BASKETBALL COMMENTARY PROJECT

Security Assessment Report

Version N.1

May 1, 2023

Security Assessment – Basketball Commentary

Table of Contents

1. Summary

The overall goal is to walk the user through a basketball game(commentary), where they can control the player's action, select an artist for half-time by name, and audience reaction using python language.

1. Assessment Scope

The tools used include Python IDEs such as PyCharm and Sublime Text with basic libraries used. And the platforms that it was performed on include Windows and Linux. Meaning it is compatible with most major operating systems. The testing of multiple OSs is needed to see if the same results occur when running the source code. A limitation was not being able to run the source code on the MacOS due to users having more desktop/gaming laptops then MacBook.

2. Summary of Findings

The security issues found include 2 ½ low issues, 1 moderate issue, and 2 high issues. Which is shown in the diagram below and the SWOT diagram shows the updated weakness, strength, etc.

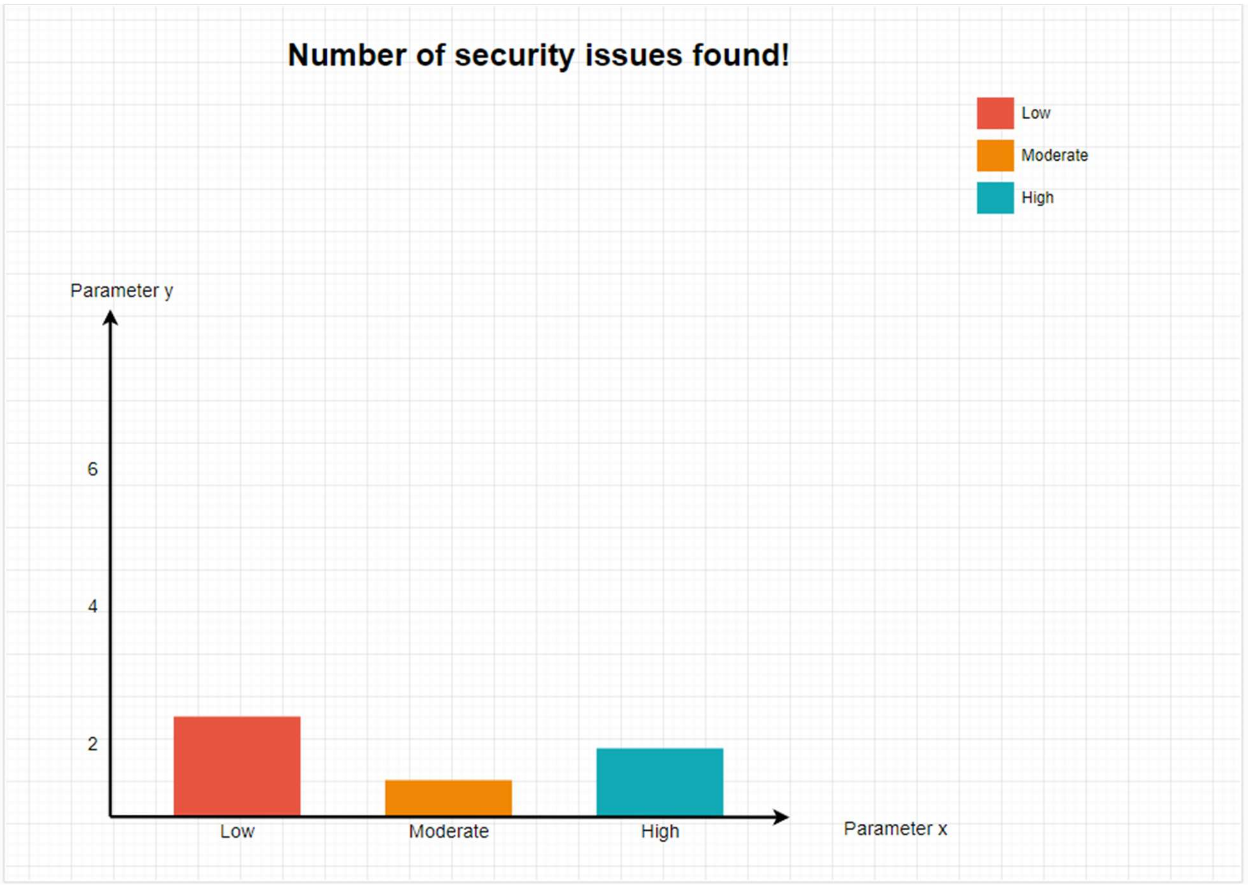
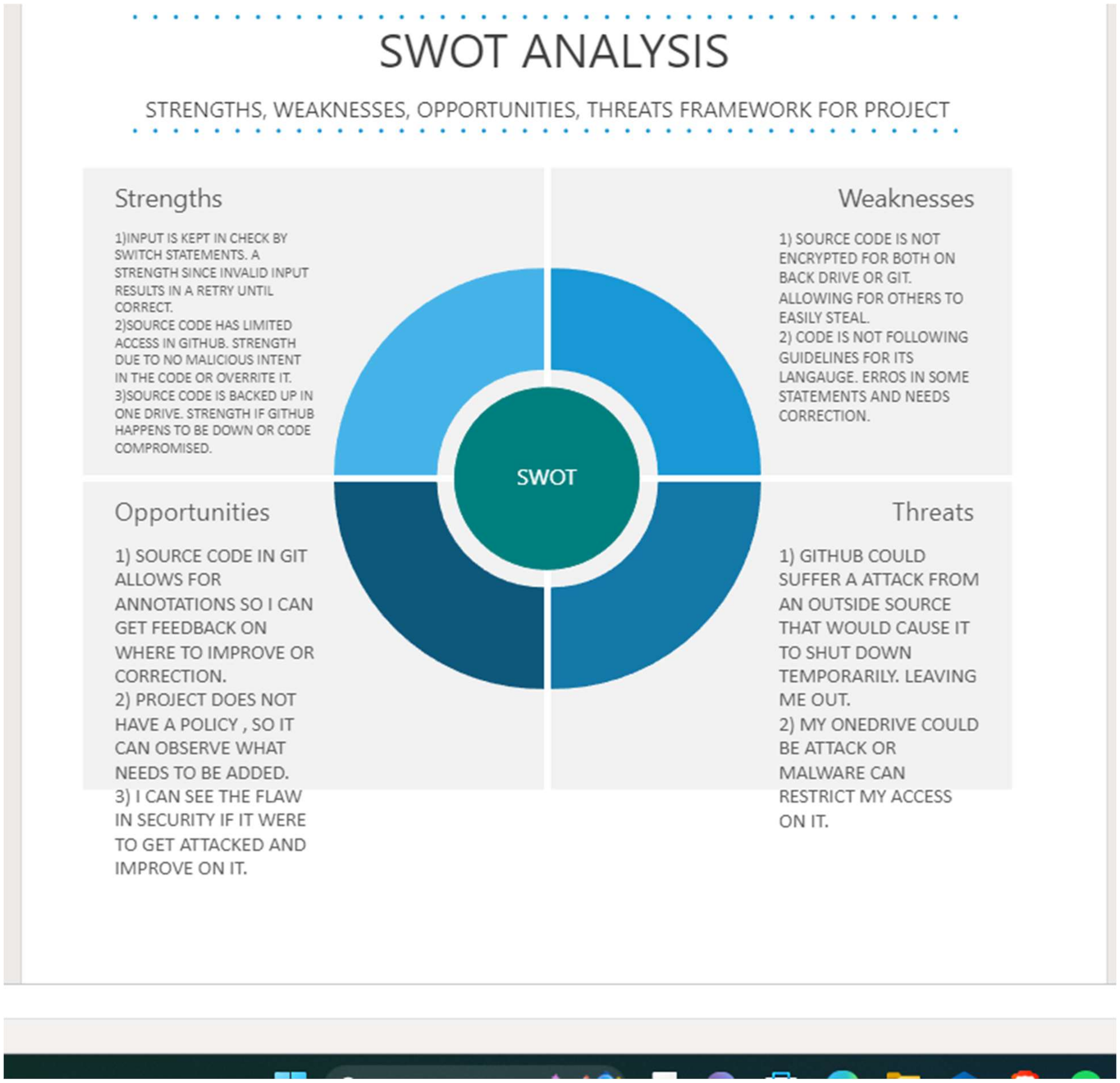


