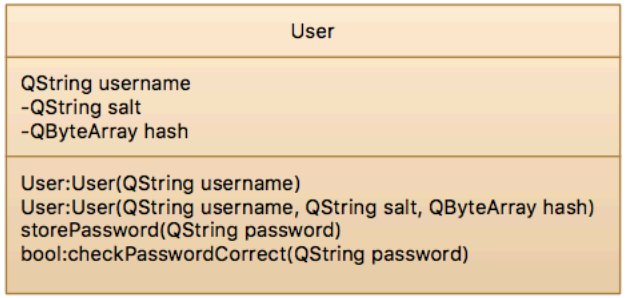

NOTE: Figure out how to store and represent in a class which episodes a user has watched.

Notes on securly handeling password

Pepper is hardcoded as a constant in the code.

On account creation:

# Salt generated and stored with each password in the db
salt = Generate a random string and hash with sha3_512

passSaltPepper = concat password, salt and pepper together
QByteArray hash = QCryptographicHash::hash(passSaltPepper.toLocal8bit(), QCryptographicHash::Sha3_512);

#Store salt and hash in db linked to this user only.

Later, when an existing user logs in, the entered password is combined with the salt found stored with the account,
and also combined with the hard coded constant pepper, and the same hash algorithm is run.
If the hash just generated, matches the hash stored in the db, then the user is permitted to log in.

#If it doesn't match, an error is reported that either the username or password is incorrect.
There should be no information disclosure on login failure. For example, indicating that the user doesn't exist,
or that the user exists, but the password was incorrect.

The below is optional for now, but the salt and pepper above should be used.
NOTE: Investigate if there is a secure, in memory way to temporarily store the passwords when they are momentarily in plaintext.
For example, C# contains securestring. Which prevents/minimizes possibility of attacks on memory.
If an equivalent to securestring is not found, there may be alternatives to properly handling the plaintext version of the
string (example, using a symmetric hash before storing it in memory for longer periods of time,
limiting number of copies in memory, and actively, overwriting with junk data, then deleting when done with it).