# VirtualSoc

Teodorescu Calin

December 7, 2022

## 1 Introduction

VirtualSoc is a client-server application which simulates a social media application. Users who want to create an account can do so by choosing a username and a password for their account. From there on, they can login into their account using the right credentials. Users can set their profile as public or private, depending on their preferences. The same can be applied to their shared posts. Users who are not logged in can only view public posts and do not have access to the application's built-in commands. There are multiple features, from adding other users to your contact list as friends / close friends / family to being able to communicate privately with a specific contact or with a group of people.

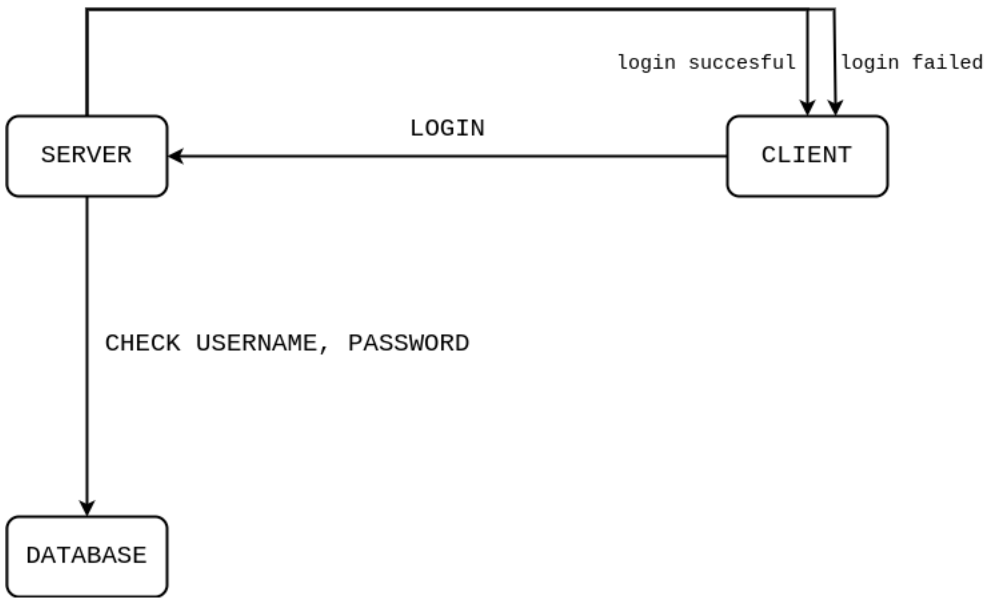
## 2 Used technologies

- TCP/IP
  The application needs a reliable protocol in order to maintain the integrity of the users' posts as well as of the messages between one another.
- SQLite
  It is used to store the credentials for each account, as well as the messages between two users.

## 3 Application architecture

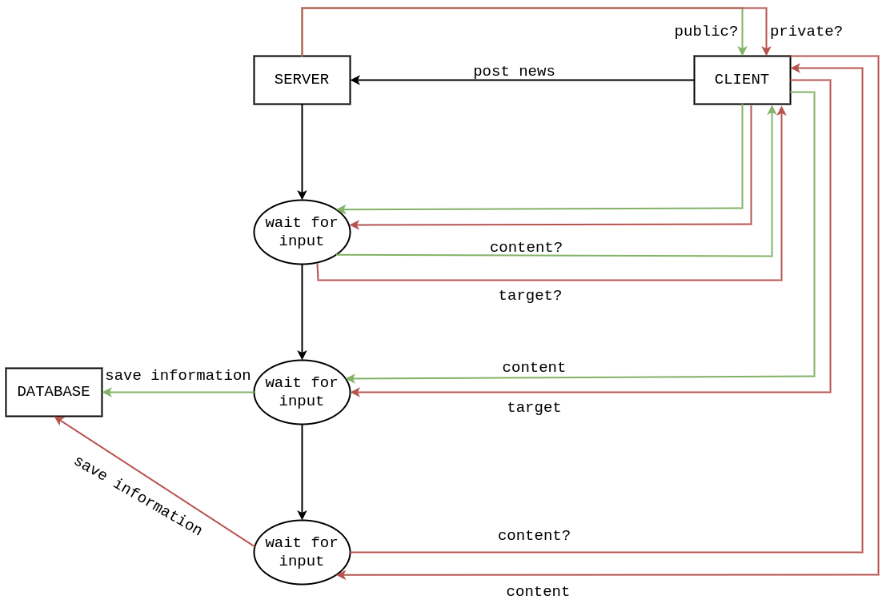
Diagrams will be used for an easier understanding on how some of the features of the application work:

