SOCKET PROGRAMMING PROJECT
“A MULTI-USER CHAT APPLICATION”

ABOUT US

Ho Phuoc Lanh
2102114

Vu Phuong Anh
2102176

Nguyen Cao Dien
2102048

Lien Hai Nam
2102115

NAME OF PROJECT
A Multi-user Chat Application

TABLE OF CONTENTS

1 OVERVIEW

2 SERVER

3 USER AUTHENTICATION

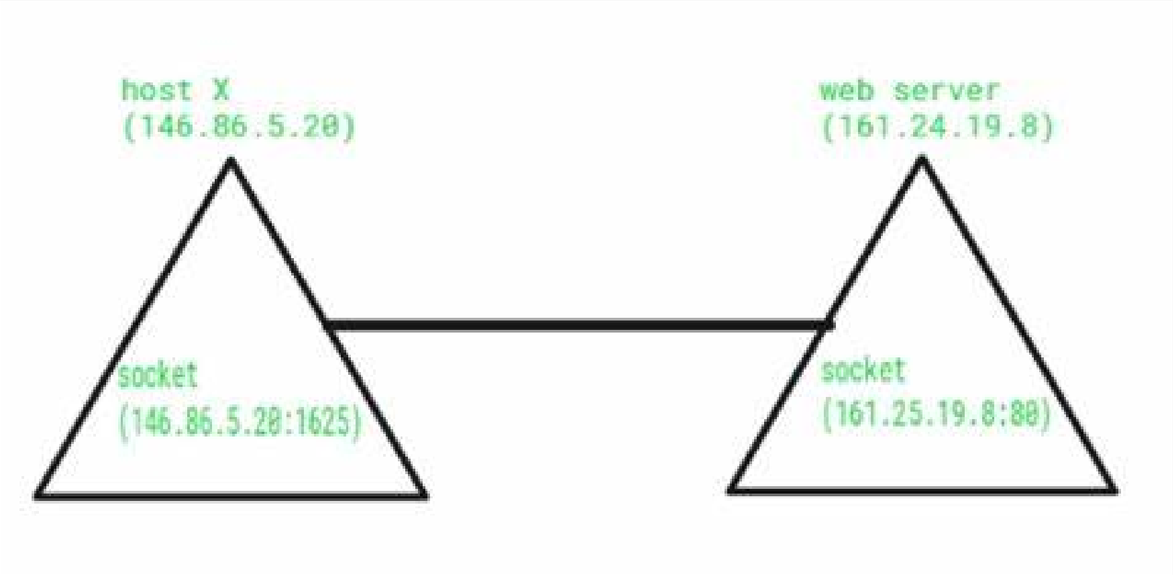
4 CHAT MANAGEMENT

5 FRONT END

6 DEMO CODE

OVERVIEW

SOCKET IN COMPUTER NETWORK



- one endpoint of a two way communication link between two programs running on the network
- a means of inter-process communication (IPC)

OVERVIEW

Socket Programming Project
“A Multi-user Chat Application”

