Project Name: ENIGMA

Problem Statement

In today's digital world, communication within an organization plays a vital role in maintaining smooth operations, improving productivity, and ensuring coordination among team members. However, many organizations face the challenge of securing sensitive internal communication from unauthorized access. Simple messaging systems may not offer the necessary protection against potential data breaches or unauthorized access.

Code:

```
#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

#include <windows.h>

#include <ctime>

#include <cctype>      //for lowercase

using namespace std;

int userCount = 0;


class User {

private:

    string organization;

    string username;

    string password;

    User* next;
    User* prev;
public:
    User()
    {
        username = " ";
        password = " ";
        next = NULL;
        prev = NULL;
    }
    User(string username, string password, string orgName)
    {
        this->username = username;
        this->password = password;
        this->organization = orgName;
        next = NULL;
        prev = NULL;
    }

    string GetUsername() {
        return username;
    }
    void SetUsername(string username) {
        this->username = username;
    }
```

```
string GetPassword() {

    return password;

}

void SetPassword(string password) {

    this->password = password;

}

string GetOrganization() {

    return organization;

}

void SetOrganization(string organization) {

    this->organization = organization;

}



User* GetNext() {

    return next;

}

void SetNext(User* next) {

    this->next = next;

}

User* GetPrev() {

    return prev;

}

void SetPrev(User* prev) {
        this->prev = prev;

    }



};

class UserList {

private:

    User* first;

    User* last;



public:

    UserList()

    {

        first = last = NULL;

    }

    User* GetFirst() {

        return first;

    }

    User* GetLast() {

        return last;

    }

    void SetFirst(User* first) {

        this->first = first;

    }

    void SetLast(User* last) {
        this->last = last;
    }


    void addUser(string org, string username, string password) {
        User* newUser = new User();
        newUser->SetUsername(username);
        newUser->SetPassword(password);
        newUser->SetOrganization(org);


        if (first == NULL)
        {
            first = newUser;
            last = newUser;
        }
        else {
            newUser->SetPrev(last);
            last->SetNext(newUser);
            last = newUser;
        }
        userCount++;
    }


};
class Admin {
```

```
private:
    string code;
    string password;
    string orgName;
    Admin* prev;
    Admin* next;
public:
    Admin() {
        code = " ";
        password = " ";
        orgName = " ";
    }
    Admin(string code, string password, string orgName) {
        this->code = code;
        this->password = password;
        this->orgName = orgName;
    }
    Admin* GetNext() {
        return next;
    }
    void SetNext(Admin* next) {
        this->next = next;
    }
    Admin* GetPrev() {
        return prev;
    }
    void SetPrev(Admin* prev) {
        this->prev = prev;
    }
    string GetOrgName() {
        return orgName;
    }
    string GetCode() {
        return code;
    }
    string GetPassword() {
        return password;
    }
};
class AdminList {
private:
    Admin* first;
    Admin* last;
public:
    AdminList()
    {
        first = last = NULL;
    }
    Admin* GetFirst() {
        return first;
    }
    Admin* GetLast() {
        return last;
    }
    void SetFirst(Admin* first) {
        this->first = first;
    }
    void SetLast(Admin* last) {
        this->last = last;
    }

};
class node
{
private:
    char data;
    node* next;
public:
    node()
    {
        data = NULL;
        next = NULL;
    }
    char getdata()
    {
        return data;
    }
    node* getnext()
    {
        return next;
    }
    void setdata(char data)
    {
        this->data = data;
    }
    void setnext(node* next)
    {
        this->next = next;
    }
};

class stack
{
private:
    node* top;
public:
```

```
stack()
{
   top = NULL;
}
void push(char data)
{
   node* newnode = new node();
   newnode->setdata(data);
   newnode->setnext(top);
   top = newnode;

}
bool isempty()
{
   if (top == NULL)
      return true;
   else
      return false;
}

char pop() {
   char a;

   a = top->getdata();
```

```
        node* temp = top;

        top = top->getnext();

        delete temp; // Free memory

        return a;

    }


};


class Enigma {

private:

    User* user;

    UserList* list = new UserList();

    AdminList* admins = new AdminList();

    stack stk;

    Admin* admin;

public:

    string ToLower(string str) {

        string newStr;

        for (char i : str) {

            i = tolower(i);

            newStr += i;

        }

        return newStr;
```

```
}
void readAdmins(const string orgName) {
    string code, password;
    ifstream file(orgName + ".txt"); // Open the file using the organization name
    if (!file.is_open()) {
        cout << "Error: Could not open file " << orgName << endl;
        return;
    }
    string line;
    getline(file, line);
    size_t delimPos = line.find(':');
    if (delimPos != string::npos) {
        code = line.substr(0, delimPos);
        password = line.substr(delimPos + 1);
    }

    file.close();
    Admin* newAdmin = new Admin(code,password,orgName);

    if (admins->GetFirst() == NULL) {
        admins->SetFirst(newAdmin);
        admins->SetLast(newAdmin);
    }
    else {
            admins->GetLast()->SetNext(newAdmin);

            newAdmin->SetPrev(admins->GetLast());

            admins->SetLast(newAdmin);

        }

    }

    bool checkAdmins(const string orgName,string code,string password) {

        Admin* temp = admins->GetFirst();

        while (temp != NULL) {

            if (temp->GetOrgName() == orgName && temp->GetCode() == code && temp-
>GetPassword() == password) {

                admin = temp;

                return true;

            }

            temp = temp->GetNext();

        }

        return false;

    }


    void AdminLogin()

    {


        string code, password, orgName;

        system("cls");

        cout << "================ ADMIN LOGIN ================\n";

        cout << "Enter organization name: ";
        getline(cin, orgName);

        cout << "Enter organization code : ";

        getline(cin, code);

        cout << "Enter admin password: ";

        getline(cin, password);


        if (checkAdmins(ToLower(orgName), code, password))

        {

            cout << endl << "Admin login successful!\n";

            cout << "Welcome "  << ToLower(orgName) << "'s Admin!\n";

            Sleep(2000);

            AdminMenu();

        }

        else

        {

            cout << "Invalid admin credentials! Returning to main menu.\n";

            Sleep(1500);

        }

    }


    bool findusername(string orgName, string username) {        //to check if the username is
already present in the list


        User* temp = list->GetFirst();

        while (temp != NULL) {
            if (temp->GetOrganization() == ToLower(orgName) && temp->GetUsername() ==
username) {
                return true;
            }
            temp = temp->GetNext();
        }
        return false;
    }


    bool checkusername(string username)         //to check if the username follows the
constraints
    {
        int size = username.length();


        if (size >= 8 && size <= 12)
        {
            bool hasUppercase = false;
            bool hasDigit = false;


            for (int i = 0; i < size; i++)
            {
                if (username[i] >= 'A' && username[i] <= 'Z') {
                    hasUppercase = true;
                }
                if (username[i] >= '0' && username[i] <= '9')
                {
                    hasDigit = true;
                }
                if (hasUppercase && hasDigit)
                {
                    return true;
                }
            }
            return false;
        }
        else
        {
            return false;
        }
    }

    bool checkpassword(string password)
    {
        int size = password.length();
        if (size >= 8 && size <= 12)
            return true;
        else
            return false;
    }
```

```
    void ShowUsers() {                        //READ

        system("cls");

        cout << "=================== USERS LIST ===================" << endl;

        cout << "--------------------------------------------------" << endl;

