

COLLEGE CODE : 9604

COLLEGE NAME : C.S.I INSTITUTE OF TECHNOLOGY

DEPARTMENT : AI & DS

**STUDENT NM ID:
332E259F6622205A5184DD2072774F0D**

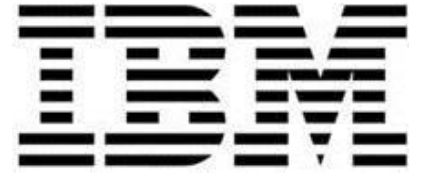
ROLL NO : 960423243006

DATE : 29/09/2025

SUBMITTED BY,

NAME : Ashick Joshua C

MOBILE NO : +91 94879 44311



Phase 4 - Enhancements & Deployment : User Registration with VAlidation

1. Additional Features:

Definition: New functionalities added to improve the capabilities of an application beyond the basic structure.

Examples:

- Show/Hide password toggle
- Confirm password match check
- Remember Me with local storage

a. Show/Hide Password Toggle

Add an eye icon to toggle password visibility:

HTML (in password input group):

Code:

HTML:

```
<i class="fas fa-eye toggle-password" id="togglePassword"></i>
```

Code:

CSS:

```
.toggle-password {
```

```
position: absolute;
right: 15px;
top: 42px;
cursor: pointer;
color: #555;
}
```

Code:

Js:

```
document.getElementById('togglePassword').addEventListener('click'
, function () {
    const passwordInput = document.getElementById('password');
    const type = passwordInput.getAttribute('type') === 'password' ?
'text' : 'password';
    passwordInput.setAttribute('type', type);
    this.classList.toggle('fa-eye-slash');
});
```

b. Confirm Password Matching Check

Add real-time password confirmation check:

Code:

JS (add this to your script):

```
document.getElementById('Confirm
Password').addEventListener('input', function () {
    const password = document.getElementById('password').value;
    const confirmPassword = this.value;

    if (confirmPassword !== password) {
        this.style.borderColor = '#f44336';
    }
});
```

```
    } else {  
      this.style.borderColor = '#4caf50';  
    }  
  });
```

c. Local Storage for “Remember Me”

```
window.addEventListener('load', () => {  
  const rememberedEmail =  
    localStorage.getItem('rememberedEmail');  
  if (rememberedEmail) {  
    document.getElementById('email').value = rememberedEmail;  
    document.getElementById('remember').checked = true;  
  }  
});
```

```
document.getElementById('loginForm').addEventListener('submit',  
function () {  
  const email = document.getElementById('email').value;  
  if (document.getElementById('remember').checked) {  
    localStorage.setItem('rememberedEmail', email);  
  } else {  
    localStorage.removeItem('rememberedEmail');  
  }  
});
```

2. UI/UX Improvements:

UI (User Interface):

The visual elements users interact with — like buttons, colors, fonts, icons, and layout.

UX (User Experience):

How users feel when interacting with your site — smooth navigation, clarity, speed, and responsiveness.

UI/UX Improvements might include:

- Cleaner layout
- Responsive design (mobile-friendly)
- Helpful animations and tooltips
- Password strength meter

Loading Spinner on Submit

HTML (inside .btn)

```
<span class="spinner" id="spinner" style="display:none;"></span>
```

CSS:

```
.spinner {  
  border: 4px solid rgba(255, 255, 255, 0.3);  
  border-left-color: #fff;  
  height: 16px;  
  width: 16px;  
  border-radius: 50%;  
  display: inline-block;  
  animation: spin 1s linear infinite;  
  margin-left: 10px;  
  vertical-align: middle;  
}  
  
@keyframes spin {  
  to { transform: rotate(360deg); }  
}
```

JS (in submit handler):

```
document.getElementById('spinner').style.display = 'inline-block';
```

```
setTimeout(() => {  
  document.getElementById('spinner').style.display = 'none';  
}, 2000);
```

3. Security Enhancements

Definition: Measures taken to protect data and prevent malicious access or behavior.

Common security practices:

- Input validation and sanitization
- Rate limiting (to avoid brute force attacks)
- Hiding passwords (eye icon toggle)
- Secure login using HTTPS
- Password strength requirements

a. Prevent XSS

- Sanitize all dynamic content before inserting into DOM (e.g., `innerHTML`, `toastMessage`).

```
function sanitize(input) {  
  const temp = document.createElement('div');  
  temp.textContent = input;  
  return temp.innerHTML;  
}  
toastMessage.innerHTML = sanitize(message);
```

b. Client-Side Validation (Enhanced)

Add stronger password validation:

```
if (password.length < 8 || ![A-Z]/.test(password) || ![0-9]/.test(password) || ![^A-Za-z0-9]/.test(password)) {  
  showToast('Password must be at least 8 characters, include an uppercase letter, number, and symbol.', 'error');
```

```
    return;  
}
```

c. Rate Limiting (Fake, Client-Side Simulation)

```
let loginAttempts = 0;  
document.getElementById('loginForm').addEventListener('submit',  
function (e) {  
    if (loginAttempts >= 5) {  
        showToast('Too many attempts. Please wait.', 'error');  
        return;  
    }  
  
    // inside wrong login block:  
    loginAttempts++;  
    setTimeout(() => {  
        loginAttempts = 0; // reset after timeout  
    }, 30000); // 30 seconds cooldown  
});
```

4. Performance Improvements

Definition: Optimizations to make your application load faster and run more efficiently.

