SRM UNIVERSITY AP
Andhra Pradesh

CSE 202

CRIME CONTROL

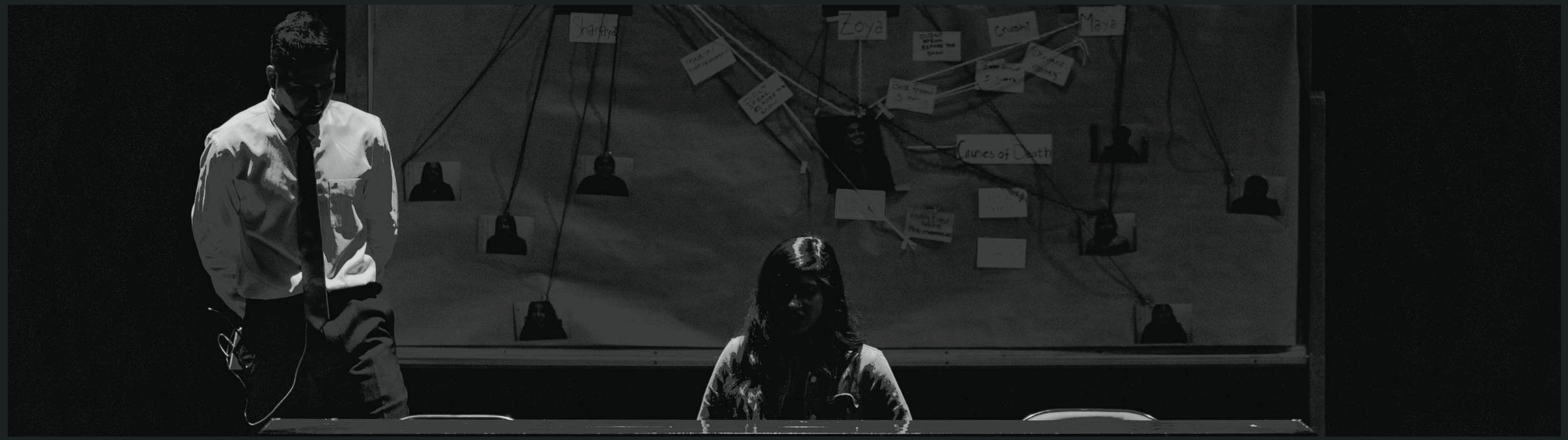
ON A MISSION TO CONTROL CRIME

BY
TEAM 7

キショア

CRIME CONTROL

ABOUT OUR PROJECT



