COLLEGE CODE: 9123

COLLEGE NAME: SACS MAVMM Engineering College

STUDENT NM-ID: 4822D33F698C3B88D634C38C8237F3F6

STUDENT NM-ID: 82DBE839B5BEB7697C643AE558BB639A

STUDENT NM-ID: 18C5FEC84FBE1F2ADD54DD0B31E5C2E1

STUDENT NM-ID: 26F7A4E3FB2E532BFC44304B2A1EBED9

ROLL NO:23CS030

ROLL NO:23CS021

ROLL NO:23CS052

ROLL NO:23CS022

DATE: 06.10.2025

COMPLETED THE PROJECT NAMED AS PHASE 5 TECHNOLOGY

PROJECT NAME: LOGIN AUTHENTIFICATION

NAME	REG NO	PHONE NO	EMAIL ID
J. MEENAKSHI	912323104027	8838321932	meenu.mailforyou@gmail.com
A.V. HARINI	912323104021	9787862865	harinisaravindababu@gmail.com
R. THIBITHRA	912323104049	8124810977	thibithra06@gmail.com
T.S. JANANI	912323104022	9043006559	janasri255@gmail.com

SUBMITTED BY,

J. Meenakshi

A.V. Harini

R. Thibithra

T.S. Janani

ENHANCEMENTS & DEPLOYMENT(Deadline – Week 9)

1. Login Authentication – Overview

Login authentication is a process that verifies a user's identity before granting access to a system, application, or website. It ensures that only authorized users can access protected resources. In your project, this typically involves:

- **Frontend:** Login form (username/email + password)
- **Backend:** API endpoint to validate credentials
- **Database:** Stores encrypted passwords & user details
- **Security:** Hashing (e.g., bcrypt) & session/token management (e.g., JWT)

2. Final Demo Walkthrough

When presenting during your final demo:

1. **Show Login Page** – Clean UI with fields for username/email & password.
2. **User Flow:**
    - New user → Registration → Stored in DB.
    - Existing user → Login → Verification → Redirect to dashboard.
3. **Error Handling:** Show messages for invalid login, wrong password, or unregistered user.
4. **Security Features:** Mention password hashing, session expiry, CAPTCHA, or multi-factor authentication (if implemented).
5. **Demo Flow:**
    - Step 1: Register a new account
    - Step 2: Logout & re-login with correct credential
    - Step 3: Attempt login with wrong password → show error
    - Step 4: Access dashboard only after authentication

3. Project Report Section

In your report, explain:

- **Objective:** Why authentication is necessary.
- **Architecture:**
  - Frontend → sends credentials to backend API
  - Backend → validates credentials with DB
  - JWT/Session → returned to client
  - Middleware → checks authentication for protected routes
- **Technologies Used:** (HTML/CSS/JS, React/Angular/Vue, Node.js/Flask/Django, MongoDB/MySQL, JWT, bcrypt)
- **Implementation Details:** How you encrypted passwords, handled sessions, and secured APIs.
- **Testing & Validation:** How you tested valid/invalid login cases.

4. Screenshots / API Documentation

- **Screenshots to include:**
  - Login Page (empty form)
  - Login Page (with error)
  - Successful login (redirect to dashboard)
  - Registration form (if included)
  - API testing in Postman (showing /login and /register endpoints)
- **API Documentation Example:**
  - **POST /register**
    - Request: { "username": "test", "email": "test@mail.com", "password": "1234" }
    - Response: {"message": "User registered successfully"}

- **POST /login**
  - Request: { "email": "test@mail.com", "password": "1234" }
  - Response: { "token": "JWT_TOKEN", "message": "Login successful" }
- **GET /profile** (protected)
  - Request Header: Authorization: Bearer <JWT_TOKEN>
  - Response: { "username": "test", "email": "test@mail.com" }

5. Challenges & Solutions

- **Challenge 1:** Storing passwords securely
  - *Solution:* Used hashing with bcrypt/argon2 instead of plain text.
- **Challenge 2:** Protecting routes
  - *Solution:* Added middleware to validate JWT before accessing protected APIs.
- **Challenge 3:** Session expiration / token invalidation
  - *Solution:* Set JWT expiry (e.g., 1 hour) and implemented refresh tokens.
- **Challenge 4:** Deployment issues (CORS, environment variables)
  - *Solution:* Configured CORS headers properly & used .env for secrets.

