter whether the entered match or not. So, if someone were to enter the correct username for one account and the correct password for an entirely different account the program will still be accessed.

[Remove() with no Array Elements](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/8)

When the interface for the program is called a series of options are presented. The remove option deletes selected items from the given array that is also selected. The issue is that when the remove function is called again and again eventually there are no more items. There is not a great risk, but if there are eventually no items to delete then when you finally have reached this point how does the user know when to stop?

[No Exception for Entries; Accounting Check](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/12)

For this error there would be a higher risk were the fix not so simple. When the program requests a yes or no a yes is checked for but a no is not so anything entered besides yes will result in a continuation. Were it not for an exception in the case of a negative, anything could continue the program regardless of what is entered.

3. Recommendations\*\*\*

Here’s where your fixes go (ensure you reference Table 2 for your ease of fix evaluation and explain why it matches that category). For further issues or information please visit the Github link to the project itself under the issues tab.

[Github Security](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/11)

To implement some Github security features I followed an instructional video to see what features were pertinent for my project. I found I could implement a security policy and branch protection rules. Not all were applicable and not all were accessible without payment. The policy and a read.me added to the project by protecting the code from damaging alterations and explanations of the code’s use. This was conclusively an easy fix.

[Namespace Std Use](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/6)

This was another fairly easy fix as it merely required the removal of all namespace lines in the code along with all related lines. These would then be replaced with ‘std::’ statements and the like where they are necessary across the program.

[Arbitrary Length of Arrays](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/10)

While more risky than previous fixes it was only moderately difficult to fix as I was not set on the solution for this issue. But in the end I found the simplest change and the most effective to be the replacement of the arrays with vectors instead. This allows the amount of data to change with the amount of data that can be stored.

[Password Protection](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/1)

A very risky function to be missing from the program and only moderately difficult to produce. I created a class of newUser and a series of functions evaluating a user’s responses. If the user is a member they need only enter their username and password. Should the user need to create an account they first would need to enter the company code which is hard coded into the program. This is meant to make it more difficult to be changed except by the company itself. Once the code is entered, the user will be made to enter their chosen username and password to be stored by designated vectors.

[Hashing Algorithm](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/2)

The issue with having passwords and usernames stored is how they are stored. If the stored sensitive information is in plain text it is much easier to access this information as a hacker or with malcontent. So, with a hashing algorithm being implemented when the password is entered and when it is checked there will be no plain text passwords stored. All passwords stored will be encrypted by the hashing algorithm and otherwise unreadable.

[getUser/getPass Issue](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/4)

This was a somewhat difficult fix for me as the solution was not inherently clear to me. It would turn out that instead of storing the usernames and passwords in separate vectors they should be stored in what is called a “pair vector”. This creates a single vector with two different paired storing locations. This was my first program using vectors let alone a pair vector so the syntax was foreign to my eyes.

[Remove() with no Array Elements](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/8)

This item was not so risky but had potential to make the program not so user friendly. Everytime the user continued to remove items from the arrays(now vectors) the program would keep removing despite the lack of items and print the menu with nothing. With this fix I create statements checking for this limit and print out a warning to the user that there are no more items to delete.

[No Exception for Entries; Accounting Check](https://github.com/45aidpre/Menu-Randomizer-Final/milestone/12)

The program was missing a catch for the positive responses so that when they were entered it did not matter what was entered into the program, it would proceed without checking for a username or password. To stop this issue one merely had to add a check for the positive values and return to the prior options when receiving the incorrect input. This was a fairly risky error as one could enter any input and still gain access however the fix itself was an easy level from table 2.

3. Methodology for the Security Control Assessment

My process began with simple analysis of the code itself and any syntactic or logical errors that could be found. I then used the Visual Studio debugger along with my own online research to attempt to explain any further errors located. Then through some peer assistance and penetration testing of my own I was able to narrow down the rest of the issues in the program.

1. Code Review- During the code review I was able to gather some idea of the initial issues with the program and begin some fixes. I had some help with peers and from professors throughout the semester.
2. Dynamic code analysis- After reviewing the code I was able to use the debugger to further analyze the code. The debugger is absolutely essential for software development and it was used by creating breakpoints where the concerning lines of code were found to be.
3. Penetration testing- Pen testing was used on a step by step basis. First I planned out how one could possibly break the program or how one could try to break it. Then, with all of the information about the program already in hand, I was able to attempt certain exploits to try to “break” the program. I accomplished this by entering different inputs that could be otherwise unexpected and using the program as it may not otherwise be meant to be used.
4. Overall results- Taking everything I had gathered I was able to, for the most part, list the issues on Github itself. By navigating one by one I was able to either solve or find potential solutions for each issue. The only reason for an issue not being fixed was either they could not be fixed at all or I did not have the knowledge or tools at my disposal at the time of this assessment.

