Proposal for security features within the Logon of The system

Jessica Erasmus

Table of Contents

[The registration of new users and login process of the system 2](#_Toc114068176)

[HTTP requests and traffic security 2](#_Toc114068177)

[Input validation 4](#_Toc114068178)

[Storing and hashing of passwords 6](#_Toc114068179)

[Maintaining authentication state 7](#_Toc114068180)

[Credential security 8](#_Toc114068181)

[The overall flow of your login process. 9](#_Toc114068182)

[Protection plan of the application against threats and attacks 10](#_Toc114068183)

[Username harvesting 10](#_Toc114068184)

[Brute force attacks 10](#_Toc114068185)

[Session jacking 10](#_Toc114068186)

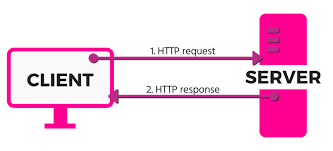
[Session fixation 11](#_Toc114068187)

[References 12](#_Toc114068188)

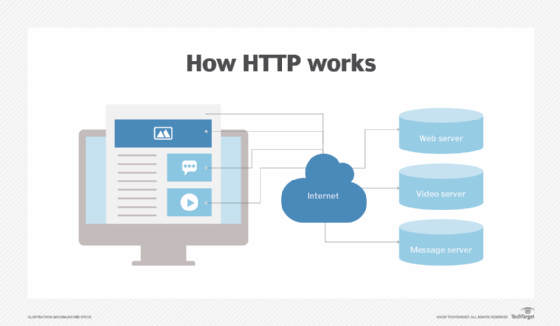
# The registration of new users and login process of the system

## HTTP requests and traffic security

Before implementing HTTP requests, one must understand the fundamentals of how HTTP works. HTTP acts as a request-response protocol and provides the user interaction towards web resource by conveying hypertext messages between clients and servers. HTTP clients usually make use of the Transmission Control Protocol (TCP) to communicate with servers. Consider the diagrams below as an example.



A simple diagram showing the communication between the client and server. The client makes the HTTP request, and the server responds to it in simple terms (RadDevon, 2022).



Client connects to servers through the internet (Chai & Ferguson, 2021).

An HTTP request has the following, according to Garzon (Garzon, 2022):

* An HTTP method (POST, GET, PUT, PATCH, and DELETE, which correspond to CRUD(create, read, update, and delete) operations)
* A host URL
* An endpoint paths
* An optional body, depending on the use of the GET method
* Optional headers
* Optional query strings

A HTTP response usually contains, according to Garzon (Garzon, 2022):

* Protocol version
* Status code
* Status text
* Headers
* An optional body, depending on the DELETE operation

The main methods or operation within the HTTP requests can provide the purposes as explained below:

* GET – retrieves information
* HEAD – retrieves resource headers
* POST – submits data to server
* PUT – saves an object at a specific location
* DELETE – deletes an object from a specific location

