Final Submission (Repo + Deployed Link) ,

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Interactive Form Validation — Demo</title>
  <style>
    :root{
      --bg:#f5f7fb;
      --card:#ffffff;
      --accent:#2563eb;
      --danger:#ef4444;
      --ok:#16a34a;
      --muted:#6b7280;
      font-family:Inter, system-ui, -apple-system, "Segoe UI", Roboto,
"Helvetica Neue", Arial;
    }

    body{
      margin:0;
      padding:40px;
      background:linear-gradient(180deg, #eef2ff 0%, var(--bg) 100%);
      display:flex;
      align-items:flex-start;
      justify-content:center;
      min-height:100vh;
    }

    .card{
      width:100%;
      max-width:760px;
      background:var(--card);
      border-radius:12px;
      box-shadow:0 8px 30px rgba(20,20,40,0.08);
      padding:28px;
    }
```

```
   h1{ margin:0 0 8px 0; font-size:20px; }
   p.lead{ margin:0 0 18px 0; color:var(--muted); }

   form { display:grid; gap:12px; }
   .row { display:flex; gap:12px; }
   .col { flex:1; min-width:0; }

   label{
     display:block;
     font-size:13px;
     margin-bottom:6px;
     color:#111827;
     font-weight:600;
   }

   input[type="text"], input[type="email"], input[type="password"],
input[type="tel"], input[type="number"], select, textarea {
     width:100%;
     padding:10px 12px;
     border-radius:8px;
     border:1px solid #e6e9ef;
     font-size:14px;
     outline:none;
     box-sizing:border-box;
     transition:box-shadow .12s, border-color .12s;
   }

   input:focus, select:focus, textarea:focus {
     border-color:var(--accent);
     box-shadow:0 6px 18px rgba(37,99,235,0.08);
   }

   .field-hint { font-size:12px; color:var(--muted); margin-top:6px; }

   .messages { font-size:13px; margin-top:6px; min-height:18px; }
   .messages .error { color:var(--danger); display:flex; gap:8px; align-
items:center; }
   .messages .success { color:var(--ok); display:flex; gap:8px; align-
items:center; }
```

```
 .meter {
     height:8px;
     width:100%;
     background:#eef2f2;
     border-radius:999px;
     overflow:hidden;
     margin-top:8px;
   }
   .meter > i {
     display:block;
     height:100%;
     width:0%;
     background:linear-gradient(90deg,#f97316,#16a34a);
     transition: width .22s ease;
   }

   .actions { display:flex; gap:10px; margin-top:8px; align-items:center; }
   button {
     background:var(--accent);
     color:white;
     border:0;
     padding:10px 14px;
     border-radius:10px;
     font-weight:600;
     cursor:pointer;
   }
   button:disabled { opacity:.56; cursor:not-allowed; }

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

   /* success outline */
   .valid { border-color: rgba(22,163,74,0.18); box-shadow:0 6px 18px
rgba(16,185,129,0.06); }
   .invalid { border-color: rgba(239,68,68,0.18); box-shadow:0 6px 18px
rgba(239,68,68,0.04); }
  </style>
</head>
<body>
```

```
<main class="card" role="main">
  <h1>Interactive Form Validation</h1>
  <p class="lead">Demo form that validates inputs in real time and on


submit. Try entering values to see feedback.</p>

  <form id="demoForm" novalidate>
   <div class="row">
    <div class="col">
     <label for="fullname">Full name</label>
     <input id="fullname" name="fullname" type="text"
autocomplete="name" required aria-describedby="fullnameMsg" />
     <div id="fullnameMsg" class="messages" aria-live="polite"></div>
    </div>

    <div class="col">
     <label for="username">Username</label>
     <input id="username" name="username" type="text" minlength="3"
maxlength="20" required aria-describedby="usernameMsg" />
     <div id="usernameMsg" class="messages" aria-live="polite"></div>
    </div>
   </div>

   <div class="row">
    <div class="col">
     <label for="email">Email</label>
     <input id="email" name="email" type="email" required aria-
describedby="emailMsg" />
     <div id="emailMsg" class="messages" aria-live="polite"></div>
    </div>

    <div class="col">
     <label for="phone">Phone (optional)</label>
     <input id="phone" name="phone" type="tel" placeholder="+91-
9876543210" aria-describedby="phoneMsg" />
     <div id="phoneMsg" class="messages" aria-live="polite"></div>
    </div>
   </div>
  <div class="row">
       <div class="col">
         <label for="password">Password</label>
         <input id="password" name="password" type="password"
minlength="8" required aria-describedby="pwdMsg" />
         <div id="pwdMsg" class="messages" aria-live="polite"></div>
         <div class="meter" aria-hidden="true"><i id="pwdMeter"></i></div>
         <div class="field-hint">At least 8 characters, include letters &
numbers. Stronger password gets higher meter.</div>
       </div>