SECURED REPORTS

BY
TEAM 7

CRIME CONTROL

DATA FLOW

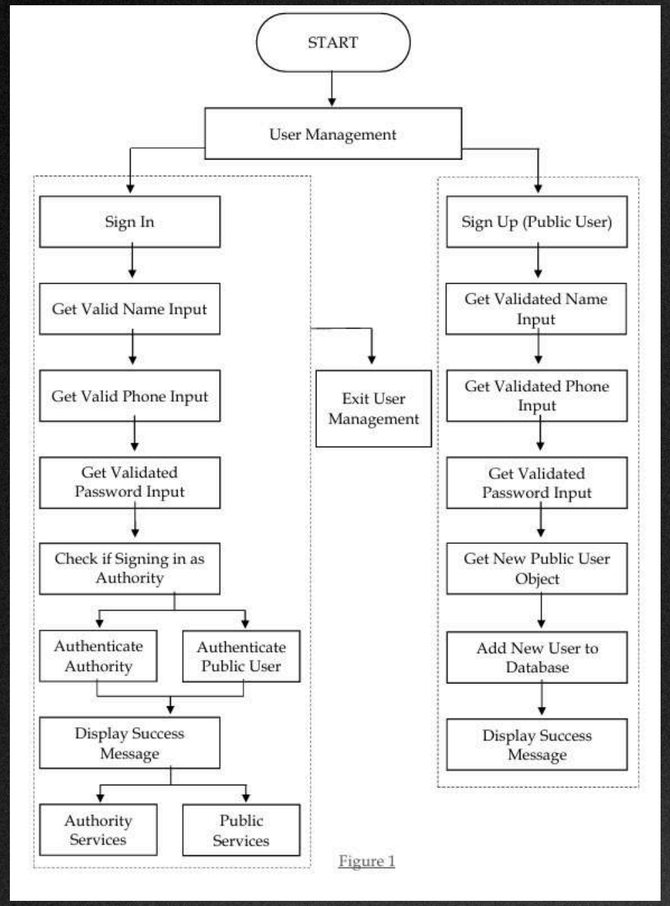
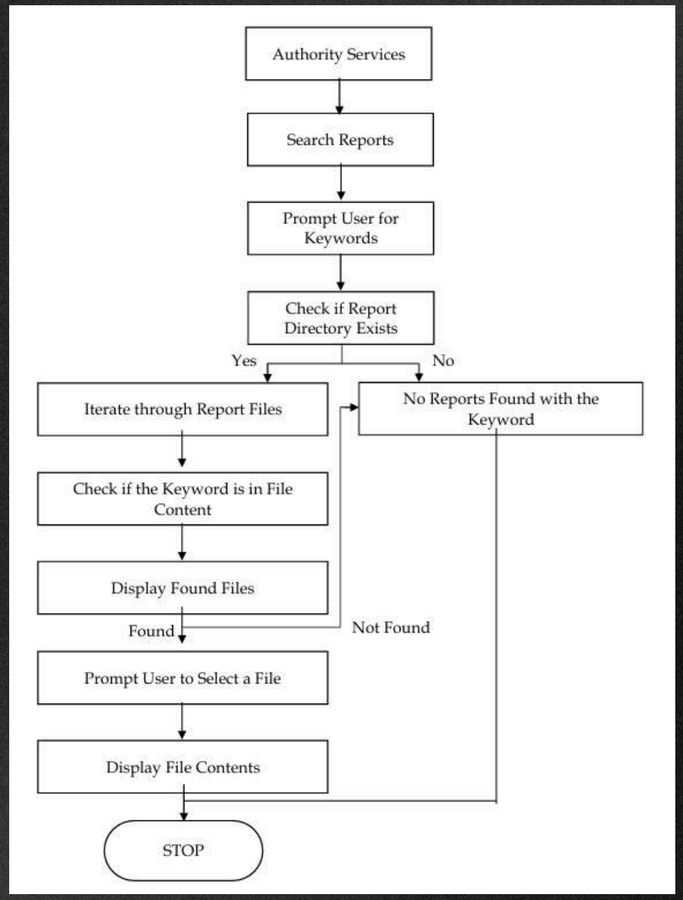
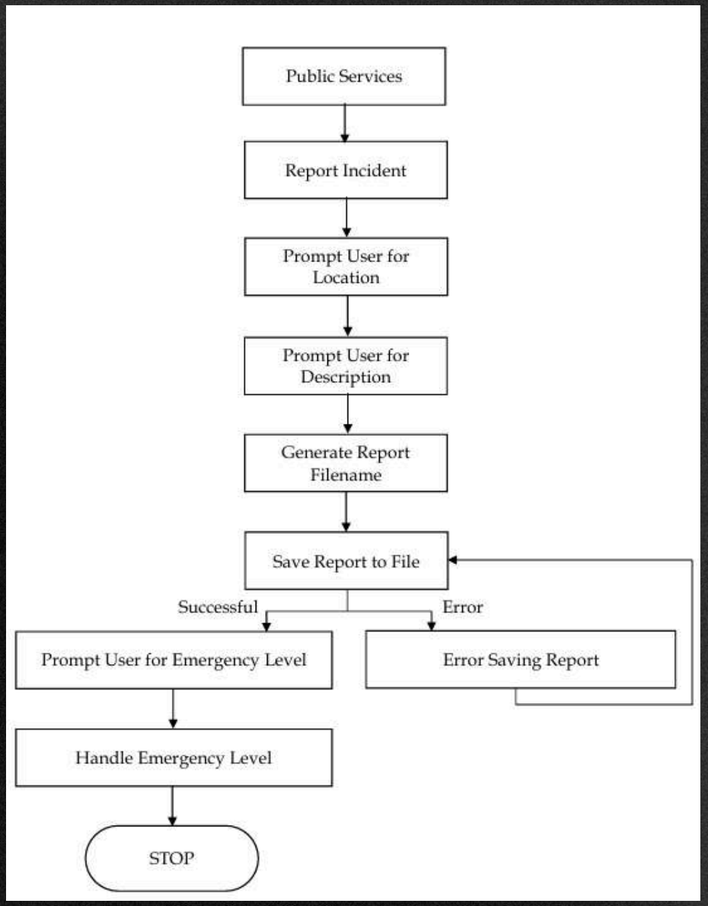


