# Introduction

Newcastle University approached me to develop a prototype for their new lottery system. The lottery system will be a web application used for the university’s Lottery Scheme which has a goal of raising money for local charities. I have been tasked with designing and the implementation of main functionalities as well as security elements to ensure the application’s vulnerabilities to security threats are minimised.

The main functionalities of the application include: The user being able to create an account either as an admin role or a public role. As admin role, the user will be able to view basic account details of all users. As a public role the user will be able to view his account details, create lottery draws and check if they have won the lottery. Security requirements have also been fulfilled to ensure threats are kept to a minimum. This include SQL injections and incorrect authorisation access.

During this report I will outline the apporachs I took, steps used to solve various issues, testing used to ensure all requirements were met, further improvements that could be made to the prototype and my reflection on how well the project went

# Approach.

**Data input**

Various data input methods were designed and implemented to ensure the application could collect sufficient user data. Examples of data inputs can be located in “account.jsp” and “userLogin.jsp”. Forms allowed the user to enter the details needed to create an account and gave the user a way to log into their account by providing necessary data ( their username and password) to the application. The security approach I choose was creating a server filter called “ServletFilter”. A servlet filter is a server-side component which is used to pre-process and post-process client’s HTTP requests (www.educba.com, 2020). Filters were mainly used in this application to perform input validation. This approach was justified as servlet filters successfully prevented SQL injections from occurring .The server filter queried the request and blocked any requests-reposnse pair from passing any further if request parameters contained any SQL statements. Another security approach was input validation. Input validation meant the value entered by the user was checked agansint a set of validation charastics. This can be seen clear as phone numbers could only contain numerical dgitis.This approach Is supported as it prevents malicious statements being inputted by the user and thus stoping security threats such as Cross-Site Scripting  from happening.

**Data creation**

Data creation can be seen visible in “account.jsp” during the creation of 6 randomly secure number. When a random number button is clicked the application populates the user numbers form with 6 numbers between 0 to 60 inclusive. A major security threat during this process of data creation can be found in insecure randomness. **Insecure Randomness** errors occur when a function that can produce predictable values is used as a source of **randomness** in security-sensitive context (https://owasp.org/www-community/vulnerabilities/Insecure\_Randomness, 2018). The approach used to deal with this threat was Secure Randomness. The justification of secure randomness can be seen clear in that the values produces will be unpredictable. This will mean the user will not be able to predict the lottery draw the random function will produce.

**Data storage**

Data storage can be seen occurring in the “AddUserNumbers” servlet when the application stores a single string received from the 6 lottery numbers entered by the user to a text file named using the first 20 characters of the user’s hashed password. Encryption was implemented during data storage to minimise security threats. Encryption is the process of encoding a information in a way that only authorized parties can access it (https://www.proofpoint.com/uk/glossary/encryption, 2018). Encyption can be justified as a secure approach as during the encryption of the user’s numbers, a key was used which encoded the data to help prevent unauthorised 3rd parties from accessing the user’s lottery numbers. Another approach used was the hashing of user passwords when storing passwords in the database. A hash function was applied to the user’s password which produced a hash value. Hashing was chosen as it ensured user password in the database could not be accessed by unauthorised 3rd parties without knowing the implementation of the hash function used.

**Data access**

The accessing of data allows the application to retrieve user data and process it. This can be seen clear in during the processing of user session attributes. The Application uses session attributes to ensure it is showing the user the their correct data. A security threat that can occur with inncorect management of sessions is the a user could ponteinally view another user’s data. Correct session management is security approached used to ensure this does not happen. Session management can be seen relevant as it means when the user logs out all relevant data about the user will be removed from the session ensuring other users cannot access the user’s data. Another security approach used during data access was Role Based Access Control. Role Based Access Control can be justified as it restricted system access to authorized users with the right authorisation level. Admins could access all user data except user’s hashed password and lottery numbers whilst public roles could only access their own data and lottery number. The final security approach used during data access authentication process during the signing process. Users can only access data if they input the correct username and password. This process allows the user to prove they are the owner of the account. In addition when the user enters the incorrect login combination 3 or more times the application prevents the user from making any more login attempts. Limiting of the user’s login attempts can be justified as it prevents any brute force methods being used during the login process.

**Data transit**

Data creation can be seen visible in “account.jsp” during the creation of 6 randomly secure number. When a random number button is clicked the application populates the user numbers form with 6 numbers between 0 to 60 inclusive. A major security threat during this process of data creation can be found in insecure randomness. **Insecure Randomness** errors occur when a function that can produce predictable values is used as a source of **randomness** in security-sensitive context (https://owasp.org/www-community/vulnerabilities/Insecure\_Randomness, 2018). The approach used to deal with this threat was Secure Randomness. The justification of secure randomness can be seen clear in that the values produces will be unpredictable. This will mean the user will not be able to predict the lottery draw the random function will produce.

a section each on data input, data creation, data storage, data access, data transit explaining the functionality and security approaches you chose and your justification for those choses. (375) words

# Problem solving

During the initial setting up of the Java web application, I had difficulties getting the application running the way I needed. I recognized my project settings were incorrect which was resulting to errors being shown and me not being able to run the lottery application on my device. To make the lottery web application runnable, the correct settings and the Tomcat Sever had to be on my device locally. To solve this problem, I first had to downloaded Tomcat Sever locally from <http://tomcat.apache.org/>. I next had to ensure that tomcat was deployed on my project and all artifcats were correctly built. Once tomcat was successfully running, a problem arose in that I was unable to connect to the database and thus make new accounts in the application. To solve this issue, I watched a step by step tutorial on how to download docker locally and make MySQL runnable on a Docker Continaer. The solution used for this problem was <https://www.youtube.com/watch?v=NzKDlUVmIyE>. The final step was then making sure the correct Database URL was used during the application. This approach resulted in both tomcat being successfully deployed on my localhost and the application being able to connect to a database using docker.

