A Project report on

Online Login and Registration System

SRM University-AP, Andhra Pradesh

This project is part of OOPs with C++ Lab evaluation

Computer Science and Engineering

School of Engineering and Sciences

Submitted by:

Harshini Ponnam	Rudwika Manne	Charitha Rayapati	Yeswanth Emani
AP22110010934	AP22110010958	AP22110010974	AP22110010918

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

Nov, 2023

Table of Contents:

INTRODUCTION:

We find Loging in and registering into any web page that is present in the Internet, where there are different platforms and every platform has it’s own users. In maximum all of the websites, the website asks the user to register themselves on the website or if they do have an account they ask us to login into our account.

The login And Registration System project in C++ involves mainly the user registration process. User Credentials like usernames and passwords are asked from the user. If the registration of the user is successful then with the given credentials a file will be created of a particular user in the database.

The purpose of this project is to demonstrate the fundamental concepts of file handling, user authentication, and basic security measures in C++. This project encompasses user registration, login functionality, and even includes a password reset feature.

The code that we are building for this purpose, has 3 options displayed on the output console to either login or register or to reset a password by giving the username as input. There is a menu displayed in the output console.

This mini project contains limited features, but the essential one. The system does not create an external file to store the user's data permanently. This system is in C++ Programming Language and different variables, strings have been used for the development of it.

This mini project demonstrates the practical use of a programming language its features as well as to generate an application which can be used interest to develop a good user friendly website where he can register himself, make himself a member and keep progressing in his website.

The scope of this project is in such a way that this can be used in any website that is being developed on the internet, who wants to keep track on their users.

‘Description on the concepts that have been used in this project for solving the program’

- Classes: Class is a fundamental concept that enables the implementation of object-oriented programming (OOP). A class is essentially a blueprint or a template for creating objects, which are instances of the class. It serves as a user-defined data type that encapsulates data and functions that operate on that data. Here are some key aspects of classes in C++.

  Data Members: Classes consist of data members, which are variables that store information relevant to the class. These can be of various data types and represent the state of an object.

```
class Person {  ---------> Syntax of class
public:       ---------> Access Specifier
   string name;    --------> Data/Class member.
   int age;
   void displayAge(); -------> Member function.
private:                  ------------> Access Specifier
        void diaplayName();
};
```

Member Functions: Classes also include member functions, which are functions that operate on the data members of the class. These functions define the behaviour associated with the objects of the class.

Encapsulation: Letting the class members with different access specifiers stay in the same class is the concept of encapsulation.

- Functions:- A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.
  - You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division is such that each function performs a specific task.
  - A function declaration tells the compiler about a function's name, return type, and parameters.
  - A function definition provides the actual body of the function.
  - The C++ standard library provides numerous built-in functions that your program can call.
  - A function can also be referred as a method or a sub-routine or a procedure, etc.
  - In this program we use functions for writing the code for the particular case required.
  - In each function we write the logic required for the case we require.