Figure 1





BY TEAM 7

CRIME CONTROL

ALGORITHM

Step 1: Start

Step 2: Initialize system variables and classes

Step 3: Display the main menu with options

Step 4: Read the user's choice

Step 5: Sign-Up Process (Public)

Step 6: Sign-In Process

Step 7: Handle incorrect inputs or invalid choices in both Sign-Up and Sign-In processes

Step 8: If the user chooses to exit

Step 9: Stop

BY
TEAM 7

CRIME CONTROL

PROJECT CODE

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <cctype>
#include <algorithm>
#include <filesystem>
#include <vector>

using namespace std;
namespace fs = std::filesystem;

// Base User Class
class User {
protected:
    string name;
    string phone;
    string password;

public:
    User(string n, string p, string pass) : name(move(n)), phone(move(p)), password(move(pass)) {}

    string getName() const { return name; }
    string getPhone() const { return phone; }
    string getPassword() const { return password; }

    // Validation functions
    static bool isValidName(const string& name) {
        return all_of(name.begin(), name.end(), ::isalpha);
    }

    static bool isValidPhone(const string& phone) {
        return phone.length() == 10 && all_of(phone.begin(), phone.end(), ::isdigit);
    }

    static bool isValidPassword(const string& password) {
        return password.length() >= 8;
    }
};
```

```cpp
// Authority Class (inherits from User)
class Authority : public User {
public:
    Authority(string n, string p, string pass) : User(move(n), move(p), move(pass)) {}

    // Search reports
    void searchReports(const string& initialKeyword) const {
        string currentKeyword = initialKeyword; // Start with the initial keyword

        while (true) {
            cout << "Searching reports for keyword: " << currentKeyword << "\n";

            string reportDir = "C++ Project"; // Directory to search

            if (!fs::exists(reportDir)) {
                cout << "Directory " << reportDir << " does not exist.\n";
                return;
            }

            vector<fs::path> foundFiles;

            // Search reports in the directory
            for (const auto& file : fs::directory_iterator(reportDir)) {
                if (file.path().filename().string().find(currentKeyword) != string::npos) {
                    foundFiles.push_back(file.path());
                    cout << "Found in filename: " << file.path().filename() << "\n";
                }

                ifstream infile(file.path());
                if (infile) {
                    string line;
                    while (getline(infile, line)) {
                        if (line.find(currentKeyword) != string::npos) {
                            foundFiles.push_back(file.path());
                            cout << "Found in content of: " << file.path().filename() << "\n";
                            break;
                        }
```

BY TEAM 7

CRIME CONTROL

PROJECT CODE

```
        }
    }
}

// Handle cases when no reports are found
if (foundFiles.empty()) {
    cout << "No reports found with the keyword \"" << currentKeyword << "\".\n";
    cout << "Do you want to search for another keyword? (y/n): ";
    char retry;
    cin >> retry;
    cin.ignore(); // Clear input buffer
    if (tolower(retry) == 'y') {
        cout << "Enter a new keyword to search: ";
        getline(cin, currentKeyword); // Update the current keyword
        continue; // Restart the search
    }
    else {
        cout << "Exiting the search.\n";
        return;
    }
}

// Handle cases when reports are found
int choice;
while (true) {
    cout << "\nFiles found:\n";
    for (size_t i = 0; i < foundFiles.size(); ++i) {
        cout << i + 1 << ". " << foundFiles[i].filename() << "\n";
    }
    cout << "Enter the number of the file you want to read (or 0 to search for another keyword, -1 to exit): ";
    cin >> choice;

    if (choice == -1) {
        cout << "Exiting the search.\n";
        return;
    }
```

