Capstone Password Manager

```
#include <bits/stdc++.h>
using namespace std;

string xorEncryptDecrypt(const string& str, char key) {
    string result = str;
    for (size_t i = 0; i < str.size(); ++i) {
        result[i] ^= key;
    }
    return result;
}

void createAccount() {
    string username, password, cPass, email;
    char key = 'K'; // Using a simple fixed key for account password encryption

    cout << "\nEnter username: ";
    cin >> username;

    cout << "Enter E-mail: ";
    cin >> email;

    cout << "Enter password: ";
    cin >> password;

    cout << "Enter password again: ";
    cin >> cPass;

    if (cPass != password) {
        cout << "Password doesn't match. Try again!" << endl;
        return; // Exit the function if passwords don't match
    }

    string encryptedPassword = xorEncryptDecrypt(password, key);

    ofstream file("accounts.txt", ios::app);
    if (file.is_open()) {
```

```
        file << username << " " << email << " " << encryptedPassword << endl;
        file.close();
        ofstream userFile(username + ".txt");
        userFile.close();
        cout << "Account created successfully\n";
    } else {
        cout << "Error: Unable to open file\n";
    }
}

bool login(string &loggedInUser) {
    string username, password, email, storedUsername, storedPassword, storedEmail;
    char key = 'K'; // Using the same fixed key for account password encryption

    cout << "\nEnter username: ";
    cin >> username;

    cout << "Enter e-mail: ";
    cin >> email;

    cout << "Enter password: ";
    cin >> password;

    ifstream file("accounts.txt");
    if (file.is_open()) {
        while (file >> storedUsername >> storedEmail >> storedPassword) {
            string decryptedPassword = xorEncryptDecrypt(storedPassword, key);
            if (storedUsername == username && storedEmail == email && decryptedPassword ==
password) {
                file.close();
                loggedInUser = username;
                return true;
            }
        }
        file.close();
    } else {
        cout << "Error: Unable to open file\n";
    }

    return false;
```

```
}

void forgotPassword() {
    string username, email, storedUsername, storedEmail, storedPassword;
    char key = 'K'; // Using the same fixed key for account password encryption

    cout << "\nEnter username: ";
    cin >> username;

    cout << "Enter e-mail: ";
    cin >> email;

    ifstream file("accounts.txt");
    if (file.is_open()) {
        while (file >> storedUsername >> storedEmail >> storedPassword) {
            if (storedUsername == username && storedEmail == email) {
                string decryptedPassword = xorEncryptDecrypt(storedPassword, key);
                cout << "Your password is: " << decryptedPassword << endl;
                file.close();
                return;
            }
        }
        cout << "No account found with the provided username and email" << endl;
        file.close();
    } else {
        cout << "Error: Unable to open file\n";
    }
}

void changePassword(string username, string newPassword) {
    string cPass;
    cout << "Enter new password again: ";
    cin >> cPass;

    if (cPass != newPassword) {
        cout << "Passwords don't match. Try again!" << endl;
        return; // Exit the function if passwords don't match
    }

    ifstream inFile("accounts.txt");
```

```
    ofstream outFile("temp.txt");
    char key = 'K'; // Using the same fixed key for account password encryption

    string storedUsername, storedEmail, storedPassword;
    bool userFound = false;

    if (inFile.is_open() && outFile.is_open()) {
        while (inFile >> storedUsername >> storedEmail >> storedPassword) {
            if (storedUsername == username) {
                string encryptedPassword = xorEncryptDecrypt(newPassword, key);
                outFile << storedUsername << " " << storedEmail << " " << encryptedPassword <<
endl;
                userFound = true;
            } else {
                outFile << storedUsername << " " << storedEmail << " " << storedPassword << endl;
            }
        }
        inFile.close();
        outFile.close();

        remove("accounts.txt");
        rename("temp.txt", "accounts.txt");

        if (userFound) {
            cout << "Password changed successfully\n";
        } else {
            cout << "User not found\n";
        }
    } else {
        cout << "Error: Unable to open file\n";
    }
}

void savePassword(const string &username) {
    string websiteName, webPass, cPass, securityKey;

    cout << "\nEnter Website name: ";
    cin >> websiteName;

    cout << "Enter Website Password: ";
```

```
    cin >> webPass;

    cout << "Enter Website password again: ";
    cin >> cPass;

    if (cPass != webPass) {
        cout << "Password doesn't match. Try again!" << endl;
        return; // Exit the function if passwords don't match
    }