        cout << "USERNAME\t\t\tPASSWORD" << endl;

        cout << "--------------------------------------------------" << endl;

        User* temp = list->GetFirst();

        while (temp != NULL) {

            if (temp->GetOrganization() == admin->GetOrgName() && temp-
>GetUsername().length()>=8) {

                cout << temp->GetUsername() << "\t\t\t" << temp->GetPassword() << endl;

            }

            temp = temp->GetNext();

        }

        cout << "\nPress any key to continue ";

        cin.ignore();

    }


    void registerUser()                      //CREATE

    {

        string username, password, orgChoice;

        system("cls");

        cout << "===================== ADD USER =====================" << endl;

        cout << "Enter a username: ";

        getline(cin, username);
```

```
for (char c : username)

{

    if (c == ' ')

    {

        cout << "Spaces are not allowed in usernames. Try again.\n";

        Sleep(3000);

        registerUser();

        return;

    }

}


if (!checkusername(username))

{

    cout << "Username must be between 8-12 characters long.\n";

    cout << "It must contain at least 1 capital letter and at least 1 digit (1-9).\n";

    Sleep(3000);

   username = “\0”


    registerUser();

    return;

}


if (findusername(admin->GetOrgName(), username))
```

```
{
   cout << "=========================================================" << endl;
   cout << "||                      SORRY!                    ||" << endl;
   cout << "||  Username already exists! Try a different one.  ||" << endl;
   cout << "=========================================================" << endl;
   Sleep(3000);
   registerUser();
   return;
}

cout << "Enter a password: ";
getline(cin, password);
for (char c : password)
{
   if (c == ' ')
   {
      cout << "Spaces are not allowed in usernames. Try again.\n";
      Sleep(2000);
       username = “\0”
      registerUser();
      return;
   }
}
```

```
while (!checkpassword(password))
{
   cout << "Password must be between 8-12 characters long.\n";
   cout << "Enter a valid password: ";
   getline(cin, password);
}

User* newUser = new User();
newUser->SetUsername(username);
newUser->SetPassword(password);
newUser->SetOrganization(admin->GetOrgName());

if (list->GetFirst() == NULL) {
   list->SetFirst(newUser);
   list->SetLast(newUser);
}
else {
   list->GetLast()->SetNext(newUser);
   list->SetLast(newUser);
}
userCount++;
cout << "        CONGRATULATION!        " << endl;
cout << "  User registered successfully!  " << endl;
```

```
    ofstream outFile;
    outFile.open(admin->GetOrgName()+".txt", ios::app);

    if (outFile.is_open()) {
        outFile << username << ":" << password << endl;
        outFile.close();
        cout << "  User data saved successfully!\n";
    }
    else {
        cout << "Error saving user data.\n";
    }
    cout << "Press any key to continue ";
    cin.ignore();
}

void deleteUser()                          //DELETE
{
    string org, username;
    system("cls");
    cout << "=================== DELETE USER ===================\n";
    cout << "Enter username to delete: ";
    getline(cin, username);

    ifstream file(admin->GetOrgName() + ".txt");
    ofstream tempFile("temp.txt");


    bool found = false;

    User* temp = list->GetFirst();

    while (temp != NULL) {

        if (temp->GetOrganization() == admin->GetOrgName() && temp->GetUsername() ==
username && username.length() >= 8) {

            if (temp == list->GetFirst()) {

                list->SetFirst(temp->GetNext());

                found = true;

                delete temp;

                break;

            }

            else {

                temp->GetPrev()->SetNext(temp->GetNext());

                found = true;

                delete temp;

                break;

            }

        }

        else {

            temp = temp->GetNext();

        }

    }

    if (file.is_open() && tempFile.is_open())
```

```
{
    string line;
    while (getline(file, line))
    {
        int delimPos = line.find(':');
        string fileUsername = line.substr(0, delimPos);

        if (fileUsername != username)
        {
            tempFile << line << endl;
        }
        else
        {
            found = true;
        }
    }
    file.close();
    tempFile.close();
    remove((admin->GetOrgName() + ".txt").c_str());
    rename("temp.txt", (admin->GetOrgName() + ".txt").c_str());
}
else {
    cout << "Error opening file";
}
```

```
    if (found)
    {
        cout << "User deleted successfully!\n";
    }
    else {
        cout << "User not found!\n";
    }
    Sleep(2000);
}

void editUser() {                          //UPDATE
    string username, oldPassword, newPassword, organization;
    system("cls");
    cout << "================= EDIT USER ==================\n";

    cout << "Enter the username to edit: ";
    getline(cin, username);
    bool userExists = false;

    ifstream file(admin->GetOrgName() + ".txt");
    if (file.is_open()) {
        string line;
        while (getline(file, line)) {
```

```
        int delimiterPos = line.find(':');

        if (delimiterPos != string::npos) {

            string fileUsername = line.substr(0, delimiterPos);

            if (fileUsername == username) {

                userExists = true;

                organization = admin->GetOrgName() + ".txt";

                break;

            }

        }

    }

    file.close();

}


if (!userExists) {

    cout << "User not found!\n";

    Sleep(3000);

    return;

}


cout << "Enter the current password for " << username << ": ";

getline(cin, oldPassword);


bool oldPasswordCorrect = false;

ifstream files(organization);
```

```
if (files.is_open()) {
    string line;
    while (getline(files, line)) {
        int delimiterPos = line.find(':');
        if (delimiterPos != string::npos) {
            string fileUsername = line.substr(0, delimiterPos);
            string filePassword = line.substr(delimiterPos + 1);
            if (fileUsername == username && filePassword == oldPassword) {
                oldPasswordCorrect = true;
                break;
            }
        }
    }
    files.close();
}

if (!oldPasswordCorrect) {
    cout << "Incorrect password!\n";
    Sleep(3000);
    return;
}

cout << "Enter the new password: ";
getline(cin, newPassword);
```

```
while (!checkpassword(newPassword)) {
    cout << "Password must be between 8-12 characters long.\n";
    cout << "Enter a valid password: ";
    getline(cin, newPassword);
}

bool updated = false;
ifstream inputFile(organization);
ofstream tempFile("temp.txt");

if (inputFile.is_open() && tempFile.is_open()) {
    string line;
    while (getline(inputFile, line)) {
        int delimiterPos = line.find(':');
        string fileUsername = line.substr(0, delimiterPos);
        string filePassword = line.substr(delimiterPos + 1);

        if (fileUsername == username && filePassword == oldPassword) {
            tempFile << fileUsername << ":" << newPassword << endl;
            updated = true;

            User* temp = list->GetFirst();
            while (temp != NULL) {
                    if (temp->GetUsername() == username && temp->GetPassword() ==
oldPassword) {
                        temp->SetPassword(newPassword);
                        break;
                    }
                    temp = temp->GetNext();
                }
            }
            else {
                tempFile << line << endl;
            }
        }
        inputFile.close();
        tempFile.close();
        remove(organization.c_str());
        rename("temp.txt", organization.c_str());
    }


    if (updated) {
        cout << "Password updated successfully for " << username << " in " << organization <<
"!\n";


        ofstream outFile("username.txt", ios::app);
        if (outFile.is_open()) {
            outFile << username << ":" << newPassword << ":" << organization << endl;
            outFile.close();
        }
        else {
            cout << "Error saving updated data to username.txt.\n";
        }
    }
    else {
        cout << "Error updating password. Please try again.\n";
    }

    Sleep(3000);
}


void loginUser()
{
    system("cls");
    cout << "==================== SIGN IN ====================\n\n";
    string username, password, orgName;
    cout << "Enter organization name: ";
    getline(cin, orgName);
    cout << "Enter username: ";
    getline(cin, username);
    cout << "Enter password: ";
    getline(cin, password);
```

```
        User* temp = list->GetFirst();

        while (temp != NULL) {

            if (temp->GetUsername() == username && temp->GetPassword() == password &&