```
                if (choice == 0) {
                    cout << "Enter a new keyword to search: ";
                    cin.ignore(); // Clear input buffer
                    getline(cin, currentKeyword); // Update the current keyword
                    break; // Restart the search loop
                }

                if (choice > 0 && choice <= static_cast<int>(foundFiles.size())) {
                    ifstream infile(foundFiles[choice - 1]);
                    if (infile) {
                        cout << "\n--- Contents of " << foundFiles[choice - 1].filename() << " ---\n";
                        string line;
                        while (getline(infile, line)) {
                            cout << line << "\n";
                        }
                        cout << "----------------------------------------\n";
                    }
                    else {
                        cout << "Error opening file.\n";
                    }
                }
                else {
                    cout << "Invalid choice. Try again.\n";
                }
            }
        }
    }
};

// Public Class (inherits from User)
class Public : public User {
public:
    Public(string n, string p, string pass) : User(move(n), move(p), move(pass)) {}

    void reportIncident() const {
```

BY TEAM 7

CRIME CONTROL

PROJECT CODE

```
string caseType, location, description;
cout << "Enter the type of case you are reporting (please enter space 1st e.g., Theft, Murder): ";
cin.ignore(); // Clear the input buffer
getline(cin, caseType);

cout << "Enter the location of the incident: ";
getline(cin, location);

cout << "Enter a detailed description of the case:\n";
getline(cin, description);

// Sanitize input for filename by replacing invalid characters with underscores
auto sanitize = [](string& str) {
    for (char& c : str) {
        if (!isalnum(c) && c != '_' && c != '-') {
            c = '_'; // Replace invalid characters with '_'
        }
    }
    };

sanitize(caseType);
sanitize(location);

// Construct the filename
string filename = "C++ Project/" + caseType + "-" + location + "-" + name + ".txt";

ofstream file(filename);
if (file) {
    file << "Reported by: " << name << "\n";
    file << "Phone: " << phone << "\n";
    file << "Location: " << location << "\n";
    file << "Description: " << description << "\n";
    file << "--------------------------------------\n";

    cout << "\nIncident report saved successfully!!\n";
    cout << "Please select the level of emergency:\n";
            cout << "1. Highly Emergency (Sending location to control room and nearest hospital)\n";
            cout << "2. Normal Crime (To be solved after investigation)\n";
            cout << "Enter your choice: ";

            int emergencyLevel;
            cin >> emergencyLevel;

            if (emergencyLevel == 1) {
                cout << "An emergency alert has been sent with your location to the control room and the nearest hospital.\n";
            }
            else if (emergencyLevel == 2) {
                cout << "Your case has been recorded and will be solved after investigation.\n";
            }
            else {
                cout << "Invalid choice! Defaulting to normal processing.\n";
            }
        }
        else {
            cout << "Error saving the report.\n";
        }
    }

};

// Public User Database
class PublicUserDatabase {
    Public** publicUsers;
    int userCount;
    int capacity;

    void resize() {
        capacity *= 2;
        Public** newUsers = new Public * [capacity];
        for (int i = 0; i < userCount; ++i) {
            newUsers[i] = publicUsers[i];
        }
```

BY TEAM 7

CRIME CONTROL

PROJECT CODE

```
        delete[] publicUsers;
        publicUsers = newUsers;
    }

public:
    PublicUserDatabase(int initialCapacity = 5) : userCount(0), capacity(initialCapacity) {
        publicUsers = new Public * [capacity];
        loadUsersFromFile();
    }

    ~PublicUserDatabase() {
        saveUsersToFile();
        for (int i = 0; i < userCount; ++i) {
            delete publicUsers[i];
        }
        delete[] publicUsers;
    }

    void addPublicUser(Public* newUser) {
        if (userCount == capacity) {
            resize();
        }
        publicUsers[userCount++] = newUser;
    }