Create chats

Join group chats

User authentication

Private messaging

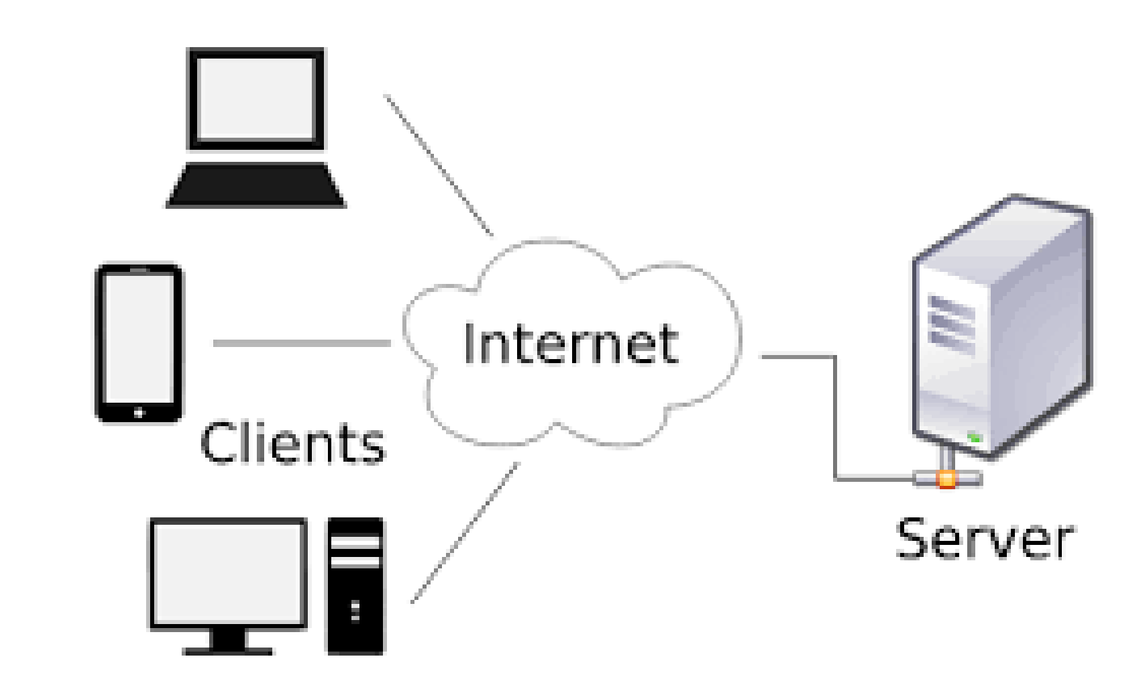
Persistent display of users on the dashboard.

