**Functional requirements:**

* Creating accounts – available after successful connection client to the server. To create account correctly, client have to type unique ID and minimum 6 letters length password.
* Deleting account – available after successful connection client to the server and login into right account. To approve deleting client have to type present password.
* Log in – available after successful connection client to the server. To log in correctly client have to type his login and password.
* Log out - available after successful connection client to the server and login into right account. When client wants to finished session he is able to logout.
* Receiving messages - available after successful connection client to the server and login into right account. Receiving is auto-called method working in background as a loop and trying to catch incoming message.
* Creating private chat groups - available after successful connection client to the server and login into right account. By creating private chat group client can sending private messages to selected client. For creating private chat client have to type chat title and other client name.
* Inviting other users to private groups - available after successful connection client to the server and login into right account. Client can add more than one other client to private chat by typing another client name.
* Sending broadcast messages - available after successful connection client to the server and login into right account. This method allows to send message to all online clients just by typing message without clients’ name.
* Listing online clients - available after successful connection client to the server and login into right account. Clients only need to call this method without parameters and that return list of online clients.

**Unfunctional requrements:**

* Command-line communicator – application will be working in command-line with text command control.
* TCP/IP communication protocol – one of the most popular communication protocol using for application to connect between users. This protocol gives certainty that message will came from source to destination.
* Store accounts credentials in encrypted file – encrypting file increase security level of data colleting on server and allows to keep in memory logins and passwords for clients.
* Defining message frame -
* Available in local network – local network is the easiest environment for running for this kind of communication application. Also available for devices with external IP address
* Creating in C++ - well prepared API for socket based programs and individual language preferences.

**Client UML:**

* Creating accounts



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number |
| Normal flow | * Connection to host with correct address and port. * Choosing from menu Create new account * Entering unique login and password |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Login is already in use – loop with request for login. * Timeout for response – disconnect from the server and restart client |

* Deleting account



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number * Connected to server * Created account * Log in to account |
| Normal flow | * Connection to host with correct address and port. * Log in to account and choose from menu Delete account * Confirm with typing password again |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Wrong login or password – loop with request for login and password * Wrong password while deleting – 3 attempts for typing password. * Timeout for response – disconnect from the server and restart client |

* Log in



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number * Connected to server |
| Normal flow | * Connection to host with correct address and port. * Choose from menu Log in and confirm it with correct login and password |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Wrong login or password – loop with request for login and password * Timeout for response – disconnect from the server and restart client |

* Log out



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number * Connected to server * Created account * Log in to account |
| Normal flow | * Connection to host with correct address and port * Log in to account and choose from menu Logout |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Wrong login or password – loop with request for login and password * Timeout for response – disconnect from the server and restart client |

* Receiving messages



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number * Connected to server * Created account * Log in to account |
| Normal flow | * Connection to host with correct address and port * Choose from menu Log in and confirm it with correct login and password * Self-called method Receive message in loop and handling it when something comes |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Wrong login or password – loop with request for login and password * Receive unrecognized message – print “Unrecognized message” and ignore handling * Timeout for response – disconnect from the server and restart client |

* Creating private chat



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number * Connected to server * Created account * Log in to account |
| Normal flow | * Connection to host with correct address and port * Choose from menu Log in and confirm it with correct login and password * Choose from menu Create private chat and confirm it with typing title of this group and other client nick |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Wrong login or password – loop with request for login and password * Group title is already in use – loop with request for another title * Wrong client name (offline or doesn’t exist) – loop with request for another client * Receive unrecognized message – print “Unrecognized message” and ignore handling * Timeout for response – disconnect from the server and restart client |

* Inviting other users to private



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number * Connected to server * Created account * Log in to account * Already created group |
| Normal flow | * Connection to host with correct address and port * Choose from menu Log in and confirm it with correct login and password * Choose from menu Invite to private group * Type name of chosen group and confirm * Type nick of client you want to add and confirm |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Wrong login or password – loop with request for login and password * Clients doesn’t belong to any group – Invite to private group isn’t visible in menu * Wrong group title – loop with request for another title * Wrong client name (offline or doesn’t exist) – loop with request for another client * Receive unrecognized message – print “Unrecognized message” and ignore handling * Timeout for response – disconnect from the server and restart client |

* Sending broadcast messages



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number * Connected to server * Created account * Log in to account |
| Normal flow | * Connection to host with correct address and port * Choose from menu Log in and confirm it with correct login and password * Choose from menu Send broadcast message * Type message and confirm |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Wrong login or password – loop with request for login and password * Receive unrecognized message – print “Unrecognized message” and ignore handling * Timeout for response – disconnect from the server and restart client |