    Public* findPublicUser(const string& name, const string& phone, const string& password) const {
        for (int i = 0; i < userCount; ++i) {
            if (publicUsers[i]->getName() == name && publicUsers[i]->getPhone() == phone &&
                publicUsers[i]->getPassword() == password) {
                return publicUsers[i];
            }
        }
        return nullptr;
    }

    void loadUsersFromFile() {
        ifstream file("public_users.txt");
        string name, phone, password;
        while (file >> name >> phone >> password) {
            addPublicUser(new Public(name, phone, password));
        }
    }

    void saveUsersToFile() const {
        ofstream file("public_users.txt");
        for (int i = 0; i < userCount; ++i) {
            file << publicUsers[i]->getName() << " " << publicUsers[i]->getPhone() << " "
                 << publicUsers[i]->getPassword() << "\n";
        }
    }
};

// Helper function
static string getValidatedInput(const string& prompt, bool (*validate)(const string&)) {
    string input;
    while (true) {
        cout << prompt;
        cin >> input;
        if (validate(input)) {
            return input;
        }
        else {
            cout << "Invalid input! Please try again.\n";
        }
    }
}

// Sign-Up function
static void signUp(PublicUserDatabase& db) {
    string name = getValidatedInput("Enter your name: ", User::isValidName);
    string phone = getValidatedInput("Enter your phone number: ", User::isValidPhone);
    string password = getValidatedInput("Enter your password: ", User::isValidPassword);

    // Check if the user already exists
    Public* existingUser = db.findPublicUser(name, phone, password);
```

BY TEAM 7

CRIME CONTROL

PROJECT CODE

```
    Public* existingUser = db.findPublicUser(name, phone, password);
    if (existingUser) {
        cout << "User already exists! Please sign in instead.\n";
        return;
    }

    // Add new user and save to file
    db.addPublicUser(new Public(move(name), move(phone), move(password)));
    cout << "Public user signed up successfully!\n";
}

// Sign-In function
static void signIn(PublicUserDatabase& db, Authority* authorityUsers, int authorityCount) {
    string name = getValidatedInput("Enter your name: ", User::isValidName);
    string phone = getValidatedInput("Enter your phone number: ", User::isValidPhone);
    string password = getValidatedInput("Enter your password: ", User::isValidPassword);

    char choice;
    cout << "Are you signing in as an Authority? (y/n): ";
    cin >> choice;
    cin.ignore();

    if (tolower(choice) == 'y') {
        for (int i = 0; i < authorityCount; ++i) {
            if (authorityUsers[i].getName() == name && authorityUsers[i].getPhone() == phone &&
                authorityUsers[i].getPassword() == password) {
                cout << "Signed in successfully as Authority!\n";

                string keyword;
                cout << "Enter a keyword to search reports: ";
                getline(cin, keyword);
                authorityUsers[i].searchReports(keyword);

                return;
            }
        }
        cout << "Invalid credentials for Authority.\n";
    }
    else {
        Public* user = db.findPublicUser(name, phone, password);
        if (user) {
            cout << "Signed in successfully as Public user!\n";
            user->reportIncident();
        }
        else {
            cout << "Invalid credentials for Public user.\n";
        }
    }
}

// Main Function
int main() {
    PublicUserDatabase publicDb;
    Authority authorityUsers[] = {
        Authority("Dr.Kavatha Rani", "1234512345", "password1"),
        Authority("Prudhvi", "1234567890", "password2"),
        Authority("Navnith", "6789067890", "password3"),
        Authority("Kishore", "0987654321", "password4"),
        Authority("Adithya", "2345678910", "password5")

    };
    int authorityCount = sizeof(authorityUsers) / sizeof(authorityUsers[0]);