Figure 1. Findings by Risk Level

Security Assessment – Basketball Commentary

The diagram shows the issues encountered when running the source code and where it was hosted. The 2 high issues were the lack of error handling in the code to handle unexpected user inputs. Another issue is the user input variable 'moves and face' are not sanitized or validated, so it could lead to unexpected results or errors. Such as an injection attack if the attacker can insert malicious input into the ser input fields. (one right). Then the 1 moderate issue is the difference variable is calculated from user input, but there is no check to ensure that the inputs are valid integers, leading to type error. Then the 2-3 low issue include access control on the GitHub platform, encryption on the source code, and no backup. Before everyone had access control of read/write but not only selected groups are able to write into the file. And the source code has been encrypted and backed up.



SWOT ANALYSIS

STRENGTHS, WEAKNESSES, OPPORTUNITIES, THREATS FRAMEWORK FOR PROJECT

Strengths

1)INPUT IS KEPT IN CHECK BY SWITCH STATEMENTS. A STRENGTH SINCE INVALID INPUT RESULTS IN A RETRY UNTIL CORRECT.
2)SOURCE CODE HAS LIMITED ACCESS IN GITHUB. STRENGTH DUE TO NO MALICIOUS INTENT IN THE CODE OR OVERRITE IT.
3)SOURCE CODE IS BACKED UP IN ONE DRIVE. STRENGTH IF GITHUB HAPPENS TO BE DOWN OR CODE COMPROMISED.

Weaknesses

1) SOURCE CODE IS NOT ENCRYPTED FOR BOTH ON BACK DRIVE OR GIT. ALLOWING FOR OTHERS TO EASILY STEAL.
2) CODE IS NOT FOLLOWING GUIDELINES FOR ITS LANGAUGE. ERROS IN SOME STATEMENTS AND NEEDS CORRECTION.

Opportunities

1) SOURCE CODE IN GIT ALLOWS FOR ANNOTATIONS SO I CAN GET FEEDBACK ON WHERE TO IMPROVE OR CORRECTION.
2) PROJECT DOES NOT HAVE A POLICY , SO IT CAN OBSERVE WHAT NEEDS TO BE ADDED.
3) I CAN SEE THE FLAW IN SECURITY IF IT WERE TO GET ATTACKED AND IMPROVE ON IT.

Threats

1) GITHUB COULD SUFFER A ATTACK FROM AN OUTSIDE SOURCE THAT WOULD CAUSE IT TO SHUT DOWN TEMPORARILY. LEAVING ME OUT.
2) MY ONEDRIVE COULD BE ATTACK OR MALWARE CAN RESTRICT MY ACCESS ON IT.

Figure 2. SWOT

The SWOT diagram demonstrate the strengths, threats, opportunities, and weaknesses found. Many relating the host platform of GitHub. Such as access control, public settings, and lack of policy. In the other hand, the source code lack encryption, user input check, etc.

3. Summary of Recommendations

The changes made was the addition of a function named "calculate_score" that includes an error handling of try and except. In order to handle the unexpected user input. And the function allows for reusability since before it was individual user input to calculate scores. In addition, the variables move, and face have been validated to ensure no attacker can modify the code if accessed. As for the GitHub platform, I changed the access control to only allow outside people to only read the code rather than full access. And created groups of trustworthy people who can edit the source code. Which allows me to track where the changes were made and trace back to the original group. A change that is still needed is seeing the user input if the code were downloaded and tried on the MacOS because it can lead unexpected errors.

2. Goals, Findings, and Recommendations

1. Assessment Goals

The purpose of this assessment was to do the following:

- Identify any potential vulnerabilities or weakness in the code that could be exploited by attackers.
- Helps ensure the code is secure and it can protect sensitive data and resources from unauthorized access/modification.
- Ensure code meets industry best practices and standards for security and prevent security breaches.

2. Detailed Findings

Ensure each vulnerability is thoroughly explained, specific risks to the continued operations are identified, and the impact of each Threat or Weakness is analyzed as a business case. Ensure these are linked to Table 1 when describing the Risk Value. This is not the fixes – it's the description of the problems found. The fixes go in the next section (for ease of lookup using TOC) - build this off your checklist, SWOT, and risk assessments.

