```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Rejection Roulette</title>
  <meta name="description" content="Generate in-person rejection challenges and track your streak. No login." />
  <style>
    :root{
      --bg:#0b0f19; --card:#121a2a; --muted:#93a4c7; --text:#e8eefc;
      --accent:#7c5cff; --accent2:#36d399; --danger:#ff5c7a; --border:#21304d;
      --shadow: 0 10px 35px rgba(0,0,0,.35);
      --radius:18px;
      --mono: ui-monospace, SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace;
      --sans: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Helvetica, Arial, "Apple Color Emoji","Segoe UI Emoji";
    }
    *{box-sizing:border-box}
    body{
      margin:0; font-family:var(--sans); color:var(--text); background:radial-gradient(1200px 600px at 20% -10%, rgba(124,92,255,.35), transparent 60%),
      radial-gradient(900px 500px at 100% 0%, rgba(54,211,153,.25), transparent 55%),
      var(--bg);
      min-height:100vh;
    }
    a{color:inherit}
    .wrap{max-width:980px; margin:0 auto; padding:28px 16px 60px}
    header{
      display:flex; align-items:flex-start; justify-content:space-between; gap:16px;
      margin-bottom:18px;
    }
    .brand{
      display:flex; flex-direction:column; gap:6px;
    }
    .title{
      display:flex; align-items:center; gap:10px;
      font-size:22px; font-weight:800; letter-spacing:.2px;
    }
    .pill{
      font-family:var(--mono);
      font-size:12px; color:rgba(232,238,252,.9);
      border:1px solid rgba(255,255,255,.12);
```

```
      padding:4px 8px; border-radius:999px;
      background:rgba(18,26,42,.55);
    }
    .sub{color:var(--muted); font-size:14px; line-height:1.35; max-width:62ch}
    .grid{display:grid; grid-template-columns:1.2fr .8fr; gap:14px; margin-
top:14px}
    @media (max-width: 900px){ .grid{grid-template-columns:1fr} header{flex-
direction:column} }
    .card{
      background:linear-gradient(180deg, rgba(18,26,42,.92),
rgba(18,26,42,.78));
      border:1px solid rgba(255,255,255,.10);
      border-radius:var(--radius);
      box-shadow:var(--shadow);
      padding:16px;
    }
    .card h2{margin:0 0 10px; font-size:15px; letter-spacing:.2px}
    .row{display:flex; gap:10px; align-items:center; flex-wrap:wrap}
    .row > *{flex: 1 1 auto}
    label{display:block; font-size:12px; color:var(--muted); margin-bottom:6px}
    select, input[type="range"], input[type="text"], textarea{
      width:100%;
      background:rgba(11,15,25,.65);
      border:1px solid rgba(255,255,255,.12);
      color:var(--text);
      border-radius:12px;
      padding:10px 12px;
      outline:none;
    }
    textarea{min-height:92px; resize:vertical}
    .btn{
      appearance:none; border:0; cursor:pointer;
      padding:10px 12px;
      border-radius:12px;
      font-weight:700;
      color:var(--text);
      background:rgba(255,255,255,.08);
      border:1px solid rgba(255,255,255,.12);
      transition:transform .05s ease, background .15s ease;
      user-select:none;
    }
    .btn:hover{background:rgba(255,255,255,.12)}
    .btn:active{transform:translateY(1px)}
    .btn.primary{background:linear-gradient(90deg, rgba(124,92,255,.95),
rgba(124,92,255,.7)); border-color:rgba(124,92,255,.6)}
    .btn.good{background:linear-gradient(90deg, rgba(54,211,153,.92),
rgba(54,211,153,.65)); border-color:rgba(54,211,153,.55)}
```

```
    .btn.bad{background:linear-gradient(90deg, rgba(255,92,122,.92),
rgba(255,92,122,.65)); border-color:rgba(255,92,122,.55)}
    .btn.small{padding:8px 10px; font-size:13px}
    .challenge{
      border:1px dashed rgba(255,255,255,.18);
      border-radius:16px;
      padding:14px;
      background:rgba(11,15,25,.45);
    }
    .challenge .headline{display:flex; align-items:center; justify-content:space-
between; gap:10px}
    .challenge .text{font-size:18px; font-weight:800; margin:8px 0 6px; line-
height:1.25}
    .challenge .meta{display:flex; gap:8px; flex-wrap:wrap; color:var(--muted);
font-size:12px}
    .tag{
      border:1px solid rgba(255,255,255,.12);
      background:rgba(18,26,42,.55);
      padding:3px 8px;
      border-radius:999px;
      font-family:var(--mono);
    }
    .statgrid{display:grid; grid-template-columns:repeat(3,1fr); gap:10px}