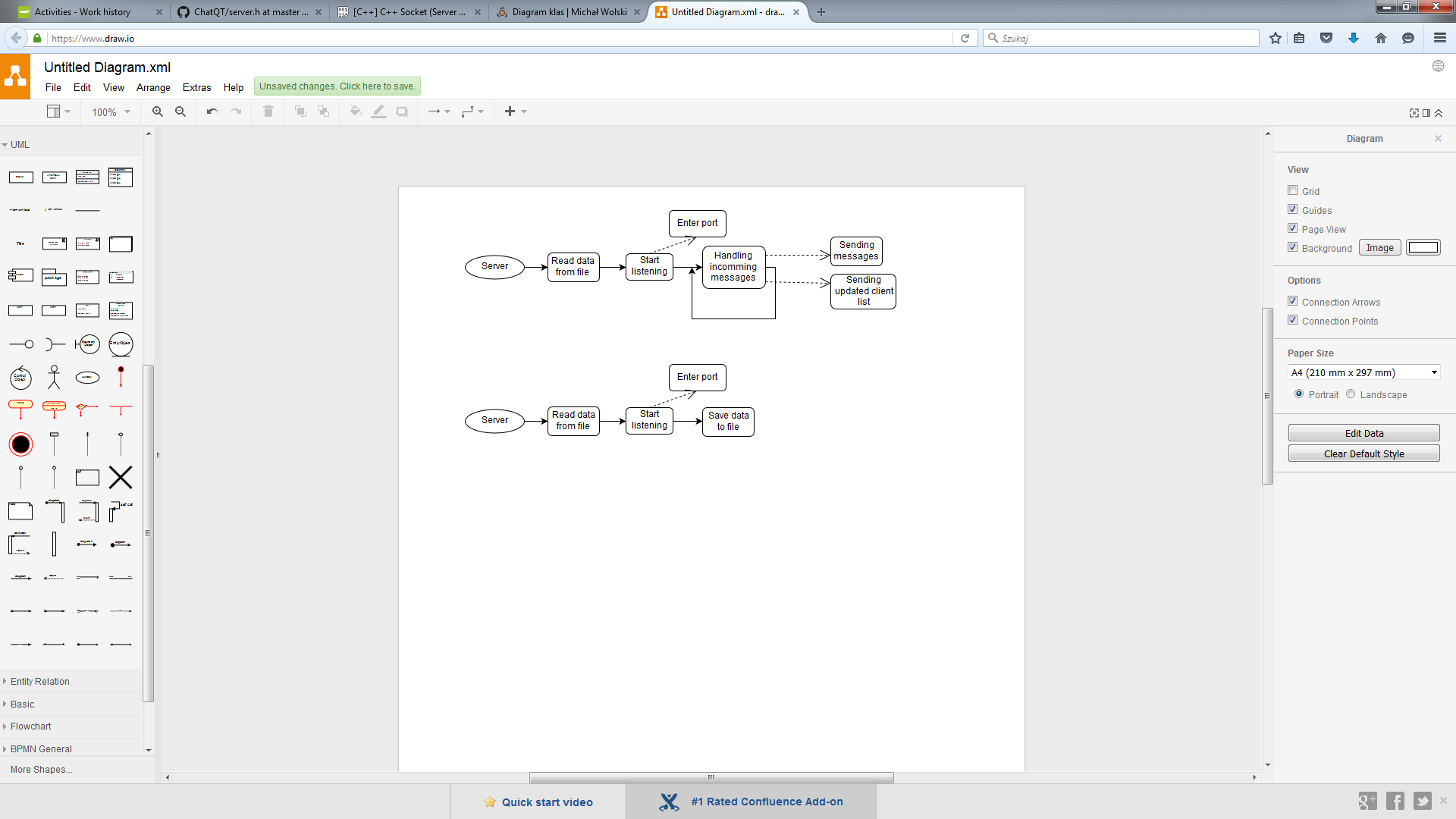
* Listing online



|  |  |
| --- | --- |
| Preconditions | * Host address * Host port number * Connected to server * Created account * Log in to account |
| Normal flow | * Connection to host with correct address and port * Choose from menu Log in and confirm it with correct login and password * Choose from menu List online clients |
| Exceptional flow | * Cannot connect to host (wrong address, wrong port, host offline) – loop with request for address and port. * Wrong login or password – loop with request for login and password * Receive unrecognized message – print “Unrecognized message” and ignore handling * Timeout for response – disconnect from the server and restart client |