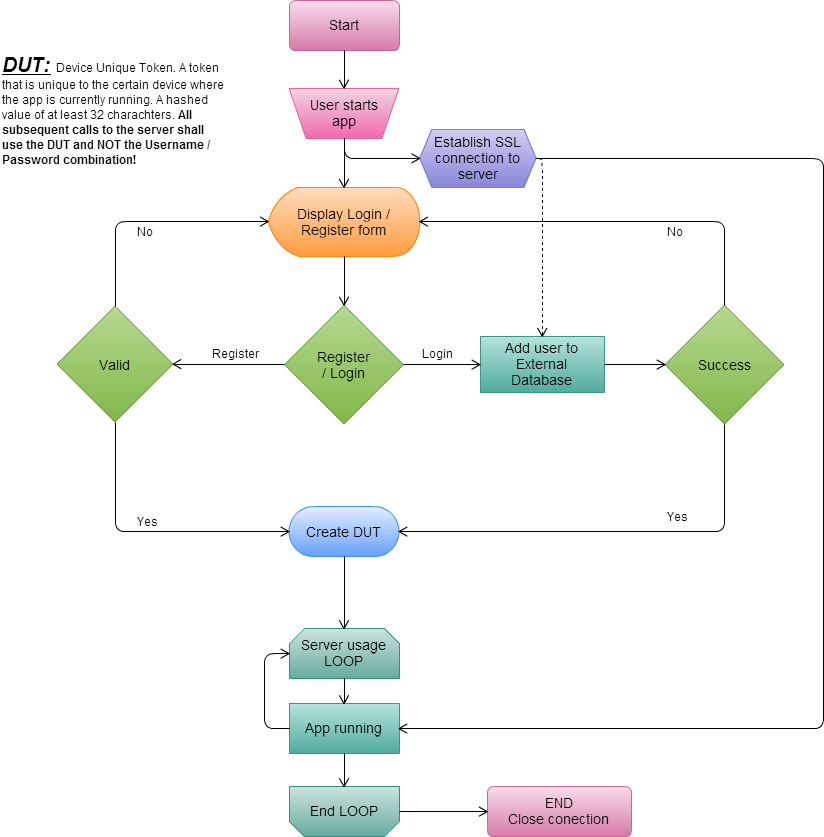
To create a user account within the system, a registration request must provide a user object as a collection of key or value properties (Backendless, 2022). The collection must contain a user identity property and a property to hold the password. Additionally, since users need to be approved to access the system because it is a government system, adding an email property would be suggestable as well. The user registration uses the POST method to add a user to the system. In terms of error handling which may recall when calling the user registration method, the server will detect and report an error to the client by returning a JSON object or message. A response will be provided for the User Registration, which was created, however security sensitive fields will not be returned within the response (FusionAuth, 2022). Therefore, the full user registration will be implemented by creating the user and registration together.

For security purposes, POST is the ideal method for the login as well, since GET will show user credentials within the public URL. The main difference between the methods is that GET carries the request parameter in a URL string, while POST carries the request parameter in a message body when transferring data from client to server in HTTP protocol (duyhdb, 2021). In simpler terms, POST /login creates user session and DELETE /logout destroys user session.

However, an overload in http requests may lead to traffic and connection errors which affects performance. This affects the time for the request to be completes, reach the server, prepare response and send/receive response (RadDevon, 2022). Hence, reducing resource size and the number of requests can reduce the amount of time it takes to load page. Consider the following approach for optimising performance, provided by RadDevon (RadDevon, 2022):

1. Eliminate unnecessary requests by eliminating unnecessary images or scripts.
2. Reduce weight by optimizing images and minifying code.
3. Reduce the number of requests by concatenating all Javascript and CSS into a single file each.

Consider the system register/login will function similarly to the following diagram (Widerberg, 2012):



## Input validation

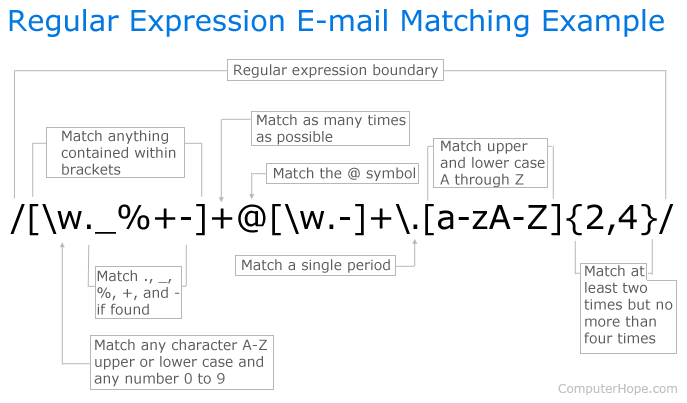
Input validation can be implemented using validation rules. Consider the following ideal validation rules:

* Validating open text inputs – since null values or empty fields should not be allowed in the system to prevent error, empty values automatically have a default value instead of a non-existent value.
* Implementing error messages into text input validation – shows error message if input field contains incorrect data

Whitelisting creates a list of permitted entities and blocks everything else (CTI, 2021). It relies on trust and the default is set to deny anything new unless it is proven to be acceptable (CTI, 2021). Hence, to summarise whitelisting (CTI, 2021):

* Whitelisting involves only allowing access for approved entities.
* The default is to block access.
* Whitelisting is trust centric.

Whitelisting should also be implemented to allow data to be read independently by applying the RegEx (regular expression) framework to validate user input successfully. RegEx accomplishes this by identifying whether the fields match to the already registered data of the user or simply finding data. Regex.IsMatch specifically uses a lookaround pattern. Below shows an example of email confirmation using RegEx provided by Computer Hope (Computer Hope, 2020).



