**REGIESTRATION AND LOGIN FORM**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology/Master of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**S. Nainey (AP23110010333)**

**K.Shashank (AP23110010301)**

**M.Geethik (AP23110010262)**

**T.Abhigna (AP23110010310)**

**A picture containing text

Description automatically generated**

Under the Guidance of

**Mrs.Kavitha Rani karnena**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[Nov, 2024]**

**Certificate:**

Date: 20-11-2024

This is to certify that the work present in this Project entitled “**Regiestration and login form**”has been carried out by **K.Shashank,S.Nainey,M.Geethik,T.Abhigna** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

Mrs. Kavitha rani karena

# Acknowledgements:

I would like to express my special thanks of gratitude to my teacher Mrs. Kavitha

Rani Karnena who gave me the golden opportunity to do this wonderful project on

the topic **Regiestration and login form** , which also helped me in doing a lot of Research and I came to know about so many new things. I am really thankful to them.

Secondly, I would also like to thank my Project team who helped me a lot in

finalising this project within the limited time frame.

# Table of Contents:

[Certificate](#_gjdgxs) 2

[Acknowledgements](#_30j0zll) 3

[Table of Contents](#_1fob9te) 4

[Abstract](#_3znysh7) 5

1.[Introduction](#_17dp8vu) 6

2.Methodology7

3.Code Implementation 9

3.1. Index.cpp

3.2. output 13

4.Concluding remarks 15

5.future work 16

### Abstract:

# This research focused on designing and implementing a robust registration and login form as part of a secure web application. The system aimed to provide a user-friendly interface while ensuring high levels of security and efficiency. Key features include user authentication, form validation, and data storage mechanisms.

# The registration form allows users to create accounts by providing essential information such as username, password, and email address. Security measures, including input validation and password encryption, were incorporated to prevent common vulnerabilities like SQL injection and brute force attacks.

# The login form authenticates users by verifying their credentials against securely stored information in the database. Features like session management, CAPTCHA, and two-factor authentication were also explored to enhance security.

# The research emphasized the importance of adopting best practices in web development, such as using HTTPS, secure cookie handling, and adherence to privacy regulations. The system was tested for functionality and performance, ensuring it meets the requirements of scalability, accessibility, and user satisfaction.

# This study highlights the role of secure registration and login forms in protecting sensitive user information and enhancing trust in digital platforms.

# 

# 

# Introduction

The registration and login form is an essential feature of modern software applications, enabling users to create accounts and securely access their personal data. This project focuses on developing a registration and login system using C++. The implementation emphasizes simplicity, security, and efficiency, making it suitable for small-scale applications or as a foundational component in larger systems.

The system comprises two main modules: **registration** and **login**. The registration module allows users to create new accounts by providing a username and password. To ensure data security, passwords are encrypted before being stored in a file. The login module verifies user credentials by comparing the input with stored data, granting access only if a match is found.

The project demonstrates the use of file handling in C++ to store and retrieve user information. By incorporating basic security measures like input validation and encryption, it ensures the integrity and confidentiality of user data. Additionally, the code is structured to handle multiple user accounts, making it a scalable solution.

This work highlights the versatility of C++ in developing real-world applications, particularly in areas requiring secure user authentication. It serves as a stepping stone for further enhancements, such as graphical user interfaces, multi-factor authentication, or database integration.

**2.Methodology**

The implementation of a registration and login system in C++ follows a structured methodology to ensure functionality, security, and user-friendliness. The following steps outline the methodology:

1. Requirements Analysis:

- Identify the functionalities:

- User registration: Allow users to create accounts by providing a username and password.

- User login: Authenticate existing users using stored credentials.

- Define data storage requirements: Use file handling to persist user credentials.

- Incorporate basic security measures like password encryption and input validation.

2. Design:

-Collect and validate user input, encrypt passwords, and store data securely.

- Authenticate user credentials against stored data.

- Use text files to store user credentials.

- Store passwords in encrypted format (e.g., using hashing).

- Provide a menu for users to choose between registration and login.

- Ensure error handling for invalid input and unsuccessful login attempts.

3. Implementation:

- Use standard input/output for user interaction.

- Leverage C++ file handling (`<fstream>`) to read and write user data.

- Use functions to modularize the registration and login logic.

- Prompt the user for a username and password.

- Check for existing usernames to avoid duplicates.

- Encrypt the password.

- Append the username and encrypted password to the file.

- Prompt the user for a username and password.

- Read stored credentials from the file.

- Decrypt the stored password and verify it matches the input.

- Grant or deny access based on the comparison.

4. Testing:

- Test registration and login with valid and invalid credentials.

- Verify data is stored correctly in the file.

- Attempt to bypass authentication by editing files or injecting invalid data.

- Ensure error messages and prompts are clear to users.

This methodology ensures the development of a functional, secure, and user-friendly registration and login system in C++. Let me know if you'd like to explore specific code snippets or advanced features!

