**Professionalism and Employability**

**Testing and Vulnerabilities**

**Adam May – 17129303**

**Adam Lewis - 17021294**

**Jake Bulmer - 17144132**

**Liam Newton - 17013658**

**Vulnerabilities**

**Authentication Vulnerabilities**

Authentication vulnerabilities can be used by hackers to find loops within the user login stage. Having a secure login page is common practice and vital to the security of a company’s data. Studies show that an alarming amount of people use the password “password” as their password, along with other overly simple phrases such as their birthday or a name. It is estimated that between 15% and 50% of passwords fall into this category[[1]](#footnote-1) and because of this, hackers can easily guess passwords and gain sensitive information.

If users within the company refrain from using simple passwords or the company provide passwords for the users, then the risk of authentication vulnerabilities arising should be reduced. It is also vital that each stage of the authentication process works correctly, if a person can enter a wrong password and still gain access to the home page then the authentication process has failed, and vulnerabilities arise.

**SQL Injection**

An SQL injection is where computer hackers interfere with database queries that an application makes. When they do this correctly, it allows the hacker to see sensitive information that they otherwise would not have been able to access. This could leave systems extremely vulnerable if a login database containing employee’s usernames and passwords were compromised, giving the hackers the ability to essentially bypass that level of security relatively easily.

A common way to combat this type of vulnerability is through defensive coding practices. It is found that the root cause of SQL injection vulnerabilities is insufficient input validation.[[2]](#footnote-2) An example of a defensive coding practice is input type checking. Input type checking is the practice of checking that a string or an integer has the correct type[[3]](#footnote-3) in doing so you increase the security of the information stored within it. With a better understanding of defensive coding practices and more attention to detail we can reduce the risk of SQL injection attack.

**Insecure Cryptographic Storage**

When information, files or data are not stored securely, they become vulnerable. Cryptographic storage is a method of securely storing something behind a wall of encryption. There are many different encryption algorithms available for people to use to secure their data, but sometimes people may use the wrong algorithms in the wrong context or for the wrong data type. Insecure Cryptographic Storage occurs when this happens, and a hacker targets the implementation or the algorithm itself.[[4]](#footnote-4)

Fortunately, there are some easy countermeasures that you can take to ensure that your encryption works well. An example of one of these is that you should only use approved public algorithms such as AES, RSA and public key. Another example of a way to decrease the vulnerability of the encryption would be to use salting hashing techniques with your own logical operations implemented. Salt is random data that is used an addition input to a verification system. Applying this technique to passwords will make them more secure[[5]](#footnote-5) and keep your data safe from hackers.

**Format String Vulnerability**

String formats are used to specify what datatype is stored within the string. Although many other datatypes exist, and the vulnerability applies to all of them. Float is a datatype used to store numbers with decimal points and integers or “int” are commonly used to store whole numbers[[6]](#footnote-6). When you use an int to store a float, issues can arise.

Format String Vulnerabilities occur when incorrect string formats are used to store data. Hackers can then overload these formats, crashing systems and cause memory leaks which will in turn expose all the data found within the compromised string.[[7]](#footnote-7) To prevent or greatly reduce the threat from hackers you must ensure that you use the correct type of format for the string that you are using.

**Buffer Overflow**

A data buffer is a temporary holder in memory within the RAM or disk, which data can be put in until it can then be processed at a later point when is available. Buffers are used mainly for input devices to dump the data they input to then be processed and sent to an output device. These can be used as vulnerability in the code as hackers can cause the buffer to overflow so therefore it will then fall into the memory holder next to the one being used as the buffer. Overflowing into the next memory holder can cause the data within that data holder to be overwritten and cause breaks within the software/code.

One of the main countermeasures which can be used to prevent buffer overflows are StackGuards.[[8]](#footnote-8) A StackGuard causes the system to detect when there is a buffer overflow about to begin and then to delete whatever process it might be to prevent the software/code running into any errors. However, if this method is used then the process that was running will be immediately deleted and unrecoverable. Furthermore, there are multiple types of StackGuards depending on what provider of StackGuards is used. The different StackGuards main differences is that one is more secure and the other one is more performant.

**Code Injection**

A vulnerability which can be found within software and code is data injection. Data injection is where a hacker/third party has injected the data that a user has required with some nasty code which can cause loops within their computer or even viruses. The worst thing about this time of attack as if you are not very familiar with computers then it is very hard to spot if your data has been infected by dangerous code.[[9]](#footnote-9) This type of attack if used on a business scale can then cause the hacker or third-party access to their servers in some cases and can cause data be taken from them or just shut down of the servers completely.

A way that code injection can be prevented is by using a random instruction set. As the attacker will need to know the key to the randomizer algorithm.[[10]](#footnote-10) If an attacker still attempts to attach code even without knowing the key will cause runtime exception. A runtime exception is where this will break down or crash the software/data when they occur. A runtime exception will then prevent the buffer from overflowing and overwriting into any other memory pointers.