NUMBER OF VULNERABILITIES:	DESCRIPTION OF VULNERABILITIES AND THE DETERMINATION VALUE
1	Code Injection Vulnerability: The code includes user input functions that are not being sanitized, which makes it vulnerable to code injection attacks.

	Meaning hackers can exploit this vulnerability by injection malicious code into the program/execute. Which is in "input" function since it can receive any arbitrary string. Making it a vulnerability and the effect is talked about in the powerpoint, causing access of sensitive data and program errors (1).
2	Lack of Encryption: The code itself is not encrypted in any way. Which could lead to data exposure and hackers can exploit this vulnerability by accessing the data directly. Making it a weakness and the attacker will find it easy to gain access or encypt it themselves and ask for ransom (2).
3	Lack of Data Validation: The code does not perform any data validation or verification which could lead to SQL injection and can be exploited by providing unexpected input if it were to be hosted online. If the attacker were to input wrong input, then it can lead to access to sensitive data through injection and gain access to modify data (3). Making it a vulnerability to the code.
4	No authentication or authorization: The code and the GitHub did not implement any authorization mechanism to restrict access to sensitive resources. Which can lead to unauthorized access to the source code/data. Which is why we must identify the actor and have access control set up and is weakness (4).
5	Buffer Overflow: The code does not have any measure to prevent buffer overflow attacks. So, the hackers can exploit this vulnerability by overflowing the buffer with data that can lead to program crashes, corruption (5).
6	No input validation: The code does not validate the input data type or format, so it is vulnerable to data injection attacks. And lead to valueError exception for the 'int' function if the input data is not an integer (6). So, it is vulnerable to the arttacks of wrong data type.
7	Lack Of Policy: There is no policy in check for others if they were to take my code or use as reference. Also lack of following guidelines. Such as the MIT license is not in the GitHub which helps protect you if others were to take the code (7).
8	Lack of error handling: The code does not implement proper error handling mechanisms which can cause unexpected crashes or undesirable consequences. So, if the user were to put an unexpected input the program may raise an exception code (8). A weakness to the code and needs attention.

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).

NUMBER	DESCRIPTION ON HOW TO FIX
1	With the program containing many "lack of input validation", leading to potential security risk such as injection attacks. To mitigate this vulnerability, I would validate and sanitize the user inputs. Such as 'isnumeric()' method to ensure that only numeric values are being entered for 'num1' and 'num2' variables.

2	To fix the error handling issues I added the use of try-except blocks to catch and handle errors that may occur during runtime.
3	I as well added proper comments and documentation to the code. As for policy I added the "MIT" license if the source code were to be distributed.
4	I as well added validation to user input when doing mathematical calculations to handle any errors that may occur.
5	As for security measure I grouped people into groups in GitHub, so I am able to trace where the last edit was made and see who it was. In addition, they have access to edit rather than the general public who only have read access.
6	I separated some parts of the program due to lack of modularity and easier to read. And the original file is not encrypted and back-up in another drive.

3. Methodology for the Security Control Assessment

3.1.1 Risk Level Assessment (delete this text: you don't have to change 3.1.1)

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 **Error! Reference source not found.** 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".

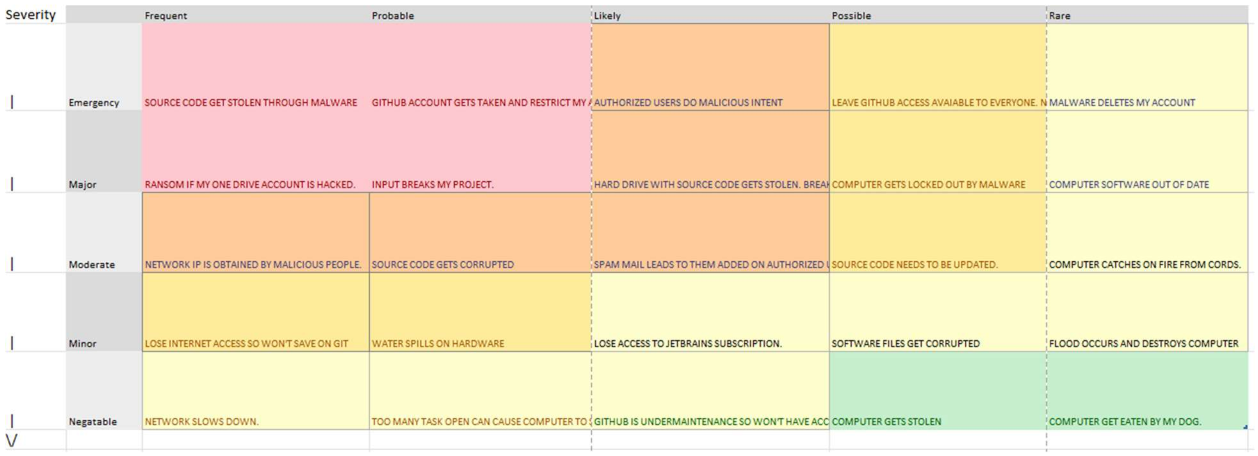
Table 1 - Risk Values

Rating	Definition of Risk Rating
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

Rating	Definition of Risk Rating
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

Rating	Definition of Risk Rating
	• 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



Severity		Frequent	Probable	Likely	Possible	Rare
|	Emergency	SOURCE CODE GET STOLEN THROUGH MALWARE	GITHUB ACCOUNT GETS TAKEN AND RESTRICT MY A	AUTHORIZED USERS DO MALICIOUS INTENT	LEAVE GITHUB ACCESS AVAIABLE TO EVERYONE. N	MALWARE DELETES MY ACCOUNT
|	Major	RANSOM IF MY ONE DRIVE ACCOUNT IS HACKED.	INPUT BREAKS MY PROJECT.	HARD DRIVE WITH SOURCE CODE GETS STOLEN. BREAK	COMPUTER GETS LOCKED OUT BY MALWARE	COMPUTER SOFTWARE OUT OF DATE
|	Moderate	NETWORK IP IS OBTAINED BY MALICIOUS PEOPLE.	SOURCE CODE GETS CORRUPTED	SPAM MAIL LEADS TO THEM ADDED ON AUTHORIZED U	SOURCE CODE NEEDS TO BE UPDATED.	COMPUTER CATCHES ON FIRE FROM CORDS.
|	Minor	LOSE INTERNET ACCESS SO WON'T SAVE ON GIT	WATER SPILLS ON HARDWARE	LOSE ACCESS TO JETBRAINS SUBSCRIPTION.	SOFTWARE FILES GET CORRUPTED	FLOOD OCCURS AND DESTROYS COMPUTER
|	Negatable	NETWORK SLOWS DOWN.	TOO MANY TASK OPEN CAN CAUSE COMPUTER TO S	GITHUB IS UNDERMAINTENANCE SO WON'T HAVE ACC	COMPUTER GETS STOLEN	COMPUTER GET EATEN BY MY DOG.
V						