6. GitHub README & Setup Guide

GITHUB LINK:

https://github.com/harinisaravindababu-hub

https://github.com/janasri255-create

https://github.com/meenumailforyou-sudo

https://github.com/Thibithra

Your README should include:

- **Project Title & Description**
- **Tech Stack**

- **Features (Login, Register, Authentication, Protected Routes)**
- **Setup Guide:**
- # Clone repo
- git clone <repo-link>
- cd project-folder
- # Install dependencies
- npm install   # or pip install -r requirements.txt
- # Setup environment variables
- # Example: create .env file with DB_URL and JWT_SECRET
- # Run project
- npm start   # or python app.py
- **API Endpoints Documentation** (login/register)
- **Screenshots/GIFs** showing working demo
- **Deployed Link** (Netlify/Vercel + Render/Heroku/Railway)

7. Final Submission (Repo + Deployed Link)

- **GitHub Repo:** Should contain
  - Source code (frontend + backend)
  - README with setup guide
  - Screenshots folder
  - API docs (markdown or Postman collection)

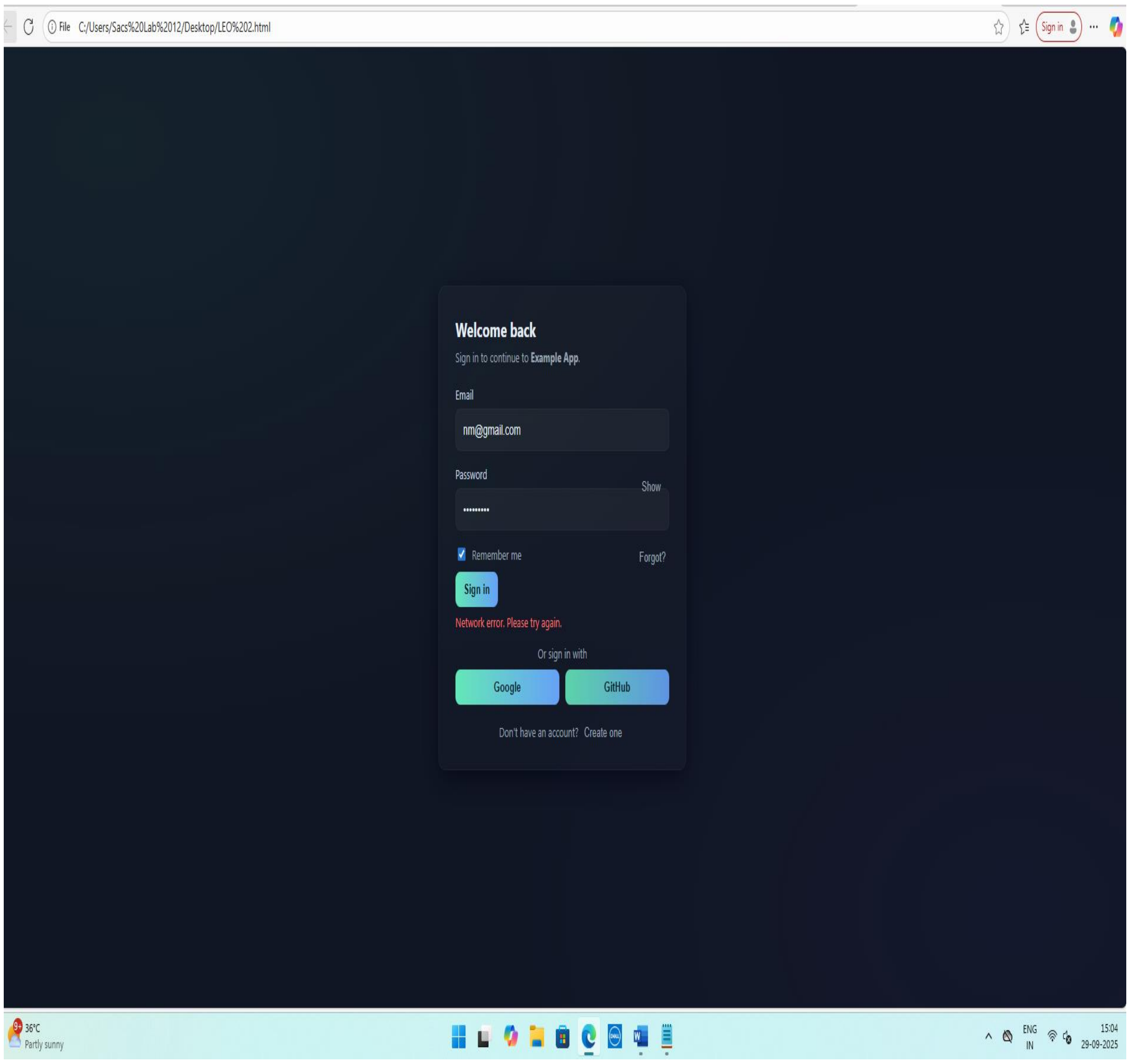
- **Deployed Link:**
  - Frontend (Netlify/Vercel)
  - Backend (Render/Railway/Heroku)
  - Provide demo credentials (test user) for reviewers

