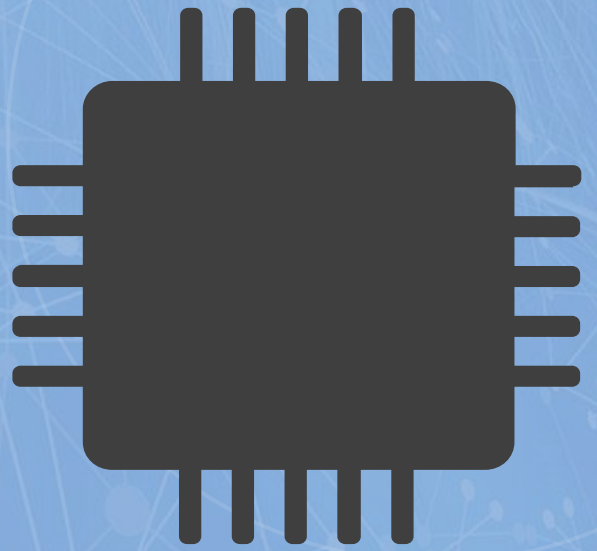
01 SOFTWARE PROJECT LAB

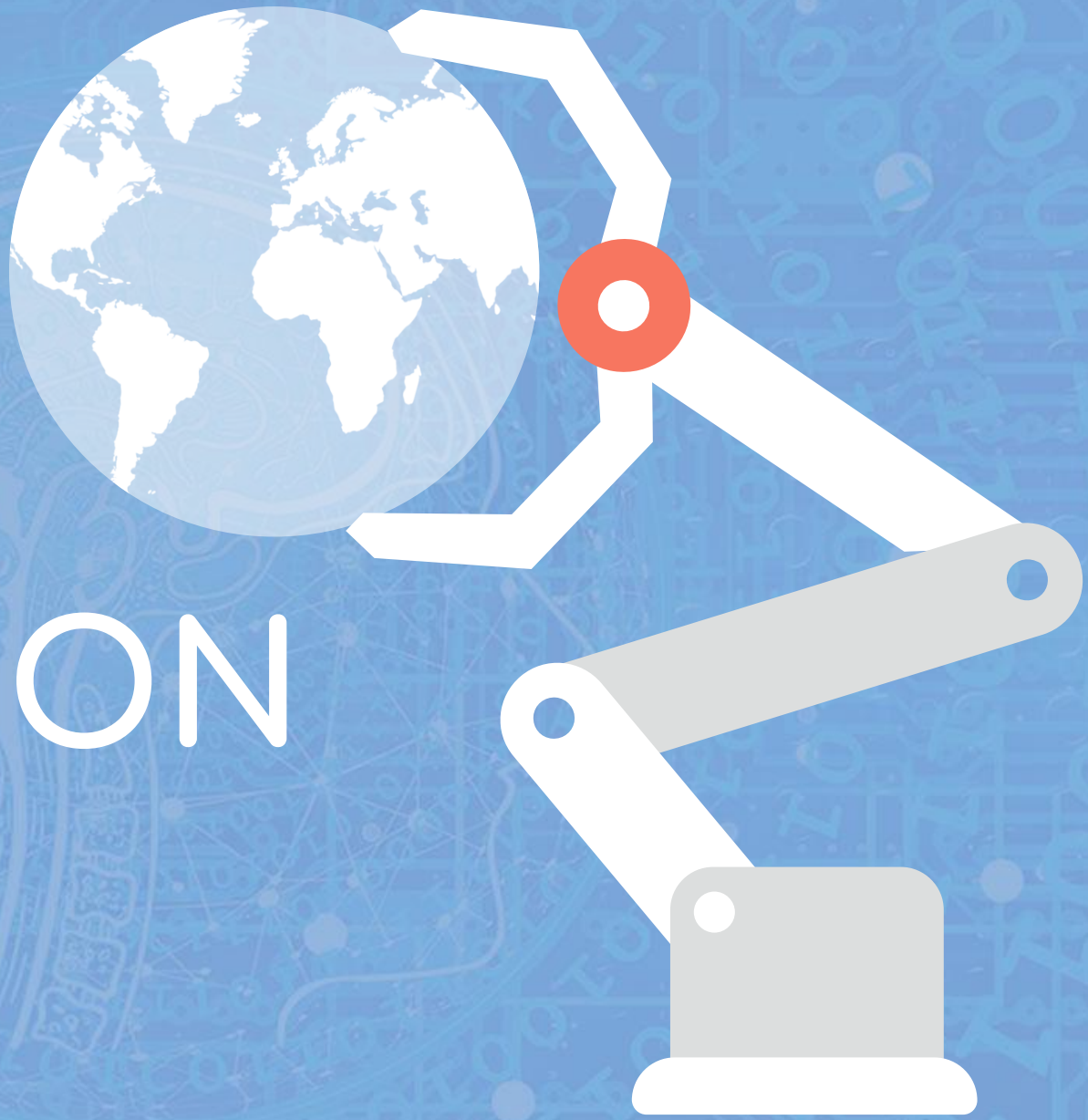


NAME : EFFAZ RAYHAN
ROLL : BSSE 1501

SUPERVISOR : DR. MD. NURUL AHAD TAWHID

Table of Contents

PROJECT DESCRIPTION



Project Description

Connecto 

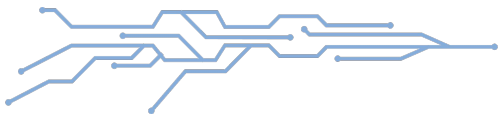
A Client-Server Based Chatting Application

The secure chat room application is a client-server based chat system that enables multiple users to communicate securely.

Project Description

03 HIGHLIGHTED FEATURES

- **Secure**
- **Chat room**
- **Client-Server**