temp->GetOrganization() == ToLower(orgName))

            {

                cout << "Login successful! Welcome, " << username << ".\n";

                cout << "Press any key to continue ";

                cin.ignore();

                user = temp;

                UserMenu();

                return;

            }

            temp = temp->GetNext();

        }



        cout << "Invalid username, password, or organization!\n";  // Show error for invalid
credentials

        cout << "Press any key to continue ";

        cin.ignore();

        return;

    }



    void AdminMenu() {

        string choice;
```

```
system("cls");
cout << "                    =========================                   " << endl;
cout << "                    ||WELCOME TO ADMIN MENU||                   " << endl;
cout << "                    =========================                   " << endl;
while (true) {
  // Show main menu
  cout << "====================" << endl;
  cout << "|1. Add user       |" << endl;
  cout << "|2. Remove user    |" << endl;
  cout << "|3. Edit user      |" << endl;
  cout << "|4. Show users     |" << endl;
  cout << "|5. Logout         |" << endl;
  cout << "====================" << endl;
  cout << endl;
  cout << "Enter your choice: ";
  getline(cin, choice);

  if (choice == "1") {
    registerUser();
    cin.ignore();
    AdminMenu();
  }
  else if (choice == "2")
  {
    deleteUser();

    AdminMenu();


}

else if (choice == "3")

{

    editUser();

    AdminMenu();

}

else if (choice == "4") {

    ShowUsers();

    AdminMenu();

}


else if (choice == "5")

{

    string confirm;

    system("cls");

    cout << "Are you sure that you want to logout? (y/n) ";

    getline(cin, confirm);

    if (confirm == "y" || confirm == "Y") {

        admin = NULL;

        Display();

        return;
```

```
            }
            else if (confirm == "n" || confirm == "N") {
                AdminMenu();
            }
            else {
                cout << "Invalid choice...returning to Admin menu...";
                Sleep(2000);
                AdminMenu();
            }
        }
        else
        {
            cout << "Invalid choice! Try again.\n";
            Sleep(1200);
            AdminMenu();
        }
    }
}

void UserMenu() {
    string choice;
    system("cls");
    cout << "                              ========================                          " << endl;
    cout << "                              || WELCOME TO USER MENU ||                          " << endl;
cout << "                    ==========================                  " << endl;

// Show main menu

cout << "======================" << endl;

cout << "|1. Send Messages     |" << endl;

cout << "|2. View Messages     |" << endl;

cout << "|3. Log Out           |" << endl;

cout << "======================" << endl;

cout << endl;

cout << "Enter your choice: ";

getline(cin, choice);


if (choice == "1") {

   sendMessage();

   cin.ignore();

   choice = "\0";

   UserMenu();

   cin.ignore();

}

else if (choice == "2") {

   viewMessages();

   cin.ignore();

   choice = "\0";

   UserMenu();

}
```

```
else if (choice == "3") {


    string confirm;

    system("cls");

    cout << "Are you sure that you want to logout? (y/n) ";

    getline(cin, confirm);

    if (confirm == "y" || confirm == "Y") {

        user = NULL;

        Display();

        return;

    }

    else if (confirm == "n" || confirm == "N") {

        choice = "\0";

        UserMenu();

    }

    else {

        cout << "Invalid choice...returning to User menu...";

        Sleep(2000);

        UserMenu();

    }

}

else {

    cout << "Invalid choice. Try again.\n";

    Sleep(1000);
```

```
        choice = "\0";

        UserMenu();

        cin.ignore();

    }


}

void sendMessage() {

    string recipient, message;

    cout << "Enter recipient's username: ";

    getline(cin, recipient);


    bool recipientFound = false;


    string organizationUserFile = user->GetOrganization() + ".txt";


    ifstream file(organizationUserFile);

    if (!file.is_open()) {

        cout << "Error: Could not open file for user validation.\n";

        cin.ignore();

        cin.get();

        return;

    }


    string line;
```

```
while (getline(file, line)) {
    int delimiterPos = line.find(':');
    if (delimiterPos != string::npos) {
        string fileUsername = line.substr(0, delimiterPos);
        if (fileUsername == recipient) {
            recipientFound = true;
            break;
        }
    }
}
file.close();

if (!recipientFound) {
    cout << "The username '" << recipient << "' does not belong to your organization!\n";
    cout << "Press any key to continue ";
    cin.ignore();
    cin.get();
    UserMenu();
    return;
}


cout << "Enter your message: ";
getline(cin, message);
```

```
Encryption(message);



time_t now = time(0);

char* timestamp = ctime(&now);

timestamp[strlen(timestamp) - 1] = '\0';



string organizationMessageFile = user->GetOrganization() + "_messages.txt";


ofstream files(organizationMessageFile, ios::app);

if (!files.is_open()) {

   cout << "Error: Could not open file to save the message.\n";

   cin.ignore();

   cin.get();

   UserMenu();

   return;

}


files << user->GetUsername() << " -> " << recipient << ":" << message

     << " | Sent: " << timestamp << endl;

files.close();
```

```
    cout << "Message sent successfully to " << recipient << "!\n";

    cout << "Press any key to continue ";

    cin.ignore();

    UserMenu();

}


bool isDigit(string inp){

    return all_of(inp.begin(), inp.end(), ::isdigit);

}

int keyInput(){


    string key = "";

    while(true){


        getline(cin, key);

        if(isDigit(key)){

            return std::stoi(key);

        }

        else {

            cout<<"Plz enter valid KEY: ";

            continue;

        }

    }
```

```
}
void viewMessages()
{
    string organizationMessageFile = user->GetOrganization() + "_messages.txt";
    ifstream file(organizationMessageFile);
    string line;
    cout << "    ========================\n";
    cout << "    || Messages for " << user->GetUsername() << " ||\n";
    cout << "    ========================\n";
    bool hasMessages = false;

    if (file.is_open())
    {
        while (getline(file, line))
        {
            if (line.find(user->GetUsername() + " ->") == 0 ||
                line.find("-> " + user->GetUsername() + ":") != string::npos)
            {

                int arrowPos = line.find("->");
                string sender = line.substr(0, arrowPos);

                int recipientStart = arrowPos + 3;
                int recipientEnd = line.find(":", recipientStart);

                string recipient = line.substr(recipientStart, recipientEnd - recipientStart);



                int messageStart = recipientEnd + 1;

                int timestampPos = line.find(" | Sent: ", messageStart);

                string encryptedMessage = line.substr(messageStart, timestampPos -
messageStart);

                string timestamp = line.substr(timestampPos + 8);



                cout << "From: " << sender << "\nTo: " << recipient;

                cout<< "\nTime: " << timestamp << endl;



                cout << "Enter decryption key to view the message: ";

                int key = keyInput();





                Decryption(encryptedMessage,key);

                cout<< "\nMessage: " << encryptedMessage << endl << endl;

                hasMessages = true;

              }

            }

            file.close();

        }

        else

        {
            cout << "Error reading messages.\n";
        }


        if (!hasMessages) {
            cout << "No messages found.\n";
        }
        cout << "Press any key to continue ";
        cin.ignore();
    }



    void Display() {
        string choice;
        system("cls");
        cout << endl;
        cout << "                    ========================================                    " << endl;
        cout << "<<<<<<<<<<<<<<<<<<<<<<||  ENIGMA! YOUR PRIVACY
PARTNER  ||>>>>>>>>>>>>>>>>>>>>>> " << endl;
        cout << "                    ========================================                    " << endl;
        cout << endl;
        cout << "                          ===========================                    " << endl;
        cout << "                          ||WELCOME TO MAIN MENU||                    " << endl;
        cout << "                          ===========================                    " << endl;
        while (true) {
            // Show main menu
```