3.1.1 Risk Level Assessment

Each Business Risk has been assigned a Risk Level value of High, Moderate, or Low. The rating is, in actuality, an assessment of the priority with which each Business Risk will be viewed. The definitions in Table 1 apply to risk level assessment values (based on probability and severity of risk). While Table 2 describes the estimation values used for a risk’s “ease-of-fix”.

Table 1 - Risk Values

| High Risk | Exploitation of the technical or procedural vulnerability will cause substantial harm to the business processes. Significant political, financial, and legal damage is likely to result |
| --- | --- |
| Moderate Risk | Exploitation of the technical or procedural vulnerability will significantly impact the confidentiality, integrity and/or availability of the system, or data. Exploitation of the vulnerability may cause moderate financial loss or public embarrassment to organization |
| Low Risk | Exploitation of the technical or procedural vulnerability will cause minimal impact to operations. The confidentiality, integrity and availability of sensitive information are not at risk of compromise. Exploitation of the vulnerability may cause slight financial loss or public embarrassment |
| Informational | An “Informational” finding, is a risk that has been identified during this assessment which is reassigned to another Major Application (MA) or General Support System (GSS). As these already exist or are handled by a different department, the informational finding will simply be noted as it is not the responsibility of this group to create a Corrective Action Plan. |
| Observations | An observation risk will need to be “watched” as it may arise as a result of various changes raising it to a higher risk category. However, until and unless the change happens it remains a low risk. |

Table 2 - Ease of Fix Definitions

| Easy | The corrective action(s) can be completed quickly with minimal resources, and without causing disruption to the system or data |
| --- | --- |
| Moderately Difficult | Remediation efforts will likely cause a noticeable service disruption  • A vendor patch or major configuration change may be required to close the vulnerability  • An upgrade to a different version of the software may be required to address the impact severity  • The system may require a reconfiguration to mitigate the threat exposure  • Corrective action may require construction or significant alterations to the manner in which business is undertaken |
| Very Difficult | The high risk of substantial service disruption makes it impractical to complete the corrective action for mission critical systems without careful scheduling  • An obscure, hard-to-find vendor patch may be required to close the vulnerability  • Significant, time-consuming configuration changes may be required to address the threat exposure or impact severity  • Corrective action requires major construction or redesign of an entire business process |
| No Known Fix | No known solution to the problem currently exists. The Risk may require the Business Owner to:  • Discontinue use of the software or protocol  • Isolate the information system within the enterprise, thereby eliminating reliance on the system  In some cases, the vulnerability is due to a design-level flaw that cannot be resolved through the application of vendor patches or the reconfiguration of the system. If the system is critical and must be used to support on-going business functions, no less than quarterly monitoring shall be conducted by the Business Owner, and reviewed by IS Management, to validate that security incidents have not occurred |

3.1.2 Tests and Analyses

This was completed using: <list and describe any type of testing you performed here>.