**Timing attack**

A timing attack is another vulnerability due to hackers being able to attack the cryptosystem. The attacker can get some data from the computer however, there are quite a few variables which will determine how much data the attacker will gain access to. Some of these variables are the how strong the systems CPU is, the algorithms being used and the design of the whole cryptosystem. The timing attack works by analysing the time it takes for the computer to execute algorithms. So, during the time that it is taking for the computer to execute the algorithms the hacker will attack and take data from the workspace.

One countermeasure that can be used to prevent any type of timing attack is implementing a cryptography circuit[[11]](#footnote-11). This is due to a lot of data being quickly stored within memory this will disguise the amount of data being used.

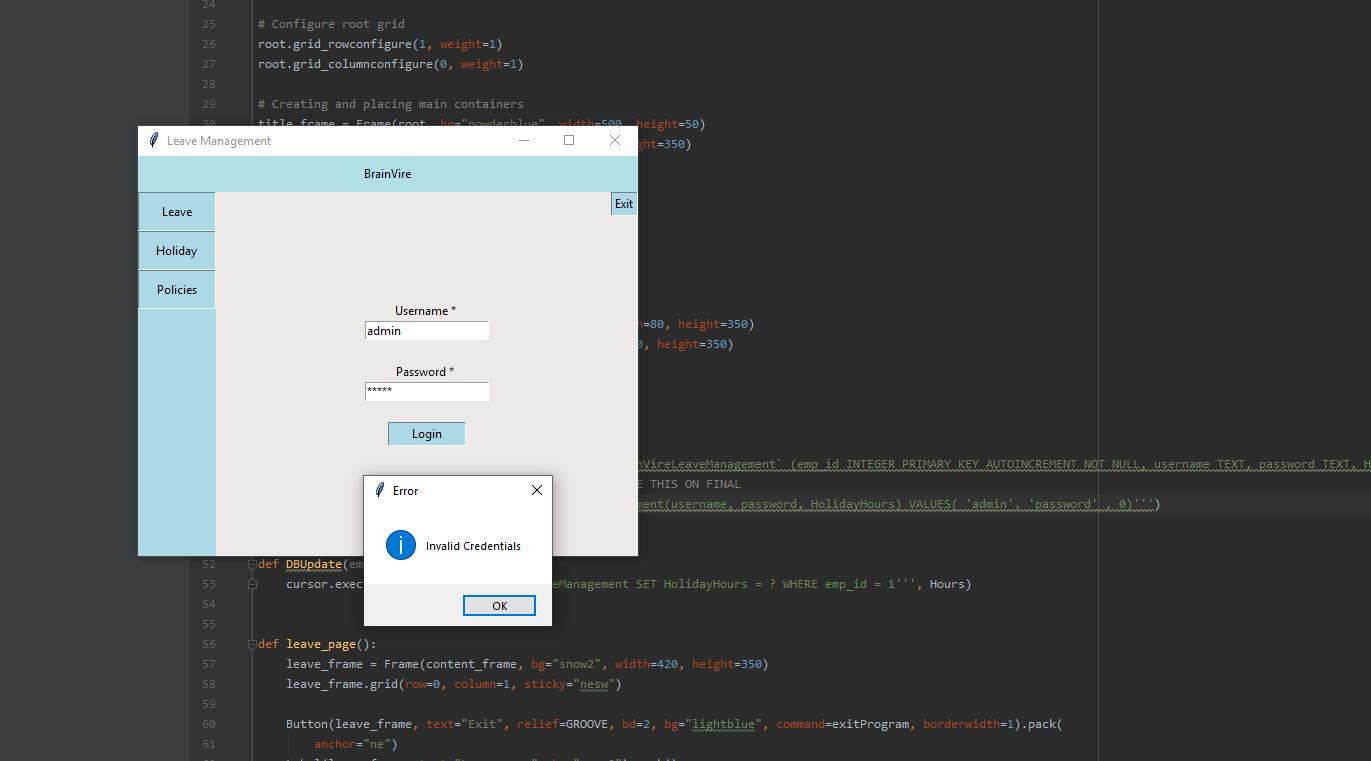
**Testing – GUI**

**Login Functionality**

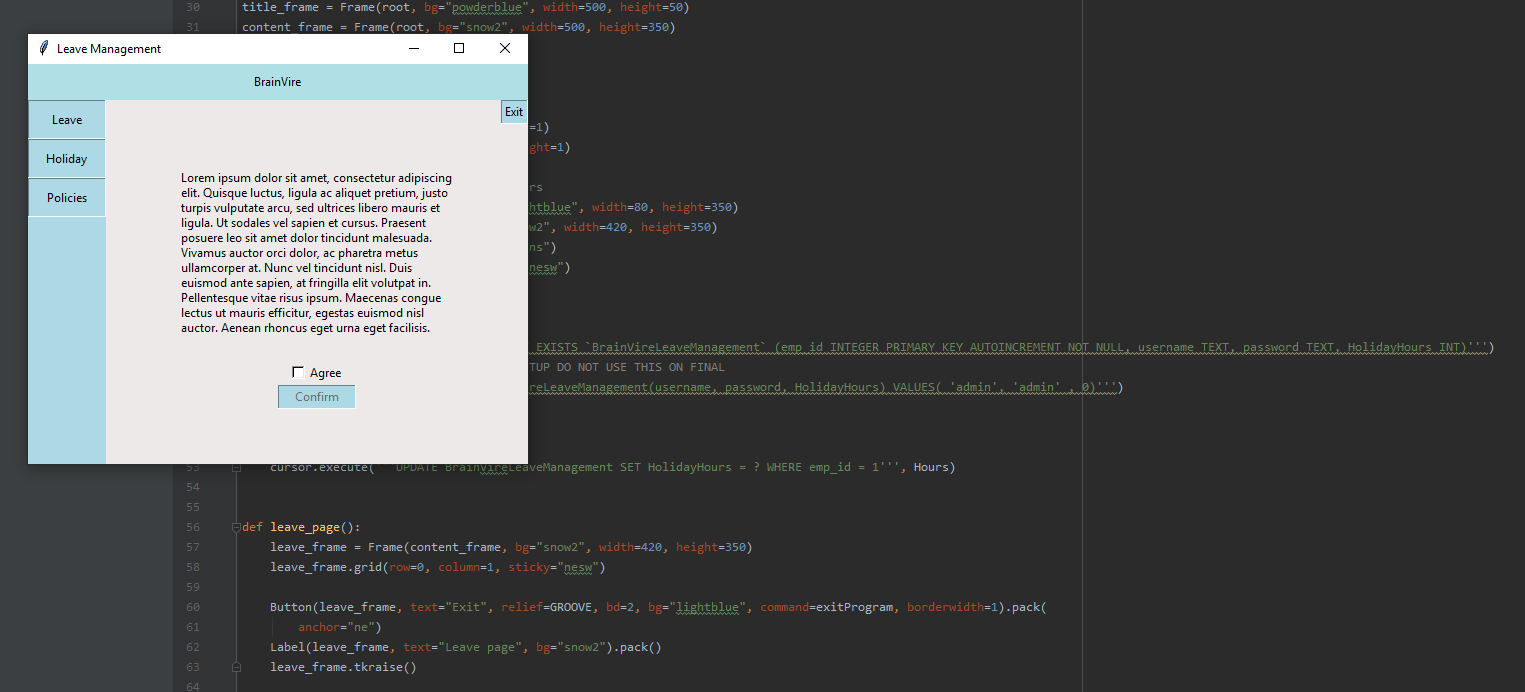
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test No. | What is being entered | Expected Result | Outcome | Proof |
| 1 | Valid Username and password | Allows me to continue to the next page | Allowed me to carry on to the policies | Appendix 1 |
| 2 | Valid username but invalid password | Gives me an error telling me credentials are invalid | Gave me an error | Appendix 2 |
| 3 | Invalid username but valid password | Gives me an error telling me credentials are invalid | Gave me an error | Appendix 3 |
| 4 | Invalid username and invalid password | Gives me an error telling me credentials are invalid | Gave me an error | Appendix 4 |

