

Client-Server Protocol

In our instant message application, client should be able to:

1. Send messages (to individual users or groups)
2. Receive messages (both online and offline)
3. View historical messages
4. Participate in multiple chats simultaneously
5. Manage chat groups (create, join, leave)
6. Handle user authentication (login, logout)

Commands and Actions

Define commands that the client can send to the server and the expected server responses. Communication between client and server is structured using JSON. Key message types include:

Registration:

Client -> Server:

```
{  
  
  "type": "register",  
  
  "username": "user1",  
  
}
```

Server -> Client:

```
{  
  
  "type": "registration_response",  
  
  "status": "register_success"  
  
}
```

Login:

Client -> Server:

```
{  
  "type": "login",  
  "username": "user1",  
}
```

Server -> Client:

```
{  
  "type": "login_response",  
  "status": "login_success",  
  "pending_messages": [  
    "type": " fetch_history ",  
    {"message_id": "msgid123", "sender": "user2", "content": "Hello!"}  
  ]  
}
```

Send and Receive Message:

Private Chat:

Client:

```
{  
  "type": " SEND_MESSAGE ",  
  "recipient": "user1",  
  "content": "Hi user1"  
}
```

Server:

```
{  
  "type": " MESSAGE_SENT ",  
}  
  
{
```

```
"type": " RECEIVE_MESSAGE ",
"sender": "user2",
"content": "Hi user1"
"timestamp": "2024-08-21T10:01:00Z"
}
```

Group Chat:

Client:

```
{
"type": " SEND_GROUP_MESSAGE ",
"groupname": "groupname1",
"sender": "user1",
"content": "Hi"
}
```

Server:

```
{
"type": " GROUP_MESSAGE_SENT ",
}
{
"type": " RECEIVE_GROUP_MESSAGE ",
"groupname": "groupname1",
"sender": "user1",
"content": "Hi"
"timestamp": "2024-08-21T10:02:00Z"
}
```

Protocol Grammar

Registration

registration_command = "register " (space) username

registrarion_response = "register_success" | "register_fail" (space) reason

Login

login_command = "login" (space) username

login_response = "login_success" | "login_fail" (space) reason

Logout

logout_command = "logout"

logout_response = "logout_success"

Messaging

send_command = "send" (space) recipient | group (space) message

send_response = "send_success" | "send_fail" (space) reason

receive_message = "receive" (space) sender | group (space) message (space) timestamp

Historical Message

pending_messages_command = "fetch_history" (space) recipient (space) start_time (space) end_time

pending_messages_response = "history" (space) messages

Create group

create_group_command = "create_group" (space) group_name

create_group_response = "create_group_success" (space) group_id | "create_group_fail" (space) reason

Join Group

join_group_command = "join_group" (space) group_id

join_group_response = "join_group_success" | "join_group_fail" (space) reason

Leave group

`leave_group_command = "leave_group" (space) group_id`

`leave_group_response = "leave_group_success" | "leave_group_fail" (space) reason`

State Management

Server-Side State

The server needs to maintain

- User Sessions: Which users are currently logged in.
- Message Queues: For delivering offline messages.
- Chat Histories: Messages sent in each chat (user-to-user or group).
- Group Memberships: Which users belong to which groups.

Client-Side State

The client should store:

- Session Information: The current logged-in user and possibly a session token.
- Message Cache: Recently received messages, especially for chats the user is actively participating in.
- Chat Memberships: List of groups the user is a part of.