    cout << "Enter a security key (Numerical): ";
    cin >> securityKey;

    ofstream file(username + ".txt", ios::app);
    if (file.is_open()) {
        string encryptedPass = xorEncryptDecrypt(webPass, securityKey[0]);
        file << websiteName << " " << encryptedPass << endl;
        file.close();
        cout << "Password saved successfully\n";
    } else {
        cout << "Error: Unable to open file\n";
    }
}

void showPassword(const string &username) {
    string websiteName, storedWebsiteName, storedPassword, securityKey;

    cout << "\nEnter your security key: ";
    cin >> securityKey;

    ifstream file(username + ".txt");
    if (file.is_open()) {
        while (file >> storedWebsiteName >> storedPassword) {
            string decryptedPass = xorEncryptDecrypt(storedPassword, securityKey[0]);
            cout << "Website: " << storedWebsiteName << ", Password: " << decryptedPass << endl;
        }
        file.close();
    } else {
        cout << "Error: Unable to open file\n";
    }
```

```
}

void generatePass() {
    int length;
    cout << "\nEnter length of password: ";
    cin >> length;

    const string charset =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^
&*()_+-={}[]|;:,.<>?";

    string generatedPassword;
    srand(time(nullptr)); // Seed the random number generator with current time

    for (int i = 0; i < length; ++i) {
        generatedPassword += charset[rand() % charset.length()];
    }

    cout << "Generated Password: " << generatedPassword << endl;
}

bool hasLowerCase(const string& str) {
    for (char c : str) {
        if (islower(c)) {
            return true;
        }
    }
    return false;
}

bool hasUpperCase(const string& str) {
    for (char c : str) {
        if (isupper(c)) {
            return true;
        }
    }
    return false;
}

bool hasDigit(const string& str) {
```

```
    for (char c : str) {
        if (isdigit(c)) {
            return true;
        }
    }
    return false;
}

bool hasSpecialChar(const string& str) {
    const string specialChars = "!@#$%^&*()_+-={}[]|;:,.<>?";
    for (char c : str) {
        if (specialChars.find(c) != string::npos) {
            return true;
        }
    }
    return false;
}

void passStrength() {
    string password;
    cout << "\nEnter password to check its strength: ";
    cin >> password;

    int score = 0;
    if (password.length() >= 8) {
        ++score;
    }
    if (hasLowerCase(password)) {
        ++score;
    }
    if (hasUpperCase(password)) {
        ++score;
    }
    if (hasDigit(password)) {
        ++score;
    }
    if (hasSpecialChar(password)) {
        ++score;
    }
```

```
    cout << "Password strength: ";
    if (score < 3) {
        cout << "Weak" << endl;
    } else if (score < 5) {
        cout << "Medium" << endl;
    } else {
        cout << "Strong" << endl;
    }
}

void dashboard(const string &username) {
    int choice;

    cout << "\nWelcome to your Dashboard, " << username << "!" <<endl;

    while (true) {
        cout << "\n1. Save new password\n";
        cout << "2. Show passwords\n";
        cout << "3. Change account password\n";
        cout << "4. Generate password\n";
        cout << "5. Check password strength\n";
        cout << "6. Exit\n";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                savePassword(username);
                break;
            case 2:
                showPassword(username);
                break;
            case 3: {
                string newPassword;
                cout << "\nEnter new password: ";
                cin >> newPassword;
                changePassword(username, newPassword);
                break;
            }
            case 4:
```

```
            generatePass();
            break;
        case 5:
            passStrength();
            break;
        case 6:
            cout << "Exiting...\n";
            exit(0);
        default:
            cout << "Invalid choice\n";
      }
   }
}

int main() {
   int choice;
   string loggedInUser;

   cout << "Welcome to the Capstone Password Manager\n";

   while (true) {
      cout << "\n1. Create Account\n";
      cout << "2. Log-in\n";
      cout << "3. Forget Password\n";
      cout << "4. Exit\n";
      cout << "\nEnter your choice: ";
      cin >> choice;

      switch (choice) {
         case 1:
            createAccount();
            break;
         case 2:
            if (login(loggedInUser)) {
               cout << "Log-in successful\n";
               dashboard(loggedInUser);
            } else {
               cout << "Log-in failed\n";
            }
            break;
```

```
            case 3:
                forgotPassword();
                break;
            case 4:
                cout << "Exiting...\n";
                exit(0);
            default:
                cout << "Invalid choice\n";
        }
    }

    return 0;
}
```