**SQL testing**

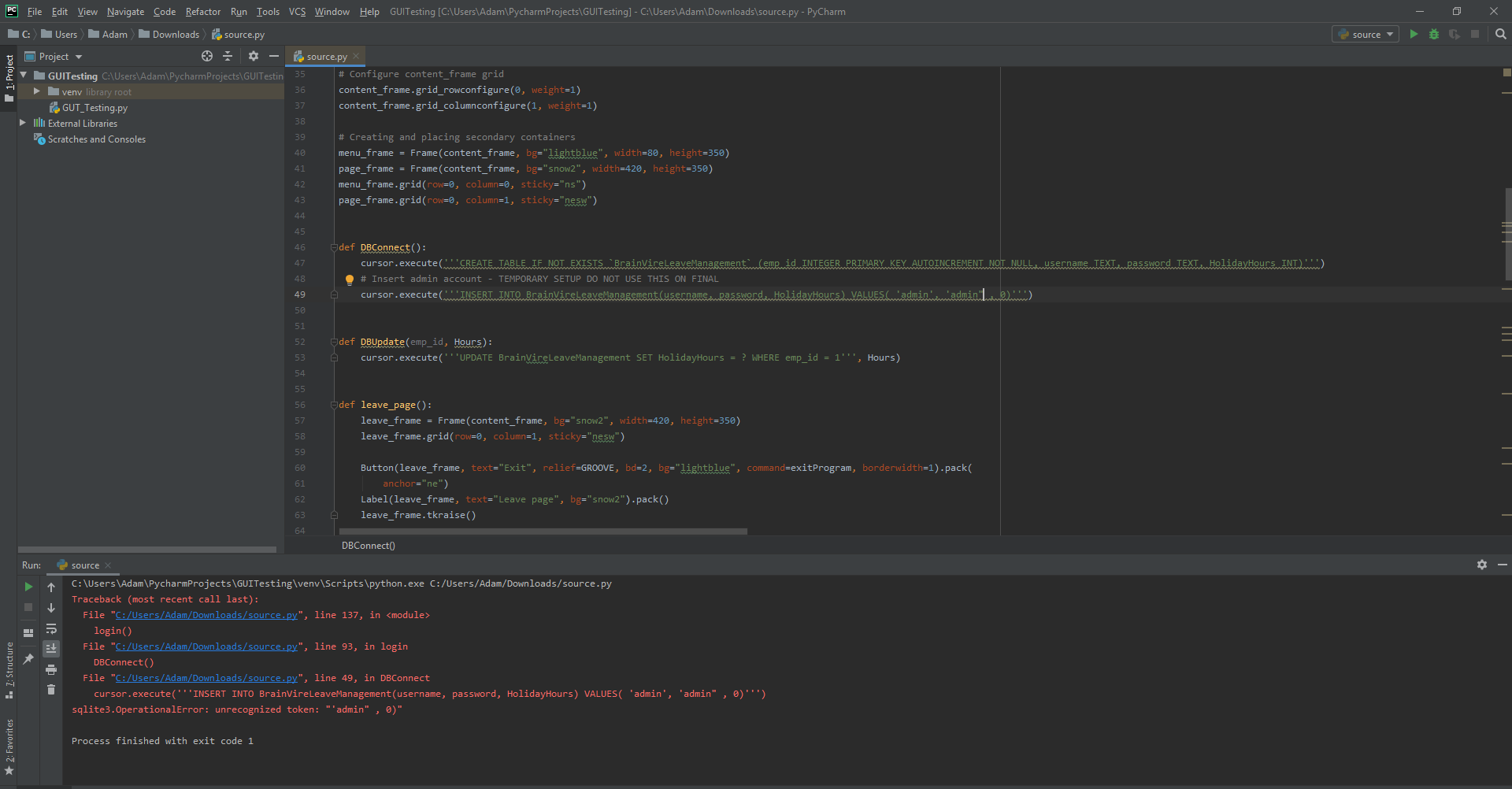


By changing the required passcode, currently hardcoded into the program, from ‘admin’ to ‘password’ the admin user no longer authenticates with the details, ‘admin’ ‘admin’ for the username and password.

Reverting the change back to ‘admin’ allows the admin user to authenticate with the correct password. This simple test proves that the correct login details are required to access the program.

