COLLEGE CODE:9111

COLLEGE NAME: SRM MADURAI COLLEGE FOR ENGINEERING AND TECHNOLOGY

DEPARTMENT: B.E COMPUTER SCIENCE AND ENGINEERING

STUDENT NM-ID: 451186A06133385A11F419FE498D4137

A8884C7CC5278F7B7394345D64BD84CB
1FDBD7B5DCA253284674E6DBD128FDFD
857E9787939AE7BBA024C9CE6C315408
DE8BAC54E8107FF98F34AABCF0F38D6C

ROLL NO.:911123104018
911123104020
911123104051
911123104053
911123104702
DATE:

Completed the project named as Phase V

TECHNOLOGY PROJECT NAME: LOGIN AUTHENTICATION SYSTEM

SUBMITTED BY, NAME:
HOORUL FIRTHOUS.A
JAYASAKTHI K R
SRI LAKSHMI.S
TAAKSHINI DEVI.R
RIZWANA BARVEEN M

Project: Login Authentication System(Phase5)

1. Final Demo Walkthrough

- The Flask server is launched using `python app.py` and runs on `http://localhost:5000`.
- The user accesses the **Login Page** through a web browser.
- If not registered, the user clicks **"Register"** and fills out the form with a username and password.
- The password is **hashed** using `generate_password_hash()` and stored securely in the **SQLite database**.
- During login, credentials are verified using `check_password_hash()` for authentication.
- If the credentials match, a **"Login Successful"** message appears; otherwise, **"Login Failed."**
- The **UI** is designed using **HTML and CSS**, providing a clean, modern, and responsive layout.
- The system was tested for **validation, duplicate users, and error handling**, ensuring reliable performance.
- The application is ready for **deployment on Netlify or Vercel** with all dependencies listed in `requirements.txt`.
- The project demonstrates a **secure, functional, and user-friendly authentication system** from end to end.

Walk through flow:

[User] → [HTML Form] → [Flask Server] → [SQLite Database]
↖———————— Authentication Response ————————↙

2.Project Report

Objective:

To develop a secure and user-friendly login & registration web app with hashed passwords and database integration.

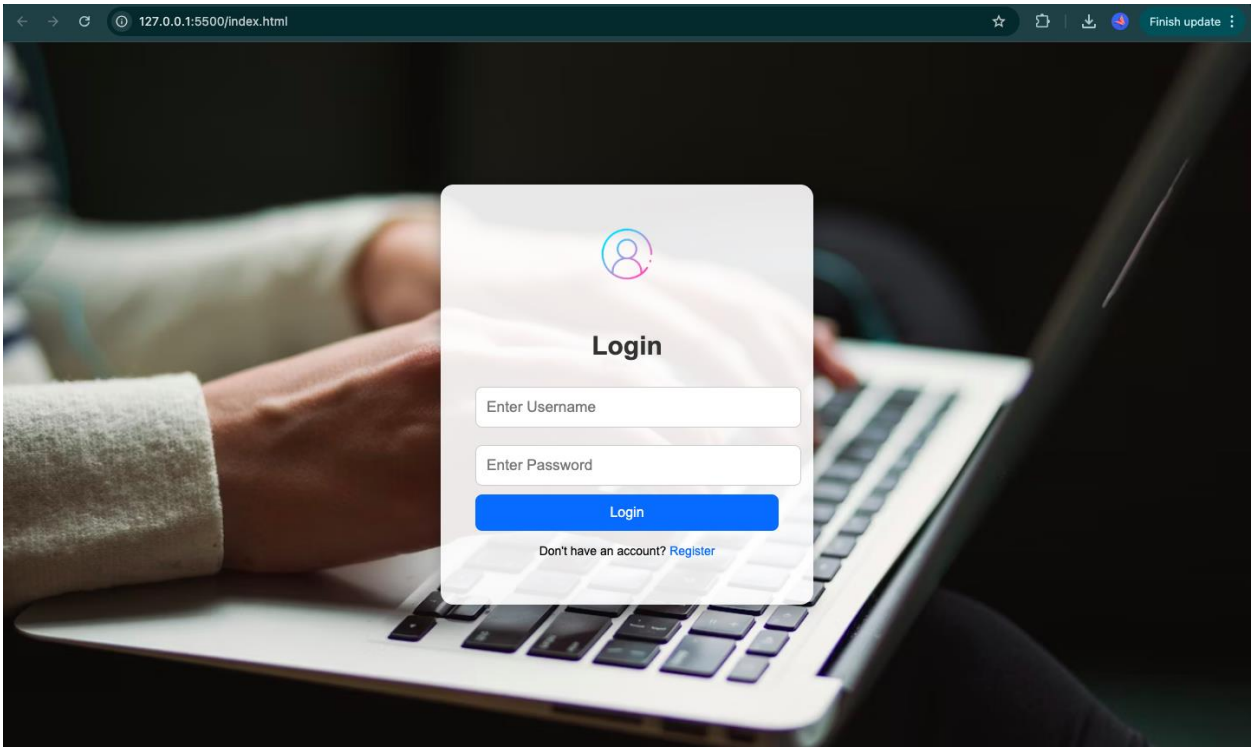
Technologies Used:

