

## Java Lagged Security Features

In our ticketing system project, we will implement extensive security measures to make sure our users are secure. We will focus our security measures to implement safeguards against brute force and dictionary attacks.

First and foremost, functionally we will make sure to only give admin privileges to only our group of developers. Next, we will make sure that all account information for the administrative accounts is never shared with anyone under any circumstances. This prevents anyone other than a select few to have access to admin privileges. Finally, all users will be made to sign a user license agreement to never share their password.

When it comes to our code, there are a couple of easy things we can do to make sure that no malicious actors can purposely exploit our software. First, utilizing 3rd party software to verify information or complete business transactions allows us to tap into a larger company's higher tech security measures without having to go through all the trouble ourselves. This includes 2 factor authentication and 3rd party payment like paypal. Next, we will make sure to separate as many classes and objects from the main as possible, so anyone in main with basic privileges can not access anything, while the classes and objects themselves can interact and change whatever is necessary. Finally, all information stored (payment information, past transactions, usernames and passwords) will all be stored in servers encrypted end to end.

There are two specific technical things we will implement directly into our code to resist any malicious actors. We will add a failsafe for malicious actors that have access gained unwarranted user account information. This will be prevented with 2 factor authentication. Next,

if a malicious user just tries to break into someone's account through means of trial and error, we will implement a limit to the amount of password attempts one person can make.

First, we will implement a 2 factor authentication system. 2 factor authentication is the easiest and safest way to make sure the identity of the user logged in matches the user that created the account. There is already infrastructure in place that allows us to do this, such as google authenticator. All we would need to do is implement the google authentication through firebase and we could verify users in an easy and safe way. Within TP D5, we have two operation contracts that describe this: `begin2FA()`, which begins after the login information is verified and generates a 2FA code to send to the user, and `verify2FA()`, which checks the code that the user inputs to see if it matches with the previously generated 2FA code.

The next thing we are implementing is a limit on the number of times a user can attempt to login to an account. This stops a specific type of attack called a brute force attack. A brute force attack is one of the most straightforward and least subtle methods of hacking: it is when a malicious user tries to guess a password by simply trying every combination. Given enough tries, they are bound to get the password correct eventually. If a user attempts to login to an account and fails to input the right password 5 times, the account will be locked and will be inaccessible. The system would then alert the user by sending an email to the corresponding email attached to the account, letting them know that their account was locked. This will be done by implementing 3 new methods as previously described in the TP D5 operation contracts: `loginAccess()`, which will count the number of failed attempts to log in, `lockAccount()`, which activates once the failed attempts reach 5 and blocks any further attempts from being made, and `alertUser()`, which sends an email to the user of the account, telling them that the account was locked.