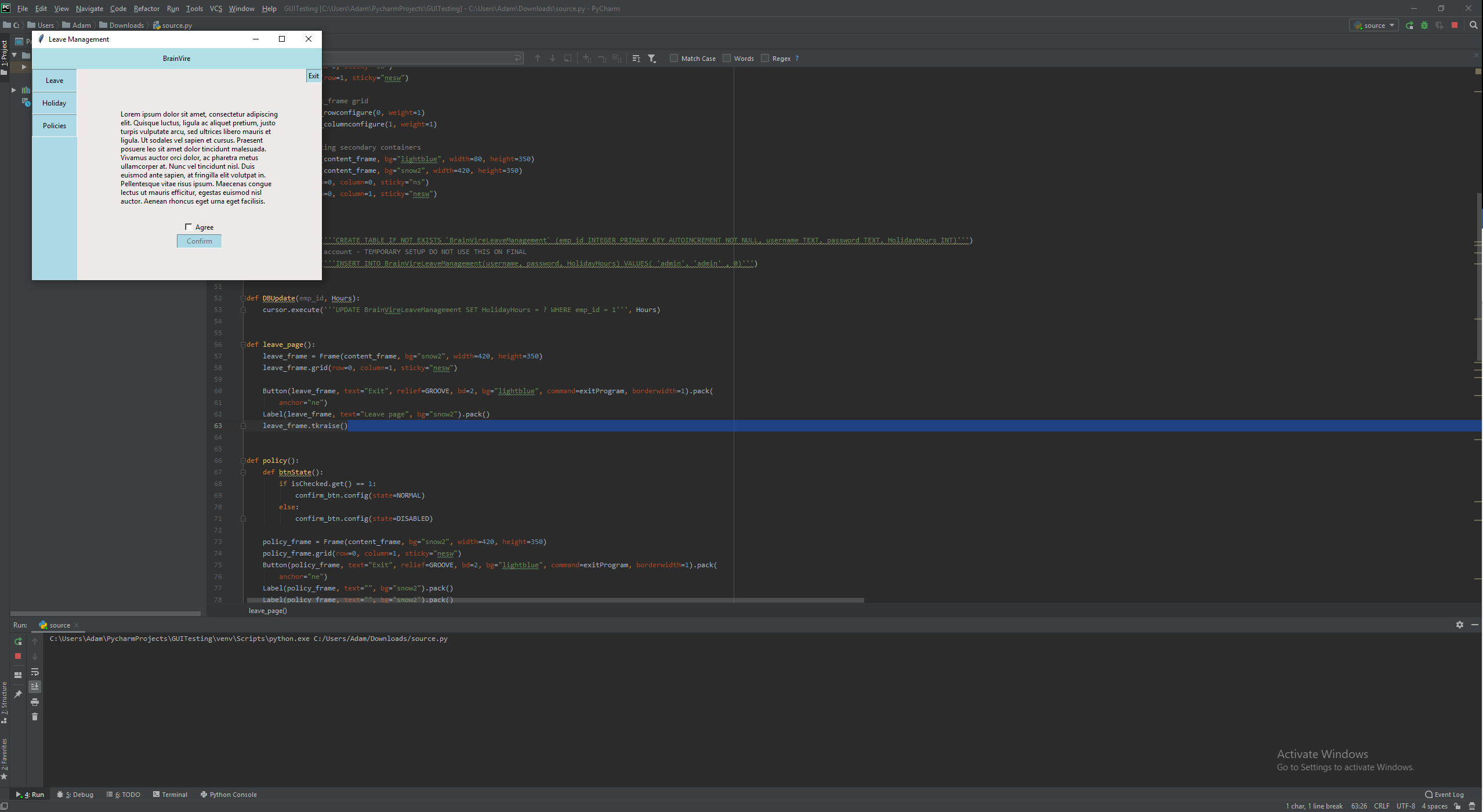


A common error is instead of having it as ‘admin’, sometimes it is put has ‘admin”, this will give a error and not allow the code to run.



**Format Strings testing**

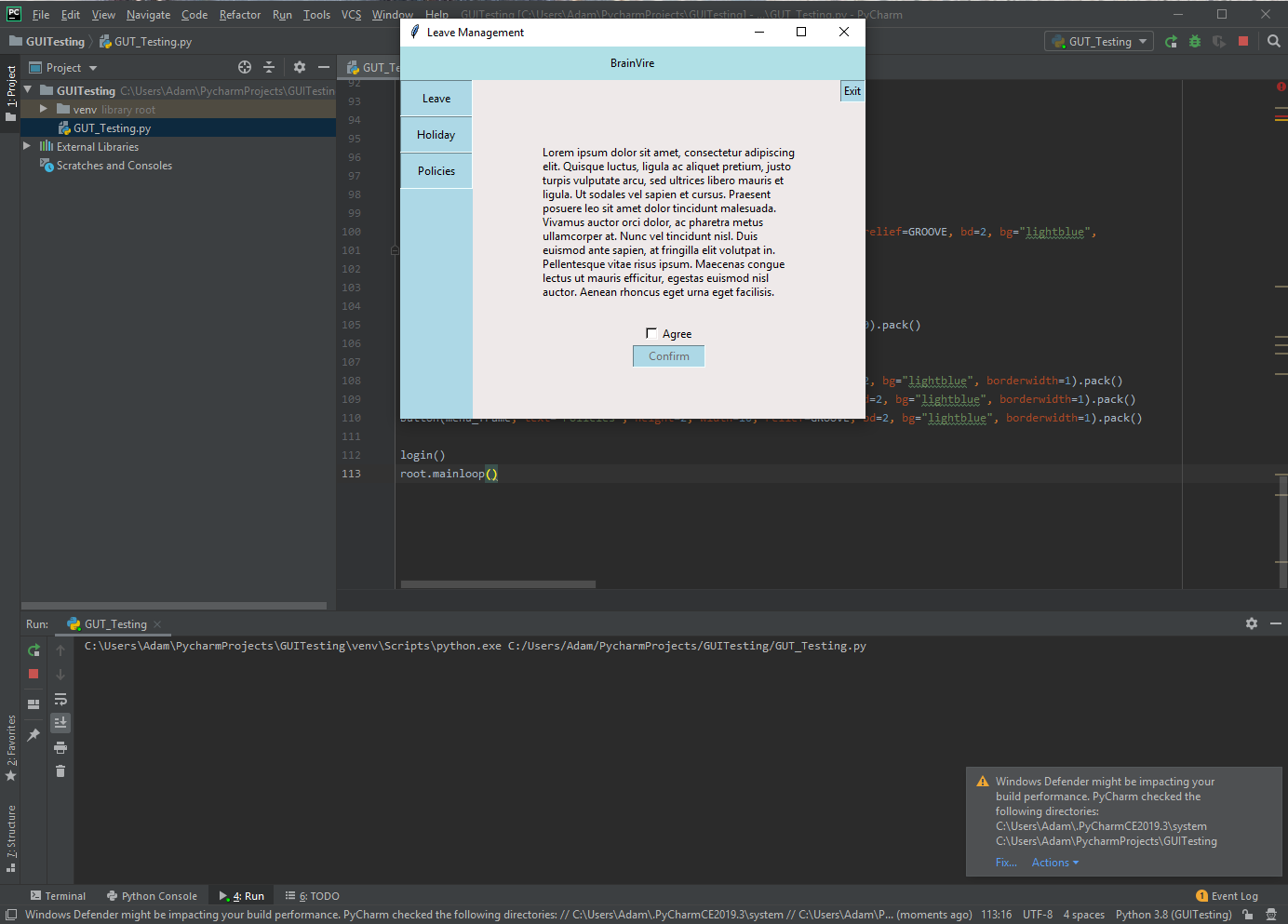
When writing code, you got to make sure you use the correct datatype, for example in our GUI we have “HolidayHours” set to “int”, but if we were to accidentally set it has a “Float” the login function would still work but for an whole number a “int” datatype would be much better to use.