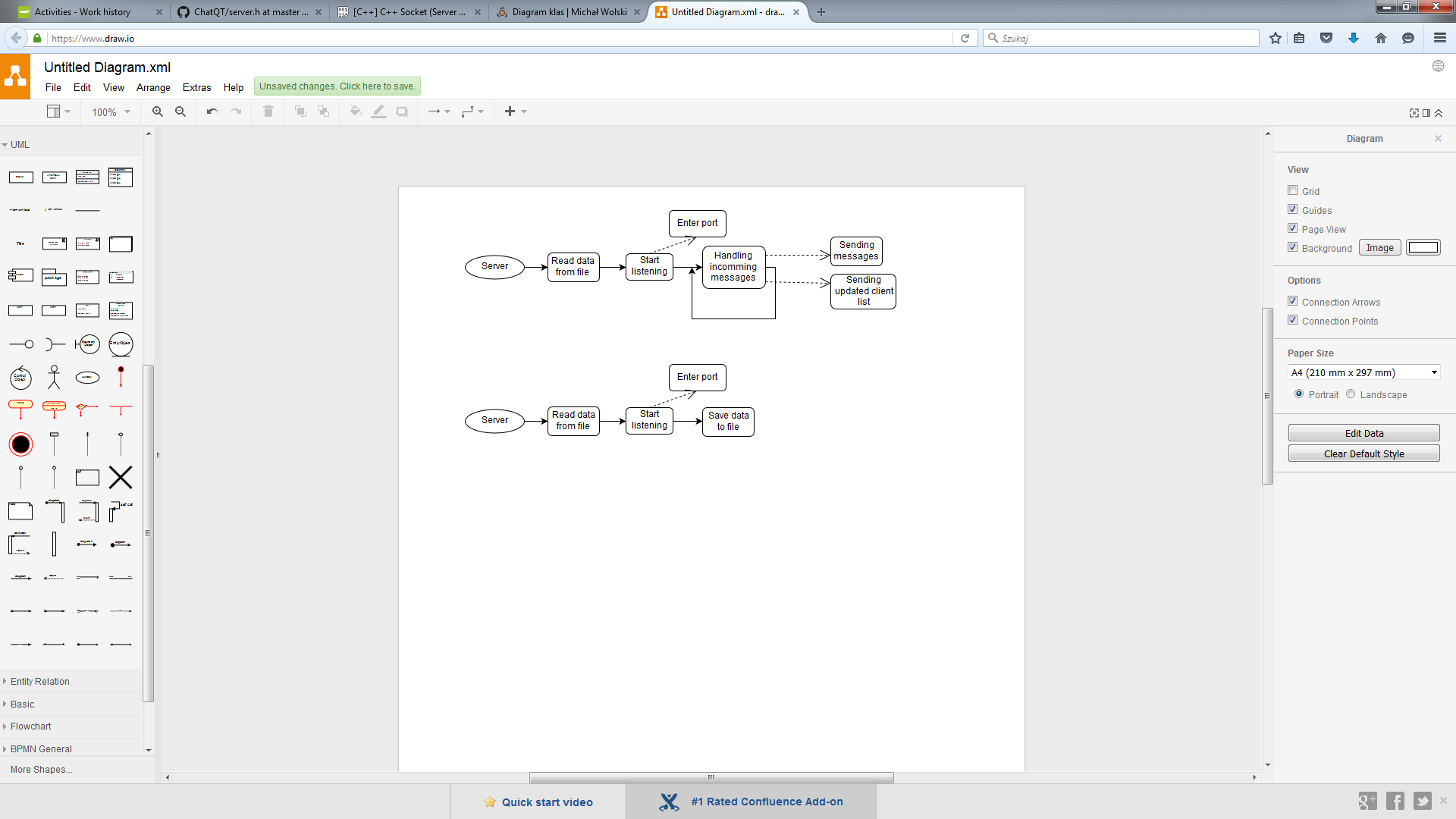
**Server UML:**

* Reading file, listening, handling and sending messages



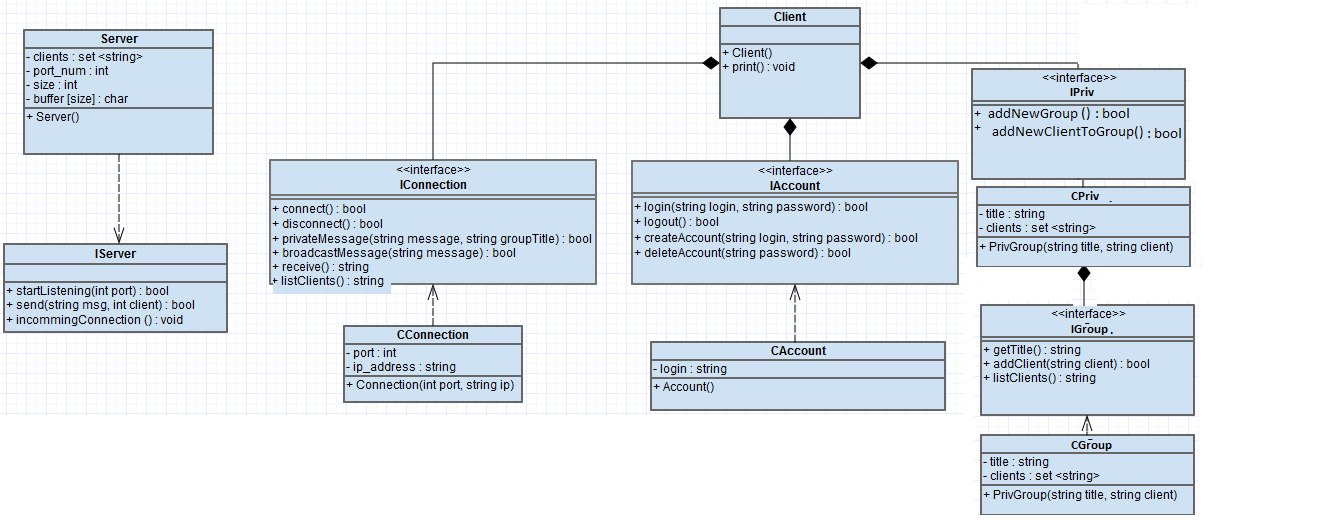
|  |  |
| --- | --- |
| Preconditions | * Available port number * External IP address or local work range * Existing file with encrypted data |
| Normal flow | * Running server with defined port number * Self-read data from file * Listening for incoming calls and messages * Handling incoming messages * Sending redirected messages and updated online clients list |
| Exceptional flow | * Port is already in use – request for type another number * File with data doesn’t exist – create new empty file * Unrecognized incoming message - print warning and ignore handling this message |