Overall, the findings were mainly correct. As the source code main errors were in handling errors which was a high risk. Due to attackers can exploit it with invalid inputs and the fix was moderately difficult because I had to remove all the individual inputs. And make its own function with a try-except block. This way it is reusable and can handle all errors. Then another accurate finding was the access control with a moderate risk. Due to the general users had access to all the code and were able to modify. But it had a rating of easy due to only had to adjust the settings and separate people with links to certain groups. Those groups may have access to edit but I am able to see who made the last changes. Another moderate risk was the source code not being encrypted and it was an easy risk because I encrypted it in drive. All the other finding were on par with the assessment ratings and only require small fixes.

3.1.2 Tests and Analyses

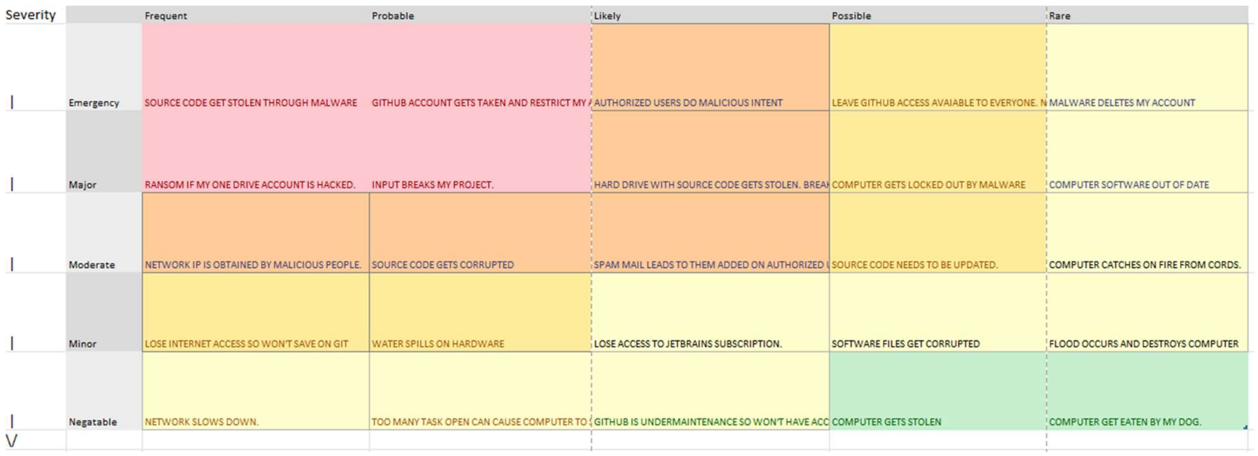
The risk assessment was a great first step in identifying and analyzing the risks that the organization or system faces. Which involved me identifying the threats, vulnerabilities, and opportunities. Paving the way of what my scope and objectives for security control. The penetration testing process was helpful in testing the security controls of the program. With white box testing: the user found out that it only consisted of the main body function. And that it was best take out input functions and separate it into its own ‘def function’ for readability and reusability. And noticed the lack of try-except block for user input. In addition, that the GitHub access controls were not optimized at all. Then for the grey box testing: the tester knew some knowledge of the structured and argued that buffer overflow was a concern when the tester attempted to insert other data type values instead of int values for ‘score_calculate’ function. Then during black box testing: the tester found many flaws in the input because it crashed the program and threw handling errors. And how the program lines were not documented well and lack of policy.

3.1.3 Tools

The use of Linux command was a great help when wanting to encrypt the files because of the commands being readily available and can put our own password. Also, it allowed me to user who had read/write access to it through ‘-ls’ and I later changed the access. Then GitHub setting gave me control on who can access and if I wanted to make it private. Great help in wanting more access control. Seeing the security policy review showed that I lacked a policy. Another tool was PyCharm since this is where the tester would input their choice and PyCharm would throw out errors with reference on where it occurred.