Project Description

WHY?

Chat room

Secure

Client-Server 

Project Description

HOW?

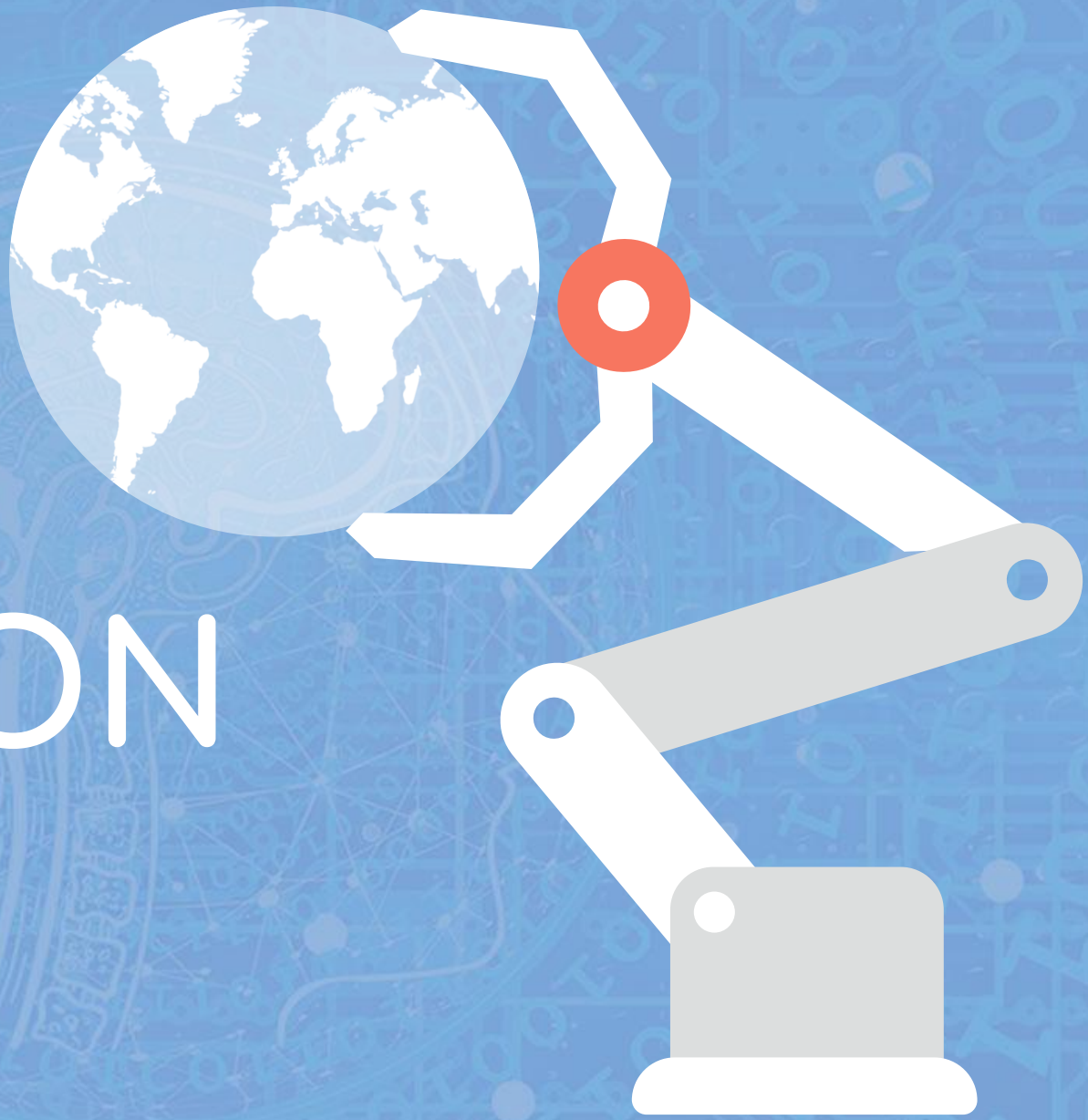
Chat room

Secure

Client-Server 

ANSWER TO WHY

PROJECT MOTIVATION



Project Motivation

Growing Need for Secure Communication
Cybersecurity threats such as data breaches and unauthorized access have highlighted the vulnerabilities of traditional communication systems. Connecto aims to address this concern by integrating robust encryption and secure user authentication, ensuring that user data and messages remain private and tamper-proof.