* Saving data to file and finishing work



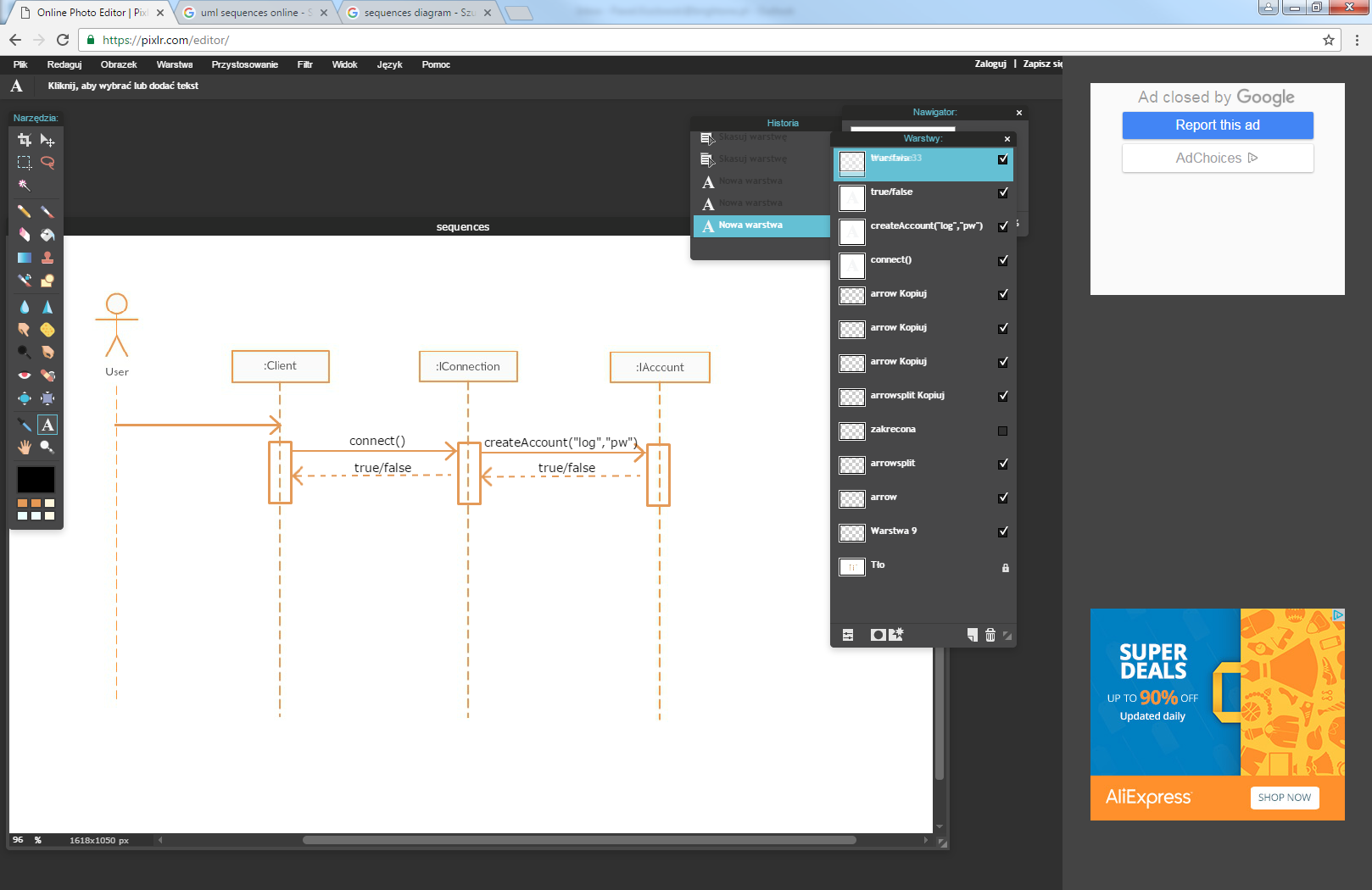
|  |  |
| --- | --- |
| Preconditions | * Available port number * External IP address or local work range |
| Normal flow | * Running server with defined port number * Self-read data from file * Listening for incoming calls and messages * Saving updated data to file |
| Exceptional flow | * Port is already in use – request for type another number * File with data doesn’t exist – create new empty file * File with data exist – override present file * Unrecognized incoming message - print warning and ignore handling this message |