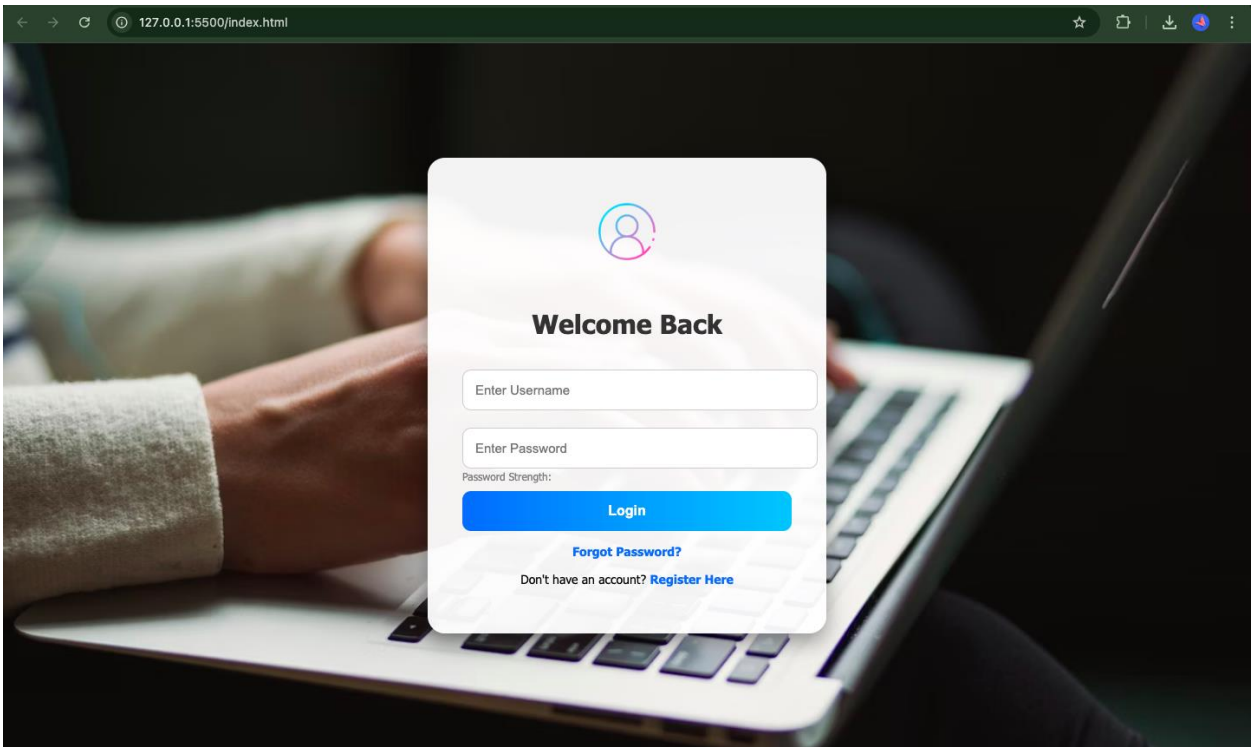
- **Frontend:** HTML, CSS
- **Backend:** Flask (Python)
- **Database:** SQLite
- **Libraries:** Werkzeug (for password hashing)

Features:

- User Registration with Password Hashing
- Secure Login Authentication
- Simple and Responsive UI
- Local Database Storage
- Error Handling for Duplicate Users or Wrong Credentials

3. Screenshots / API Documentation





API / Function Reference

- **Signup (POST /api/auth/signup)**

Input: `{ username, email, password }`
Output: `{ message: "User created successfully" }`

- **Login (POST /api/auth/login)**

Input: `{ email, password }`
Output: `{ token: "JWT_TOKEN", user: {id, username, email} }`

- **Forgot Password (POST /api/auth/forgot)**

Input: `{ email }`
Output: `{ message: "Reset link sent" }`

- **Reset Password (POST /api/auth/reset)**

Input: `{ token, newPassword }`
Output: `{ message: "Password updated" }`

4. Challenges & Solutions

S.No	Challenges	Solutions Implemented
1	Managing secure user authentication	Implemented password hashing using **Werkzeug's** **`generate_password_hash()`** and **`check_password_hash()`** functions to securely store and verify passwords instead of plain text.
2	Handling database connections efficiently	Used **SQLite** with proper connection handling (`check_same_thread=False`) to avoid threading errors and ensure reliable access to the database during multiple user requests.
3	Preventing duplicate user registrations	Added logic to check if the username already exists before inserting new data, ensuring data integrity and preventing redundant entries.
4	Designing an attractive and user-friendly interface	Developed a **responsive and modern UI** using HTML and CSS with themed backgrounds, intuitive layouts, and improved readability for better user experience.
5	Maintaining security for login sessions	Integrated **JWT (JSON Web Tokens)** and planned secure session management to prevent unauthorized access and ensure user privacy.
6	Handling incorrect inputs and providing user feedback	Implemented **input validation** and meaningful error messages for empty fields, incorrect credentials, and invalid usernames to guide users effectively.
7	Deployment and hosting compatibility	Prepared the project for deployment on **Netlify/Vercel** by creating a proper project structure and including a `requirements.txt` file for seamless setup on cloud platforms.