Simplified and Intuitive Interaction
By providing a command-based user interface, Connecto caters to users seeking a lightweight, efficient alternative to resource-intensive graphical interfaces. This approach supports seamless navigation and interaction, particularly for those comfortable with terminal-based systems.

Support for Collaborative Spaces
Modern workflows often require team-based discussions in dedicated environments. Connecto's chatroom management feature facilitates real-time collaboration, offering a persistent and structured platform for discussions.

Accessibility and Availability
Connecto incorporates features such as online availability checking, enabling users to see who is active and facilitating instant communication. This real-time connectivity fosters productivity and minimizes delays.

ANSWER TO HOW

WORKING METHODOLOGY

WORKING METHODOLOGY

EASY TO OPERATE

TO ESTABLISH COMMUNICATION BETWEEN CO-WORKERS AND FRIENDS

WORKS ON SAME LAN

NO INTERNET DEPENDENCY

SECURE

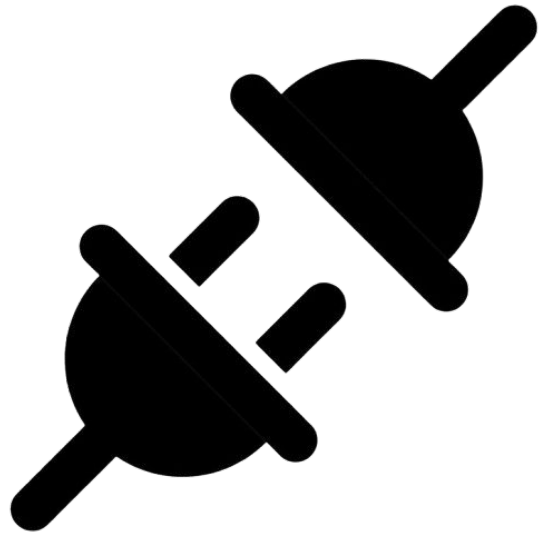
WORKING METHODOLOGY