       <div class="col">
         <label for="confirm">Confirm password</label>
         <input id="confirm" name="confirm" type="password" required aria-
describedby="confirmMsg" />
         <div id="confirmMsg" class="messages" aria-live="polite"></div>
       </div>
     </div>

     <div>
       <label for="age">Age</label>
       <input id="age" name="age" type="number" min="13" max="120"
required aria-describedby="ageMsg" />
       <div id="ageMsg" class="messages" aria-live="polite"></div>
     </div>

     <div>
       <label for="country">Country</label>
       <select id="country" name="country" required aria-
describedby="countryMsg">
         <option value="">Choose...</option>
         <option>India</option>
         <option>United States</option>
         <option>United Kingdom</option>
         <option>Australia</option>
         <option>Other</option>
       </select>
       <div id="countryMsg" class="messages" aria-live="polite"></div>
     </div>
  <div style="display:flex;align-items:center;gap:10px;">
       <input id="terms" name="terms" type="checkbox" />
       <label for="terms" class="small">I agree to the <a href="#"
onclick="return false">terms</a></label>
     </div>
     <div id="termsMsg" class="messages" aria-live="polite"></div>

     <div class="actions">
       <button id="submitBtn" type="submit">Create account</button>
       <div class="small" id="formStatus" aria-live="polite"></div>
     </div>
   </form>
  </main>

  <script>
   // ---- helper validators ----
   const qs = sel => document.querySelector(sel);
   const isFilled = v => v.trim().length > 0;
   const emailRx = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
   const phoneRx = /^\+?\d{7,15}$/; // simple international-ish
   const usernameRx = /^[a-zA-Z0-9._-]{3,20}$/;

   // elements
   const form = qs('#demoForm');
   const submitBtn = qs('#submitBtn');
   const formStatus = qs('#formStatus');

   const fields = {
     fullname: { el: qs('#fullname'), msg: qs('#fullnameMsg') },
     username: { el: qs('#username'), msg: qs('#usernameMsg') },
     email: { el: qs('#email'), msg: qs('#emailMsg') },
     phone: { el: qs('#phone'), msg: qs('#phoneMsg') },
     password: { el: qs('#password'), msg: qs('#pwdMsg'), meter:
qs('#pwdMeter') },
     confirm: { el: qs('#confirm'), msg: qs('#confirmMsg') },
     age: { el: qs('#age'), msg: qs('#ageMsg') },
     country: { el: qs('#country'), msg: qs('#countryMsg') },
     terms: { el: qs('#terms'), msg: qs('#termsMsg') }
   };
```

```
 // set field state helpers
   function setError(field, text) {
     const { el, msg } = fields[field];
     el.classList.remove('valid');
     el.classList.add('invalid');
     msg.innerHTML = `<span class="error">⚠ ${text}</span>`;
     el.setAttribute('aria-invalid', 'true');
   }

   function setSuccess(field, text = 'Looks good') {
     const { el, msg } = fields[field];
     el.classList.remove('invalid');
     el.classList.add('valid');
     msg.innerHTML = `<span class="success">✓ ${text}</span>`;
     el.setAttribute('aria-invalid', 'false');
   }

   function clearState(field) {
     const { el, msg } = fields[field];
     el.classList.remove('valid', 'invalid');
     msg.textContent = '';
     el.removeAttribute('aria-invalid');
   }

   // validation functions
   function validateFullname() {
     const v = fields.fullname.el.value;
     if (!isFilled(v)) { setError('fullname', 'Full name is required'); return false;
}
     if (v.length < 3) { setError('fullname', 'Enter at least 3 characters'); return
false; }
     setSuccess('fullname', 'Nice name');
     return true;
   }

   function validateUsername() {
     const v = fields.username.el.value;
     if (!isFilled(v)) { setError('username', 'Username required'); return false; }
     if (!usernameRx.test(v)) { setError('username', 'Use letters, numbers, . _ -
(3–20 chars)'); return false; }
```

```
 // Example additional check: disallow 'admin' or 'root'

 if (['admin','root','support'].includes(v.toLowerCase())) { setError('username',
'Choose a different username'); return false; }
      setSuccess('username', 'Username available');
      return true;
    }

    function validateEmail() {
      const v = fields.email.el.value;
      if (!isFilled(v)) { setError('email', 'Email is required'); return false; }
      if (!emailRx.test(v)) { setError('email', 'Enter a valid email'); return false;
}
      setSuccess('email', 'Valid email');
      return true;
    }

    function validatePhone() {
      const v = fields.phone.el.value.trim();
      if (!v) { clearState('phone'); return true; } // optional
      if (!phoneRx.test(v)) { setError('phone', 'Invalid phone number'); return
false; }
      setSuccess('phone', 'Phone looks good');
      return true;
    }

    function passwordStrength(pwd) {
      // simple scoring rules
      let score = 0;
      if (pwd.length >= 8) score += 1;
      if (/[A-Z]/.test(pwd)) score += 1;
      if (/[0-9]/.test(pwd)) score += 1;
      if (/[^A-Za-z0-9]/.test(pwd)) score += 1;
      // return 0..4
      return score;
    }

    function validatePassword() {
      const v = fields.password.el.value;
      const meter = fields.password.meter;
```