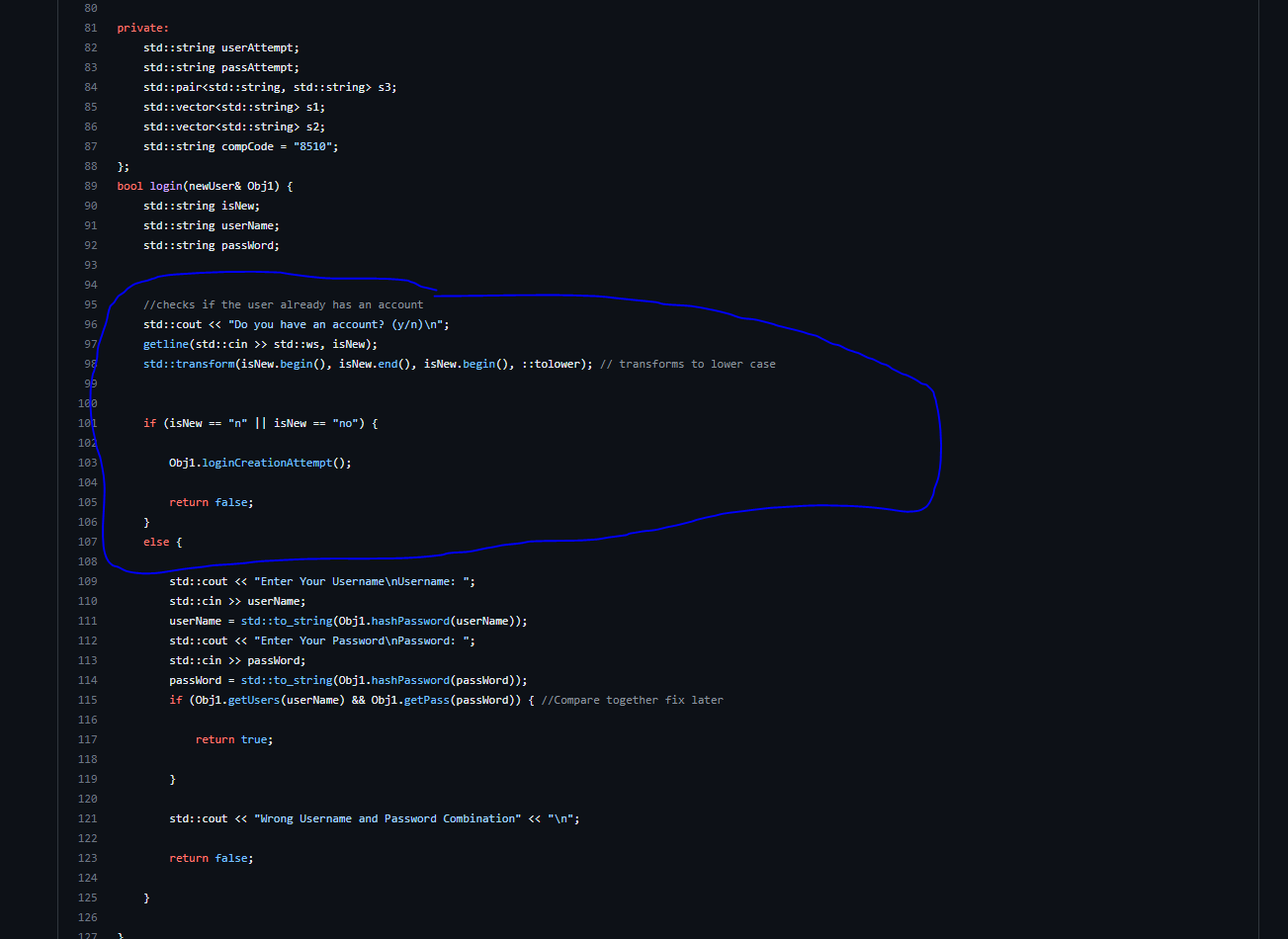
* Penetration testing- by simply using the program and trying to enter unexpected input or using the program how it should not be.
* Threat modeling- through identifying threats and analyzing how they could be dangerous in different areas of the code I could help to mitigate them.
* Code review- I had peers review my code and see what they could find in terms of vulnerabilities or errors.
* Dynamic code analysis- I was able to use the debugger in Visual Studio to identify multiple errors and handle them as they became apparent.