    int choice;
    do {
        cout << "\nCrime Control System\n";
        cout << "1. Sign Up\n";
        cout << "2. Sign In\n";
        cout << "0. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
```

BY TEAM 7

CRIME CONTROL

PROJECT CODE

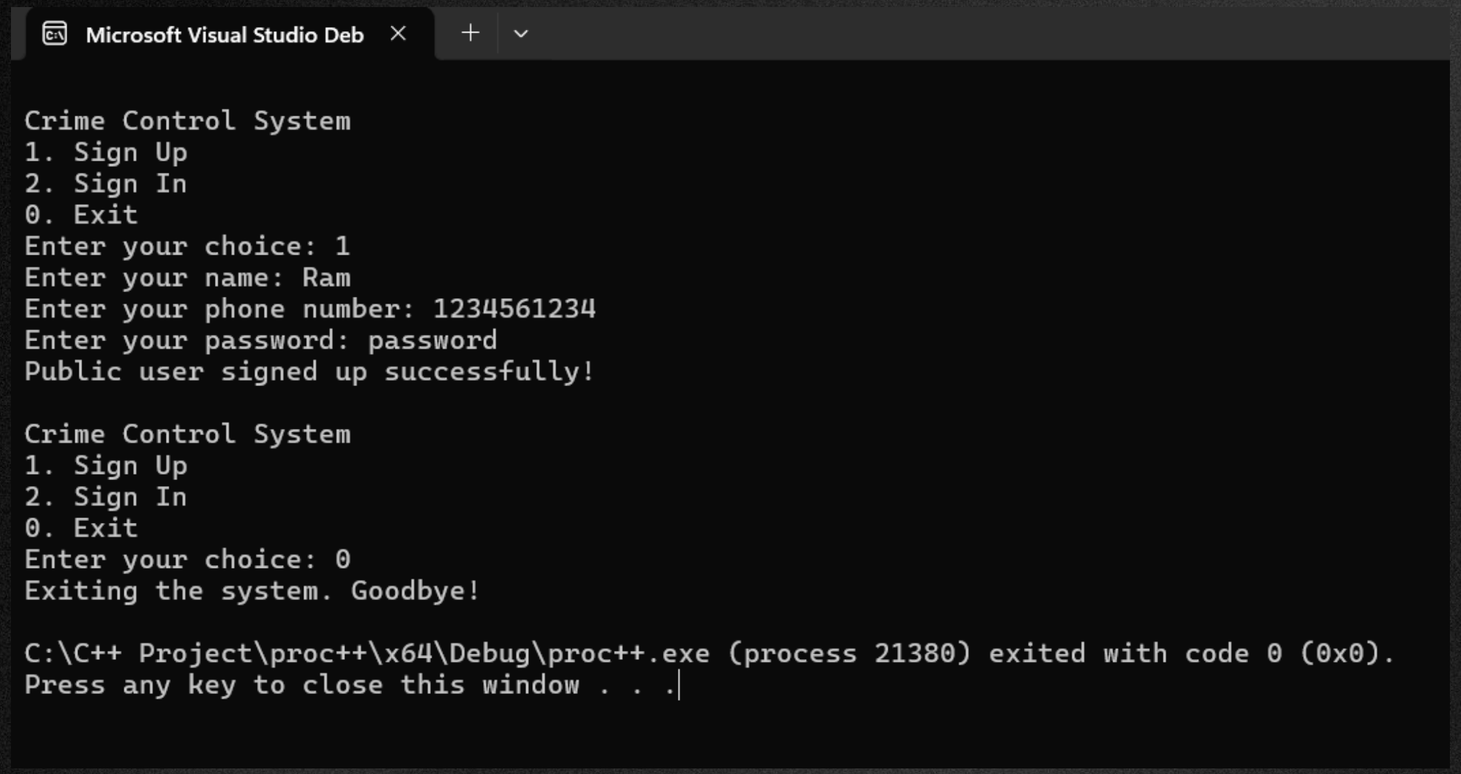
```
        switch (choice) {
        case 1:
            signUp(publicDb);
            break;
        case 2:
            signIn(publicDb, authorityUsers, authorityCount);
            break;
        case 0:
            cout << "Exiting the system. Goodbye!\n";
            break;
        default:
            cout << "Invalid choice. Try again.\n";
        }
    } while (choice != 0);