Source Code:

```
#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib> // For rand() and srand()

#include <ctime>   // For time()

using namespace std;

class User {

private:

   string username;

   string password;

   string email;

   bool isVerified;

   int loginAttempts;

public:

   // Constructors

   User() : isVerified(false), loginAttempts(0) {}

   User(const string& uname, const string& pwd, const string& mail)

       : username(uname), password(pwd), email(mail), isVerified(false), loginAttempts(0) {}

   // Getter functions

   string getUsername() const { return username; }

   string getPassword() const { return password; }

   string getEmail() const { return email; }

   bool getIsVerified() const { return isVerified; }
```

```
    int getLoginAttempts() const { return loginAttempts; }

    // Setter functions

    void setVerified(bool value) { isVerified = value; }

    void incrementLoginAttempts() { loginAttempts++; }

    // Member function to display user information

    void displayUserInfo() const {

        cout << "Username: " << username << "\n";

        cout << "Email: " << email << "\n";

        // For security reasons, we don't display the password here

    }

};

class UserManager {

private:

bool isUsernameTaken(const string& username) const {

    ifstream userFile("userDatabase.txt");

    string storedUsername;

    while (userFile >> storedUsername) {

        if (storedUsername == username) {

            userFile.close();

            return true; // Username already exists

        }

    }

    userFile.close();

    return false; // Username is available
```

```
}
    void sendVerificationEmail(const User& user) const {
        cout << "Verification email sent to: " << user.getEmail() << "\n";
    }
public:
    void registerUser() {
    string username, password, email;
    cout << "Enter a username: ";
    cin >> username;
    // Check if the username is already taken
    if (isUsernameTaken(username)) {
        cout << "Username already taken. Please choose a different username.\n";
        return;
    }
    cout << "Enter an email address: ";
    cin >> email;
    cout << "Enter a password: ";
    cin >> password;
    User newUser(username, password, email);
    // Send verification email
    sendVerificationEmail(newUser);
    newUser.setVerified(true);  // Set verification status to true
    // Open file in append mode
    ofstream userFile("userDatabase.txt", ios::app);
    // Check if the file is open
    if (!userFile.is_open()) {
        cerr << "Error opening file for writing.\n";
        return;
    }
    // Write user information to the file
    userFile << newUser.getUsername() << ' ' << newUser.getPassword() << ' '
             << newUser.getEmail() << ' ' << newUser.getIsVerified() << ' '
             << newUser.getLoginAttempts() << '\n';
    cout << "Registration successful! Please check your email for verification.\n";
    // Close the file
    userFile.close();
}
    bool loginUser() const {
        string username, password;
        cout << "Enter your username: ";
        cin >> username;
        cout << "Enter your password: ";
        cin >> password;
        // Open file in read mode
        ifstream userFile("userDatabase.txt");
        string storedUsername, storedPassword, storedEmail;
        bool storedIsVerified;
        int storedLoginAttempts;
```

```
// Check if the file is open
if (!userFile.is_open()) {
    cerr << "Error opening file for reading.\n";
    return false;
}
// Read user credentials from the file
bool userFound = false;
while (userFile >> storedUsername >> storedPassword >> storedEmail >>
       storedIsVerified >> storedLoginAttempts) {
    if (storedUsername == username) {
        userFound = true;
        // Check if the account is verified
        if (!storedIsVerified) {
            cout << "Account not verified. Please check your email for verification.\n";
            userFile.close();
            return false;
        }
        // Check if the account is locked
        if (storedLoginAttempts >= 3) {
            cout << "Account locked. Too many unsuccessful login attempts.\n";
            userFile.close();
            return false;
        }
        // Check if the password is correct
```

```
            if (storedPassword == password) {
                // Close the file
                userFile.close();
                return true; // Login successful
            } else {
                cout << "INCORRECT PASSWORD!!! TO RESET PASSCODE ENTER 3.\n";
                return false;
            }
        }
    }
    // If the loop completes and the username wasn't found
    if (!userFound) {
        cout << "ERROR, no account found.\n";
    }
    // Close the file
    userFile.close();
    return false; // Login failed
}
void resetPassword(User& user) {
    string newPassword;
    cout << "Enter a new password: ";
    cin >> newPassword;
    // Update the password in the user database
    ifstream userFile("userDatabase.txt");
```