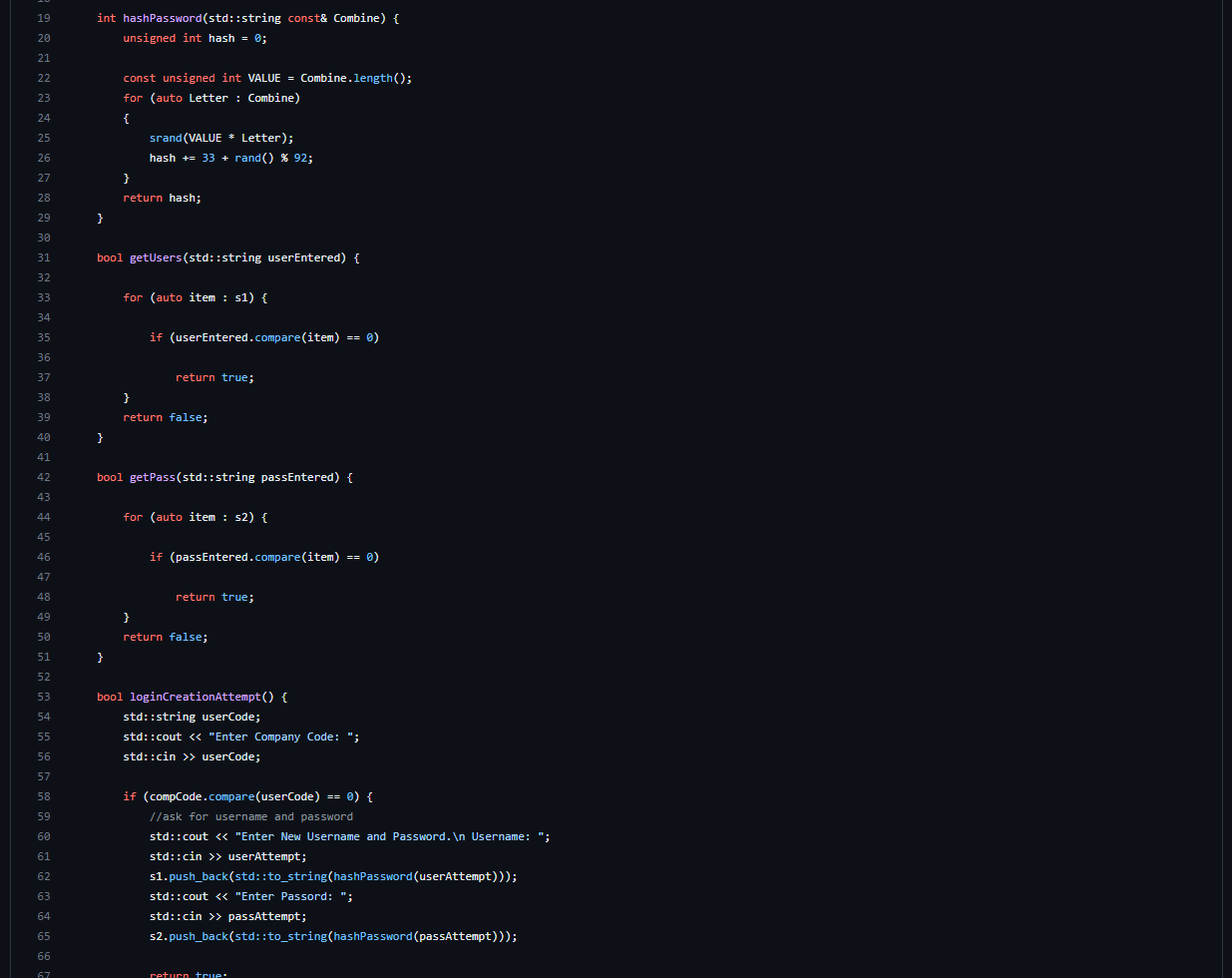
3.1.3 Tools\*\*\*

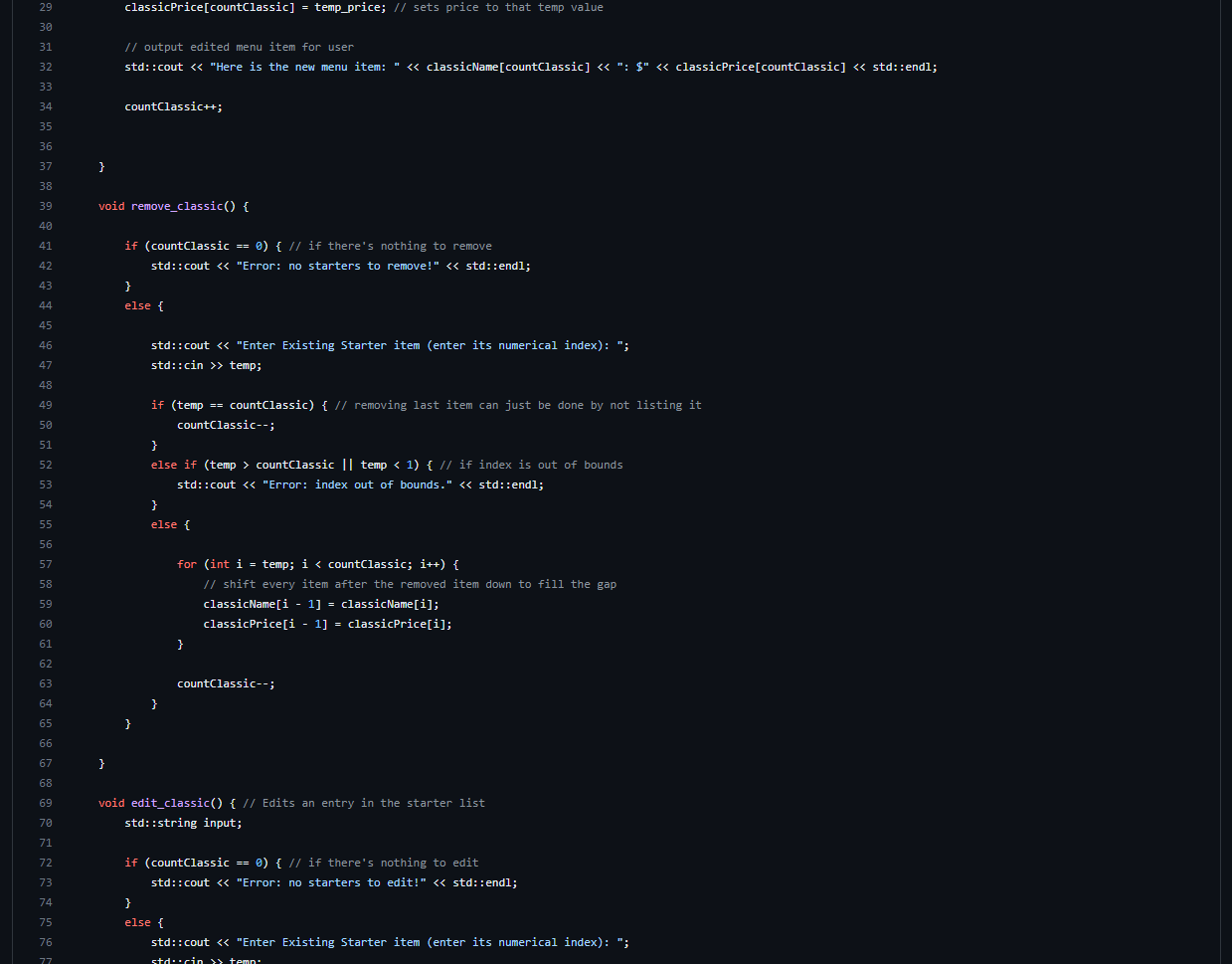
1. Visual Studios Debugger
2. Openstack.com
3. Geeksforgeeks.com
4. Github.com
5. Opengenus.org
6. Microsoft.com
7. Stackoverflow.com