- Diagram for the login function:



The username and the password for each account will be stored in a database,

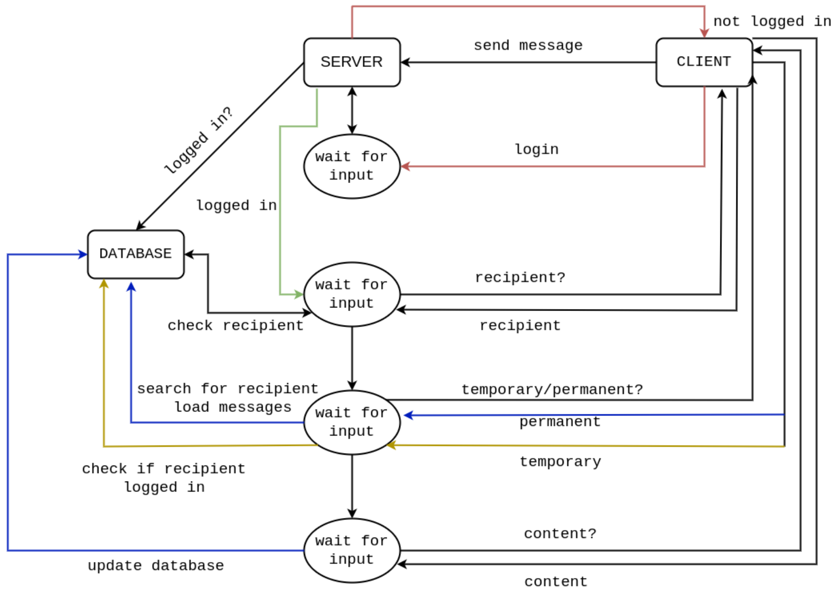
- Diagram for the post function:



The red colored arrow corresponds to a private post, while a green colored arrow corresponds to a public post.
For each post, the user can decide whether it is public or not. If the post is set to public, everyone can see it, including users who are not logged

in. For a private post, a target audience can be selected, which can range from a single user to a category of people from the contact list, or a group of users selected manually.

- Diagram for the message function:



Users can send messages to one another while they are logged in. When sent a message, the recipient can decide whether or not to accept the message. There are two types of private messaging:

1) **Temporary**
   For this feature to work, the recipient must be logged in as well, for the whole duration of the conversation. During this period, the messages will be saved, but if one of the users logs out of their account, the conversation will be deleted from the database.

2) **Permanent**
   This type of conversations will be stored in the application's database. Not both of the users have to be logged in order to send messages. When the user types the recipient's username, the server will search the database for the conversation between the two users and load the messages.

In the case of live conversations, the client will ask the server to search the database for new messages periodically.

## 4 Implementation

The application uses a concurrent server, since multiple users can be logged in at the same time. They should be able to perform different actions concurrently, without having to wait for commands from other users to be processed, which can happen while using an iterative server.
For each input from the client, the server will create a child using the *fork()* command, who will be assigned a specific task.

## 5 Conclusions

### 5.1 Improvements

Regarding the live messaging feature, if the user sends messages faster than the set refresh rate of the server, there might occur delays in the display of the messages.

## References

[1] Project requirements
`https://profs.info.uaic.ro/~computernetworks/files/homework2_ro.txt`

[2] TCP concurrent server
`https://profs.info.uaic.ro/~gcalancea/Laboratorul_7.pdf`