Examples:

- Reducing animation intensity or frequency
- Defer loading of non-critical CSS (like icons)
- Minimizing DOM updates
- Reducing memory use from JavaScript animations

a. Defer FontAwesome

Replace:

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">
```

With:

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" media="all"
onload="this.media='all';">
```

b. Minimize CSS Animations

Your background animation runs **every 0.2s infinitely**:

```
animation: gradientBG 0.2s ease infinite;
```

Consider increasing to reduce CPU usage:

```
animation: gradientBG 10s ease infinite;
```

5. Testing of Enhancements








Definition: Techniques and tools used to ensure your application works correctly under various conditions.

Types of Testing:

- **Manual Testing:** Interacting with the app to test features.
- **Unit Testing:** Testing specific functions or components in isolation.
- **End-to-End Testing:** Automated test flows that simulate real user actions.

Manual Test Checklist

Feature	Tested?
Responsive layout	

Feature	Tested?
Password strength	
Confirm password match	
Validation errors	
Toasts appear properly	
"Remember Me" works	
Forgot password handler	
Invalid login behavior	

Unit Testing (suggested with Jest for API logic)

If you plan to connect to an API, write tests like:

```
test('Returns success for valid user', () => {  
  const result = validateUser("user@example.com", "Password123");  
  expect(result).toBeTruthy();  
});
```

6. API Integration (Next Step)

Definition: Improving how the front-end communicates with back-end servers using APIs (Application Programming Interfaces).

API Enhancements include:

- Sending login data via `fetch()` or `axios` to a server
- Using JWT for authentication tokens
- Handling server responses securely and clearly
- Validating credentials against a real database

Eventually, shift away from hardcoded users:

Example API POST request:

```
fetch('https://yourdomain.com/api/login', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ email, password })
})
.then(res => res.json())
.then(data => {
  if (data.success) {
    showToast("Login successful", 'success');
  } else {
    showToast("Invalid credentials", 'error');
  }
})
.catch(err => {
  showToast("Network error", 'error');
});
```

FINAL OUTPUT:



User Registration

LOGIN AUTHENTICATION

Full Name



Enter your full name

Email Address



Enter your email

Password



Enter your password

confirm password



Enter your confirm password

Phone number

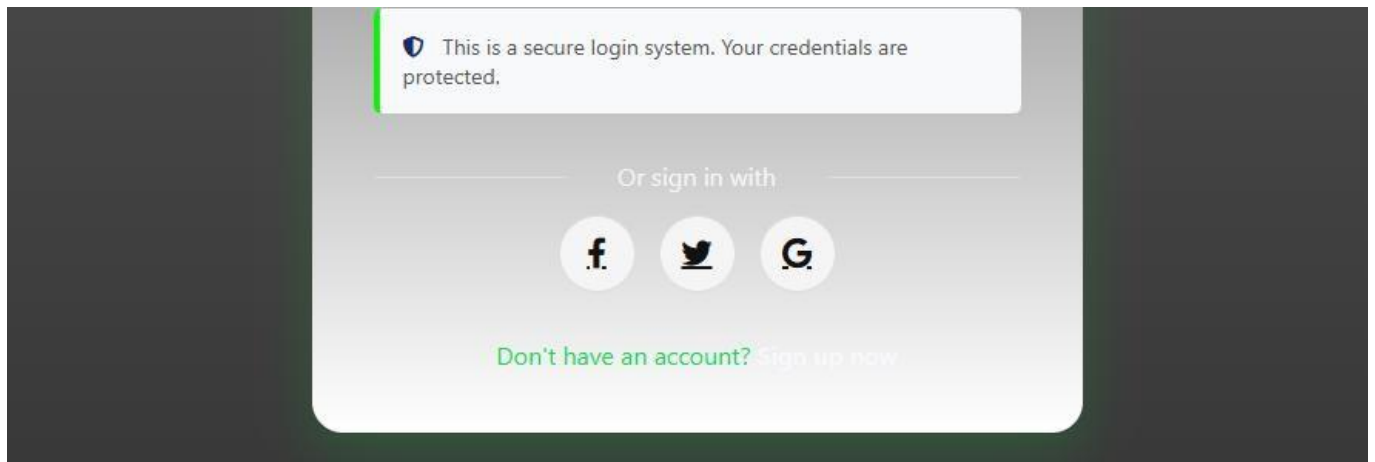


Enter your phone number

☐ Remember me

[Forgot password?](#)

Sign In



7. Deployment

Recommended Stack:

- **Frontend:** This HTML + JS page
- **Backend:** Node.js/Express or PHP (optional for handling auth)
- **Hosting:**
 - Static: GitHub Pages, Netlify, Vercel
 - Dynamic (API): Render, Railway, or Heroku

Deployment Steps (Static):

1. Push project to GitHub.
2. Use Netlify:
 - Connect GitHub repo
 - Netlify auto builds and deploys

OR

3. Use GitHub Pages:
 - Go to repo → Settings → Pages → Deploy from main or docs folder