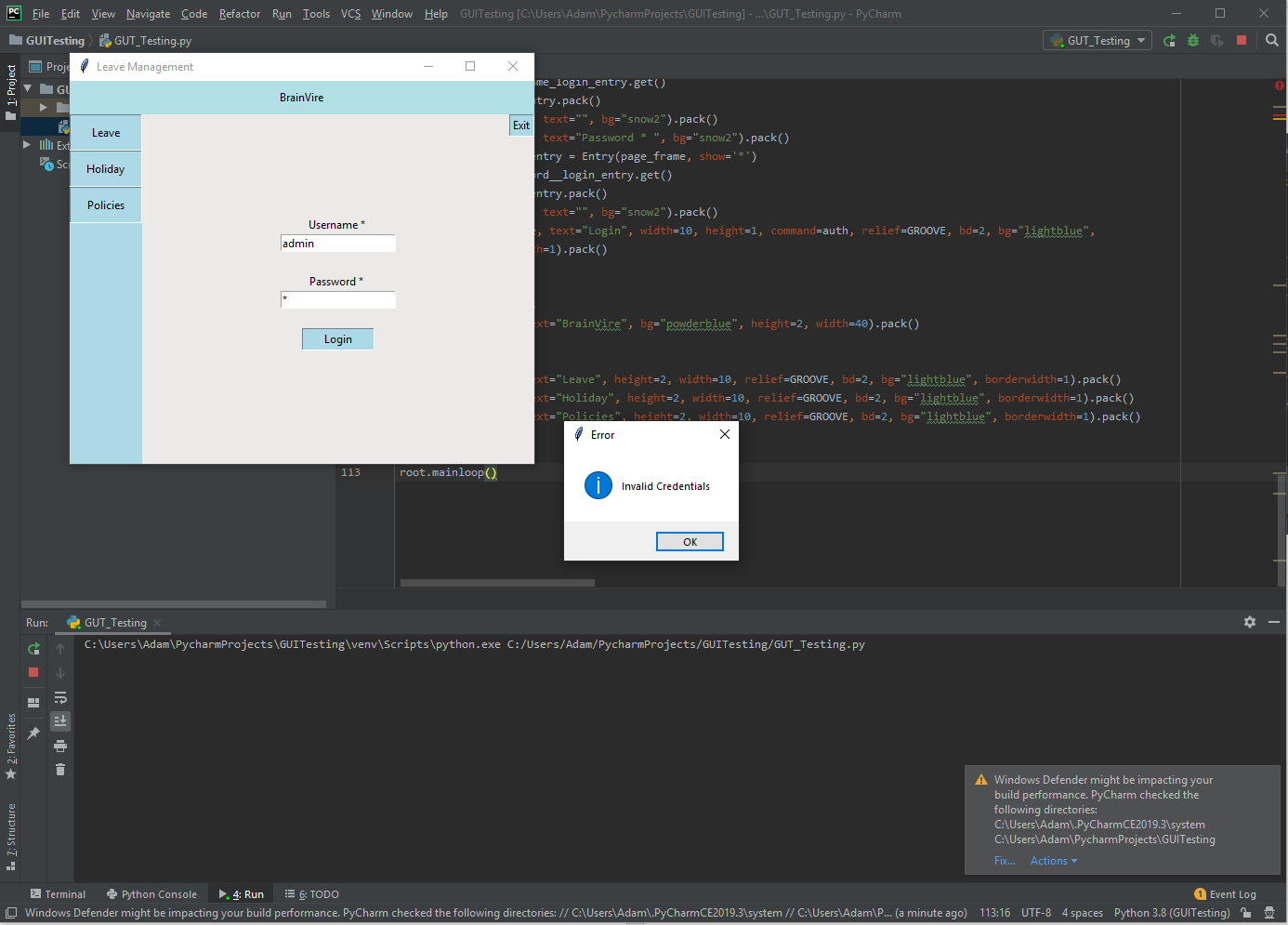
When I change it to a “Float” datatype still works, but this could be exploited easy by adding in extra data which could be dumped into unused memory.