```
  if (!isFilled(v)) { setError('password', 'Password is required');
meter.style.width = '0%'; return false; }
      if (v.length < 8) { setError('password', 'Password must be 8+ characters');
meter.style.width = '15%'; return false; }

      const s = passwordStrength(v);
      meter.style.width = ((s / 4) * 100) + '%';
      if (s <= 1) { setError('password', 'Password is weak — add numbers or
symbols'); return false; }

      setSuccess('password', ['Weak','Okay','Good','Strong','Excellent'][s]);
      // also re-validate confirm if filled
      if (fields.confirm.el.value) validateConfirm();
      return true;
    }

    function validateConfirm() {
      const v = fields.confirm.el.value;
      if (!isFilled(v)) { setError('confirm', 'Please confirm password'); return
false; }
      if (v !== fields.password.el.value) { setError('confirm', 'Passwords do not
match'); return false; }
      setSuccess('confirm', 'Passwords match');
      return true;
    }

    function validateAge() {
      const v = fields.age.el.value;
      if (!isFilled(v)) { setError('age', 'Age is required'); return false; }
      const n = Number(v);
      if (Number.isNaN(n) || n < 13 || n > 120) { setError('age', 'Enter a valid age
(13–120)'); return false; }
      setSuccess('age', 'OK');
      return true;
    }

    function validateCountry() {
      const v = fields.country.el.value;
```

```
 if (!isFilled(v)) { setError('country', 'Please pick your country'); return false;
}
     setSuccess('country', 'Thanks');
     return true;
   }

   function validateTerms() {
     if (!fields.terms.el.checked) { fields.terms.msg.innerHTML = `<span
class="error">⚠ You must agree to the terms</span>`; return false; }
     fields.terms.msg.innerHTML = `<span class="success">✓
Accepted</span>`;
     return true;
   }

   // wire events - real-time validation
   fields.fullname.el.addEventListener('input', debounce(validateFullname,
250));
   fields.username.el.addEventListener('input', debounce(validateUsername,
250));
   fields.email.el.addEventListener('input', debounce(validateEmail, 250));
   fields.phone.el.addEventListener('input', debounce(validatePhone, 250));
   fields.password.el.addEventListener('input', debounce(validatePassword,
120));
   fields.confirm.el.addEventListener('input', debounce(validateConfirm,
120));
   fields.age.el.addEventListener('input', debounce(validateAge, 200));
   fields.country.el.addEventListener('change', validateCountry);
   fields.terms.el.addEventListener('change', validateTerms);

   // on blur also validate
   Object.keys(fields).forEach(key => {
     const f = fields[key];
     if (f && f.el) f.el.addEventListener('blur', () => {
       // call appropriate validator
       switch(key){
         case 'fullname': validateFullname(); break;
         case 'username': validateUsername(); break;
         case 'email': validateEmail(); break;
```

```
        case 'phone': validatePhone(); break;
        case 'password': validatePassword(); break;
        case 'confirm': validateConfirm(); break;
        case 'age': validateAge(); break;
        case 'country': validateCountry(); break;
      }
    });
  });

  // form submit handler
  form.addEventListener('submit', function(e){
    e.preventDefault();

    formStatus.textContent = '';
    submitBtn.disabled = true;

    // validate everything
    const ok =
      validateFullname() &
      validateUsername() &
      validateEmail() &
      validatePhone() &
      validatePassword() &
      validateConfirm() &
      validateAge() &
      validateCountry() &
      validateTerms();

    // note: bitwise & used above to ensure all functions run (so users see
every error)
    if (ok) {
      // simulate server processing
      formStatus.style.color = 'var(--ok)';
      formStatus.textContent = 'All good — submitting...';

      // In a real app: send via fetch() here. We'll just show success.
      setTimeout(() => {
        formStatus.textContent = 'Account created successfully ✅';
        form.reset();
```

```
 // clear visual state
         Object.keys(fields).forEach(k => clearState(k));
         fields.password.meter && (fields.password.meter.style.width = '0%');
         submitBtn.disabled = false;
       }, 700);
     } else {
       formStatus.style.color = 'var(--danger)';
       formStatus.textContent = 'Please fix the errors above';
       submitBtn.disabled = false;
     }
   });

   // small debounce utility
   function debounce(fn, wait=200){
     let t;
     return function(...args){ clearTimeout(t); t =
setTimeout(()=>fn.apply(this,args), wait); };
   }

   // initial accessibility: announce form instructions
   formStatus.textContent = 'Please fill the form. Errors will show as you
type.';
   formStatus.style.color = 'var(--muted)';
  </script>
</body>
</html>
```