**Class diagram:**

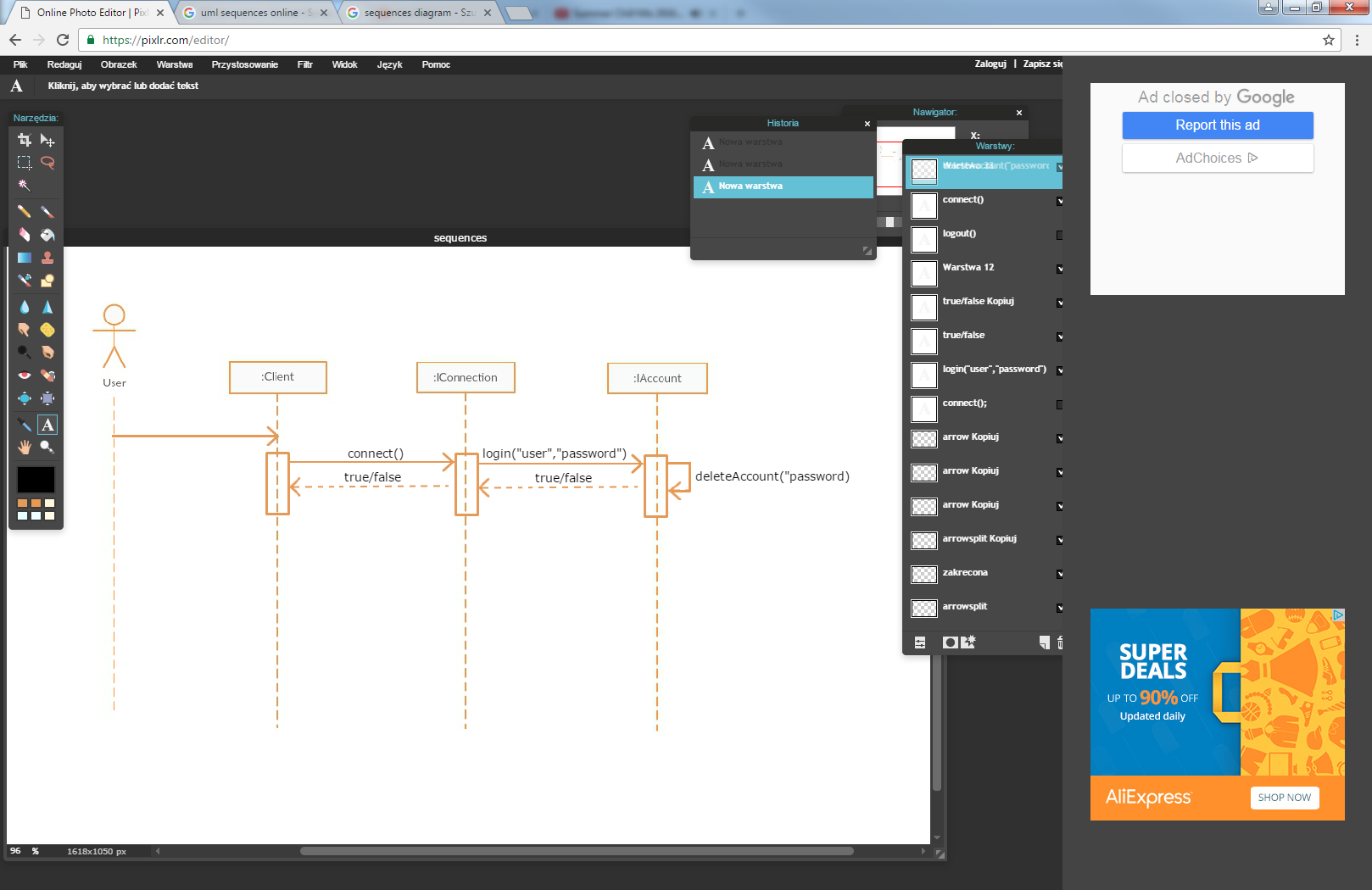
****

**Sequence diagram:**

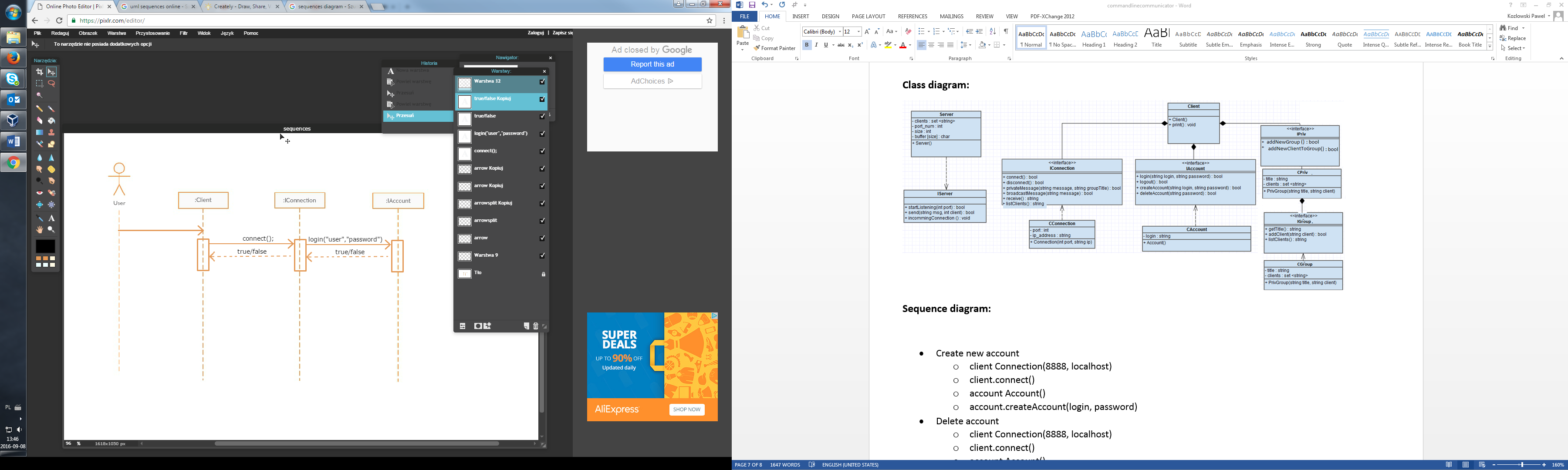
* Create new account
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.createAccount(login, password)



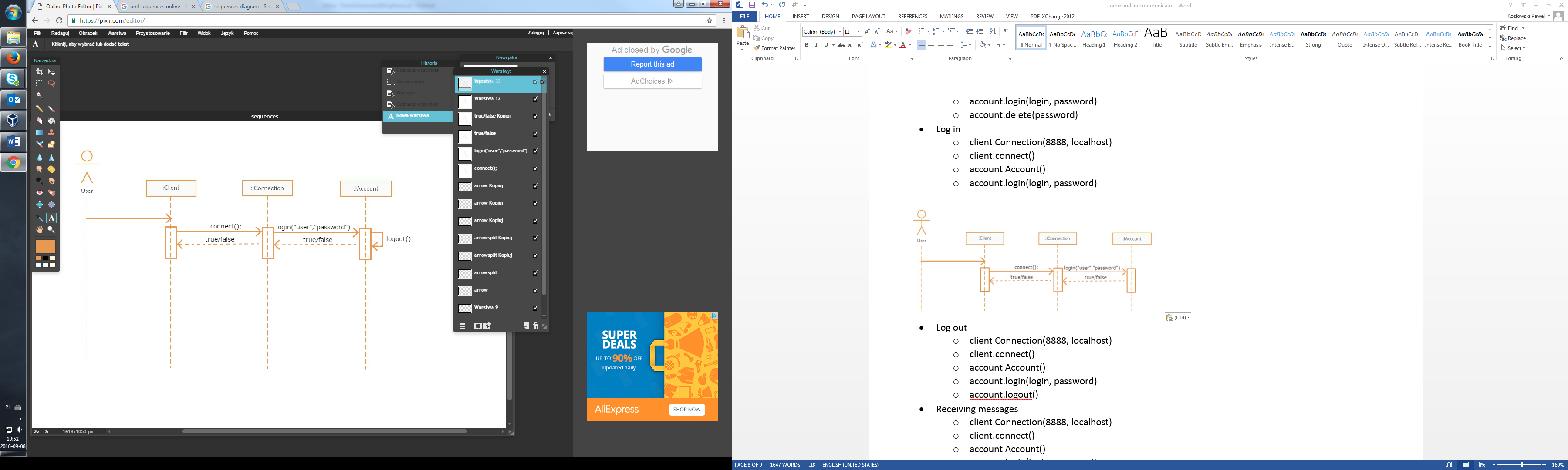
* Delete account
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)
  + account.delete(password)