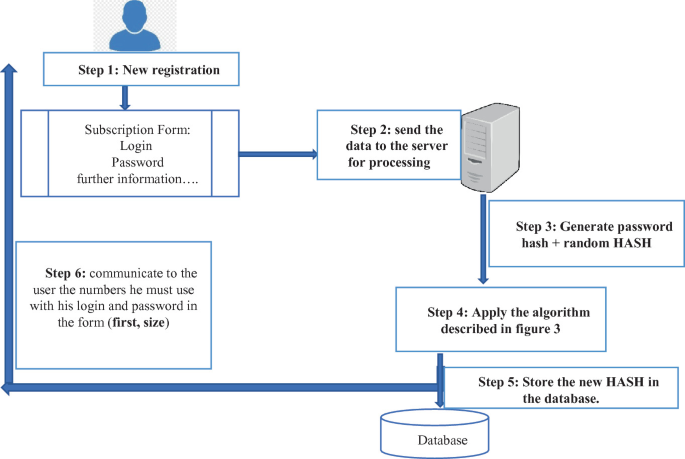
## Storing and hashing of passwords

Passwords will be stored within a cipher and generate a hash using SALT which is added to the cipher once password is stored. The system will be encrypted using a private-key and the system will use substitution cyphers to encrypt confidential user information which will replace character bits in plaintext with alternate character bits in ciphertext. See the figure below from Nohe (Nohe, 2018).

Logo, company name

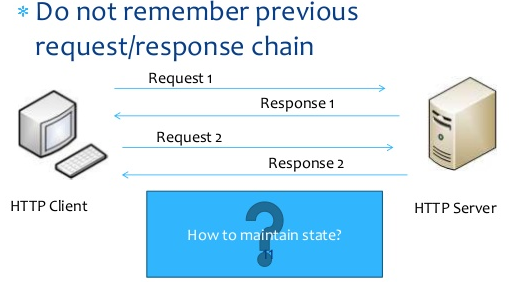
Description automatically generated

A salt is then added to the hashing process to force their uniqueness, increase their complexity without increasing user requirements, and to mitigate password attacks (Arias, 2021). Consider the diagram below representing the process of a hash transformation of a new registered user (Touil, et al., 2021). This diagram used the MD5 hashing algorithm however it is not the algorithm we are certain of using.



## Maintaining authentication state

Since HTTP is a stateless protocol, it will not know what the previous request performed because of separate connections (Boyini, 2019). The following figure (Paralogarajah, 2017) shows the representation of the initial HTTP dilemma within the application.



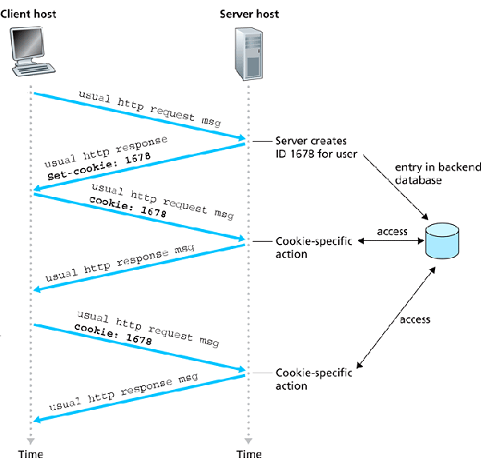
How can the application maintain authentication state? There are two probable options to solve the application in this scenario.

A session cookie can be provided to the client so that the client remains logged in for a period before timeout. Cookies are less than 4kb in size and are useful for session management.

Consider the following example, provided from Paralogarajah (Paralogarajah, 2017), of the flow of session management using cookies:

1. Browser sends a new login request R1 to the Facebook server.
2. Then Facebook will check whether there is any cookie is sent by the browser.
3. Since this is the new request, there will be no cookie value in the database.
4. So, Facebook will send a cookie to your browser and Facebook will store its ID.
5. Then the browser will set that cookie for that Facebook domain.
6. With each request for Facebook domain, your browser should send the cookie in the HTTP header.
7. Then Facebook will check whether it has the ID sent by the browser. If that exists, then Facebook will use the session indicated by the cookie.

The figure (Paralogarajah, 2017) below represents the workflow of cookie sessions (Paralogarajah, 2017).