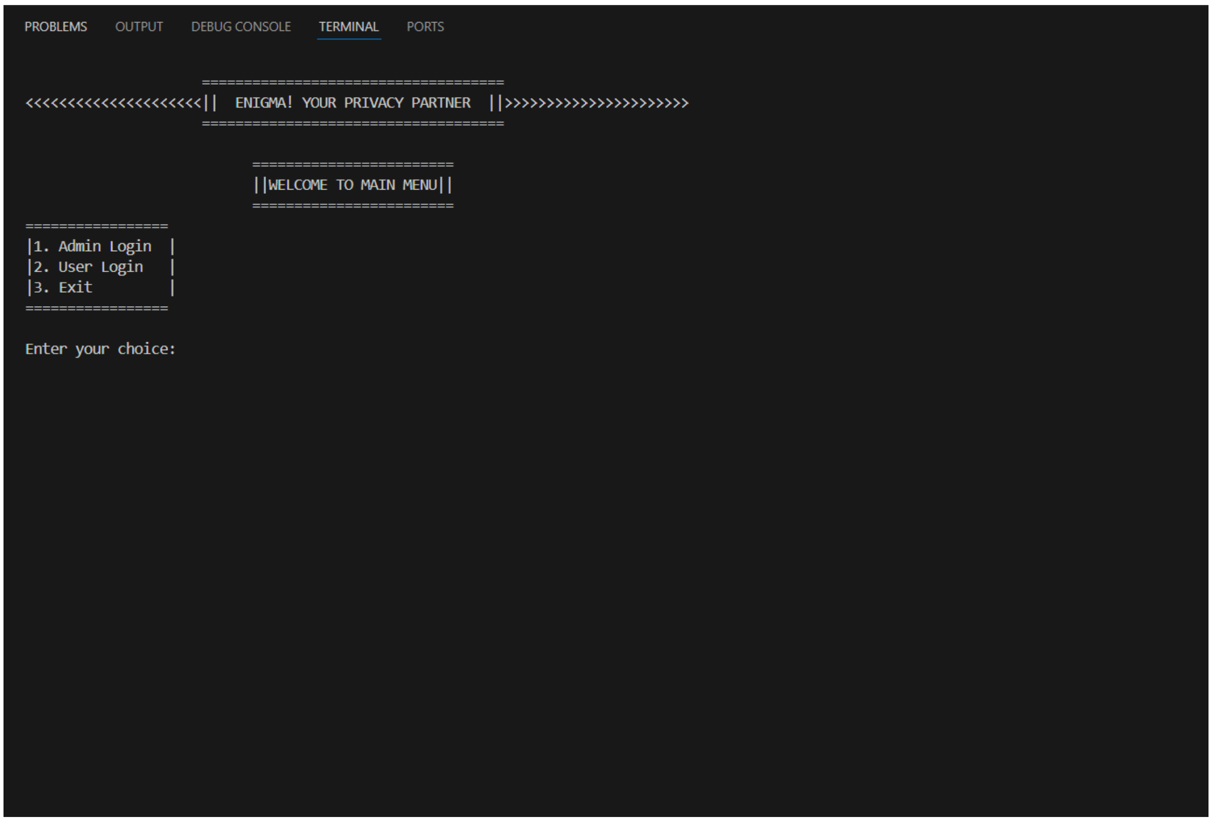
```
cout << "=================" << endl;

cout << "|1. Admin Login  |" << endl;

cout << "|2. User Login   |" << endl;

cout << "|3. Exit         |" << endl;

cout << "=================" << endl;

cout << endl;

cout << "Enter your choice: ";

getline(cin, choice);


if (choice == "1") {

    AdminLogin();

    Display();

}

else if (choice == "2") {

    loginUser();

    Display();

}

else if (choice == "3") {

    cout << "Goodbye!\n";

    Sleep(1200);

    return;


}

else {
```

```
                cout << "Invalid choice. Try again.\n";

                Sleep(1000);

                Display();

            }

        }

    }

    string Encryption(string& text)

    {

        cout<<"Enter encryption key: ";

        int key = keyInput();

for (int i=0;i<text.size();i++)

        {

            text[i]=text[i]+key;

        }




        for (char i : text)

        {

            stk.push(i);

        }


        string reversedMessage = "";

        char topChar;

        while (!stk.isempty()) {
```

```
        topChar = stk.pop();

        reversedMessage += topChar;

    }

    text = reversedMessage;

    return text;

}


string Decryption(string& text,int key)

{

    stack stk;

    for (char i: text)

    {

        stk.push(i);

    }


    string originalEncryptedMessage;

    char topChar = NULL;


    while (!stk.isempty()) {

        char topChar = stk.pop();

        originalEncryptedMessage += topChar;

    }


    for (int i = 0; i < originalEncryptedMessage.size(); i++) {
        originalEncryptedMessage[i] = originalEncryptedMessage[i] - key;
    }

    text = originalEncryptedMessage;
    return text;
}
void readFromFile(const string& orgName) {
    ifstream file(orgName + ".txt");
    if (!file.is_open()) {
        cout << "Error: Could not open file " << orgName << endl;
        return;
    }

    string line;
    while (getline(file, line)) {
        size_t delimPos = line.find(':');

        if (delimPos != string::npos) {
            string username = line.substr(0, delimPos);
            string password = line.substr(delimPos + 1);
            // Add the organization name along with the username and password to the linked list
            list->addUser(orgName, username, password);
        }
        }


        file.close();
    }



};


int main() {


    cout<<"\n\t##      ## ####### ##      ######  ###### ####  #### #######";

    cout<<"\n\t##      ## ##      ##      ##      ##  ## ##  ##  ## ##       ";

    cout<<"\n\t##  ##  ## #####   ##      ##      ##  ## ##  ##  ## #####      ";

    cout<<"\n\t##  ##  ## ##      ##      ##      ##  ## ##      ## ##           ";

    cout<<"\n\t####  #### ####### ####### ######  ###### ##      ## #######     ";

    cout<<"\n\n\t\t\t         #######  #####        ";

    cout<<"\n\t\t\t           ##     #   #      ";

    cout<<"\n\t\t\t           ##     #####    ";

    cout<<endl;

    cout<<"\n\t#######  ###      ## ###### ######### ###             ###       #####";

    cout<<"\n\t##       ## ##    ##   ##   ##        ## ##         ####      ##   ##";

    cout<<"\n\t##       ##  ##   ##   ##   ##        ##  ##       ## ##     ##     ##";

    cout<<"\n\t#####    ##   ##  ##   ##   ##  ##### ##   ##     ##  ##    ##       ##";

    cout<<"\n\t##       ##    ## ##   ##   ##  ## ## ##    ##   ##   ##   #############";
    cout<<"\n\t##      ##     ####   ##   ##     ## ##     ## ##    ##  ##          ##";

    cout<<"\n\t#######  ##      ### ###### ######### ##      ###     ## ##            ##";


    Sleep(1500);

    Enigma;

    enigma.readFromFile("army");

    enigma.readFromFile("business");

    enigma.readFromFile("bank");

    enigma.readFromFile("education");

    enigma.readAdmins("army");

    enigma.readAdmins("business");

    enigma.readAdmins("bank");

    enigma.readAdmins("education");

    enigma.Display();

    return 0;

}
```