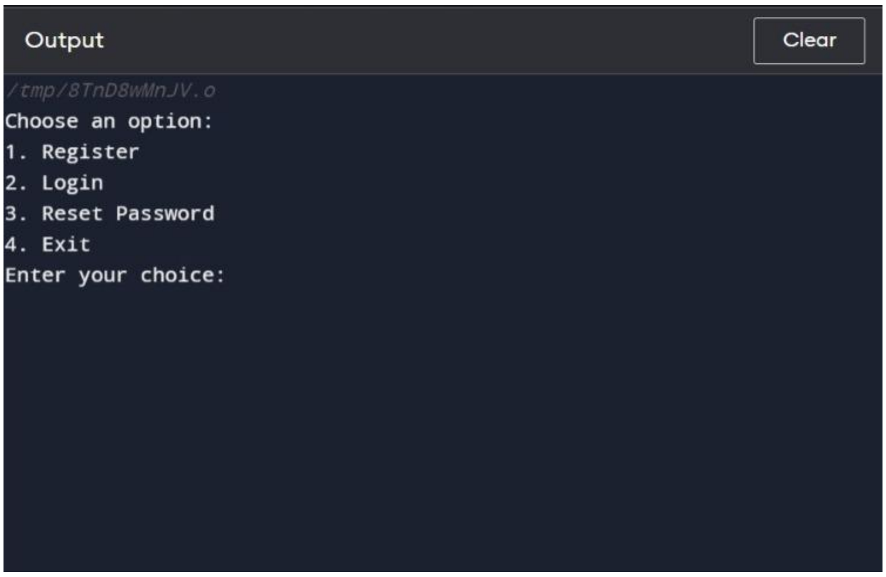
```
        ofstream tempFile("tempDatabase.txt");
        string storedUsername, storedPassword, storedEmail;
        bool storedIsVerified;
        int storedLoginAttempts;
        while (userFile >> storedUsername >> storedPassword >> storedEmail >>
               storedIsVerified >> storedLoginAttempts) {
            if (storedUsername == user.getUsername()) {
                tempFile << storedUsername << ' ' << newPassword << ' ' << storedEmail << ' '
                         << storedIsVerified << ' ' << storedLoginAttempts << '\n';
            } else {
                tempFile << storedUsername << ' ' << storedPassword << ' ' << storedEmail << ' '
                         << storedIsVerified << ' ' << storedLoginAttempts << '\n';
            }
        }
        // Close the files
        userFile.close();
        tempFile.close();
        // Rename the temp file to userDatabase.txt
        remove("userDatabase.txt");
        rename("tempDatabase.txt", "userDatabase.txt");
        cout << "Password reset successful!\n";
    }
};
int main() {
```

```
int choice;

UserManager userManager;

srand(static_cast<unsigned>(time(0))); // Seed for random number generation

do {

  cout << "Choose an option:\n";

  cout << "1. Register\n";

  cout << "2. Login\n";

  cout << "3. Reset Password\n";

  cout << "4. Exit\n";

  cout << "Enter your choice: ";

  cin >> choice;

  switch (choice) {

    case 1:

      userManager.registerUser();

      break;

    case 2:

      if (userManager.loginUser()) {

        cout << "Login successful!\n";

      }

      break;

    case 3: {

      string username;

      cout << "Enter your username: ";

      cin >> username;
      // Check if the username exists
      ifstream userFile("userDatabase.txt");
      string storedUsername, storedPassword, storedEmail;
      bool storedIsVerified;
      int storedLoginAttempts;
      while (userFile >> storedUsername >> storedPassword >> storedEmail >>
             storedIsVerified >> storedLoginAttempts) {
        if (storedUsername == username) {
            User currentUser(storedUsername, storedPassword, storedEmail);
            currentUser.setVerified(storedIsVerified);
            // Close the file
            userFile.close();
            userManager.resetPassword(currentUser);
            break;
        }
      }
      // Close the file
      userFile.close();
      break;
    }
    case 4:
      cout << "Exiting program.\n";
      break;
    default:
```

```
            cout << "Invalid choice. Try again.\n";
        }
    } while (choice != 4);
    return 0;
}
```