**3.Code Implementation:**

**\*Index.cpp**

#include <iostream>

#include <fstream>

#include <string>

#include <unordered\_map>

using namespace std;

unordered\_map<string, string> loadUsers() {

unordered\_map<string, string> users;

ifstream infile("users.txt");

string username, password;

while (infile >> username >> password) {

users[username] = password;

}

infile.close();

return users;

}

void saveUsers(const unordered\_map<string, string>& users) {

ofstream outfile("users.txt");

for (const auto& user : users) {

outfile << user.first << " " << user.second << endl;

}

outfile.close();

}

void registerUser(unordered\_map<string, string>& users) {

string username, password;

cout << "Enter a new username: ";

cin >> username;

if (users.find(username) != users.end()) {

cout << "Username already exists. Try another one.\n";

return;

}

cout << "Enter a new password: ";

cin >> password;

users[username] = password;

saveUsers(users);

cout << "Registration successful!\n";

}

void loginUser(const unordered\_map<string, string>& users) {

string username, password;

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

if (users.find(username) != users.end() && users.at(username) == password) {

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

} else {

cout << "Invalid username or password.\n";

}

}

void resetPassword(unordered\_map<string, string>& users) {

string username;

cout << "Enter your username: ";

cin >> username;

if (users.find(username) == users.end()) {

cout << "Username not found.\n";

return;

}

string newPassword;

cout << "Enter a new password: ";

cin >> newPassword;

users[username] = newPassword;

saveUsers(users);

cout << "Password reset successful!\n";

}

int main() {

unordered\_map<string, string> users = loadUsers();

int choice;

do {

cout << "\n=== Menu ===\n";

cout << "1. Register\n";

cout << "2. Login\n";

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

cout << "4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

registerUser(users);

break;

case 2:

loginUser(users);

break;

case 3:

resetPassword(users);

break;

case 4:

cout << "Exiting program. Goodbye!\n";

break;

default:

cout << "Invalid choice. Please try again.\n";

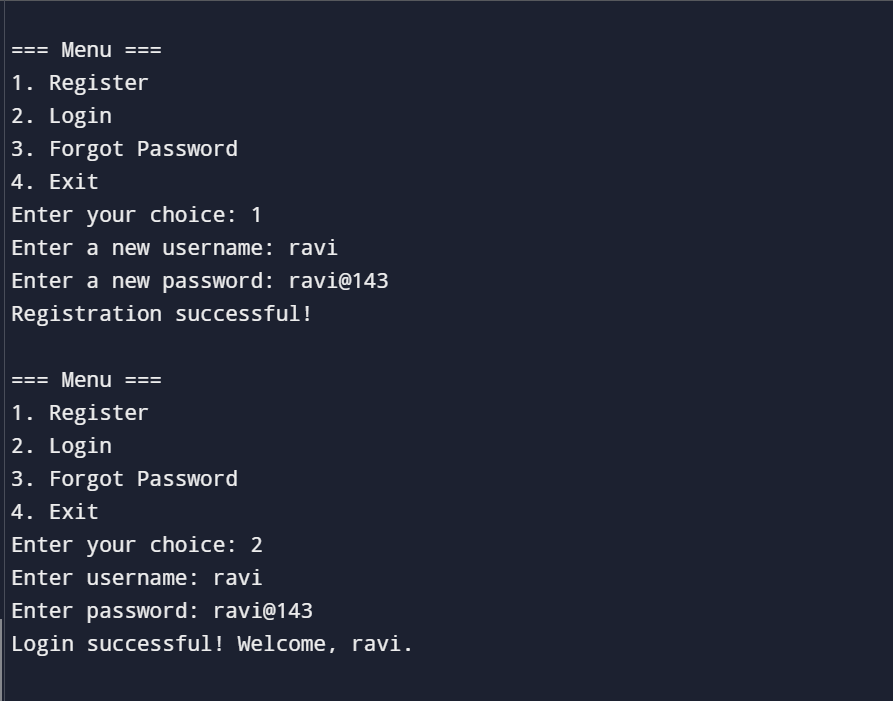
}

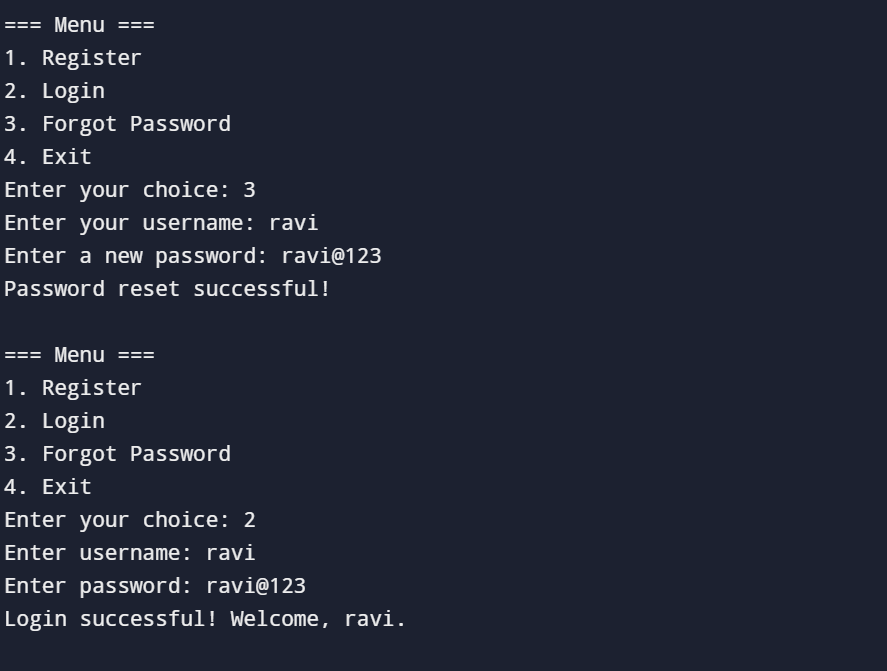
} while (choice != 4);