Server

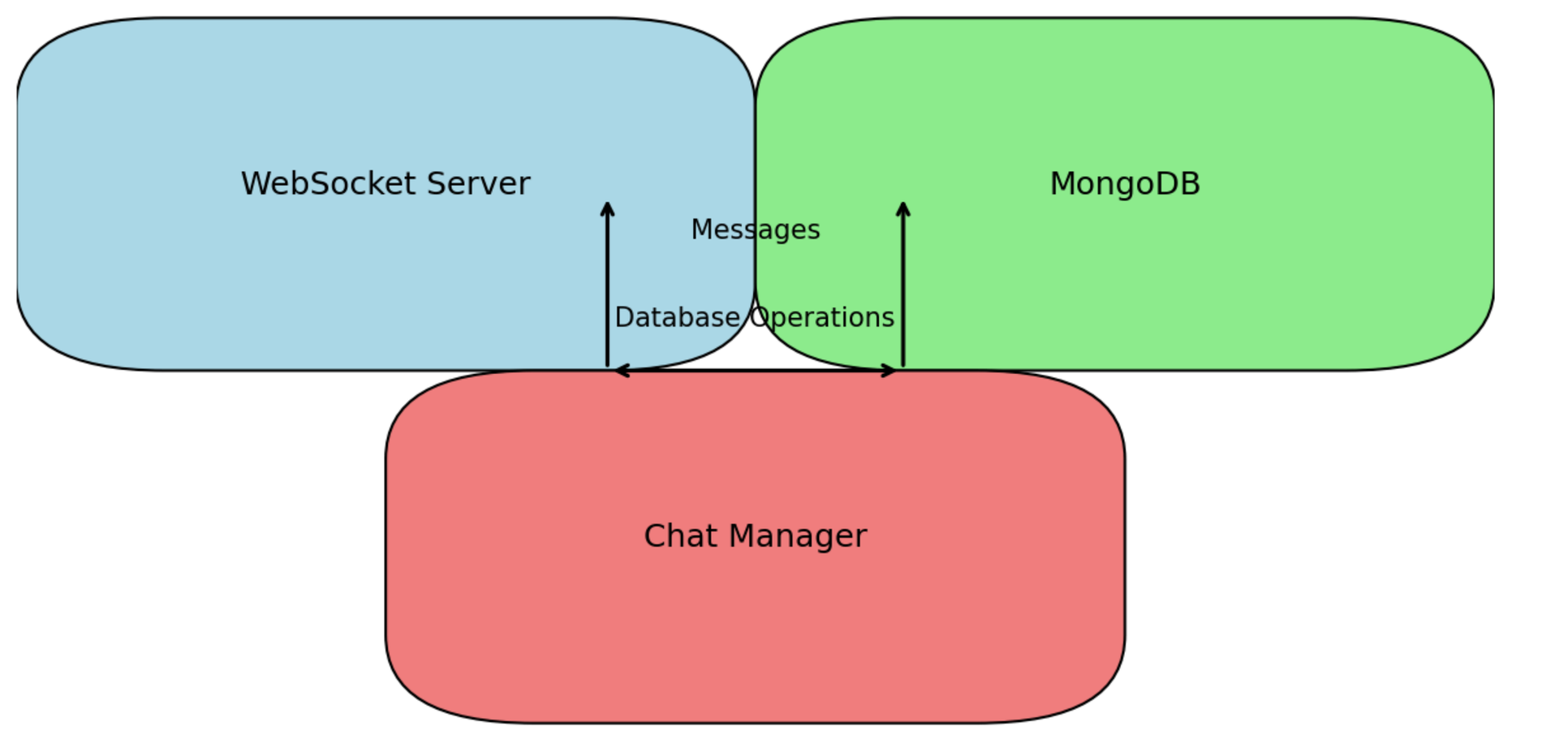
Overview of server implementation

Key components and functionalities

Importance of real-time communication

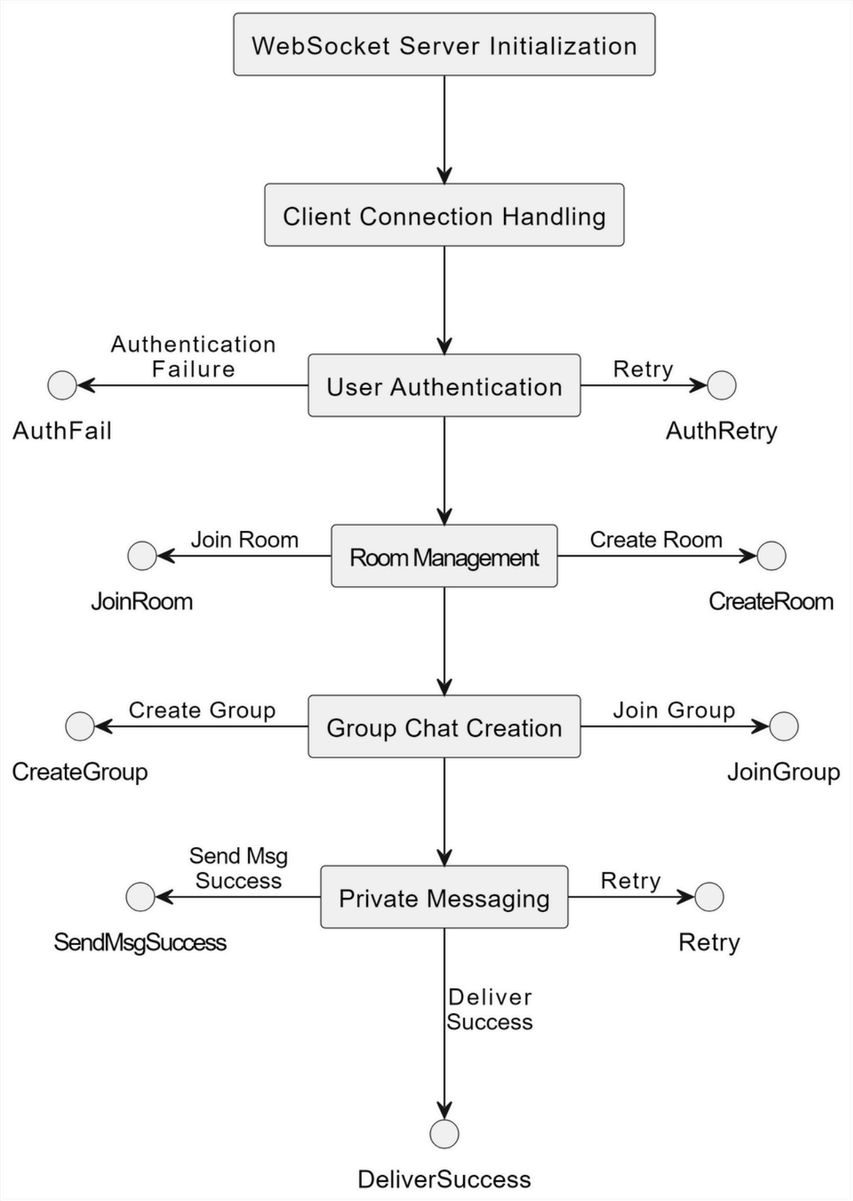


Server

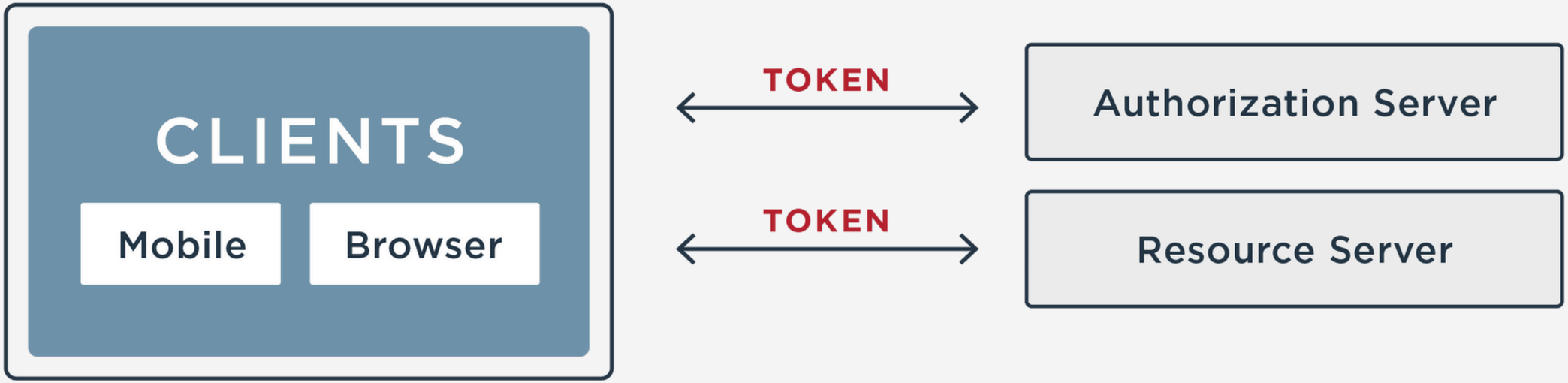

WebSocket Server
MongoDB
Messages
Database Operations
Chat Manager

Server

- WebSocket Server Initialization
- Client Connection Handling
- User Authentication
- Private Messaging
- Room Management
- Group Chat Creation