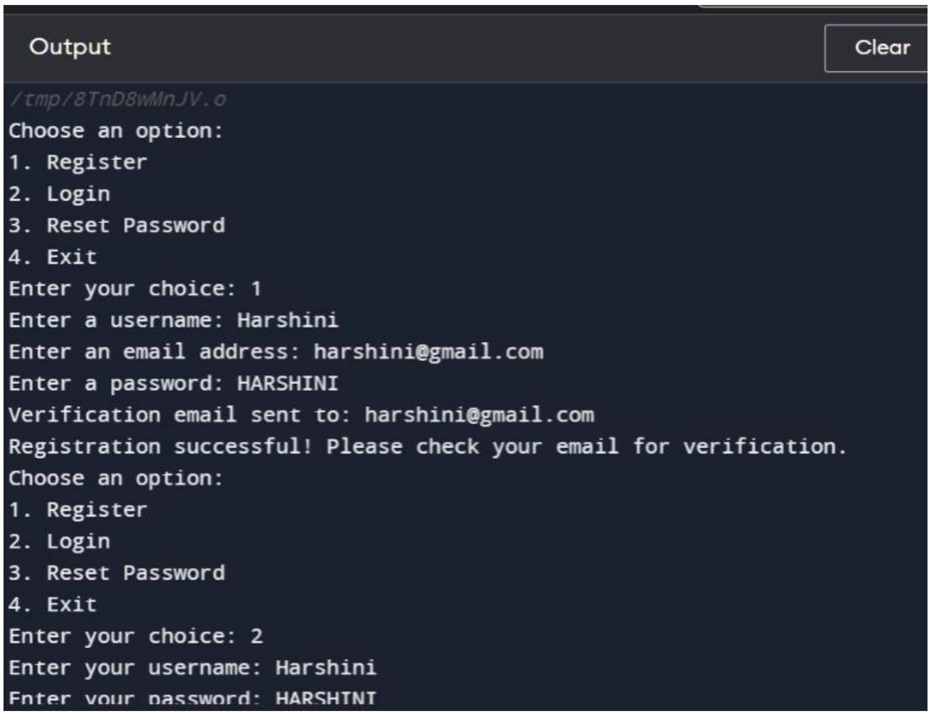
Output:

The output initially shows:

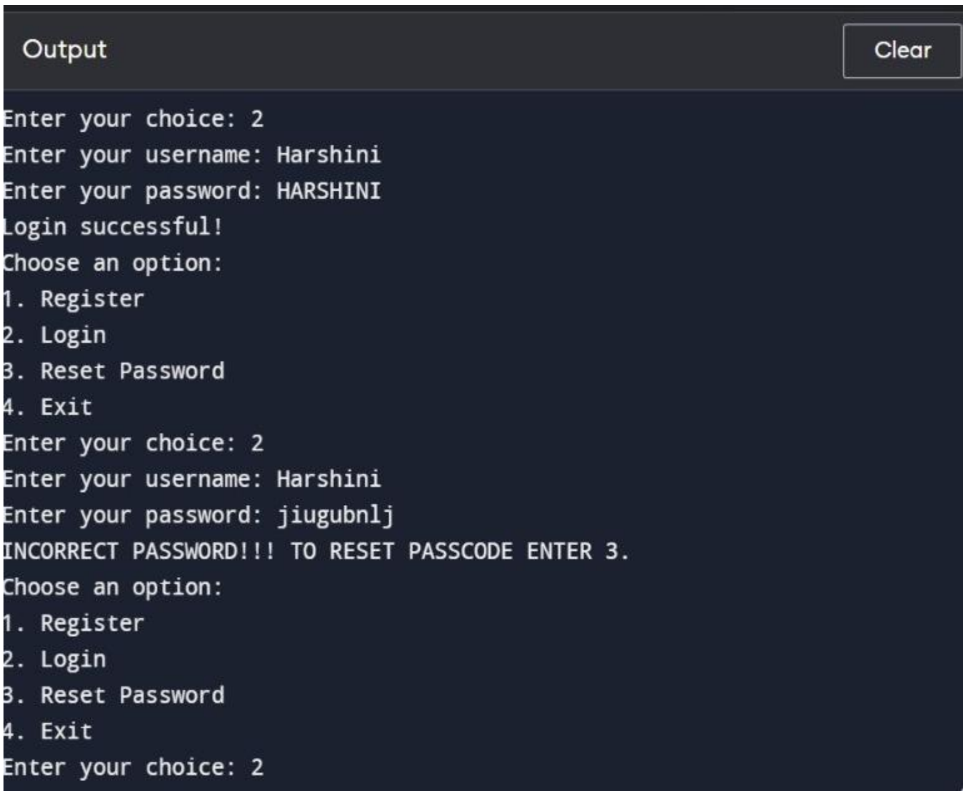


```
/tmp/8TnD8wMnJV.o
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice:
```