- SOCKET PROGRAMMING
- SHA-512 HASHING ALGORITHM
- TERMINAL BASED EASY USER INTERFACE

TECHNOLOGY

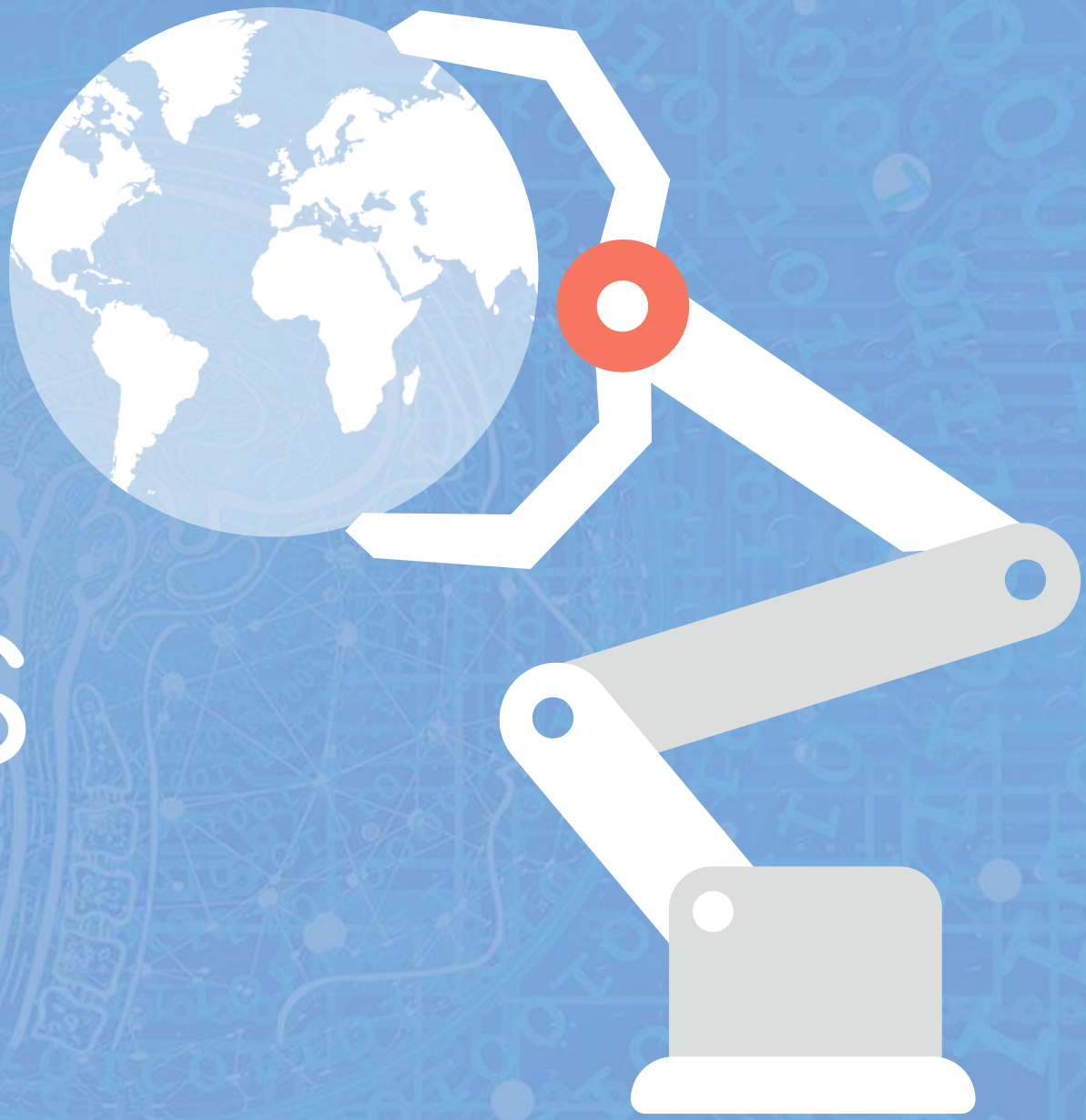
TECHNOLOGY

C++ PROGRAMMING LANGUAGE

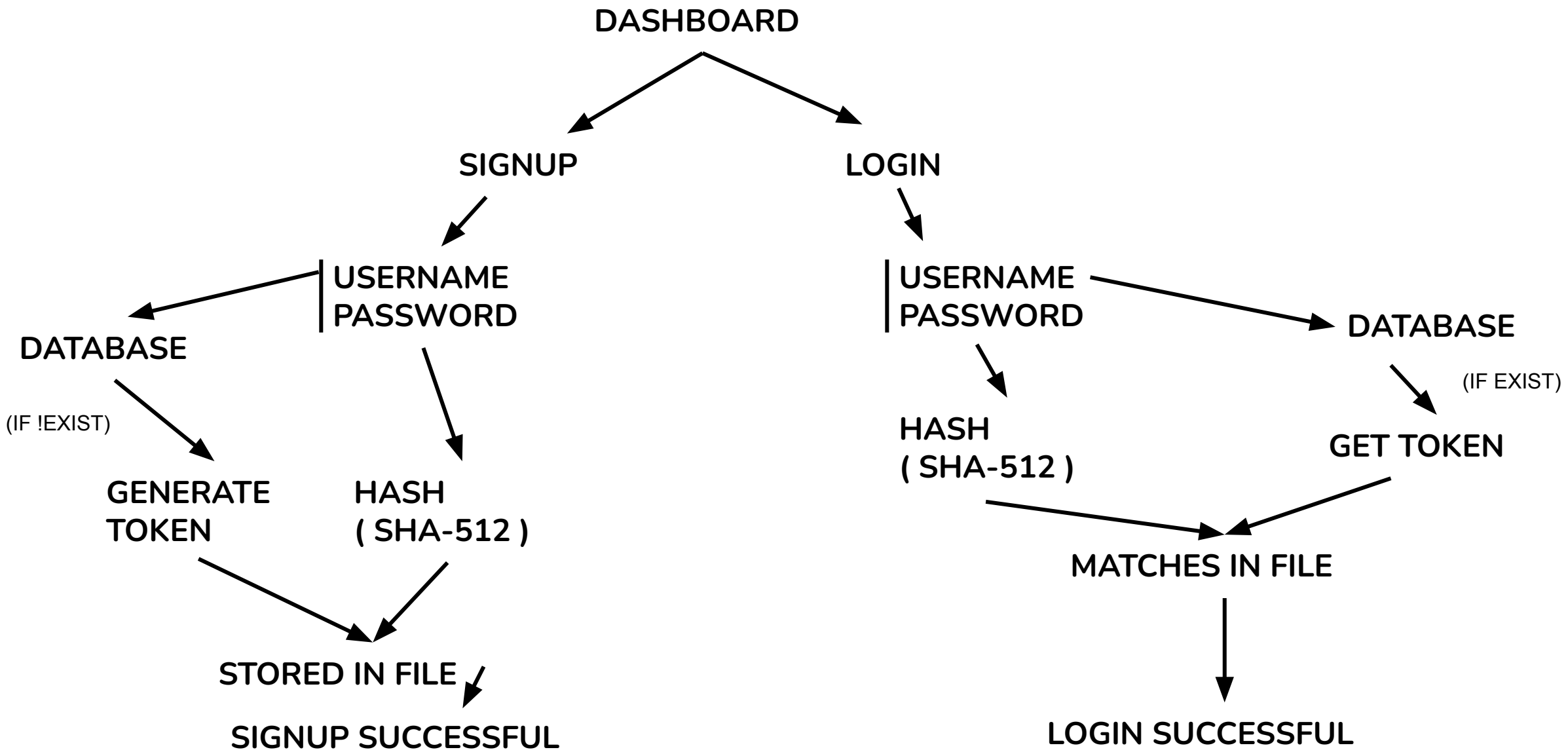


SOCKET PROGRAMMING

PROGRESS



PROGRESS


DASHBOARD
SIGNUP
LOGIN
USERNAME
PASSWORD
USERNAME
PASSWORD
DATABASE
DATABASE
(IF !EXIST)
(IF EXIST)
GENERATE
TOKEN
HASH
(SHA-512)
HASH
(SHA-512)
GET TOKEN
MATCHES IN FILE
STORED IN FILE
SIGNUP SUCCESSFUL
LOGIN SUCCESSFUL

PROGRESS

```
void Connecto()
{
    int choice;
    cout << "Enter Your Choice" <<
endl;
    cout << "1. Login\n2. Signup\n";
    cin >> choice;
    switch (choice)
    {
    case 1:
        clientLogin();
        break;
    case 2:
        clientSignup();
        break;
    default:
        cout << "Invalid input. Try
again!" << endl;
        Connecto();
        break;
    }
}
```

```
// Token
string generateToken()
{
    srand(time(0));
    string token = to_string(rand());
    return token;
}
```

```
// Hash
string hashdata(string s)
{
    vector<unsigned long> block;
    block = convert_to_binary(s);
    block = pad_to_512bits(block);
    block = resize_block(block);
    string hash = compute_hash(block);
    return hash;
}
```

```
// Username Check
bool userNameExists(string user)
{
    ifstream
userFile("user_tokens.txt");
    string username, token;
    while (userFile >> username >>
token)
    {
        if (username == user)
        {
            return true;
        }
    }
    return false;
}
```

PROGRESS

```cpp
// Store the username and password
bool storeUser(string user, string
pass)
{
    if (userNameExists(user))
    {
        cout << "Username already
exists!" << endl;
        return false;
    }

    // Generate and store token
    string token = generateToken();
    ofstream
userFile("user_tokens.txt", ios::app);
    userFile << user << " " << token <<
endl;
    userFile.close();

    // Store hashed password and token
    ofstream
passFile("token_passwords.txt",
ios::app);
    passFile << token << " " <<
hashdata(pass) << endl;
    passFile.close();

    cout << "Signup successful!" <<
endl;
    return true;
}
```