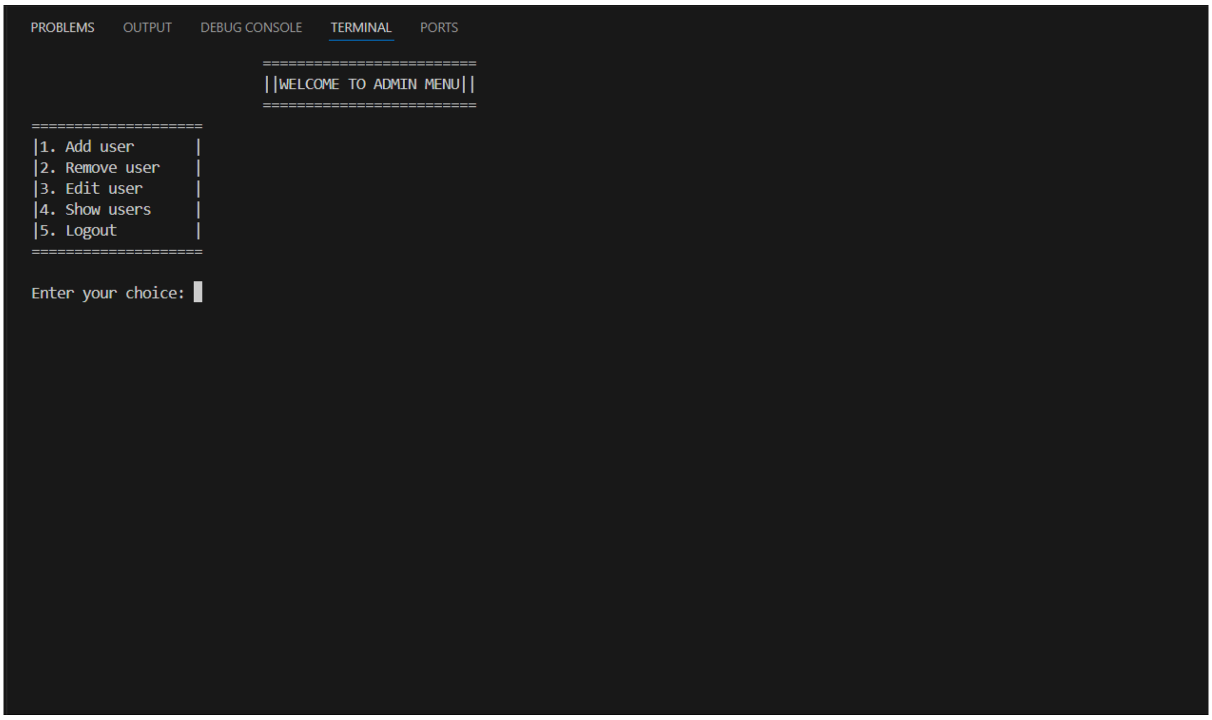
SCREEN SHOTS

MAIN MENU



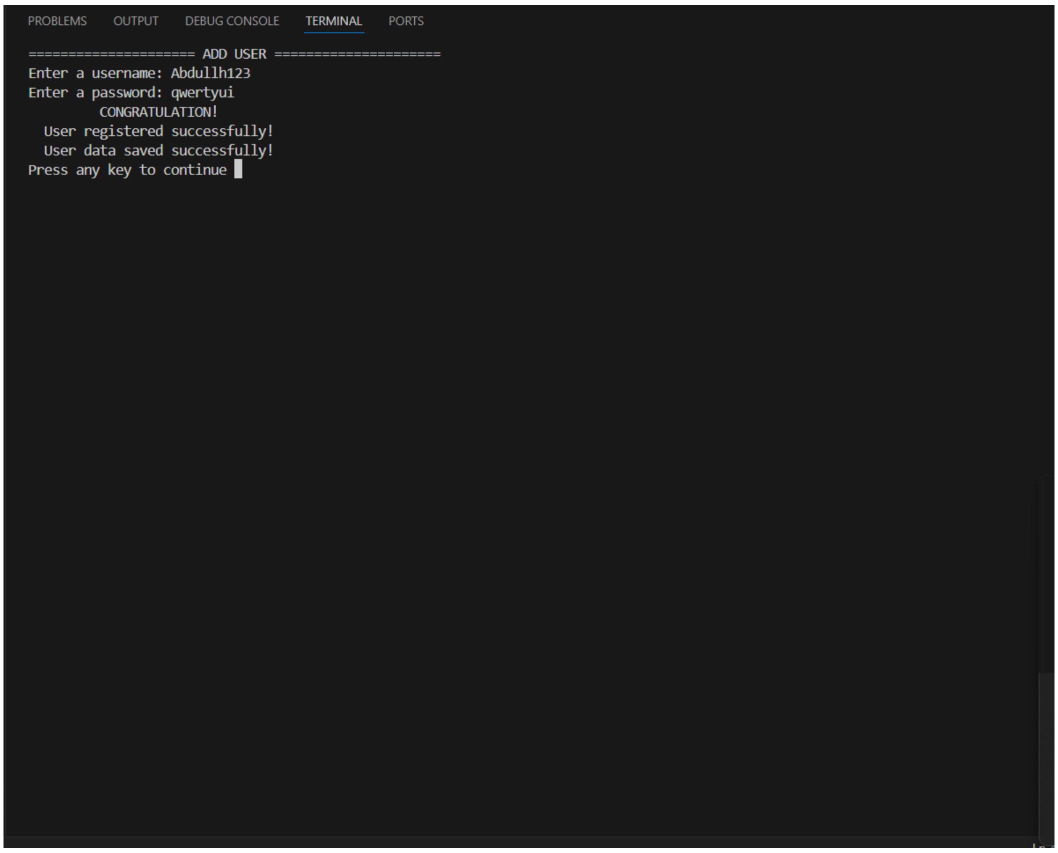
```
                     ====================================
<<<<<<<<<<<<<<<<<<<<<||  ENIGMA! YOUR PRIVACY PARTNER  ||>>>>>>>>>>>>>>>>>>>>>>
                     ====================================


                           ========================
                           ||WELCOME TO MAIN MENU||
                           ========================

=================
|1. Admin Login  |
|2. User Login   |
|3. Exit         |
=================

Enter your choice:
```