    return 0;
}
```

BY TEAM 7

CRIME CONTROL

TEST CASES



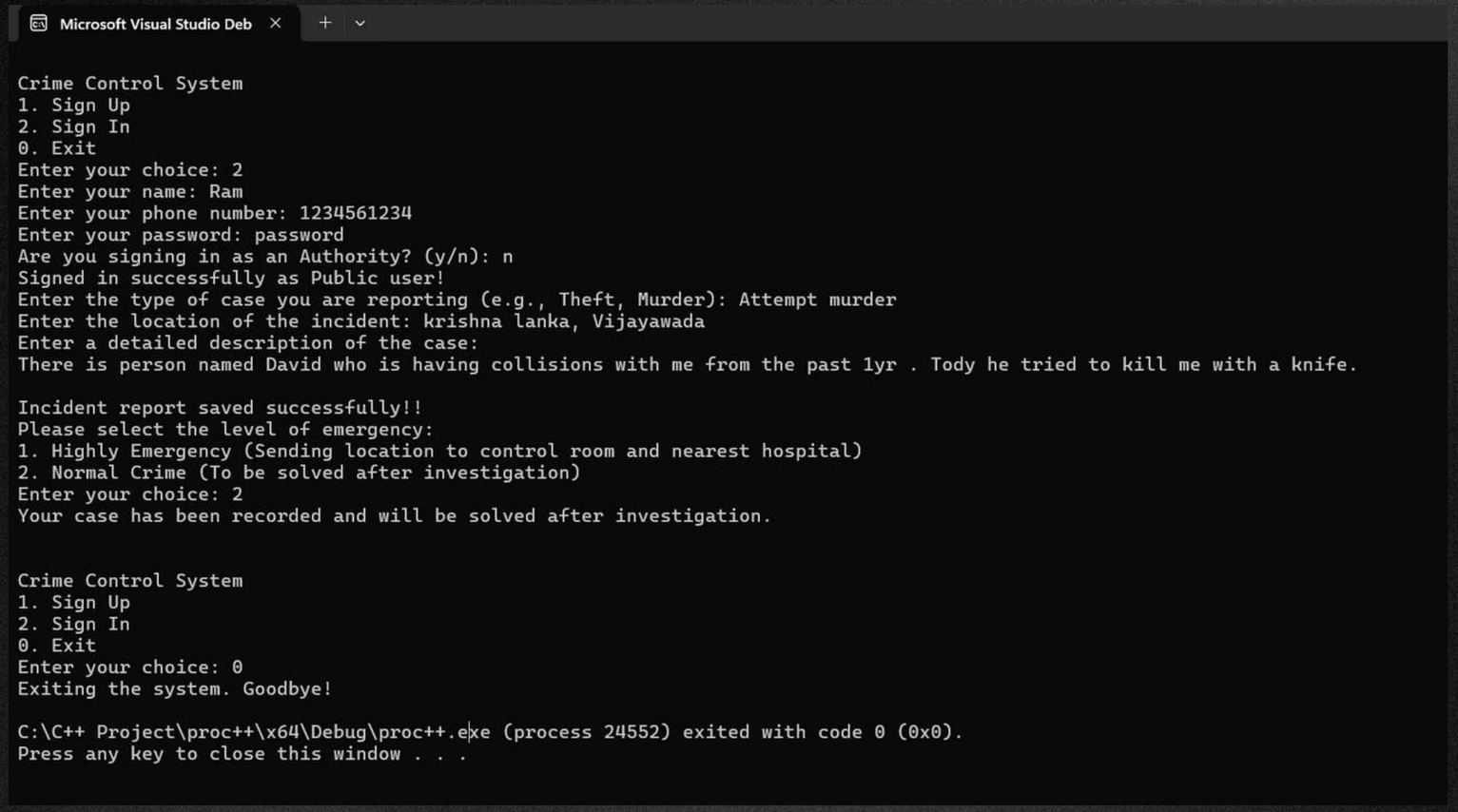
THE SIGN UP

The sign up page is only for the public users and the authority can't sign in using the sign up present here.

BY TEAM 7

CRIME CONTROL

TEST CASES



SIGN IN AND REPORTING

This is how the user can sign in with exact details and then report the incident.

BY TEAM 7

CRIME CONTROL

TEST CASES

C:\C++ Project\proc++\x64'

```
Crime Control System
1. Sign Up
2. Sign In
0. Exit
Enter your choice:
2
Enter your name: Prudhvi
Enter your phone number: 1234567890
Enter your password: password2
Are you signing in as an Authority? (y/n): y
Signed in successfully as Authority!
Enter a keyword to search reports: Kidnapping
Searching reports for keyword: Kidnapping
No reports found with the keyword "Kidnapping".
Do you want to search for another keyword? (y/n): y
Enter a new keyword to search: Murder
Searching reports for keyword: Murder
Found in filename: "Murder-Prakasham_barrage_-Ram.txt"