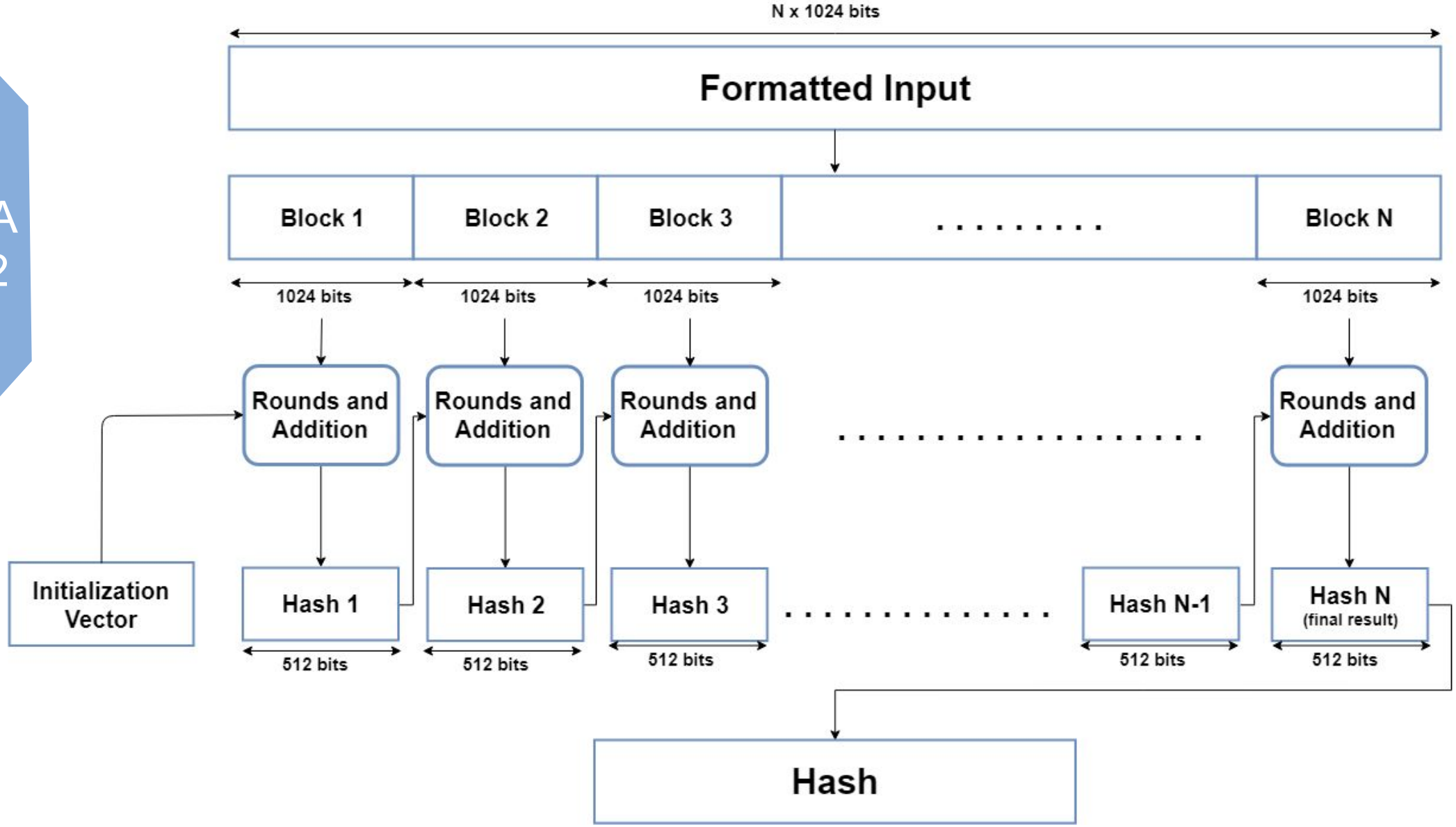
```cpp
// Check Pass
bool checkPass(int token, string hash)
{
    ifstream
passFile("token_passwords.txt");
    string storedToken, storedHash;
    while (passFile >> storedToken >>
storedHash)
    {
        if (to_string(token) ==
storedToken && hash == storedHash)
        {
            return true;
        }
    }
    return false;
}
```

```cpp
// Signup
void clientSignup()
{
    string user, pass;
    cout << "Enter Username: ";
    cin >> user;
    cout << "Enter Password: ";
    cin >> pass;

    if (storeUser(user, pass))
    {
        cout << "Signup successful. You
can now login." << endl;
    }
    else
    {
        cout << "Signup failed. Try
again." << endl;
    }
}
```