4. Figures and Code

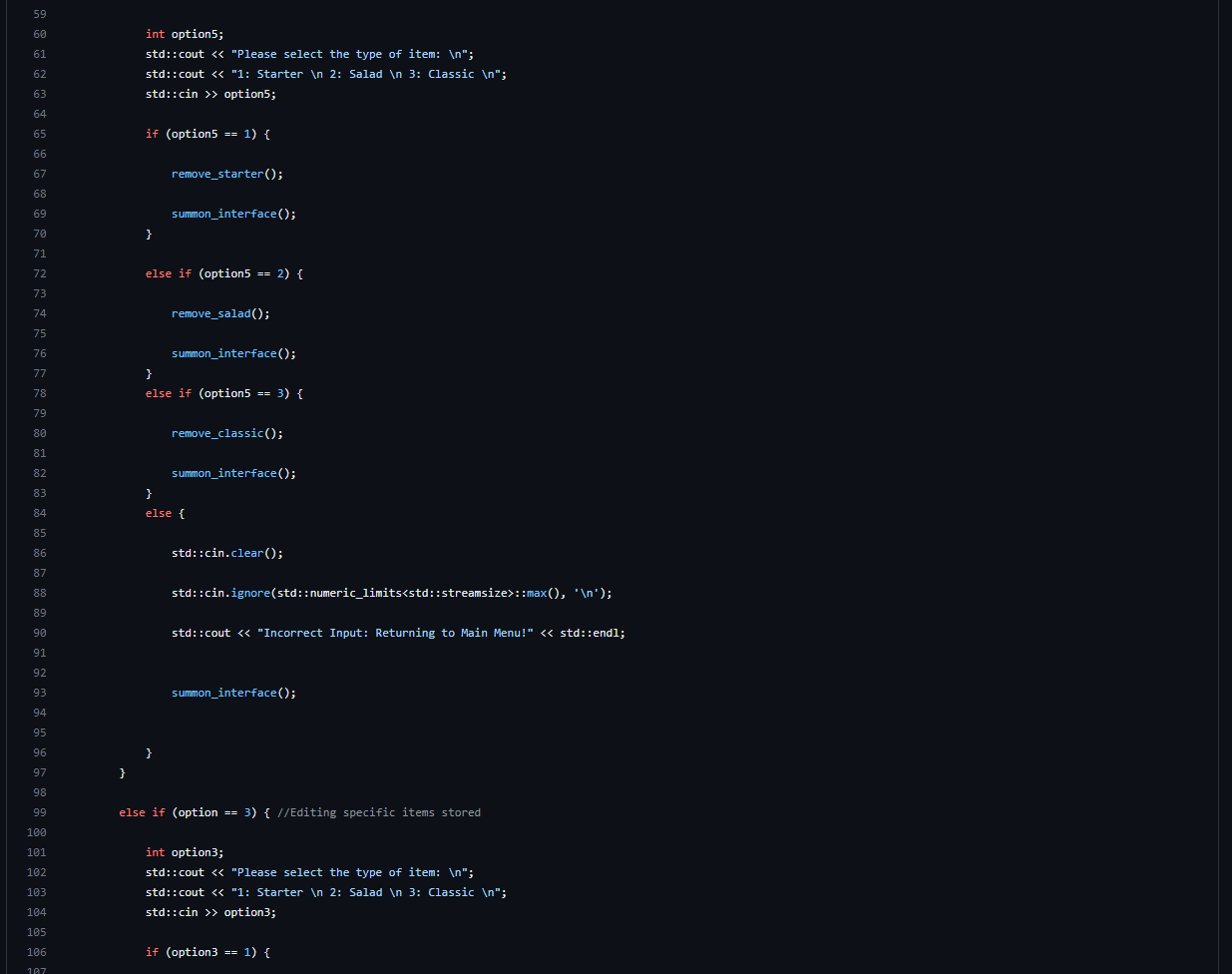
[.cpp link](https://github.com/45aidpre/Menu-Randomizer-Final/blob/master/Menu%20Randomizer%20Final/MenuRandom.cpp)