CODING:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Login — Example Authentication</title>
  <style>
    :root{
      --bg:#0f1724;
      --card:#0b1220;
      --accent:#6ee7b7;
      --muted:#9aa4b2;
      --danger:#ff6b6b;
      --glass: rgba(255,255,255,0.04);
      font-family: Inter, ui-sans-serif, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue",
Arial;
    }
    *{box-sizing:border-box}
    html,body{height:100%}
    body{
      margin:0;
      background: radial-gradient(1200px 600px at 10% 10%, rgba(110,231,183,0.06), transparent),
                  radial-gradient(900px 500px at 90% 90%, rgba(99,102,241,0.04), transparent),
                  var(--bg);
      color:#e6eef6;
      -webkit-font-smoothing:antialiased;
      -moz-osx-font-smoothing:grayscale;
      display:flex;
      align-items:center;
      justify-content:center;
      padding:24px;
    }

    .card{
      width:100%;
      max-width:420px;
      background:linear-gradient(180deg, rgba(255,255,255,0.02), rgba(255,255,255,0.01));
      border-radius:12px;
      padding:28px;
      box-shadow: 0 8px 30px rgba(2,6,23,0.7);
      border:1px solid rgba(255,255,255,0.03);
    }

    h1{margin:0 0 6px;font-size:20px}
    p.lead{margin:0 0 20px;color:var(--muted);font-size:13px}

    label{display:block;font-size:13px;margin-bottom:6px;color:#cbd6e3}
    .input{
      width:100%;
      padding:12px 12px;
      border-radius:8px;
      background:var(--glass);
      border:1px solid rgba(255,255,255,0.03);
      color:inherit;
      outline:none;
      font-size:14px;
    }
    .input:focus{box-shadow:0 0 0 4px rgba(110,231,183,0.06);border-color:rgba(110,231,183,0.18)}
    .row{display:flex;gap:10px}

    .field{margin-bottom:14px}
    .actions{display:flex;align-items:center;justify-content:space-between;margin-top:6px}

    .btn{
      appearance:none;
      border:0;
      padding:10px 14px;
      border-radius:8px;
      background:linear-gradient(90deg,var(--accent), #60a5fa);
      color:#042027;
      font-weight:600;
      cursor:pointer;
    }
    .btn:disabled{opacity:0.6;cursor:not-allowed}

    .link{background:none;border:0;color:var(--muted);cursor:pointer;font-size:13px}

    .small{font-size:13px;color:var(--muted)}

    .error{color:var(--danger);font-size:13px;margin-top:6px}
    .success{color:#8ef0a8;font-size:13px;margin-top:6px}

    .pw-wrap{position:relative}
    .pw-toggle{
      position:absolute;right:8px;top:8px;padding:6px;border-
radius:6px;border:0;background:none;color:var(--muted);cursor:pointer;font-size:13px
    }

    .footer{margin-top:18px;text-align:center;color:var(--muted);font-size:13px}

    @media (max-width:480px){.card{padding:18px}}
  </style>
</head>
<body>
  <main class="card" role="main" aria-labelledby="login-title">
    <h1 id="login-title">Welcome back</h1>
    <p class="lead">Sign in to continue to <strong>Example App</strong>.</p>

    <form id="loginForm" autocomplete="on" novalidate>
      <div class="field">
```

```
        <label for="email">Email</label>
        <input id="email" name="email" type="email" inputmode="email" class="input"
placeholder="you@example.com" required aria-required="true">
      </div>

      <div class="field pw-wrap">
        <label for="password">Password</label>
        <input id="password" name="password" type="password" class="input" placeholder="Enter your
password" required aria-required="true" minlength="6">
        <button type="button" id="togglePw" class="pw-toggle" aria-label="Show
password">Show</button>
      </div>

      <div class="field row" style="align-items:center;justify-content:space-between;margin-bottom:2px">
        <label style="display:flex;align-items:center;gap:8px">
          <input type="checkbox" id="remember" name="remember"> <span class="small">Remember
me</span>
        </label>
        <button type="button" class="link" id="forgotBtn">Forgot?</button>
      </div>

      <div class="field">
        <button class="btn" id="submitBtn" type="submit">Sign in</button>
        <div id="status" role="status" aria-live="polite"></div>
      </div>