Files found:
1. "Murder-Prakasham_barrage_-Ram.txt"
Enter the number of the file you want to read (or 0 to search for another keyword, -1 to exit): 1

--- Contents of "Murder-Prakasham_barrage_-Ram.txt" ---
Reported by: Ram
Phone: 1234561234
Location: Prakasham_barrage_
Description: Some deadbody is present here. It's covered in blood and has multiple stabs
----------------------------------------
----------------------------------------

Files found:
1. "Murder-Prakasham_barrage_-Ram.txt"
Enter the number of the file you want to read (or 0 to search for another keyword, -1 to exit): 0
Enter a new keyword to search: Chain
Searching reports for keyword: Chain
Found in filename: "Chain snaching-benz circle-Gnana.txt"
Found in filename: "Chain_snatching-Benz_circle-Gnana.txt"

Files found:
1. "Chain snaching-benz circle-Gnana.txt"
2. "Chain_snatching-Benz_circle-Gnana.txt"
Enter the number of the file you want to read (or 0 to search for another keyword, -1 to exit): 1

--- Contents of "Chain snaching-benz circle-Gnana.txt" ---
Reported by: Gnana
Phone: 1234567890
Location: benz circle
Description: someone has robbed my chain when I was walking near the benz circle beside the trendset mall
----------------------------------------
```

AUTHORITY SIGN IN AND REPORT SEARCHING

The authority can search for any key word until he finds the desired file, if exists.

BY TEAM 7

CRIME CONTROL

TEST CASES

```
Enter a new keyword to search: Ram
Searching reports for keyword: Ram
Found in filename: "Attempt murder-krishna lanka, Vijayawada-Ram.txt"
Found in content of: "Attempt murder-krishna lanka, Vijayawada-Ram.txt"
Found in filename: "Murder-Prakasham_barrage_-Ram.txt"
Found in content of: "Murder-Prakasham_barrage_-Ram.txt"
Found in filename: "Theft-Balaji_nagar__Vijayawada-Ram.txt"
Found in content of: "Theft-Balaji_nagar__Vijayawada-Ram.txt"

Files found:
1. "Attempt murder-krishna lanka, Vijayawada-Ram.txt"
2. "Attempt murder-krishna lanka, Vijayawada-Ram.txt"
3. "Murder-Prakasham_barrage_-Ram.txt"
4. "Murder-Prakasham_barrage_-Ram.txt"
5. "Theft-Balaji_nagar__Vijayawada-Ram.txt"
6. "Theft-Balaji_nagar__Vijayawada-Ram.txt"
Enter the number of the file you want to read (or 0 to search for another keyword, -1 to exit): |
```

AUTHORITY SIGN IN AND REPORT SEARCHING

The keywords can include any word name of the person who reported or place or type of case that's been reported

BY TEAM 7

CRIME CONTROL

TEST CASES

```
Files found:
1. "Attempt murder-krishna lanka, Vijayawada-Ram.txt"
2. "Attempt murder-krishna lanka, Vijayawada-Ram.txt"
3. "Murder-Prakasham_barrage_-Ram.txt"
4. "Murder-Prakasham_barrage_-Ram.txt"
5. "Theft-Balaji_nagar__Vijayawada-Ram.txt"
6. "Theft-Balaji_nagar__Vijayawada-Ram.txt"
Enter the number of the file you want to read (or 0 to search for another keyword, -1 to exit): -1
Exiting the search.

Crime Control System
1. Sign Up
2. Sign In
0. Exit
Enter your choice: 0
Exiting the system. Goodbye!

C:\C++ Project\proc++\x64\Debug\proc++.exe (process 8920) exited with code 0 (0x0).
Press any key to close this window . . .
```

The Exit

To exit the search we should press -1 and to exit completely we should press 0.

BY TEAM 7

CRIME CONTROL

THANK YOU

RASAMSETTI GNANA PRUDHVI AP23110010542
VUPPU KISHORE AP23110010563
KODUR ADITHYA AP23110010573
NAVANIT REDDY TURPINTI AP23110010578

BY
TEAM 7