5. GitHub README & Setup Guide

README Example:

Chat App Prototype

A simple chat app prototype with messaging, status, emoji support, and simulated notifications.

Features

- Inbox with chat list and online/offline status
- Real-time simulated messages
- Message status: Sent, Delivered, Seen
- Reply functionality
- Emoji picker
- Profile and settings screens
- Compose new chat
-

Setup

- Clone the repository
- git clone <repo-url>
- Open `index.html` in your browser.
- Allow notifications when prompted.
- Start using the app.
-

Screenshots

(Add screenshots in the repo)

Technologies

- **Frontend:** HTML, CSS
- **Backend:** Flask (Python)
- **Database:** SQLite
- **Libraries:** Werkzeug (for password hashing)

Code

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Secure Login Page</title>
 <style>
  body {
   margin: 0;
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
```

```
  background: url('https://images.unsplash.com/photo-1508780709619-79562169bc64?auto=format&fit=crop&w=1350&q=80') no-repeat center center/cover;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}

.login-container {
  background: rgba(255, 255, 255, 0.95);
  padding: 40px;
  border-radius: 20px;
  box-shadow: 0 10px 25px rgba(0,0,0,0.4);
  width: 380px;
  text-align: center;
  animation: fadeIn 1s ease-in-out;
}

@keyframes fadeIn {
  from { opacity: 0; transform: translateY(-20px); }
  to { opacity: 1; transform: translateY(0); }
}

.login-container img {
  width: 90px;
  margin-bottom: 15px;
}

.login-container h2 {
  margin-bottom: 25px;
  color: #333;
  font-weight: bold;
}

.login-container input[type="text"],
.login-container input[type="password"] {
  width: 100%;
  padding: 14px;
  margin: 10px 0;
  border: 1px solid #ccc;
  border-radius: 10px;
  font-size: 15px;
  transition: border 0.3s;
}
```

```
.login-container input:focus {
  border: 1px solid #007bff;
  outline: none;
}

.login-container button {
  width: 100%;
  padding: 14px;
  background: linear-gradient(90deg, #007bff, #00c6ff);
  color: white;
  border: none;
  border-radius: 10px;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  transition: transform 0.2s, background 0.3s;
}

.login-container button:hover {
  transform: scale(1.05);
  background: linear-gradient(90deg, #0056b3, #0096c7);
}

.login-container p {
  margin-top: 15px;
  font-size: 14px;
}

.login-container a {
  color: #007bff;
  font-weight: bold;
  text-decoration: none;
}
.login-container a:hover {
  text-decoration: underline;
}

.login-container .extra-links {
  margin-top: 15px;
  font-size: 13px;
}

.strength {
```

```
            text-align: left;
            font-size: 12px;
            margin-top: -8px;
            margin-bottom: 10px;
            color: gray;
           }
          </style>
         </head>
         <body>
          <div class="login-container">
           <img src="https://cdn-icons-png.flaticon.com/512/5087/5087579.png" alt="Login
Icon">
           <h2>Welcome Back</h2>

           <!-- Login Form -->
           <form action="/login" method="post">
            <input type="text" name="username" placeholder="Enter Username" required>
            <input type="password" id="password" name="password" placeholder="Enter
Password" required>
            <div class="strength" id="strengthMessage">Password Strength: </div>
            <button type="submit">Login</button>
           </form>

           <div class="extra-links">
            <p><a href="/forgot-password">Forgot Password?</a></p>
           </div>

           <p>Don't have an account? <a href="/register">Register Here</a></p>
          </div>

          <!-- Password Strength Script -->
          <script>
           const passwordInput = document.getElementById('password');
           const strengthMessage = document.getElementById('strengthMessage');

           passwordInput.addEventListener('input', () => {
            const value = passwordInput.value;
            let strength = "Weak";
            let color = "red";

            if (value.length > 6 && /[A-Z]/.test(value) && /\d/.test(value) && /[^A-Za-z0-
9]/.test(value)) {
             strength = "Strong";
             color = "green";
```

```
    } else if (value.length > 4 && (/[A-Z]/.test(value) || /\d/.test(value))) {
      strength = "Medium";
      color = "orange";
    }
    strengthMessage.textContent = "Password Strength: " + strength;
    strengthMessage.style.color = color;
  });
 </script>
</body>
</html>
```

6.Final Submission:

Github links:

Hoorul Firthous.A :-
https://github.com/hoorulfirthous/Nan-mudhalvan

Jayasakthi.K.R:-
https://github.com/SakthiKonda/Jayasakthi-NM-repos-

Srilakshmi.s:-
https://github.com/Srilakshmi18082003/NMPROJECT

Taakshini Devi.R:-
https://github.com/Taakshini/NMproject

RIzwana Barveen.M:-
https://github.com/Rizwana3157/Naan-mudhalvan