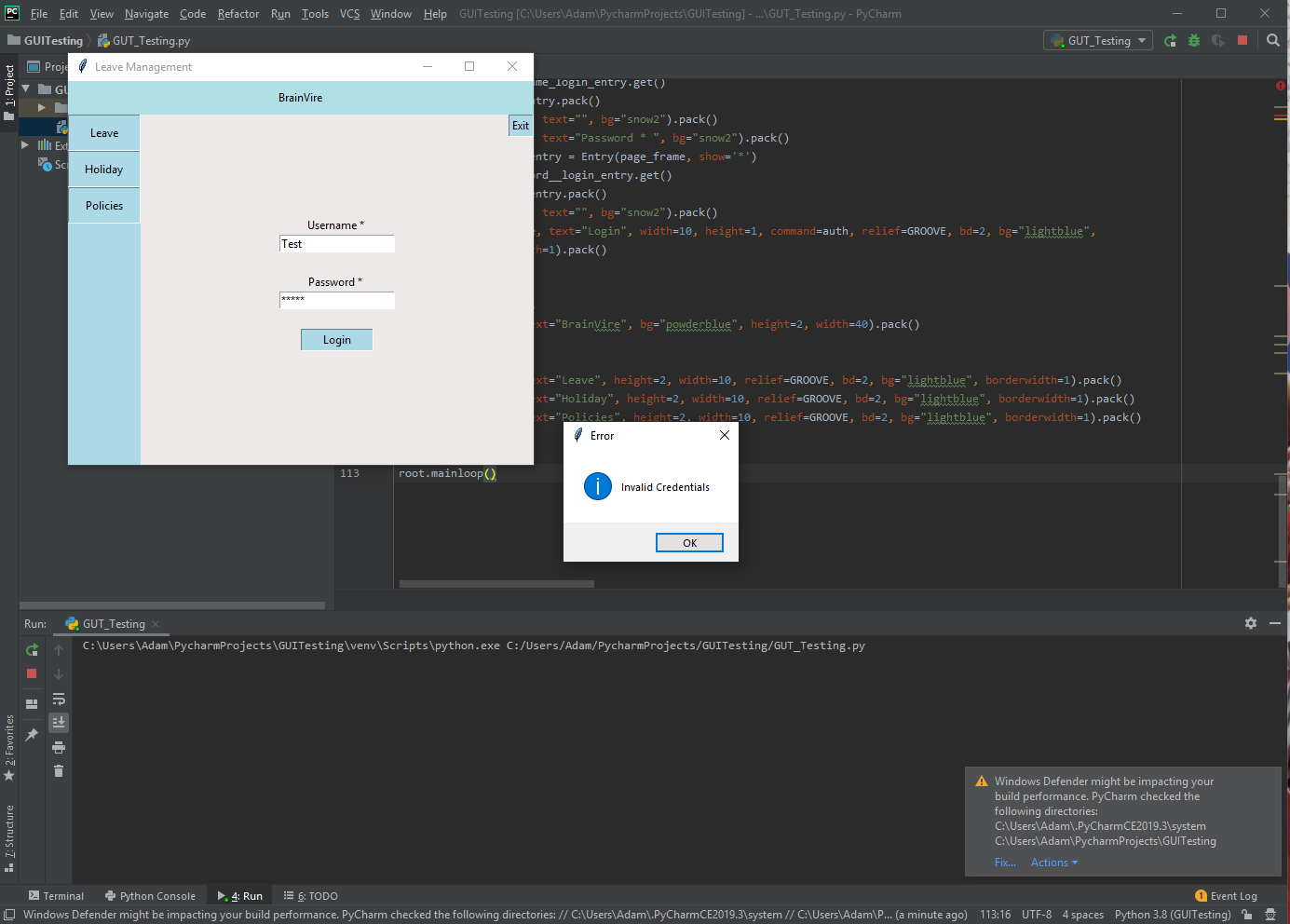
**Appendices**

Appendix 1

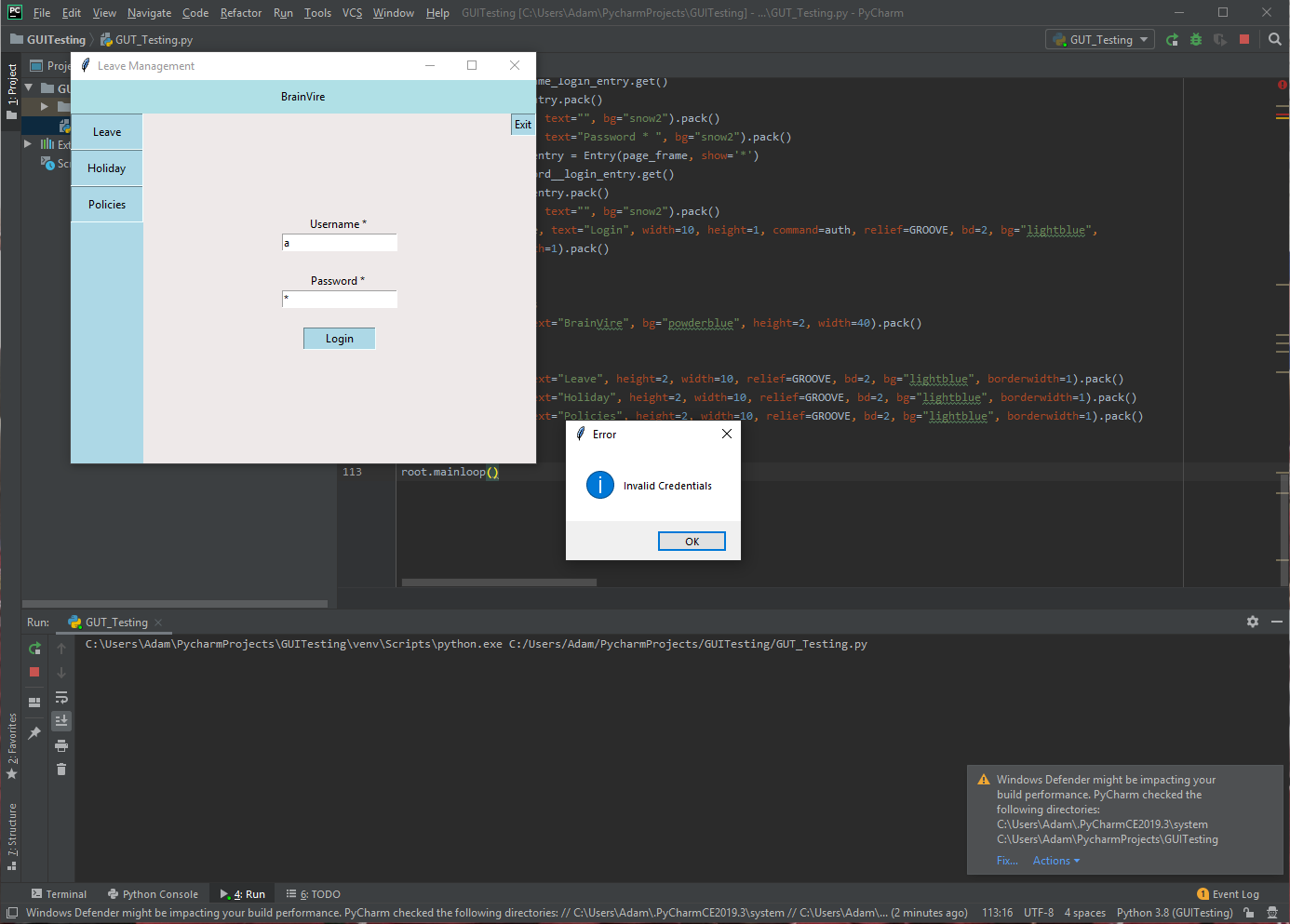


Appendix 2



Appendix 3

Appendix 4



**References**

James E. Weber, Dennis Guster, Paul Safonov, and Mark B. Schmidt - Information Security Journal: A Global Perspective – 2008

William G.J. Halfond, Jeremy Viegas, and Alessandro Orso - A Classification of SQL Injection Attacks and Countermeasures – 2006

Chris Anley - Advanced SQL Injection In SQL Server Applications – 2002

Assist. Prof. Parvin V. Ami, Assist. Prof. S. C. Malav - Top Five Dangerous Security Risks over Web Application – 2013

S Kharod - An improved hashing based password security scheme using salting and differential masking – 2015

M Huenerfauth – An Introduction to Python – 2009

CS Wright - Exploiting Format Strings with Python – 2011

C Cowan – Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks. - 1998

PV Kanakapura – Detecting code injections through cryptographic methods – 2015

AD Keromytis – Countering code-injection attack with instruction-set randomization - 2003

X Qiu – countermeasures to power attack and timing attack on cryptographic operations - 2004