ADMIN MENU



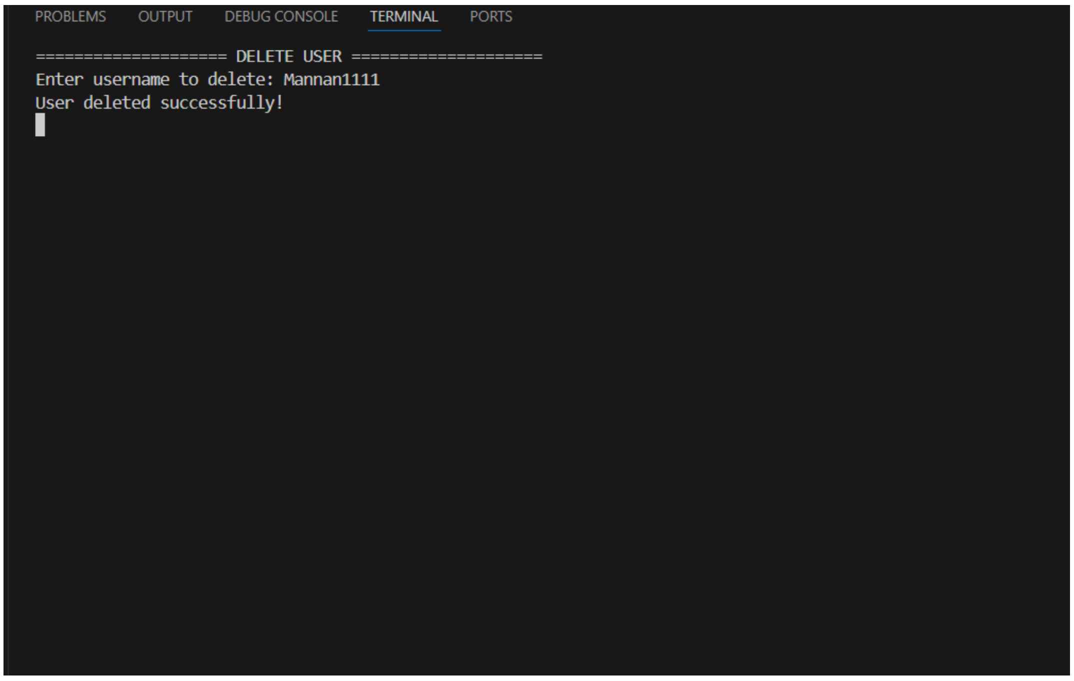
```
                           =========================
                           ||WELCOME TO ADMIN MENU||
                           =========================

====================
|1. Add user        |
|2. Remove user     |
|3. Edit user       |
|4. Show users      |
|5. Logout          |
====================

Enter your choice:
```

ADMIN MENU->ADD USER



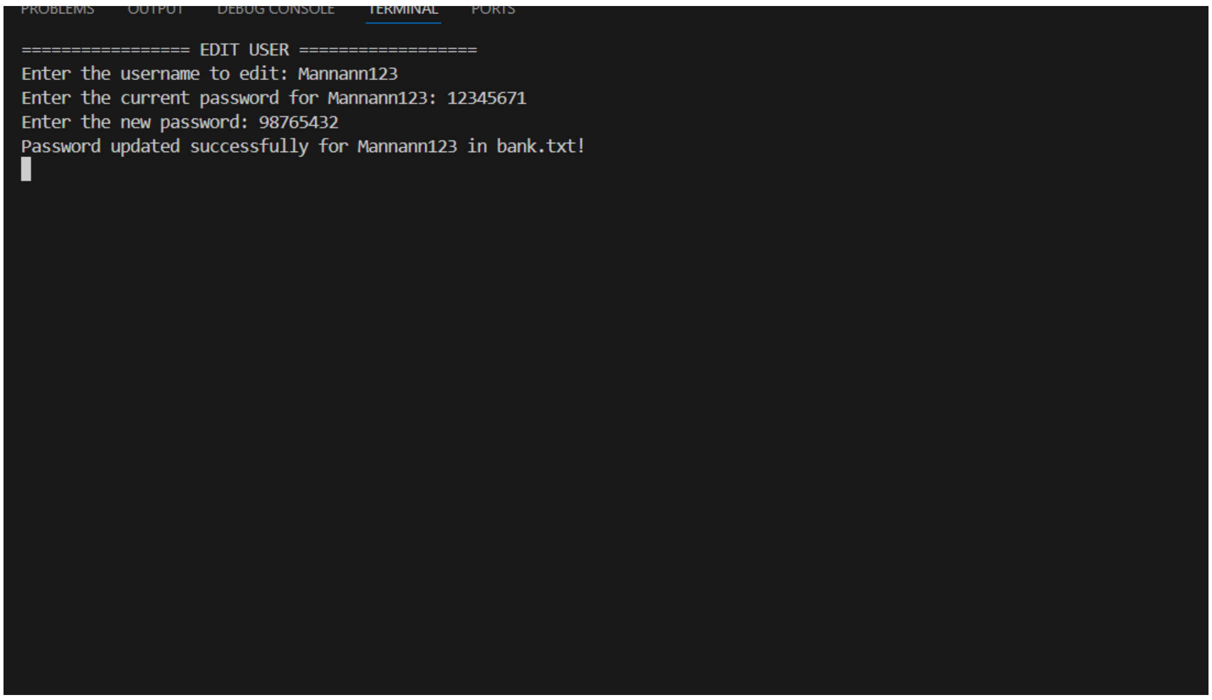
```
==================== ADD USER ====================
Enter a username: Abdullh123
Enter a password: qwertyui
         CONGRATULATION!
  User registered successfully!
  User data saved successfully!
Press any key to continue
```

ADMIN MENU->REMOVE USER



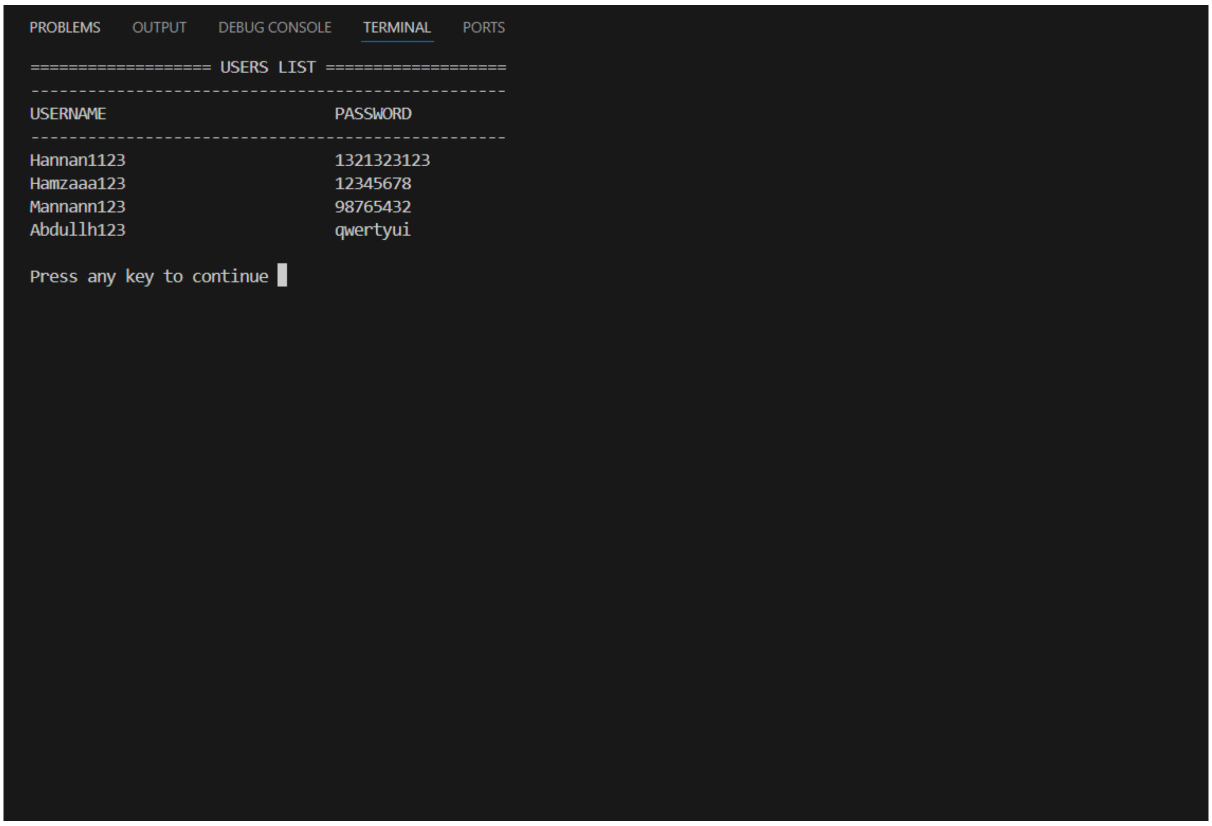
```
==================== DELETE USER ====================
Enter username to delete: Mannan1111
User deleted successfully!
```