The client sends the first request to the server; however, the http request will not have a cookie value initially (Paralogarajah, 2017). The server then sets a new cookie for that session and sends it back to the client along with the response where the browser stores the cookie for that domain (Paralogarajah, 2017).

URL rewriting could also be ideal to maintain user sessions. Extra data is attached to the end of each URL. This data identifies the session. The server associates the session identifier with the data stored from that session (Boyini, 2019). That session identifier then is assigned a unique session ID which is accessed at the server to identify the client (Boyini, 2019).

## Credential security

The password will be stored within a cypher. Another suggestion to ensure credential security is to set up and enforce a password policy to create strong passwords. Passwords will also be checked within the cypher when user logs in to verify whether the user credentials are correct.

A password storage scheme is preferable towards the application, potentially SHA512. Oracle states that : “The Salted SHA512 Password Storage Scheme provides a mechanism for encoding user passwords using a salted form of the 512-bit SHA-2 message digest algorithm” (Oracle, 2019).

Hence, passwords will be stored and checked within a cipher with salt added to each password. With the use of salt, the password will be hashed in the cipher such as show in the figure below, which was found within a blog post from Cyphere (Cyphere, 2018).

Diagram

Description automatically generated with low confidence

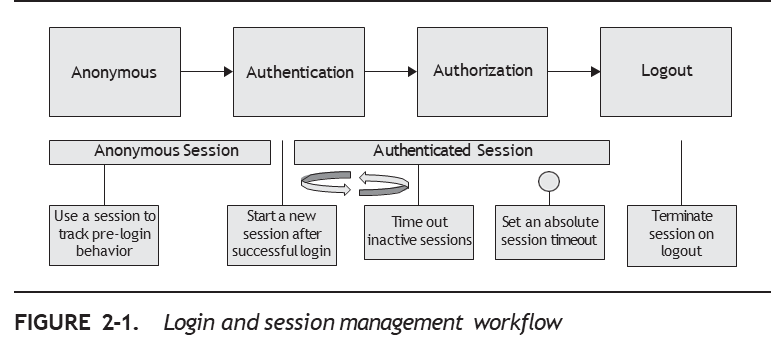
This figure provides a visual representation of the salt hashing (Cyphere, 2018).

## The overall flow of your login process.

This is the basic flow of the login process (Manico & Detlefsen, 2015):

1. Anonymous Session is created on First Hit
2. Starting HTTPS and Encryption in Transit
3. Processing and Verifying Credentials
4. Start the User’s Authenticated Session
5. Do Cool Things (Explore the system)
6. Potential Re-Authentication for Sensitive Operations
7. Idle Timeout
8. Absolute Timeout
9. Logout

The login workflow is altogether represented as shown below (Manico & Detlefsen, 2015).



# Protection plan of the application against threats and attacks

## Username harvesting

Username harvesting is an attack that occurs when threat agents try to identify if a username is valid or not (Manico & Detlefsen, 2015). It is usually conducted by inspecting error messages for failed logins. A way to prevent username harvesting is to implement error messages in a safe and secure manner. A more generic login error message would be : “Invalid username/password combination.” Is more suitable to reduce chances of these attacks.

## Brute force attacks

A brute force attack occurs when the victim attempts multiple passwords at high capacity against the victim’s account to discover their authentication credentials through trial and error (Manico & Detlefsen, 2015). The main defence towards such attacks are account lockout (Manico & Detlefsen, 2015). Limiting the amount of login attempts help minimise the probability of the attacks.

## Session jacking

The threat of session jacking/hijacking rises when the threat agent can guess or steal the victim’s session ID or simply if the user leaves their account logged in to a public computer (Manico & Detlefsen, 2015). Sessions are only created once the user logs on. So as a defence, the system must persist authentication information and users must ensure that they log out of a public computer once complete with their tasks (Manico & Detlefsen, 2015).

## Session fixation

Session fixation is a form of session hijacking jokers when the attacker tricks the victim into logging into a site using the session ID which is known to be attacker (Manico & Detlefsen, 2015). The only way to prevent session fixation is by regenerating the session once the user successfully logs in and invalidate the current session and create a new user session (Manico & Detlefsen, 2015).

# References

Arias, D., 2021. *Adding Salt to Hashing: A Better Way to Store Passwords.* [Online]   
Available at: https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/  
[Accessed 18 Aug 2022].