    @media (max-width: 520px){ .statgrid{grid-template-columns:1fr} }
    .stat{
      padding:12px;
      border-radius:14px;
      background:rgba(11,15,25,.45);
      border:1px solid rgba(255,255,255,.10);
    }
    .stat .k{font-size:11px; color:var(--muted)}
    .stat .v{font-size:20px; font-weight:900; margin-top:4px}
    table{width:100%; border-collapse:collapse; margin-top:8px; font-size:13px}
    th,td{padding:10px 8px; border-bottom:1px solid rgba(255,255,255,.08);
vertical-align:top}
    th{color:var(--muted); font-weight:700; text-align:left; font-size:12px}
    .muted{color:var(--muted)}
    .toast{
      position:fixed; left:50%; bottom:18px; transform:translateX(-50%);
      background:rgba(18,26,42,.92);
      border:1px solid rgba(255,255,255,.12);
      padding:10px 12px; border-radius:999px;
      box-shadow:var(--shadow);
      display:none;
      font-size:13px;
    }
    .footer{
```

```
      margin-top:14px;
      color:var(--muted);
      font-size:12px;
      display:flex;
      justify-content:space-between;
      gap:10px;
      flex-wrap:wrap;
    }
    .kbd{
      font-family:var(--mono);
      border:1px solid rgba(255,255,255,.14);
      background:rgba(11,15,25,.55);
      padding:2px 6px;
      border-radius:8px;
    }
  </style>
</head>
<body>
  <div class="wrap">
    <header>
      <div class="brand">
        <div class="title">🎯 Rejection Roulette <span class="pill">no login • shareable • exportable</span></div>
        <div class="sub">
          Generate an in-person “ask” designed to get you out of your comfort zone. Log what happened, build a streak, export proof.
        </div>
      </div>
      <div class="row" style="justify-content:flex-end; min-width:260px">
        <button class="btn small" id="resetBtn" title="Clears logs and streak">Reset</button>
        <button class="btn small" id="exportBtn" title="Download CSV of your logs">Export CSV</button>
        <button class="btn small" id="shareAppBtn" title="Share this app">Share App</button>
      </div>
    </header>

    <div class="grid">
      <div class="card">
        <h2>New challenge</h2>
        <div class="row">
          <div style="min-width:220px">
            <label for="category">Category</label>
            <select id="category">
              <option value="any">Any</option>
              <option value="food">Food & café</option>
```

```
            <option value="social">Social & compliments</option>
            <option value="help">Asking for help</option>
            <option value="negotiation">Negotiation</option>
            <option value="weird">Wholesome-weird</option>
            <option value="career">Career & networking</option>
          </select>
        </div>
        <div style="min-width:220px">
          <label for="difficulty">Difficulty: <span id="diffLabel" class="pill">3</
span></label>
          <input id="difficulty" type="range" min="1" max="5" value="3" />
        </div>
        <div style="min-width:220px">
          <label for="timebox">Timebox (minutes)</label>
          <select id="timebox">
            <option value="0">No timer</option>
            <option value="5">5</option>
            <option value="10">10</option>
            <option value="15" selected>15</option>
            <option value="30">30</option>
          </select>
        </div>
      </div>

      <div class="challenge" style="margin-top:12px">
        <div class="headline">
          <div class="meta">
            <span class="tag" id="tagCat">any</span>
            <span class="tag" id="tagDiff">difficulty: 3</span>
            <span class="tag" id="tagRule">in person</span>
          </div>
          <div class="row" style="flex:0 0 auto">
            <button class="btn small" id="copyLinkBtn">Copy link</button>
            <button class="btn primary" id="newBtn">Generate</button>
          </div>
        </div>
        <div class="text" id="challengeText">Click “Generate” to get your first
challenge.</div>
        <div class="muted" id="timerText" style="margin-top:6px"></div>
      </div>

      <div style="margin-top:12px">
        <label for="note">Quick note (optional)</label>
        <textarea id="note" placeholder="What did you say? What did they say?