      <div class="field" style="text-align:center;margin-top:8px">
        <div class="small">Or sign in with</div>
        <div class="row" style="margin-top:8px">
          <button type="button" class="btn" style="flex:1">Google</button>
          <button type="button" class="btn" style="flex:1;opacity:0.9">GitHub</button>
        </div>
      </div>
    </form>

    <div class="footer">Don't have an account? <button class="link" id="signupBtn">Create
one</button></div>
  </main>

  <script>
    // ======= Simple front-end auth demo ========
    // This file demonstrates an accessible, responsive login form with client-side validation
    // and a fetch() call to a backend endpoint (/api/auth/login). The backend should perform
    // real authentication (password hashing, rate-limits, issuing JWT or setting secure HttpOnly cookie).
    //
    // IMPORTANT security notes (backend responsibilities):
    // - Store passwords using Argon2 / bcrypt with a strong work factor.
    // - Use HTTPS only and set cookies with Secure; HttpOnly; SameSite=Strict (or Lax where
appropriate).
    // - Implement rate-limiting and account lockouts for repeated failed attempts.
    // - For SPAs, prefer setting refresh token in an HttpOnly cookie and returning short-lived access token.
    const form = document.getElementById('loginForm');
    const email = document.getElementById('email');
    const pw = document.getElementById('password');
    const toggle = document.getElementById('togglePw');
    const status = document.getElementById('status');
    const submitBtn = document.getElementById('submitBtn');

    // Toggle password visibility (accessible)
    toggle.addEventListener('click', () => {
      const isPw = pw.type === 'password';
      pw.type = isPw ? 'text' : 'password';
      toggle.textContent = isPw ? 'Hide' : 'Show';
      toggle.setAttribute('aria-pressed', String(isPw));
    });

    // Basic client-side validation helper
    function validate() {
      status.textContent = '';
      if (!email.value) { status.textContent = 'Please enter your email.'; status.className='error'; return false
}
      if (!pw.value) { status.textContent = 'Please enter your password.'; status.className='error'; return
false }
      if (pw.value.length < 6) { status.textContent = 'Password must be at least 6 characters.';
status.className='error'; return false }
      return true;
    }

    // On submit -> send credentials to backend
    form.addEventListener('submit', async (e) => {
      e.preventDefault();
      if (!validate()) return;

      submitBtn.disabled = true;
      const originalText = submitBtn.textContent;
      submitBtn.textContent = 'Signing in...';
      status.textContent = '';

      try {
        // Example payload. In production only send what is required.
        const payload = {
          email: email.value.trim().toLowerCase(),
          password: pw.value,
          remember: document.getElementById('remember').checked
        };

        // Use fetch to call your backend authentication endpoint
        const res = await fetch('/api/auth/login', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          credentials: 'include', // include cookies if your backend sets HttpOnly cookie
          body: JSON.stringify(payload)
        });

        const data = await res.json().catch(()=>({}));

        if (!res.ok) {
          // Backend should return useful error codes/messages like 401, 429, etc.
          status.className = 'error';
          status.textContent = data?.message || ('Login failed (' + res.status + ')');
          submitBtn.disabled = false;
          submitBtn.textContent = originalText;
          return;
        }

        // On success: backend may set HttpOnly refresh cookie and return a short-lived access token
        // For SPA: store access token in memory (not localStorage) and use it for API calls; refresh via
cookie.
        status.className = 'success';
        status.textContent = 'Signed in successfully — redirecting...';

        // Optionally, backend returns { redirect: '/dashboard' }
        const redirectTo = data?.redirect || '/dashboard';
        setTimeout(()=>{ window.location.href = redirectTo }, 700);

      } catch (err) {
        console.error(err);
        status.className = 'error';
        status.textContent = 'Network error. Please try again.';
        submitBtn.disabled = false;
        submitBtn.textContent = originalText;
      }
    });

    // Small handlers for demo buttons (replace with real flows)
    document.getElementById('forgotBtn').addEventListener('click', (;)=>{ alert('Open forgot-password
flow (backend)') });
    document.getElementById('signupBtn').addEventListener('click', ()=>{ window.location.href =
'/signup' });

    // Accessibility: focus first field on load
    window.addEventListener('load', ()=> email.focus());
  </script>
</body>
</html>
```

OUTPUT:


Welcome back

Sign in to continue to **Example App**.

Email

nm@gmail.com

Password

Show

••••••••••

Remember me

Forgot?

Sign in

Network error. Please try again.

Or sign in with

Google

GitHub

Don't have an account? Create one