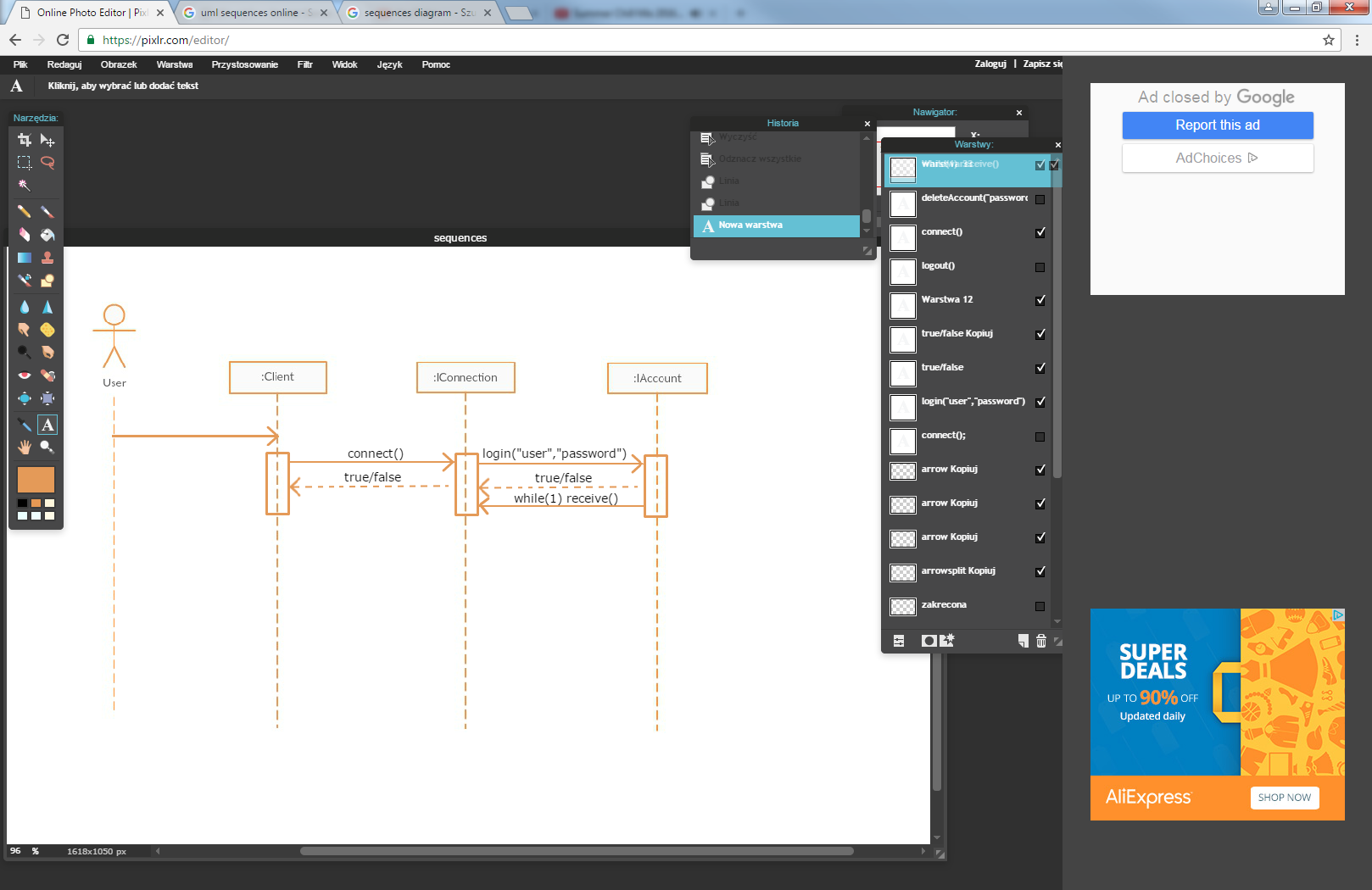
* Log in
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)



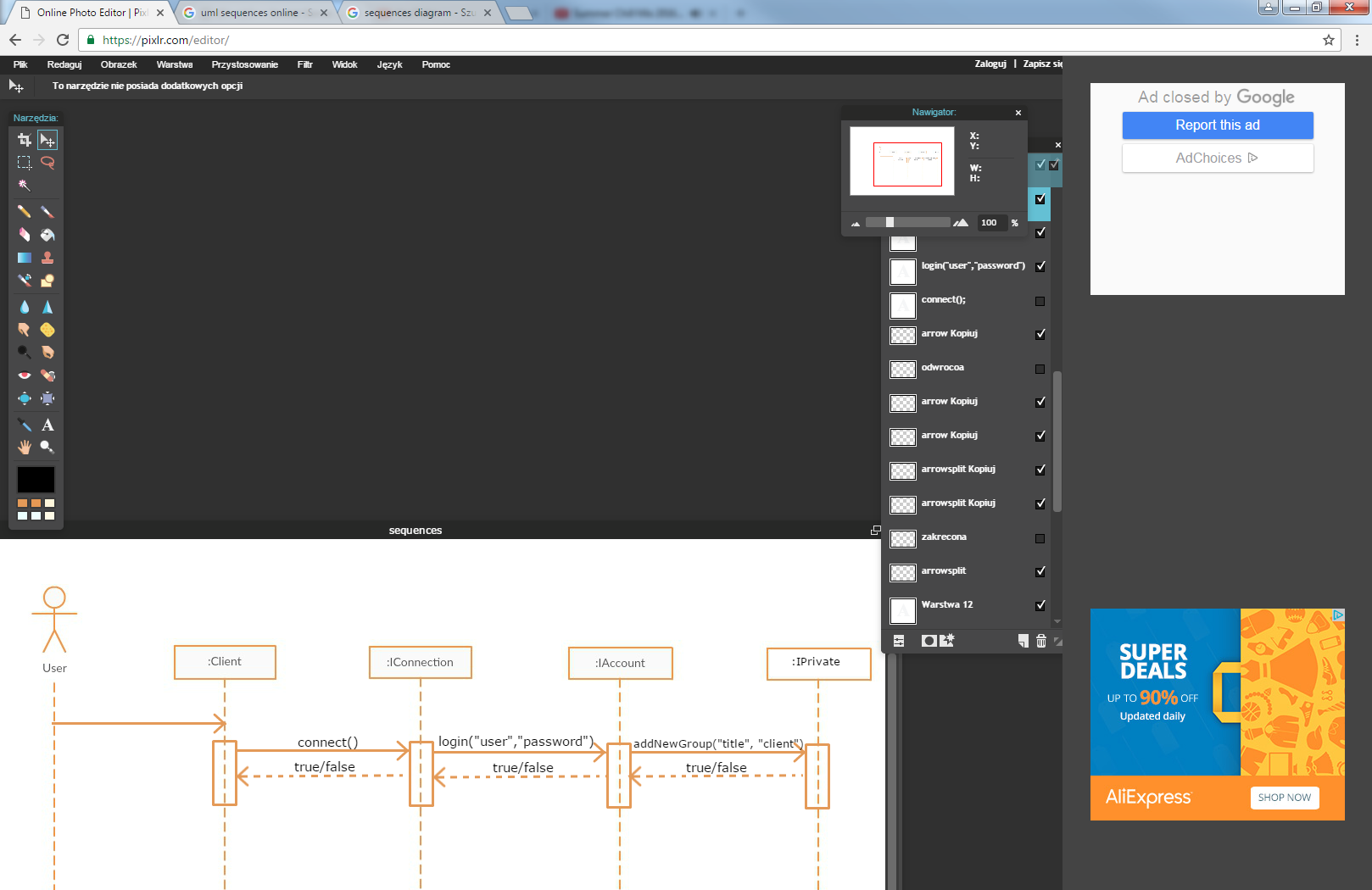
* Log out
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)
  + account.logout()