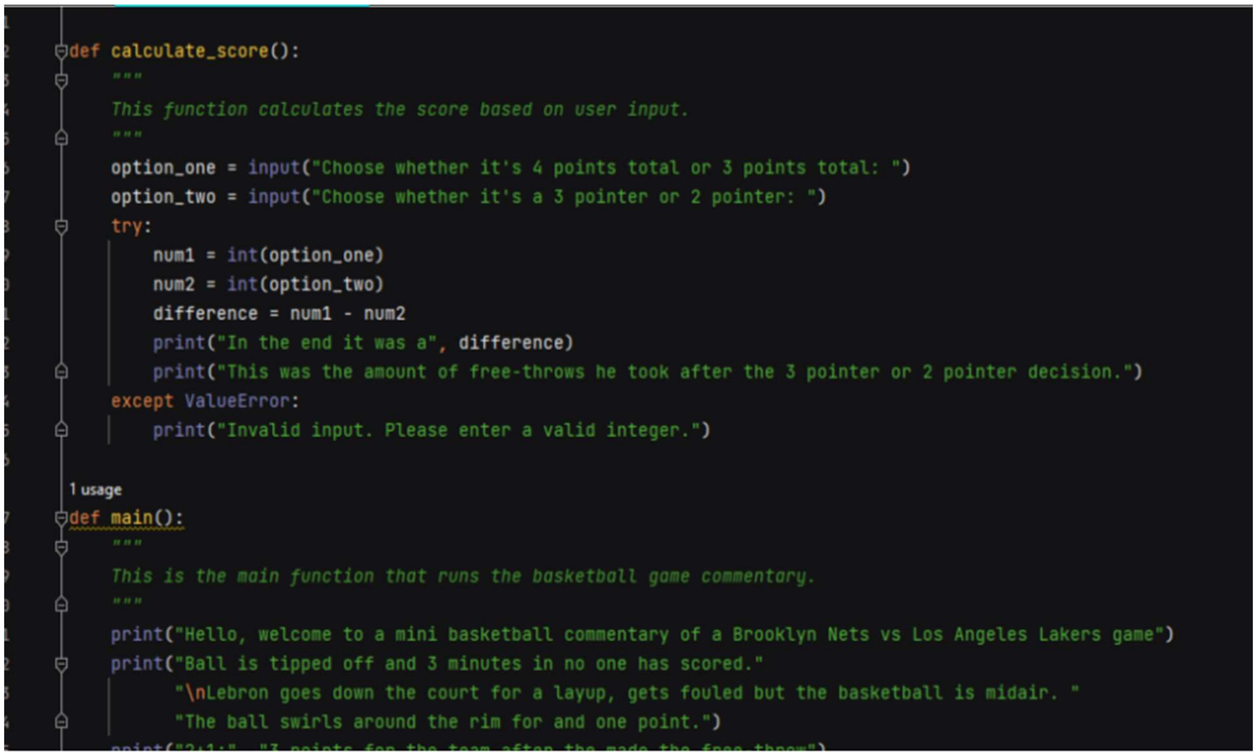
4. Figures and Code

Insert any pictures here (including of major code issues or code that was used as a tool – can just screenshot and add link to github). This section must include at least 4 figures or code portions:



Severity		Frequent	Probable	Likely	Possible	Rare
|	Emergency	SOURCE CODE GET STOLEN THROUGH MALWARE	GITHUB ACCOUNT GETS TAKEN AND RESTRICT MY	AUTHORIZED USERS DO MALICIOUS INTENT	LEAVE GITHUB ACCESS AVAIABLE TO EVERYONE. N	MALWARE DELETES MY ACCOUNT
|	Major	RANSOM IF MY ONE DRIVE ACCOUNT IS HACKED.	INPUT BREAKS MY PROJECT.	HARD DRIVE WITH SOURCE CODE GETS STOLEN. BREA	COMPUTER GETS LOCKED OUT BY MALWARE	COMPUTER SOFTWARE OUT OF DATE
|	Moderate	NETWORK IP IS OBTAINED BY MALICIOUS PEOPLE.	SOURCE CODE GETS CORRUPTED	SPAM MAIL LEADS TO THEM ADDED ON AUTHORIZED	SOURCE CODE NEEDS TO BE UPDATED.	COMPUTER CATCHES ON FIRE FROM CORDS.
|	Minor	LOSE INTERNET ACCESS SO WON'T SAVE ON GIT	WATER SPILLS ON HARDWARE	LOSE ACCESS TO JETBRAINS SUBSCRIPTION.	SOFTWARE FILES GET CORRUPTED	FLOOD OCCURS AND DESTROYS COMPUTER
|	Negatable	NETWORK SLOWS DOWN.	TOO MANY TASK OPEN CAN CAUSE COMPUTER TO	GITHUB IS UNDERMAINTENANCE SO WON'T HAVE ACC	COMPUTER GETS STOLEN	COMPUTER GET EATEN BY MY DOG.
V						

RISK ASSESTMENT: Tool used to identify what needed attention and where to start.

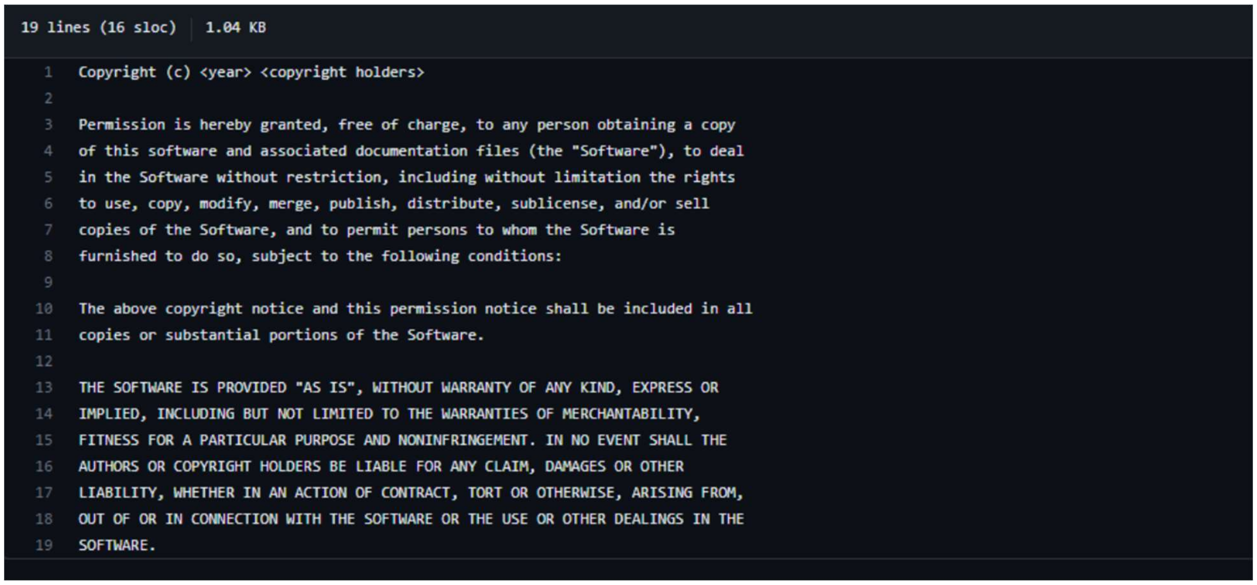


```python
def calculate_score():
    """
    This function calculates the score based on user input.
    """
    option_one = input("Choose whether it's 4 points total or 3 points total: ")
    option_two = input("Choose whether it's a 3 pointer or 2 pointer: ")
    try:
        num1 = int(option_one)
        num2 = int(option_two)
        difference = num1 - num2
        print("In the end it was a", difference)
        print("This was the amount of free-throws he took after the 3 pointer or 2 pointer decision.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")

1 usage
def main():
    """
    This is the main function that runs the basketball game commentary.
    """
    print("Hello, welcome to a mini basketball commentary of a Brooklyn Nets vs Los Angeles Lakers game")
    print("Ball is tipped off and 3 minutes in no one has scored."
          "\nLebron goes down the court for a layup, gets fouled but the basketball is midair. "
          "The ball swirls around the rim for and one point.")
    print("2:1:", "3 points for the team after the made the free throw")
```

CODE CHANGE: The implementation of function calculate_score() created for reusability. And inside it had the try-except to handle any handling errors.