Any funny detail?"></textarea>
        <div class="row" style="margin-top:10px">
          <button class="btn good" id="acceptedBtn">Accepted 😳</button>
```

```
        <button class="btn bad" id="rejectedBtn">Rejected ✅</button>
      </div>
      <div class="footer">
        <div>Tip: aim for “polite, specific, and uncomfortable.”</div>
        <div>Shortcut: press <span class="kbd">G</span> to generate.</div>
      </div>
    </div>
  </div>

  <div class="card">
    <h2>Your stats</h2>
    <div class="statgrid">
      <div class="stat">
        <div class="k">Current streak (days)</div>
        <div class="v" id="streak">0</div>
      </div>
      <div class="stat">
        <div class="k">Total logs</div>
        <div class="v" id="total">0</div>
      </div>
      <div class="stat">
        <div class="k">Rejection rate</div>
        <div class="v" id="rate">0%</div>
      </div>
    </div>

    <h2 style="margin-top:14px">Recent logs</h2>
    <div class="muted" style="font-size:12px; margin-top:-6px">
      Stored in your browser (localStorage). No account required.
    </div>

    <div style="overflow:auto; margin-top:6px">
      <table>
        <thead>
          <tr>
            <th style="min-width:140px">Time</th>
            <th>Challenge</th>
            <th style="min-width:110px">Result</th>
          </tr>
        </thead>
        <tbody id="logBody">
          <tr><td colspan="3" class="muted">No logs yet. Go get rejected
(respectfully).</td></tr>
        </tbody>
      </table>
    </div>
  </div>
```

```
    </div>

    <div class="toast" id="toast"></div>
  </div>

<script>
(() => {
  const $ = (id) => document.getElementById(id);

  const difficulty = $("difficulty");
  const diffLabel = $("diffLabel");
  const category = $("category");
  const timebox = $("timebox");

  const tagCat = $("tagCat");
  const tagDiff = $("tagDiff");
  const challengeText = $("challengeText");
  const timerText = $("timerText");

  const note = $("note");

  const newBtn = $("newBtn");
  const acceptedBtn = $("acceptedBtn");
  const rejectedBtn = $("rejectedBtn");
  const exportBtn = $("exportBtn");
  const resetBtn = $("resetBtn");
  const copyLinkBtn = $("copyLinkBtn");
  const shareAppBtn = $("shareAppBtn");

  const streakEl = $("streak");
  const totalEl = $("total");
  const rateEl = $("rate");
  const logBody = $("logBody");
  const toast = $("toast");

  const STORAGE_KEY = "rr_logs_v1";
  const STREAK_KEY = "rr_streak_v1";
  const LAST_LOG_DAY_KEY = "rr_last_log_day_v1";

  let currentChallenge = null;
  let timer = null;
  let timerEndsAt = null;

  const challenges = [
    // Food & café
    {cat:"food", diff:1, text:"Ask a barista if they can recommend something *not*
on the menu."},
```

```
    {cat:"food", diff:2, text:"Ask for a sample of a pastry before buying (politely accept no)."},
    {cat:"food", diff:3, text:"Ask if they can make your drink ‘half sweet’ and explain you’re testing taste sensitivity."},
    {cat:"food", diff:4, text:"Ask the cashier for a ‘surprise’ under $5 and let them choose."},
    {cat:"food", diff:5, text:"Ask a restaurant if they can do a custom off-menu swap (e.g., change a side) and take the no gracefully."},

    // Social & compliments
    {cat:"social", diff:1, text:"Give a sincere compliment to a stranger (then walk away, no follow-up)."},
    {cat:"social", diff:2, text:"Ask someone where they got an item (shoes/bag/jacket) because you like it."},
    {cat:"social", diff:3, text:"Ask a stranger to rate your fit from 1–10. Thank them either way."},
    {cat:"social", diff:4, text:"Ask someone for a quick photo of you (then offer to take one for them too)."},