A problem that arose during the development was the decryption of the user numbers in the textfile. I recongized the whole text file could not be decrypted in one go as it would cause a IllegalBlockSizeException when multiple sets of the user number are append to a textfile.To clear up this problem I researched about how the bytes were being stored in the textfile using <https://nullbeans.com/how-to-encrypt-decrypt-files-byte-arrays-in-java-using-aes-gcm/>. Once I this concept, I was able to come up with an algorithm with could split the textfile into separate blocks of 256 bytes. I knew I had to have a counter which counted to 256 bytes. This would then indicate a user number has been read. Next step taken was then ensuring the correct decryption key was used in order to prevent InvalidKeyException occurring. The problem was fixed by keeping a list of all encryption keys that have been used to encrypt the user’s text file and giving each one an ID so the application knows which key is to be used to decrpty a part of the textfile. This solution resuled in my main goal of making the user numbers secure.

# Testing

The testin strategy used during the development of this project was functionality testing. Black box testing was carried out to test functionality of the application. Before I began testing the application, I created a Test Case document which outlined the list of tests to be conducted. Each requirement was treated as a functionality and thus treated as a single item to be tested on the list. A set of input values were designed to test the functionality produced the desired results. One of the specific Black Box techqiue I used was boundary values. This can be seen evident in “account.jsp” where an input validation was implemented and tested to ensure the lottery draws numbers entered by the user were between 0 and 60 inclusively. Another black box technique used to ensure requirement were met was State-based testing. The application state changes were tested on provision of an input. An example of this was the testing of Role Based Access Control. I ensured program restricted users from being able to access certain pages if they held the wrong authorisation role. In addition, Security requirement were also tested using White Box techniques. This ensured the control flow of the application branch conditions were tested for both being true and false. This meant that all statements can be covered. An example of this setting was ensuring the user could not attempt to log in again if they have 3 failed login attempts. After testing was completed I created a Test summary which analysis to conclude if the software is ready to lunch.

# Recommendations

One of the main focus of this application was making sure it was secure. Although this was achieved at a basic level, the application security could have improved. One way security could be improved Is by changing the storage of the encryption keys to ensure it is securely stored. Attackers tend to target the weakest point of a security system and will not borther trying to crack an encryption algorithm if it is trivially easy to steal the keys. At the moment, the keys are stored in a session attribute. If the session attributes are comprismed the cyber attacker will be able decrypt all user’s numbers. To cover this security weekeness it would be better to store the encryption keys in the cloud using a third party cloud privser. Devices such as cloud encryption gateways that handle the encryption to and from the cloud automatically can help companies achieve this sort of security. Another security enchacnement is the use of High level Authentication. Currently the access the application users only to provide their username and password. To make the access of data more secure the use of a static password and dynamic One Time Passcode could be implemented. OTP​ is a string of characters or numbers automatically generated to be used for one single login attempt (www.cm.com, 2020). OTP’s add an extra layer of authentication and will minimize the risk of fraudulent login attempts as well as the the risk of stolen data. The finally way the security recommendation I would implemente is the addition of a Transport layer security. This will mean all data sent server and application will be encrypted. Including this layer will minimise the threat of Man in the middle attacks and SSL hijacking to allow all data being intercepted by the attacker. In addition the an improvement in the login attemps block could be made. At the moment attackers can use brute force methods to lock users out of their accounts on a shared PC to improve this. To prevent this the application, in addition If user reaches the login attempt the application could ask users to respond to a simple question. These questions are designed to differentiate humans from spambots slide a button form left to right. This will stop any brute force method as the task is easy for humans to complete while the tool remains invisible to spambots.

# Reflections

During this project most of the requirements were achieved. I thought the project was challenging but I manged to get the all 4 task completed. The easiest part of the project for me was the implementation of numerous data inputs to get data from the user but also validation to ensure any input from the user was secure. In addition, I also found the creation og data and storage of data easy to impletment. This was made easy as a result of the successful implementation of the data inputs. I understood the structure of the code and this allowed to quickly come up with the right algorithm to securely implement encryption and input validations to ensure the data was securely stored. A part of the project I found diffulct was the initial setting up of Tomcat and Docker. I had watch any tutorial on both softwares and use many external resource to make both runnable. I have learnt a lot from the setting up of this application and my focus of improvation would be making sure I understand documentations of difffernct softwares before I attempt to use them and configure them. Through the creation of this application I have learnt new security measure such as the hashing of password, random secureness and role based access control. If I was to create the application again in the future, I would improve on what didn’t work well such making a way for the application to hold on to the user’s encryption keys once the tomcat server stops and reruns again. This could be solved using a 3rd party cloud server where all user decryption keys would be stored.