Security Assessment – Basketball Commentary

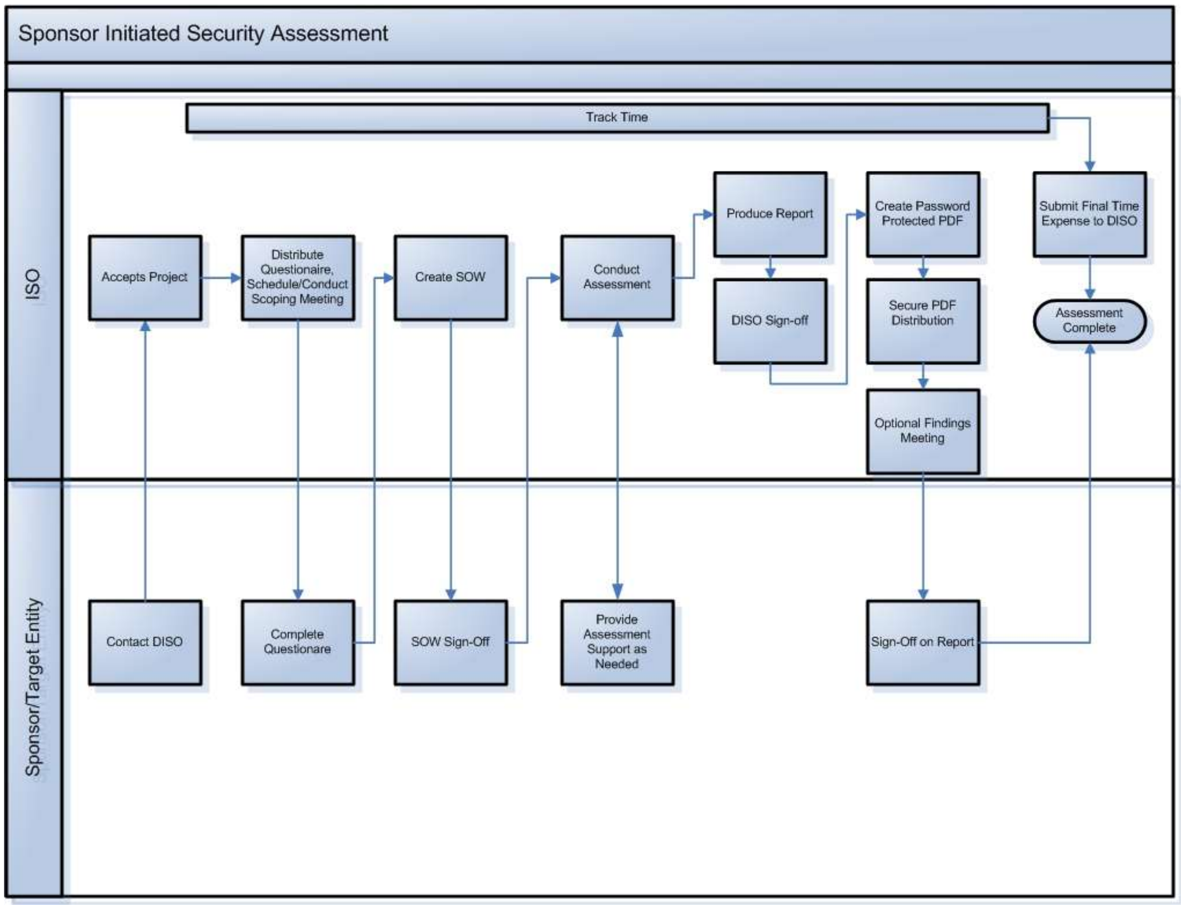


```
19 lines (16 sloc)  |  1.04 KB
```

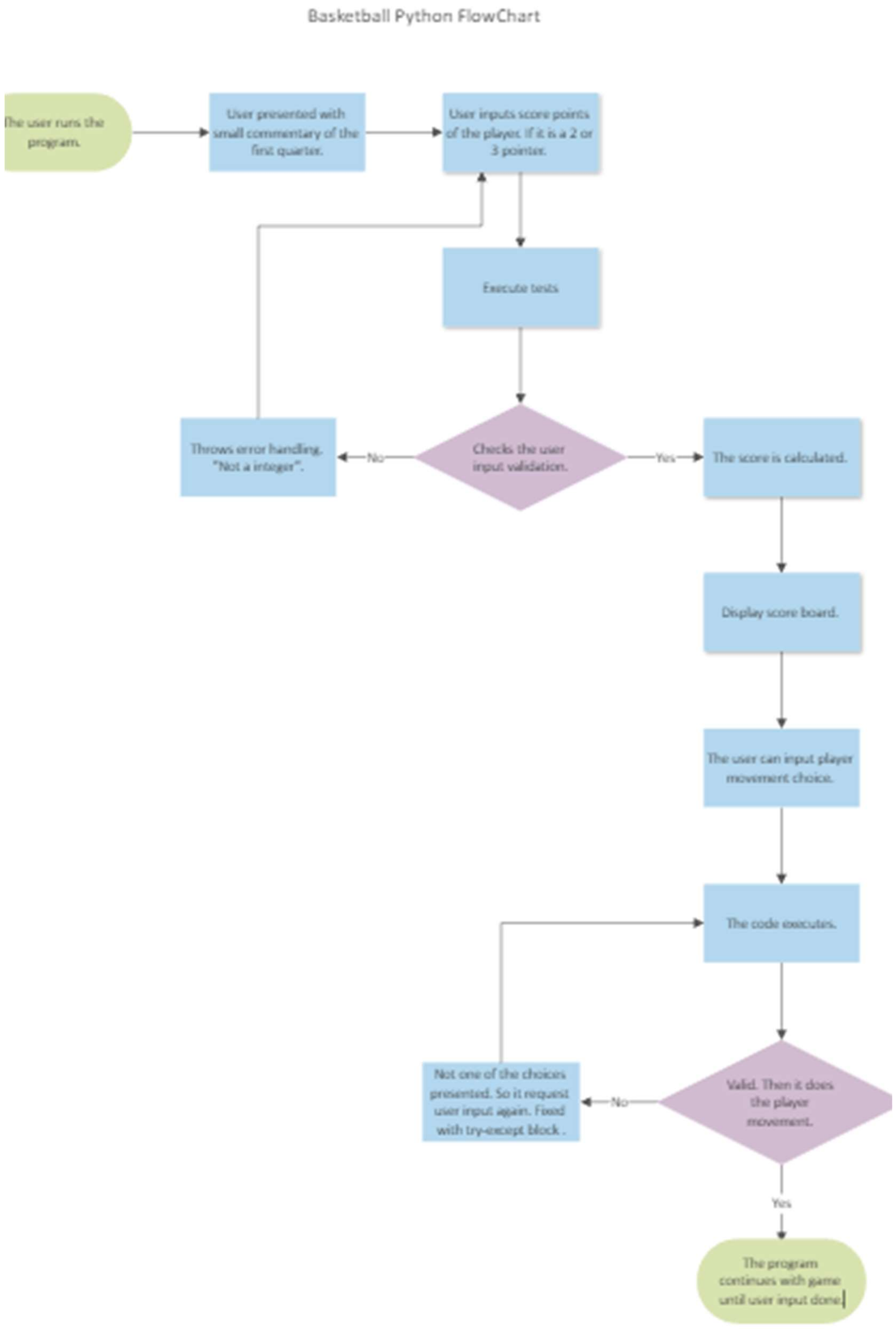
MIT LICENSE: Was added for GitHub if source code was distributed elsewhere.