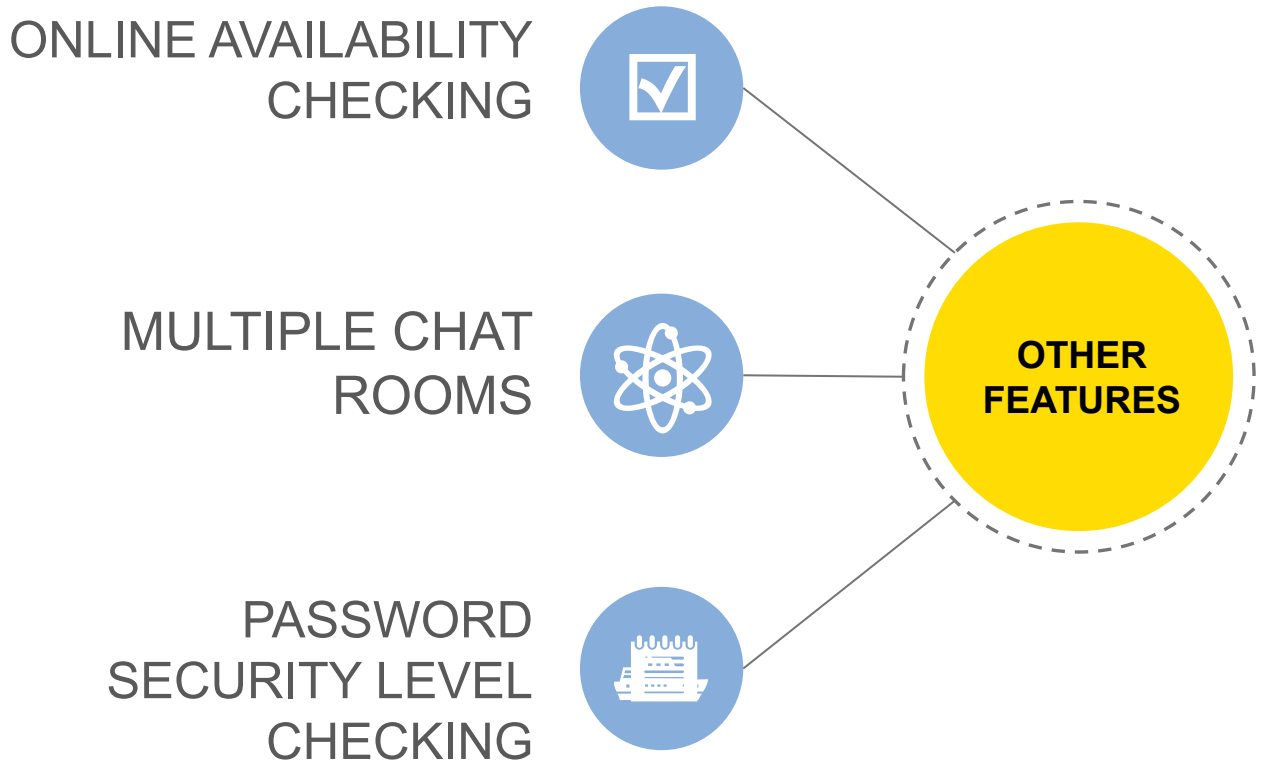
PROGRESS

SHA 512

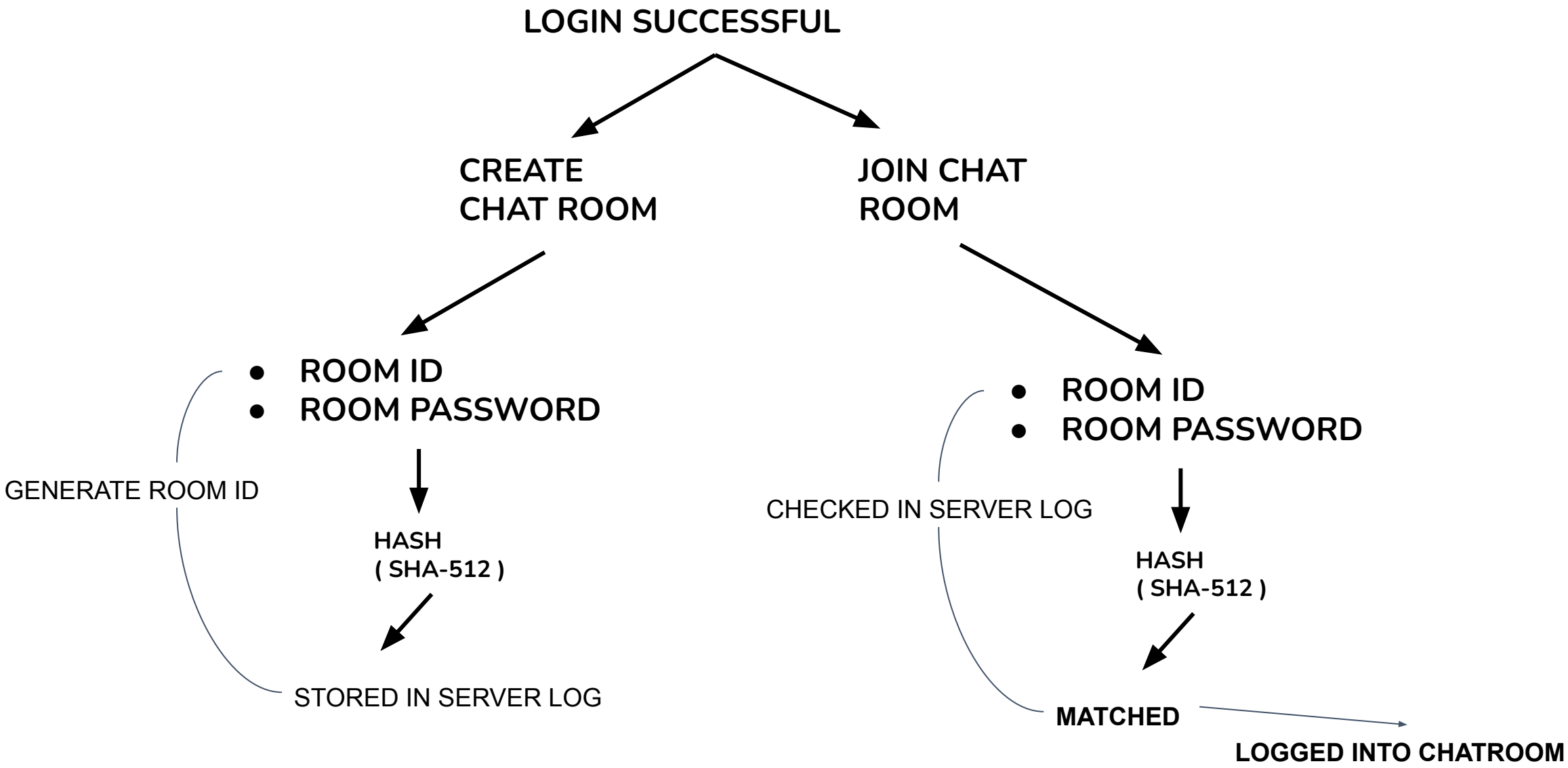


FURTHER PLAN

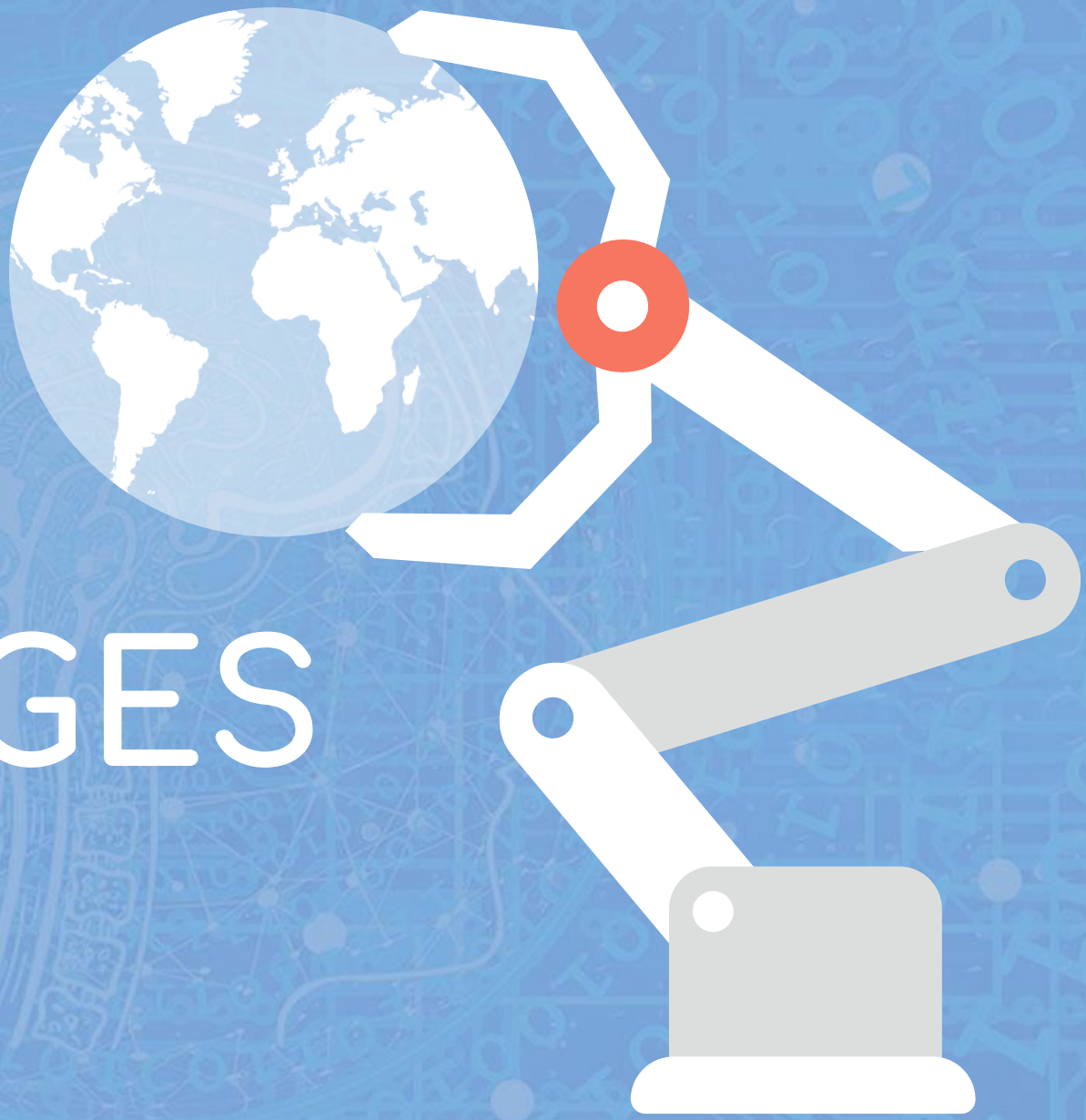
FURTHER PLAN



FURTHER PLAN



CHALLENGES



CHALLENGES

CHALLENGES SO FAR

- First Time Working on a Big Project
- Explore Socket Programming
- Study encryption and decryption algorithms
- Implement RAW SHA-512 code

FURTHER CHALLENGES

- Learn MultiThreading
- Implement Secure Encryption Algorithm For Secure Communication
- Add Password Security Level Checking
- Merge The Dashboard with server-client

THANK YOU

MY GITHUB REPOSITORY :
https://github.com/effazrayhan/spl_one

NAME : EFFAZ RAYHAN
ROLL : BSSE 1501