1. James E. Weber, Dennis Guster, Paul Safonov, and Mark B. Schmidt - Information Security Journal: A Global Perspective - 2008 [↑](#footnote-ref-1)
2. William G.J. Halfond, Jeremy Viegas, and Alessandro Orso - A Classification of SQL Injection Attacks and Countermeasures - 2006 [↑](#footnote-ref-2)
3. Chris Anley - Advanced SQL Injection In SQL Server Applications - 2002 [↑](#footnote-ref-3)
4. Assist. Prof. Parvin V. Ami, Assist. Prof. S. C. Malav - Top Five Dangerous Security Risks over Web Application - 2013 [↑](#footnote-ref-4)
5. S Kharod - An improved hashing based password security scheme using salting and differential masking - 2015 [↑](#footnote-ref-5)
6. M Huenerfauth – An Introduction to Python – 2009 [↑](#footnote-ref-6)
7. CS Wright - Exploiting Format Strings with Python – 2011 [↑](#footnote-ref-7)
8. C Cowan – Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks. - 1998 [↑](#footnote-ref-8)
9. PV Kanakapura – Detecting code injections through cryptographic methods – 2015 [↑](#footnote-ref-9)
10. AD Keromytis – Countering code-injection attack with instruction-set randomization - 2003 [↑](#footnote-ref-10)
11. X Qiu – countermeasures to power attack and timing attack on cryptographic operations - 2004 [↑](#footnote-ref-11)