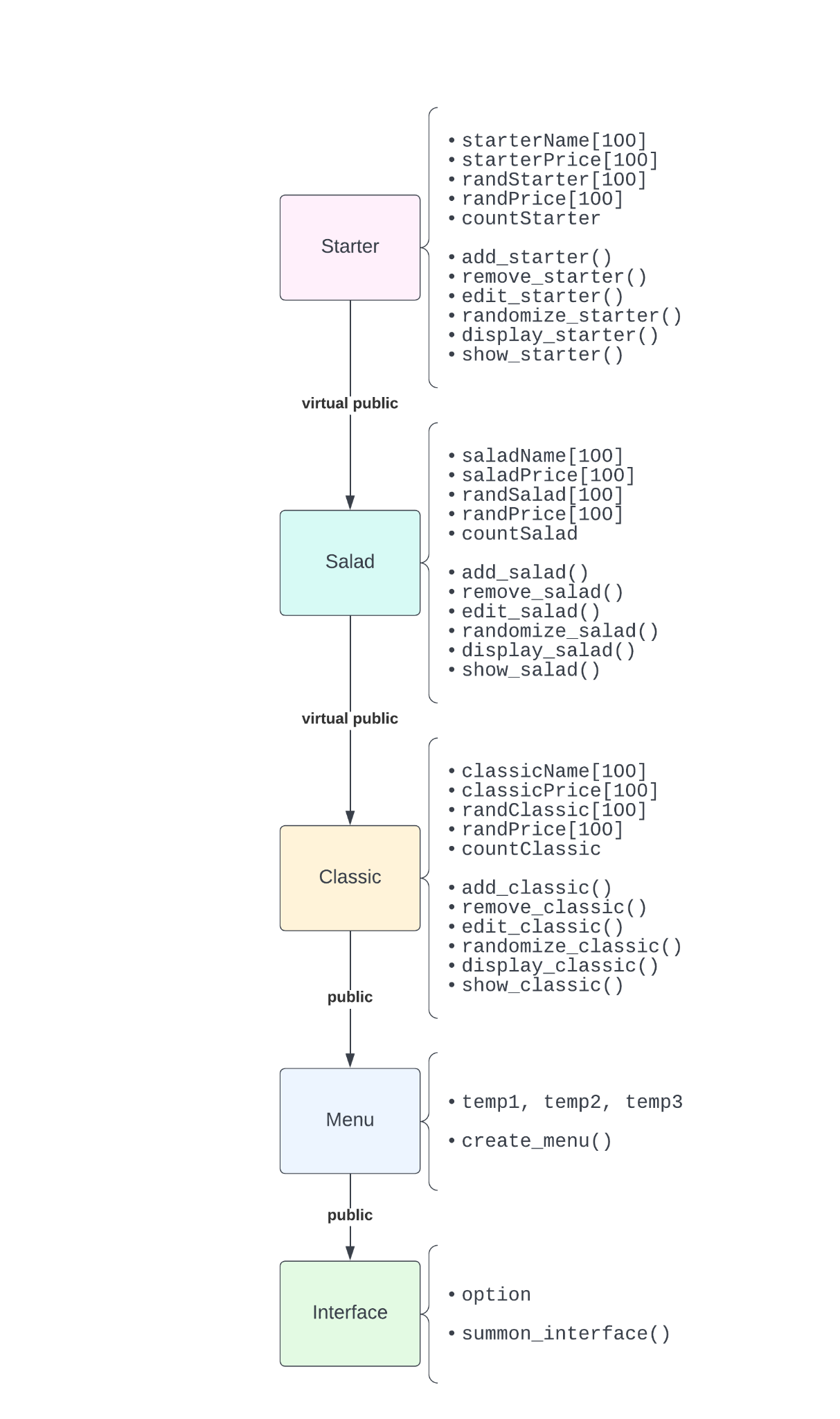


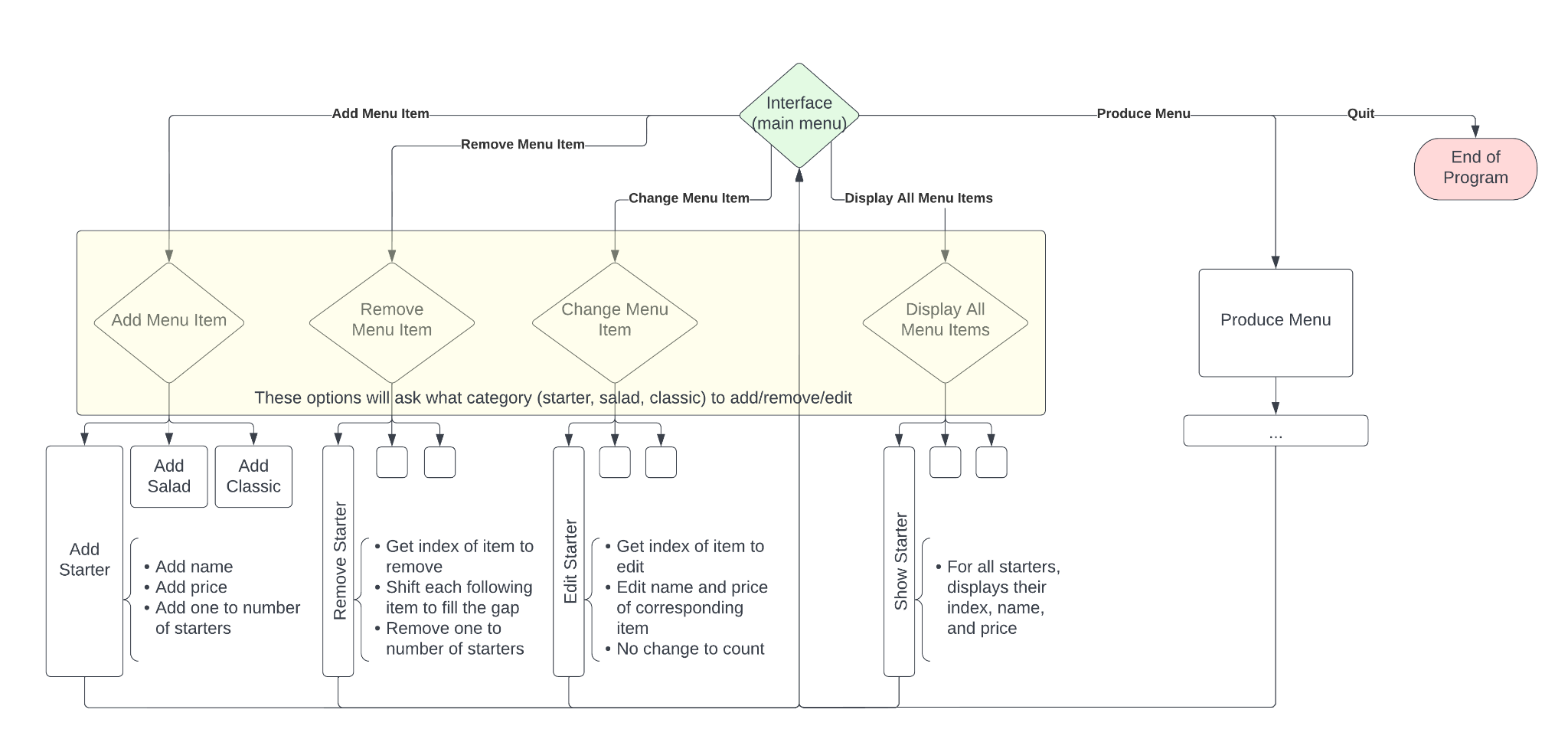


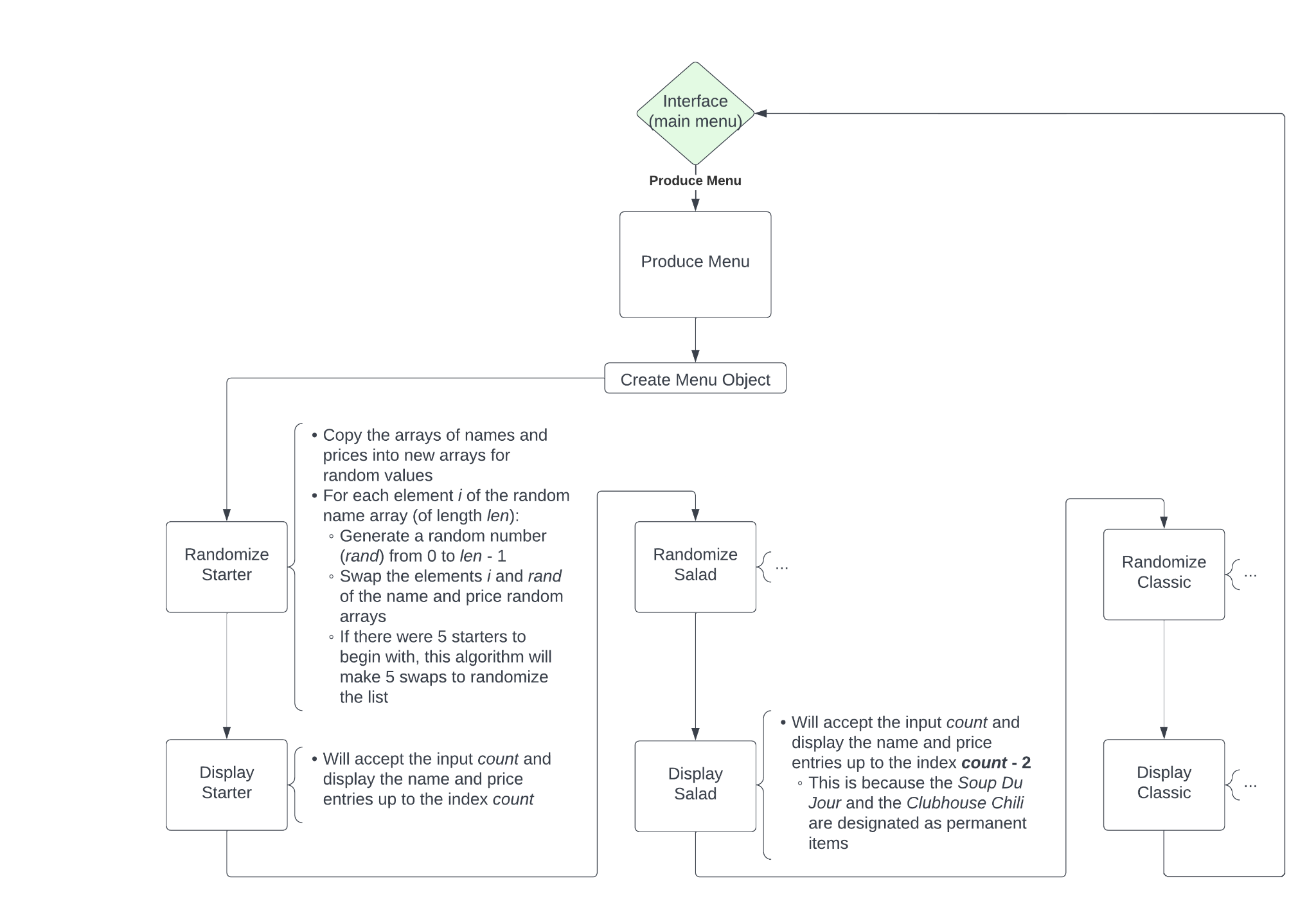
[Classics.h](https://github.com/45aidpre/Menu-Randomizer-Final/blob/master/Menu%20Randomizer%20Final/Classics.h)



4.1.1 Process or Data flow of System







5. Works Cited:

badges, JohnBobSmithJohnBobSmith 16911. “Syntax - How to Catch Invalid Input in C++? - Stack Overflow.” *Stack Overflow*, https://stackoverflow.com/questions/19696442/how-to-catch-invalid-input-in-c. Accessed 1 May 2023.