USER AUTHENTICATION



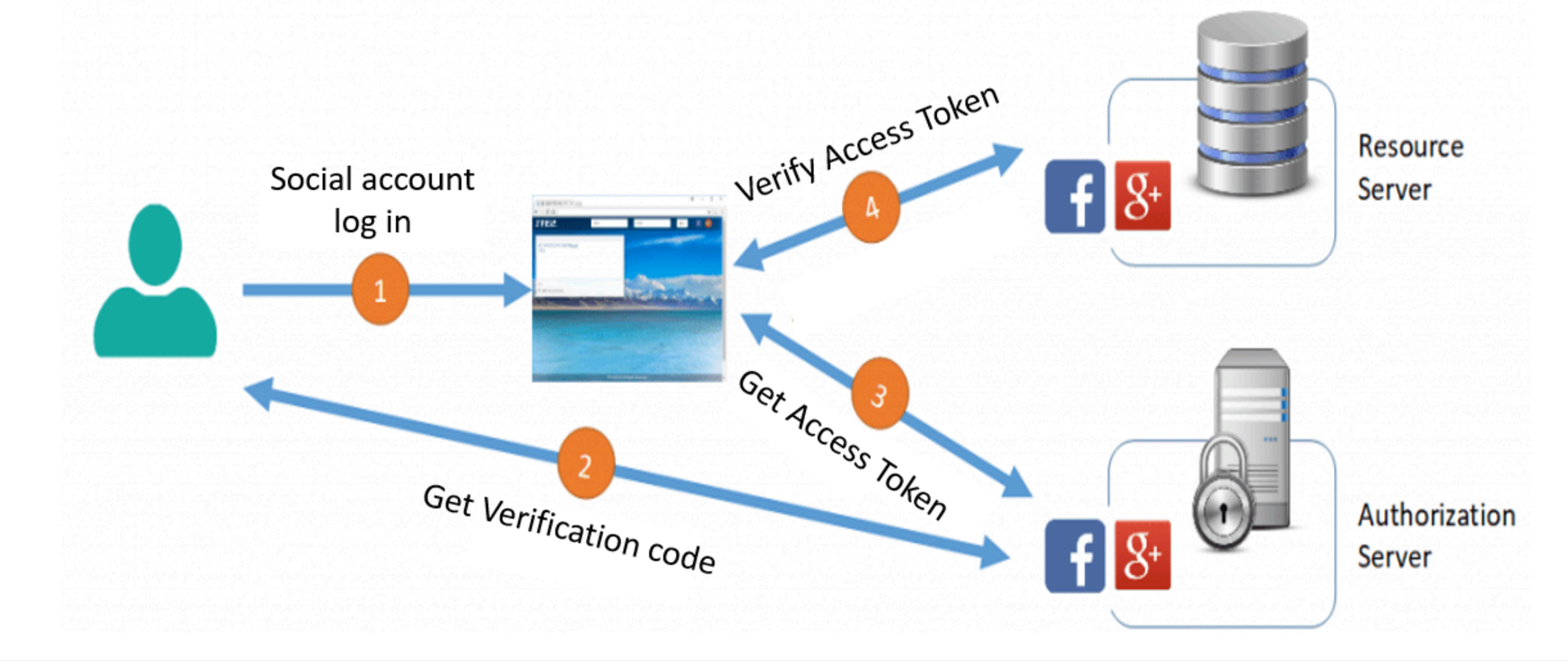
USER AUTHENTICATION

DEFINITION

Verifying User's identity and Credentials

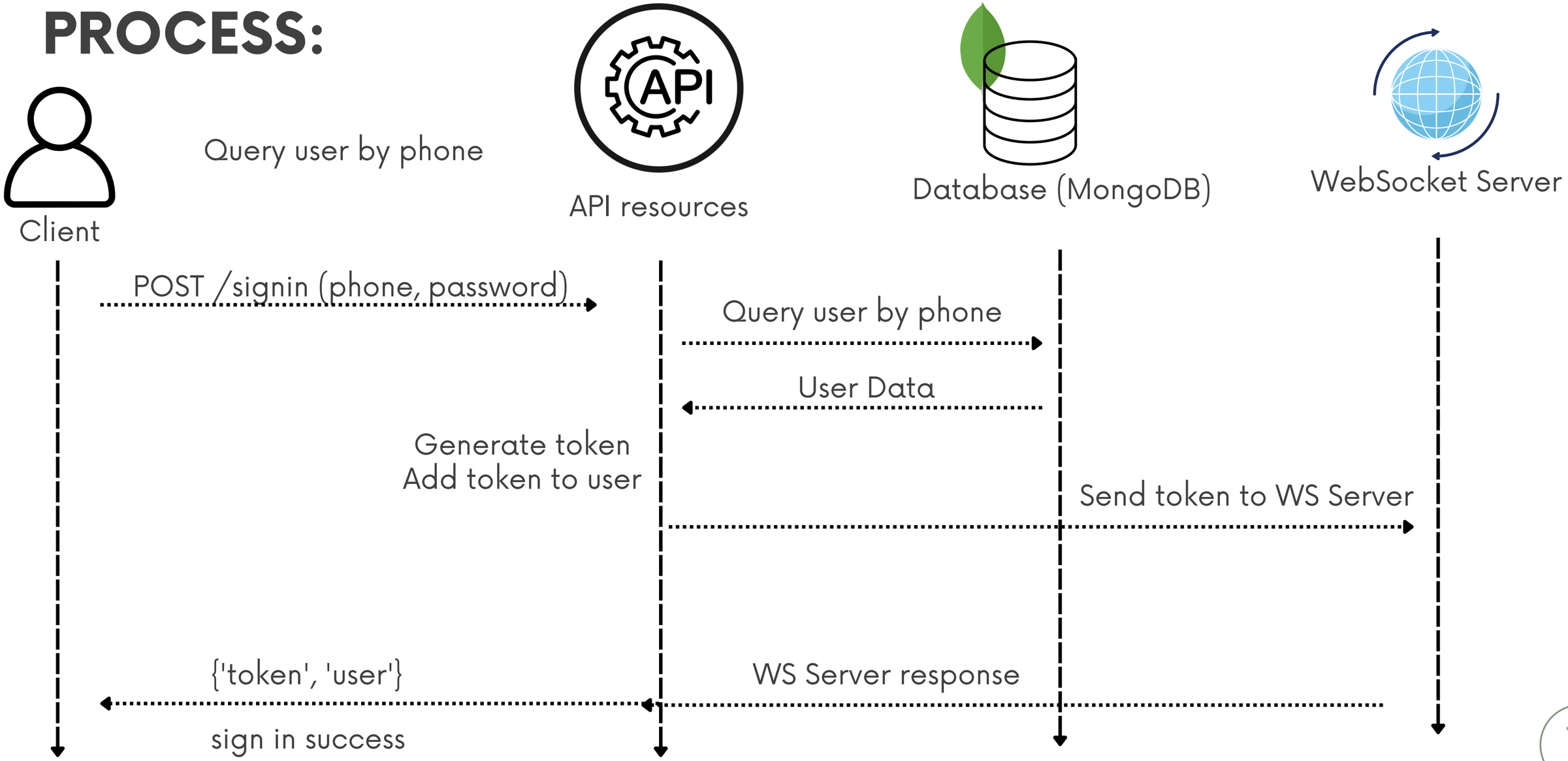
Security token

Accept users to use resources



USER AUTHENTICATION

PROCESS:



CHAT MANAGEMENT

IMPORTANCE OF CHAT MANAGEMENT

- Chat management refers to the administrative and operational processes involved in maintaining a chat system such as joining, leaving, creating, removing functions

- Enables the distribution of messages to all participants in a chat room, ensures that messages are properly routed and delivered to each user in the room.

CREATE ROOM CHAT

- The create_room function initializes a new chat room in the system
- The details of the new room, such as its name, creation timestamp, and participants, are stored in the MongoDB database.
- The newly created room is added to an in-memory list of available rooms.
- The room's details are stored in a dictionary or similar data structure within the ChatManager class, enabling fast lookups and modifications.

```
{
"name": "unique_room_name",
"created_at": "2024-06-15T12:34:56Z",
"users": [],
"metadata": {
   "description": "This is a sample chat room",
   "settings": {
      "is_private": false,
      "max_participants": 50
   }
```

```
{
   "unique_room_name": {
      "created_at": "2024-06-15T12:34:56Z",
      "users": []
   }
```

JOIN ROOM CHAT

- Allowing users to enter existing chat rooms and participate in discussions
- Once the room's existence is confirmed, the user’s WebSocket connection and username are added to the room’s participant list, which is stored in-memory.