ENCRYPTION: The source code was back-up in a encrypted hard drive.

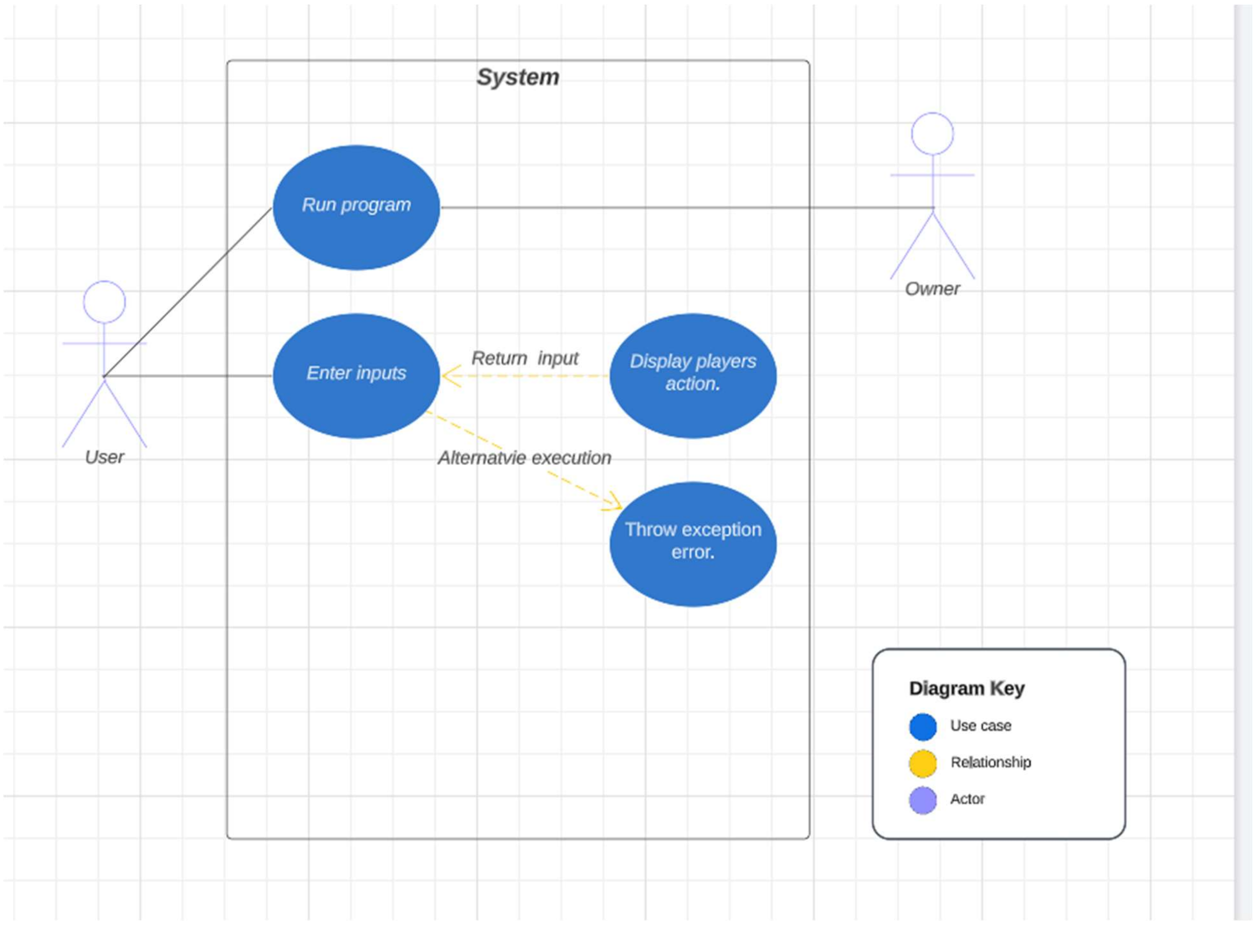
4.1.1 Process or Data flow of System (this one just describes the process for requesting), use-cases, security checklist, graphs, etc.