badges, MarJamRobMarJamRob 3., 39577. “Security - Is the C++ Hash Function Reasonably Safe for Passwords? - Stack Overflow.” *Stack Overflow*, https://stackoverflow.com/questions/16877976/is-the-c-hash-function-reasonably-safe-for-passwords. Accessed 1 May 2023.

badges, NowibananatzkiNowibananatzki 74611. “C++ - Visual Studio Console Program Crashes after Control-c - Stack Overflow.” *Stack Overflow*, https://stackoverflow.com/questions/60014305/visual-studio-console-program-crashes-after-control-c. Accessed 1 May 2023.

Greenwell, Josiah. *Web.Microsoftstream.Com*, https://web.microsoftstream.com/video/f48ac5f7-a367-4881-a994-5188e4382cfe. Accessed 1 May 2023.

Premeau, Aidan. “Menu Randomizer Project.” *Github.Com*, Aidan Premeau, 1 May 2023, https://github.com/45aidpre/Menu-Randomizer-Final.

Tewari, Anubhav. “Vector of Pair in C++.” *OpenGenus IQ: Computing Expertise & Legacy*, OpenGenus IQ: Computing Expertise &amp; Legacy, 7 Oct. 2021, https://iq.opengenus.org/vector-of-pairs-in-cpp/.

TylerMSFT. “Warning C6262 | Microsoft Learn.” *Microsoft Learn: Build Skills That Open Doors in Your Career*, https://learn.microsoft.com/en-us/cpp/code-quality/c6262?view=msvc-170. Accessed 1 May 2023.