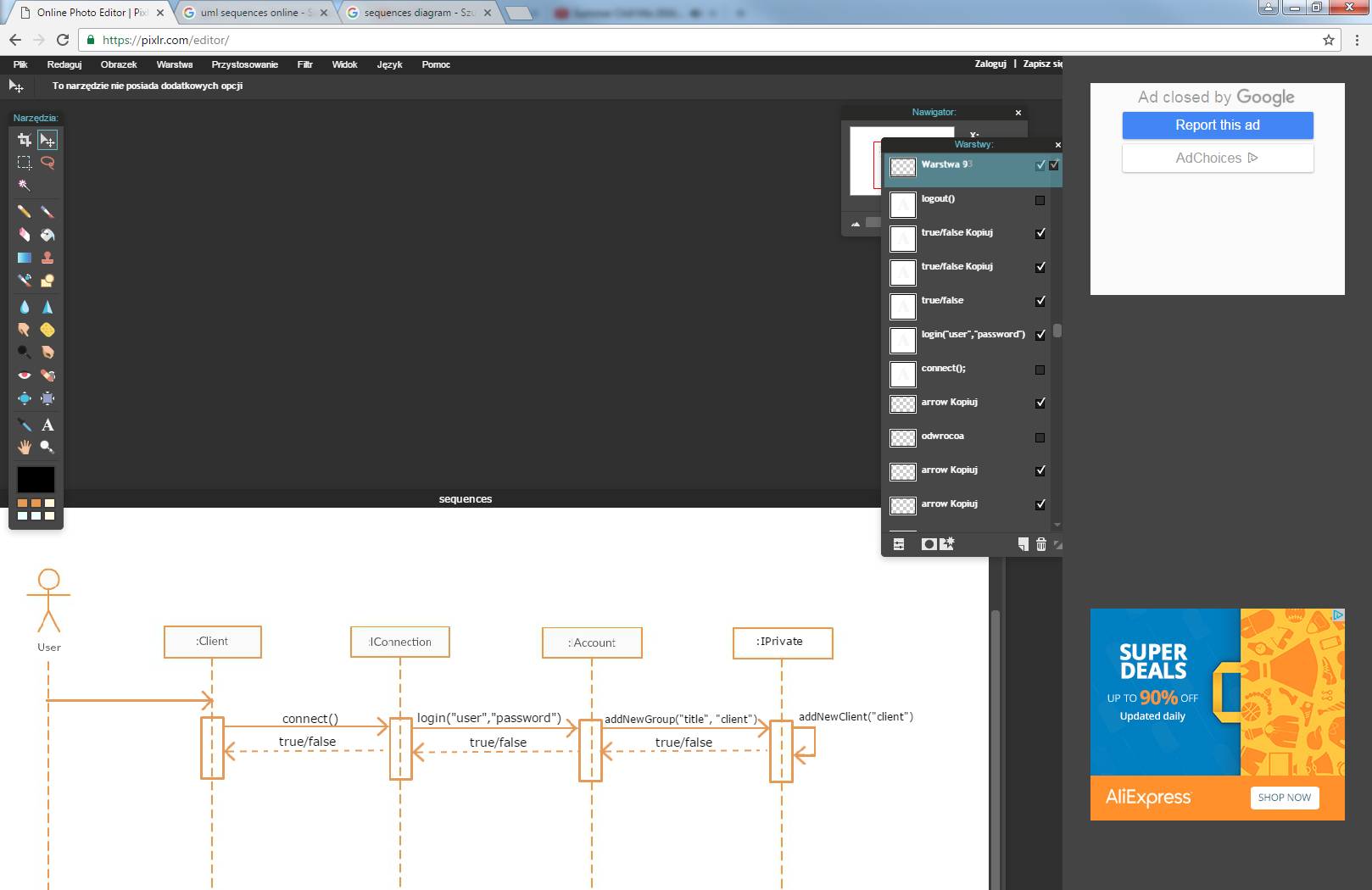
* Receiving messages
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)
  + while(1) client.receive()



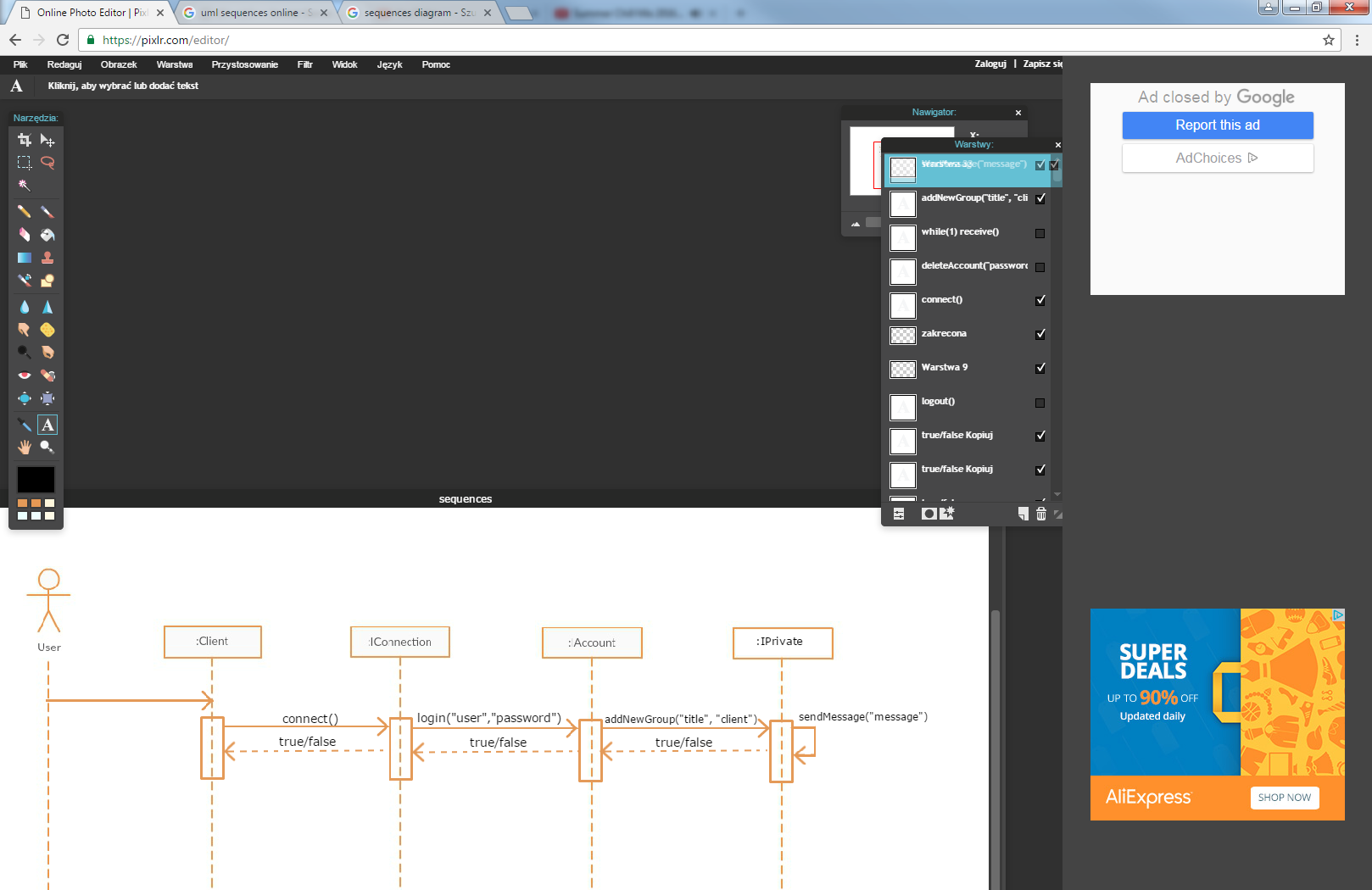
* Creating private group
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)
  + priv Private();
  + priv.addNewGroup(Title, clientName)