ADMIN MENU->EDIT USER



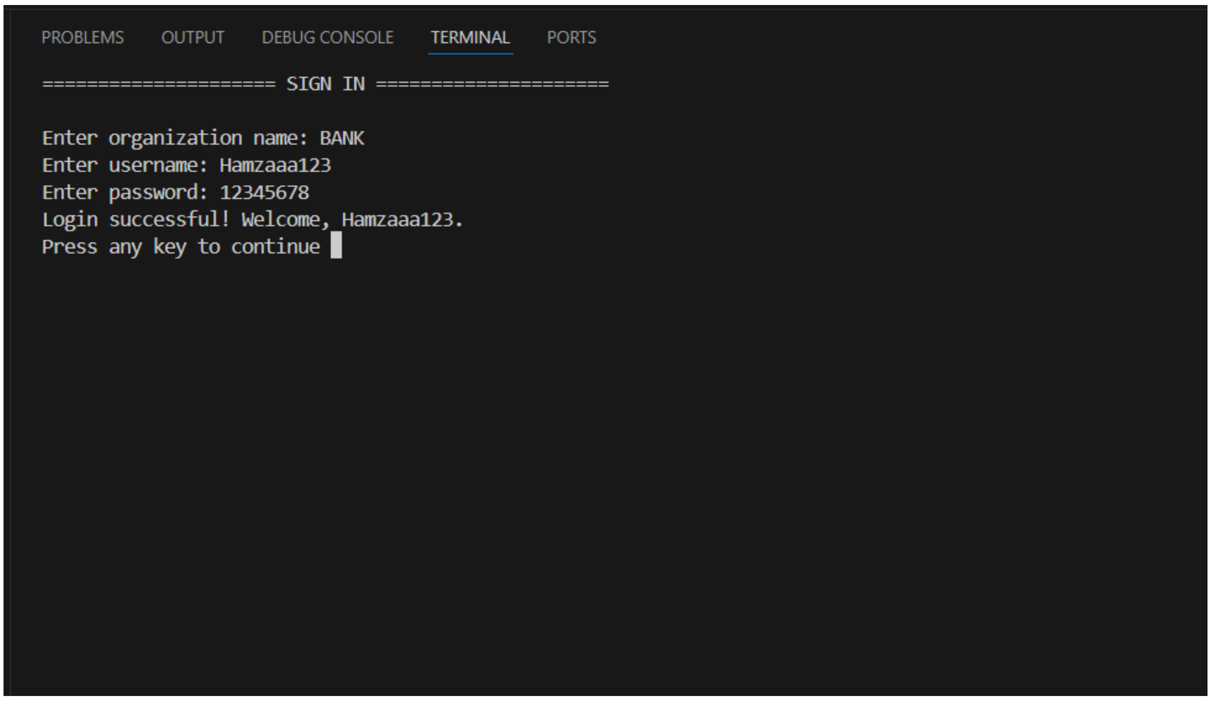
```
================= EDIT USER ==================
Enter the username to edit: Mannann123
Enter the current password for Mannann123: 12345671
Enter the new password: 98765432
Password updated successfully for Mannann123 in bank.txt!
```

ADMIN MENU->SHOW USERS

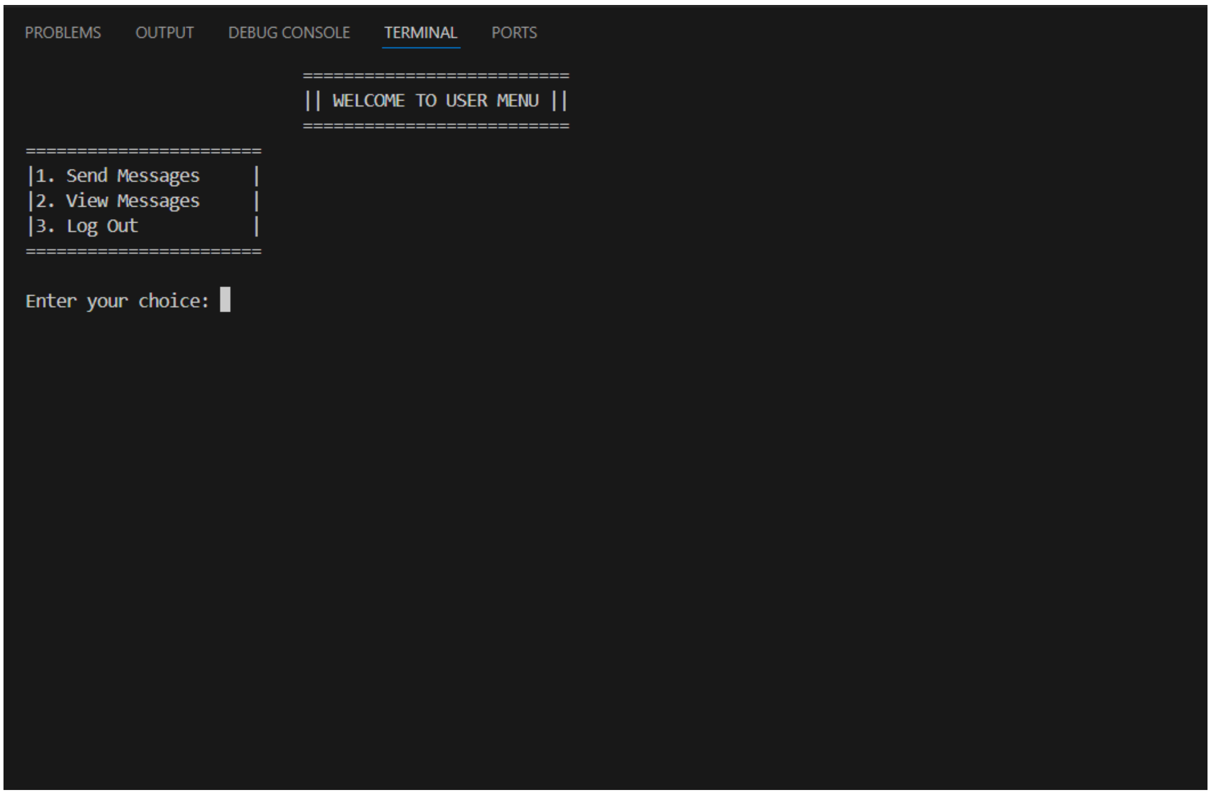


```
=================== USERS LIST ===================
--------------------------------------------------
USERNAME                        PASSWORD
--------------------------------------------------
Hannan1123                      1321323123
Hamzaaa123                      12345678
Mannann123                      98765432
Abdullh123                      qwertyui

Press any key to continue
```