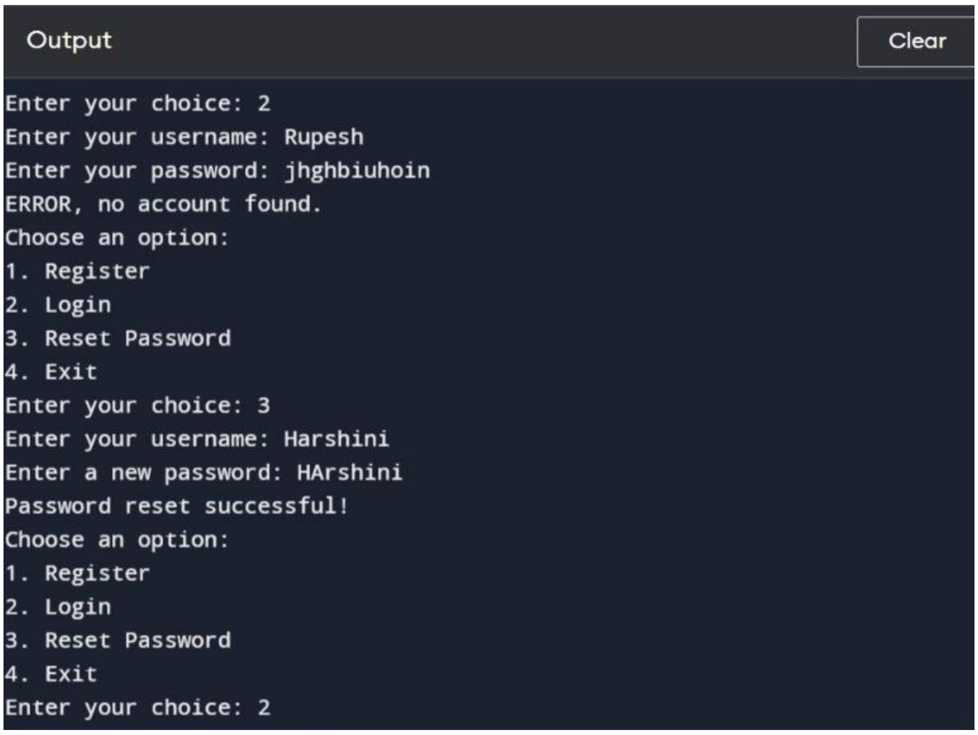
Going on with the code:



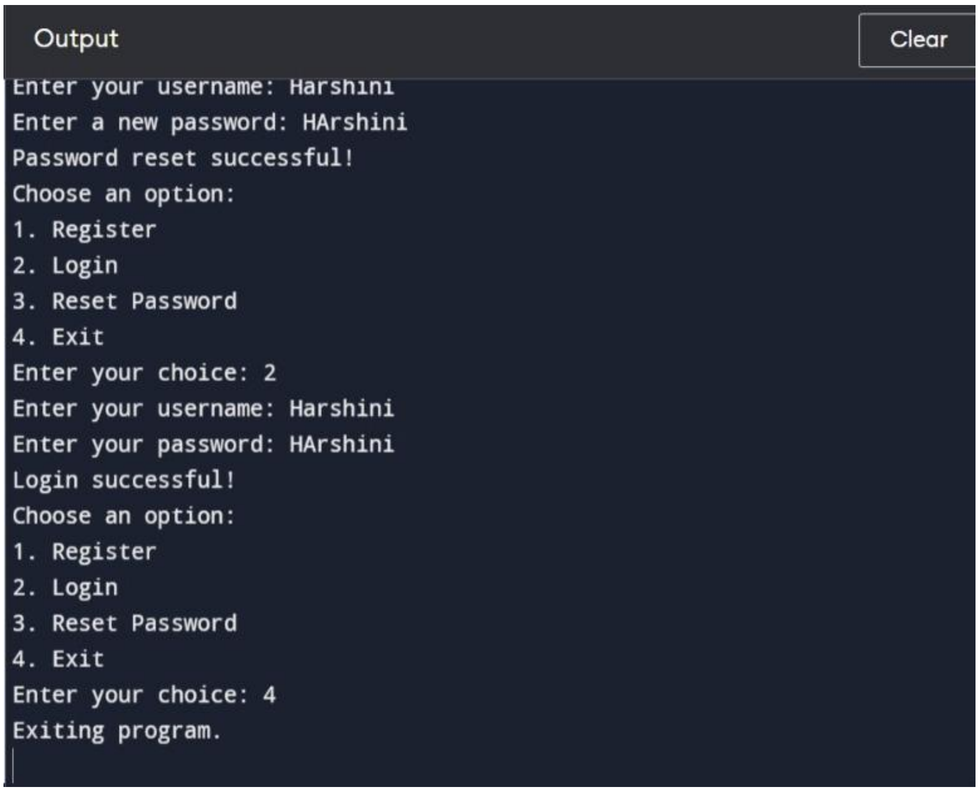
```
/tmp/8TnD8wMnJV.o
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 1
Enter a username: Harshini
Enter an email address: harshini@gmail.com
Enter a password: HARSHINI
Verification email sent to: harshini@gmail.com
Registration successful! Please check your email for verification.
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 2
Enter your username: Harshini
Enter your password: HARSHINI
```



```
Enter your choice: 2
Enter your username: Harshini
Enter your password: HARSHINI
Login successful!
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 2
Enter your username: Harshini
Enter your password: jiugubnlj
INCORRECT PASSWORD!!! TO RESET PASSCODE ENTER 3.
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 2
```



```
Enter your choice: 2
Enter your username: Rupesh
Enter your password: jhghbiuhoin
ERROR, no account found.
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 3
Enter your username: Harshini
Enter a new password: HArshini
Password reset successful!
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 2
```



```
Enter your username: Harshini
Enter a new password: HArshini
Password reset successful!
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 2
Enter your username: Harshini
Enter your password: HArshini
Login successful!
Choose an option:
1. Register
2. Login
3. Reset Password
4. Exit
Enter your choice: 4
Exiting program.
```