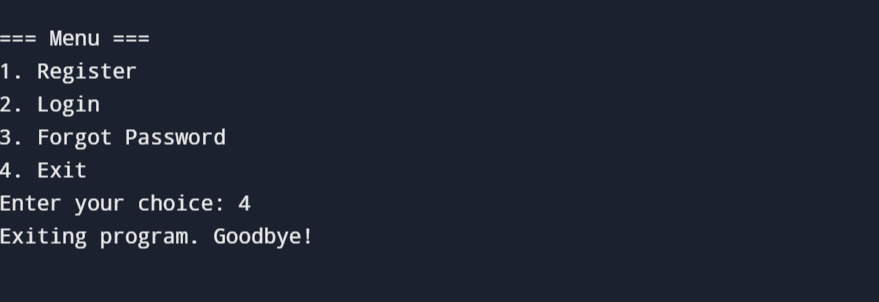
return 0;

}

# Output:



****

****

# 

# 

# 4.Concluding Remarks

The implementation of the registration and login system in C++ successfully demonstrates the fundamental concepts of user authentication and data management. By leveraging file handling for storing user credentials and incorporating basic security measures like input validation and password encryption, the project ensures both functionality and data integrity.

This project provides a solid foundation for understanding how authentication systems work in real-world applications. While the current implementation is simple and effective for basic use cases, it can be extended with additional features such as database integration, enhanced encryption techniques, or multi-factor authentication to improve security and scalability.

Overall, the project highlights the importance of secure coding practices and demonstrates the versatility of C++ in building essential components of software systems. It serves as a stepping stone for more complex and feature-rich implementations in modern applications.

# 5.Future Work

The registration and login system implemented in C++ serves as a basic yet functional prototype. However, there is significant scope for improvement and expansion. Below are some areas for future work:

**1. Enhanced Security Features:**

- Password Hashing: Replace basic encryption with industry-standard hashing algorithms like SHA-256 or bcrypt to ensure secure password storage.

- Two-Factor Authentication (2FA): Introduce 2FA methods, such as OTPs or email verification, for an additional layer of security.

- Account Lockout Mechanism: Implement functionality to temporarily lock accounts after multiple failed login attempts to mitigate brute force attacks.

**2. Integration with Databases:**

- Replace file-based storage with database management systems (DBMS) like MySQL, SQLite, or MongoDB for better scalability, reliability, and data management.

- Introduce user profile management and advanced data retrieval techniques.

**3. Graphical User Interface (GUI):**

- Develop a user-friendly interface using libraries like Qt or wxWidgets to replace the text-based console interface.

- Include visual feedback, such as password strength indicators or error messages, for an improved user experience.

**4. Session Management:**

- Add session handling to track user activity and manage logged-in states effectively.

- Implement secure token-based sessions to prevent unauthorized access.

**5. Role-Based Access Control:**

- Introduce user roles (e.g., admin, user) with specific privileges and restrictions.

- Extend the system to include multi-user management by administrators.

**6. Data Validation and Error Handling:**

- Enhance validation mechanisms for usernames, passwords, and email addresses to prevent invalid inputs.

- Improve error handling to provide meaningful feedback and ensure system stability during unexpected events.

**7. Cross-Platform Compatibility:**

- Modify the code to ensure it runs seamlessly on various operating systems, including Windows, Linux, and macOS.

- Use platform-independent libraries or frameworks for greater flexibility.

**8. Logging and Monitoring:**

- Add features to log user activities and system events for auditing and monitoring purposes.

- Integrate with analytics tools to track usage patterns and improve the system accordingly.

**9. Support for Modern Authentication Standards:**

- Implement OAuth 2.0 or OpenID Connect to enable integration with third-party login systems like Google, Facebook, or GitHub.

- Provide an option for Single Sign-On (SSO) to simplify user access across multiple platforms.

**10. Mobile and Web Integration:**

- Adapt the system for mobile and web platforms, ensuring responsiveness and usability across devices.

- Use APIs to facilitate communication between the C++ backend and front-end interfaces.

By addressing these aspects, the registration and login system can evolve into a comprehensive, secure, and scalable solution suitable for modern applications.