USER MENU

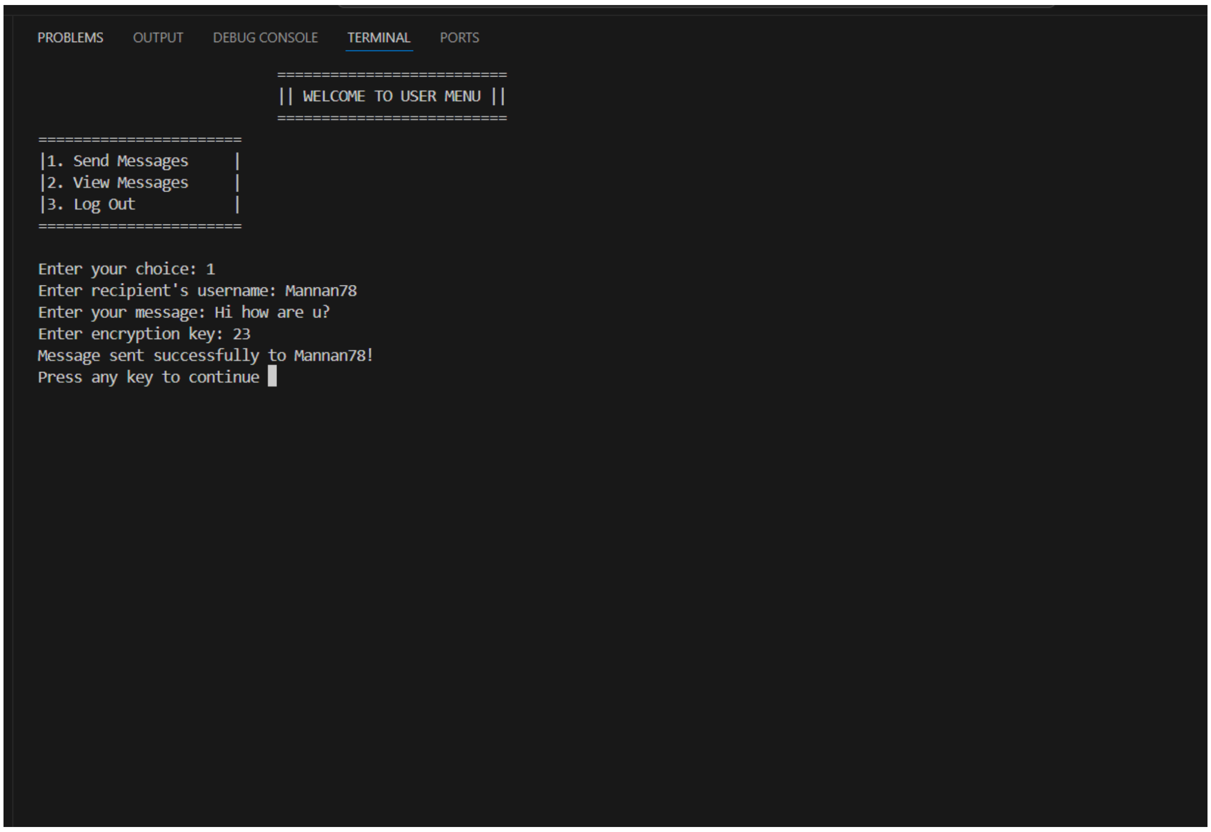


```
==================== SIGN IN ====================

Enter organization name: BANK
Enter username: Hamzaaa123
Enter password: 12345678
Login successful! Welcome, Hamzaaa123.
Press any key to continue
```



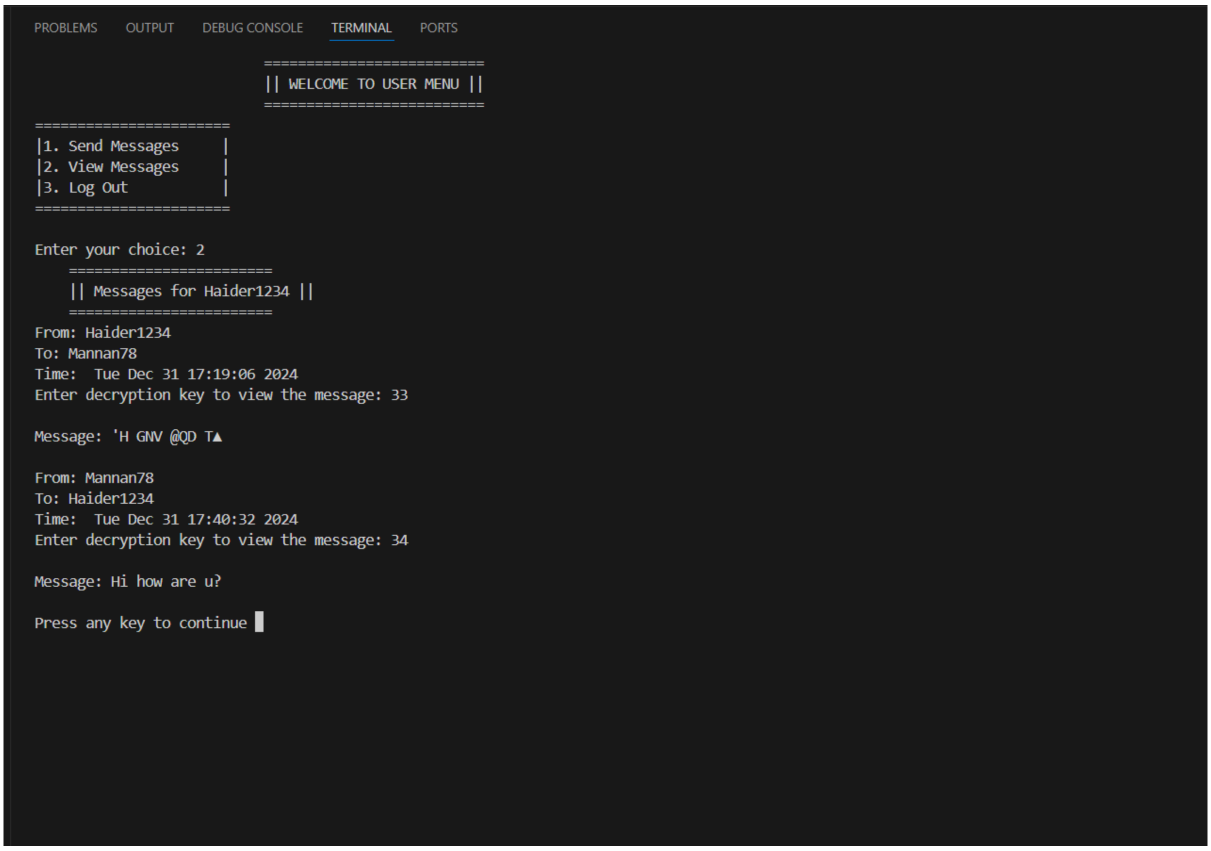
```
                              ==========================
                              || WELCOME TO USER MENU ||
                              ==========================
=======================
|1. Send Messages     |
|2. View Messages     |
|3. Log Out           |
=======================

Enter your choice:
```

USER MENU->SEND MESSAGE



```
                           ==========================
                           || WELCOME TO USER MENU ||
                           ==========================
=======================
|1. Send Messages     |
|2. View Messages     |
|3. Log Out           |
=======================

Enter your choice: 1
Enter recipient's username: Mannan78
Enter your message: Hi how are u?
Enter encryption key: 23
Message sent successfully to Mannan78!
Press any key to continue
```

USER MENU->VIEW MESSAGE



```
                           ==========================
                           || WELCOME TO USER MENU ||
                           ==========================
=======================
|1. Send Messages     |
|2. View Messages     |
|3. Log Out           |
=======================

Enter your choice: 2
    =======================
    || Messages for Haider1234 ||
    =======================
From: Haider1234
To: Mannan78
Time:  Tue Dec 31 17:19:06 2024
Enter decryption key to view the message: 33

Message: 'H GNV @QD T▲

From: Mannan78
To: Haider1234
Time:  Tue Dec 31 17:40:32 2024
Enter decryption key to view the message: 34

Message: Hi how are u?

Press any key to continue
```