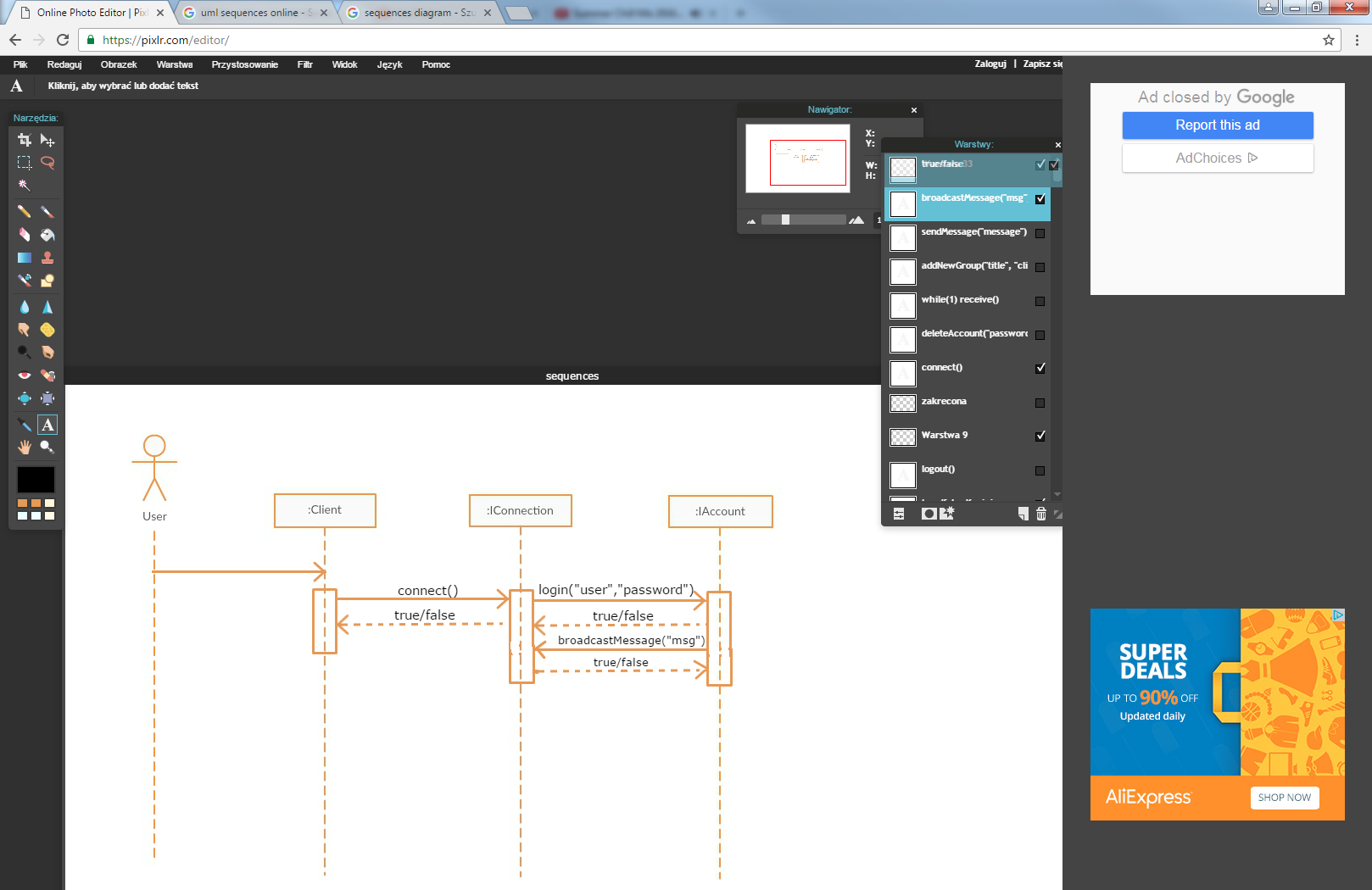
* Add other client to group
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)
  + priv Private();
  + priv.addNewGroup(Title, clientName)
  + priv.addNewClientToGroup( clientName)



* Send message to private group
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)
  + priv Private();
  + priv.addNewGroup(Title, clientName)
  + client.sendPrivateMessage(Title, TextMessage)



* Send broadcast message
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)
  + client.broadcastMessage(textMessage)



* Listing online clients
  + client Connection(8888, localhost)
  + client.connect()
  + account Account()
  + account.login(login, password)
  + client.listClients()