Backendless, 2022. *Backendless REST API Documentation.* [Online]   
Available at: https://backendless.com/docs/rest/users\_user\_registration.html#email-confirmations  
[Accessed 16 Aug 2022].

Boyini, k., 2019. *As HTTP is a stateless then how to maintain the session between web browser and web server?.* [Online]   
Available at: https://www.tutorialspoint.com/as-http-is-a-stateless-then-how-to-maintain-the-session-between-web-browser-and-web-server  
[Accessed 19 Aug 2022].

Chai, W. & Ferguson, K., 2021. *HTTP (Hypertext Transfer Protocol).* [Online]   
Available at: https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol  
[Accessed 16 Aug 2022].

Computer Hope, 2020. *Regex.* [Online]   
Available at: https://www.computerhope.com/jargon/r/regex.htm  
[Accessed 17 Aug 2022].

CTI, 2021. *Blacklisting vs. Whitelisting.* [Online]   
Available at: https://consoltech.com/blog/blacklisting-vs-whitelisting/  
[Accessed 20 Aug 2022].

Cyphere, 2018. *Difference Between Encryption And Hashing |Salt Cryptography.* [Online]   
Available at: data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAQMAAADCCAMAAAB6zFdcAAAB7FBMVEVScf////8AAABPb//czcZMbf/Z2dnXPyRKa/9Tcv9JcP/3qwBVdf//kznh0cjg0MSpqeD/rgCCfdJUbcOSkqbhoh4vVn4yWIQmU4Go4fSGrbpFaP+rjqTeokuHgc3e3deLhMj/lC3zkVWTmuk8Urn/kj8/h499stBad/vN1f4jL2kzRp17e  
[Accessed 18 Aug 2022].

duyhdb, 2021. *Login and Register Requests in API's.* [Online]   
Available at: https://stackoverflow.com/questions/66874983/login-and-register-requests-in-apis  
[Accessed 16 Aug 2022].

FusionAuth, 2022. *User Registration APIs.* [Online]   
Available at: https://fusionauth.io/docs/v1/tech/apis/registrations  
[Accessed 16 Aug 2022].

Garzon, C. R., 2022. *HTTP Request Methods – Get vs Put vs Post Explained with Code Examples.* [Online]   
Available at: https://www.freecodecamp.org/news/http-request-methods-explained/  
[Accessed 16 Aug 2022].

Manico, J. & Detlefsen, A., 2015. *Iron-Clad Java: Building Secure Web Applications.* 1st ed. NY, NY: McGraw-Hill Education.

Nohe, P., 2018. *The difference between Encryption, Hashing and Salting.* [Online]   
Available at: https://www.thesslstore.com/blog/difference-encryption-hashing-salting/  
[Accessed 18 Aug 2022].

Oracle, 2019. *Salted SHA512 Password Storage Scheme.* [Online]   
Available at: https://docs.oracle.com/cd/E19476-01/821-0507/salted-sha512-password-storage-scheme.html  
[Accessed 18 Aug 2022].

Oxford, 2021. *Lexico.* [Online]   
Available at: https://www.lexico.com/  
[Accessed 23 June 2022].

Paralogarajah, P., 2017. *Sessions and cookies.* [Online]   
Available at: https://piraveenaparalogarajah.medium.com/sessions-and-cookies-2c0919552f29  
[Accessed 17 Aug 2022].

RadDevon, 2022. *Why should I care about HTTP?.* [Online]   
Available at: https://raddevon.com/articles/why-should-i-care-about-http/  
[Accessed 16 Aug 2022].

Touil, H., Akkad, N. E. & Satori, K., 2021. *Securing the Storage of Passwords Based on the MD5 HASH Transformation.* [Online]   
Available at: https://media.springernature.com/lw685/springer-static/image/chp%3A10.1007%2F978-3-030-73882-2\_45/MediaObjects/507212\_1\_En\_45\_Fig1\_HTML.png  
[Accessed 20 Aug 2022].

Widerberg, 2012. *Android: Some questions about security and user authentication (register, login) through HTTP.* [Online]   
Available at: https://stackoverflow.com/questions/13865296/android-some-questions-about-security-and-user-authentication-register-login  
[Accessed 16 Aug 2022].