- A JSON message is created to notify participants of the new user's entry.

```
{
    "type": "system",
    "event": "join",
    "message": "username has joined the room room_name."
}
```

- The broadcast_message function uses asyncio to handle asynchronous message sending, ensuring non-blocking communication

LEAVE ROOM CHAT

- Enabling users to exit chat rooms, with proper notifications sent to other participants.
- It identifies the user's session that needs to leave the room by matching the WebSocket connection with the room's participant list and removing that user from the list.
- The function would call a method to update the participant list in the MongoDB database and after that, a system message is broadcasted to the remaining participants, informing them of the user's departure

```
{
    "type": "system",
    "event": "leave",
    "message": "username has left the room room_name."
}
```

- The broadcast_message function uses asyncio to handle asynchronous message sending, ensuring non-blocking communication

REMOVE ROOM CHAT

- Deleting chat rooms when they are no longer needed, ensuring that resources are freed up
- To confirm the room's existence in the database and retrieve its details, the function calls the get_room_by_name method of the MongoDB class.
- A JSON message is created to inform participants about the room's deletion

```
{
    "type": "system",
    "event": "delete",
    "message": "The room room_name has been deleted."
}
```

- The broadcast_message function uses asyncio to handle asynchronous message sending, ensuring non-blocking communication

FRONT END

Key technologies used: HTML, CSS, JavaScript, jQuery, Bootstrap

Let's see demo!

THANK YOU!