Sponsor Initiated Security Assessment
ISO
Sponsor/Target Entity
Track Time
Accepts Project
Distribute Questionaire, Schedule/Conduct Scoping Meeting
Create SOW
Conduct Assessment
Produce Report
DISO Sign-off
Create Password Protected PDF
Secure PDF Distribution
Optional Findings Meeting
Submit Final Time Expense to DISO
Assessment Complete
Contact DISO
Complete Questionare
SOW Sign-Off
Provide Assessment Support as Needed
Sign-Off on Report

The user gets in contact with the sponsor on an agreed ‘target entity’ that needs attention to fix. Once reach a mutable agreement they accept the project. From here we create questionnaires, interview, weekly meetings, and others. To get an understanding of what the client wants. Or the what the product should look like, and it would require multiple meetings to get a defined requirement. Once I gathered data on what is needed, I create SOW and have it signed off. Then we conduct an assessment on the system that includes security, guidelines, regulations, etc. Then produce a report on what it is needed to change and have it sign off. From here to keep information confidential between the two we create password to protect, and we can use asymmetric encryption where it would require our unique keys to gain access. Or use certificate once mailing to see the file has not been tampered with. Then the assessment is complete.

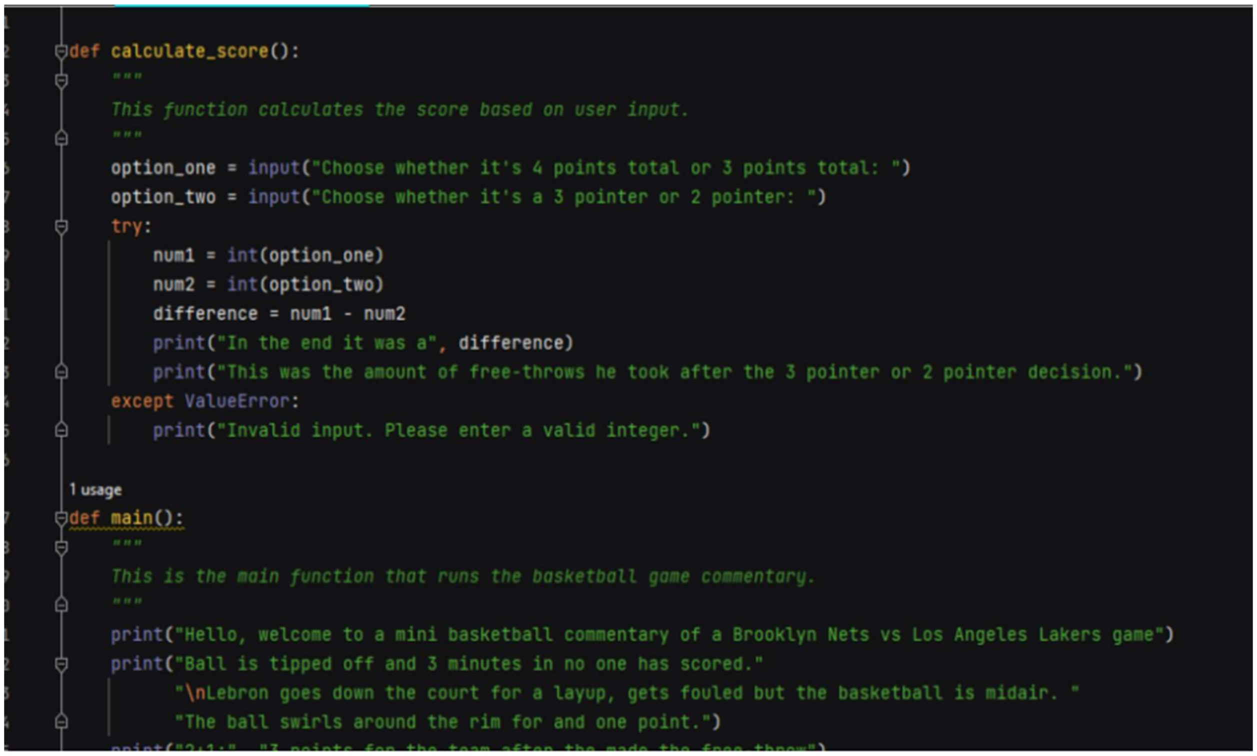

Basketball Python FlowChart
The user runs the program.
User presented with small commentary of the first quarter.
User inputs score points of the player. If it is a 2 or 3 pointer.
Execute tests
Checks the user input validation.
No
Throws error handling. "Not a integer".
Yes
The score is calculated.
Display score board.
The user can input player movement choice.
The code executes.
Valid. Then it does the player movement.
No
Not one of the choices presented. So it request user input again. Fixed with try-except block .
Yes
The program continues with game until user input done.

Flowchart: On the program input and output.



Use Case Diagram: On the user/owner interaction with program.

4.1.2 Other figure of code



```python
def calculate_score():
    """
    This function calculates the score based on user input.
    """
    option_one = input("Choose whether it's 4 points total or 3 points total: ")
    option_two = input("Choose whether it's a 3 pointer or 2 pointer: ")
    try:
        num1 = int(option_one)
        num2 = int(option_two)
        difference = num1 - num2
        print("In the end it was a", difference)
        print("This was the amount of free-throws he took after the 3 pointer or 2 pointer decision.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")

1 usage
def main():
    """
    This is the main function that runs the basketball game commentary.
    """
    print("Hello, welcome to a mini basketball commentary of a Brooklyn Nets vs Los Angeles Lakers game")
    print("Ball is tipped off and 3 minutes in no one has scored."
          "\nLebron goes down the court for a layup, gets fouled but the basketball is midair. "
          "The ball swirls around the rim for and one point.")
    print("2+1:", "3 points for the team after the made the free-throw")
```

5. Works Cited

Security Assessment – Basketball Commentary

1. Greenwell, Josiah, March 27, 2023, Code Injection

2. Greenwell, Josiah, February 20, 2023, Encryption Introduction

3. SQL Injection. TryHackMe. (n.d.). Retrieved May 2, 2023, from https://tryhackme.com/room/sqlinjectionlm

4. Greenwell, Josiah, January 18, 2023, Physical Security and SDL

5. Greenwell, Josiah, March 20, 2023, Software Part1

6. Validation. (n.d.). https://cisserv1.towson.edu/~cssecinj/modules/cs0/cs0-input-validation-python/.

7. MIT. (n.d.-b). https://opensource.org/license/mit/.

8. Python Docs. (n.d.). https://docs.python.org/3.4/c-api/exceptions.html.