Quackmessage specs

Noah Hallows

December 2025

Contents

1 Authentication **1**
- 1.1 Login 1
- 1.2 Account creation 1
- 1.3 Database 1

1 Authentication

Authentication is handled using helper functions in *auth.py*. Passwords are transmited to the server in plan text and hashed on the server using argon2 hashing. Client authentication is checked using stateless JWT tokens (for end to end encryption they also need to locally have the appropriate keys). This JWT token is checked using an interceptor.

1.1 Login

To log in the client sends the username and password over a encrypted (tls) connection to the server. The main function of the server then uses functions in *auth.py* which looks up the password hash in the database using the username and compares it with the users plan text password provided (using argon2 verify). If successful *auth.py* generates a jwt token and returns it and the boolean true. Which is then sent back to the client in the main function.

1.2 Account creation

To create an account the user needs to provide an email address (which will be checked by sending a code to it), password and a username. Account creation is handled using helper functions in *auth.py*.

1.3 Database

User accounts are stored in the table **users**. The collum **email** is used as the key.

Name	Type	Limits	Description
email	varchar	254 characters [1, p. 1]	Email address of user, verified at signup.
username	varchar	126 characters	Store usernames.
password_hash	bytea	no limit	Stores hashed password.
account_creation_date	timestamptz	4713 BC to 5874897 AD	Store account creation date and timezone.
messages_sent	integer	$-2,147,483,648$ to $2,147,483,647$	Store number of messages sent.
messages_received	integer	$-2,147,483,648$ to $2,147,483,647$	Store number of messages received.

References

[1] J. Klensin, *Rfc 3696 errata 3*, Rfc-editor.org, /07/2005. 23/12/2025. [Online]. Available: https://www.rfc-editor.org/errata_search.php?rfc=3696.