‘Discussion of efficiency and usability of this code by the user’

➢ Time Complexity (for all the functions designed in the code):

1. User Registration (registerUser):

Reading from the file (isUsernameTaken): O(N), where N is the number of users in the file.

Opening the file for writing: O(1)

Writing to the file: O(1)

Overall: O(N)

2. User Login (loginUser):

Opening the file for reading: O(1)

Reading from the file: O(N), where N is the number of users in the file.

Checking if the account is verified: O(1)

Checking if the account is locked: O(1)

Comparing passwords: O(1)

Overall: O(N)

3. Password Reset (resetPassword):

Opening the file for reading: O(1)

Opening the temp file for writing: O(1)

Reading from and writing to the file: O(N), where N is the number of users in the file.

Renaming files: O(1)

Overall: O(N)

➢ Space Complexity:

The space complexity is primarily determined by the storage required for the user database file. If there are N users, the space complexity would be O(N).

➢ Efficiency for Website Use:

1. File-Based Storage:

The code uses a simple text file as a database. While this is straightforward, it may not be the most efficient for a website with a large number of users. Databases like MySQL or MongoDB are more scalable and provide better performance.

2. Security:

Storing passwords in plain text in a file is a security risk. Passwords should be hashed and salted for secure storage.

The code could be enhanced to incorporate secure password storage practices.

3. Verification Process:

The verification process is simplistic. In a real-world scenario, email verification might involve more sophisticated mechanisms and security measures.

4.Lockout Mechanism:

The lockout mechanism after a certain number of unsuccessful login attempts is present but could be enhanced for better security.

5.Error Handling:

The code lacks comprehensive error handling, which is crucial for a production environment. It should handle exceptions gracefully and provide meaningful error messages.

➢ Scalability:

As the number of users increases, the linear search through the user database file could become a performance bottleneck. Using a more efficient data structure or database system could improve scalability.

CONCLUSION:

Concluding, the code provides a basic implementation for production-level website use where this code can be enhanced furtherly to improve security, scalability, and overall robustness for real-world deployment.

Our team believes that the design yields a simple, easy to use console application. The user is prompted through the system at every step of the way with sufficient error checking to prevent invalid input.

We(our team) have done this project with the help of Reference books and online reference links and executed the program with all required cases in the program to demonstrate every function in the program.