    {cat:"social", diff:5, text:"Ask a group if you can join for 2 minutes just to practice small talk, then leave."},

    // Asking for help
    {cat:"help", diff:1, text:"Ask a stranger for directions to a nearby place you already know."},
    {cat:"help", diff:2, text:"Ask someone to recommend a nearby spot they genuinely like."},
    {cat:"help", diff:3, text:"Ask if you can borrow a pen for 30 seconds."},
    {cat:"help", diff:4, text:"Ask a store employee to help you pick between two options *and* explain why."},
    {cat:"help", diff:5, text:"Ask someone to practice a 30-second intro with you (‘I’m working on confidence—can I try?’)."},

    // Negotiation
    {cat:"negotiation", diff:1, text:"Ask if there’s a student discount (even if you’re not sure)."},
    {cat:"negotiation", diff:2, text:"Ask if they can waive an add-on fee ‘just this once’ (accept no)."},
    {cat:"negotiation", diff:3, text:"Ask for a small upgrade (bigger size, better seat, etc.) politely."},
    {cat:"negotiation", diff:4, text:"Ask for a $1–$2 discount on a small purchase ‘for fun’ (smile, accept no)."},
    {cat:"negotiation", diff:5, text:"Ask for a free item that’s clearly optional (e.g., extra topping). No arguing."},

    // Wholesome-weird
    {cat:"weird", diff:1, text:"Ask someone if they think pigeons have a leader."},
    {cat:"weird", diff:2, text:"Ask a cashier to pick: ‘Is today more of a 7 or a
```

```
9?’"},
    {cat:"weird", diff:3, text:"Ask someone for a 10-second pep talk (tell them you’ll time it)."},
    {cat:"weird", diff:4, text:"Ask a stranger for a ‘life hack’ that saves time."},
    {cat:"weird", diff:5, text:"Ask if you can record a 5-second video of them saying ‘you’ve got this’ (consent matters)."},

    // Career & networking
    {cat:"career", diff:1, text:"Ask someone what they do for work (then ask what they *actually* enjoy about it)."},
    {cat:"career", diff:2, text:"Ask for one piece of advice for someone early in their career."},
    {cat:"career", diff:3, text:"Ask if they’d be open to a 10-minute coffee chat sometime (accept no warmly)."},
    {cat:"career", diff:4, text:"Ask if they know one person you should meet and why."},
    {cat:"career", diff:5, text:"Ask a professional for feedback on your 15-second self-intro—on the spot."},
  ];

  function nowISO() { return new Date().toISOString(); }
  function dayKey(d = new Date()) { return d.toISOString().slice(0,10); }

  function loadLogs(){
    try { return JSON.parse(localStorage.getItem(STORAGE_KEY) || "[]"); }
    catch { return []; }
  }
  function saveLogs(logs){ localStorage.setItem(STORAGE_KEY, JSON.stringify(logs)); }

  function loadStreak(){
    try { return JSON.parse(localStorage.getItem(STREAK_KEY) || "0"); }
    catch { return 0; }
  }
  function saveStreak(n){ localStorage.setItem(STREAK_KEY, JSON.stringify(n)); }

  function toastMsg(msg){
    toast.textContent = msg;
    toast.style.display = "block";
    clearTimeout(toast._t);
    toast._t = setTimeout(()=> toast.style.display="none", 1800);
  }

  function pickChallenge(cat, diff){
    let pool = challenges.filter(c => (cat === "any" ? true : c.cat === cat) && c.diff === diff);
```

```
  if (!pool.length){
    pool = challenges.filter(c => (cat === "any" ? true : c.cat === cat));
  }
  const chosen = pool[Math.floor(Math.random() * pool.length)];
  return chosen;
}

function encodeChallenge(ch){
  const payload = {cat:ch.cat, diff:ch.diff, text:ch.text};
  const b64 =
btoa(unescape(encodeURIComponent(JSON.stringify(payload))));
  return b64.replaceAll("=", "");
}

function decodeChallenge(str){
  try{
    const pad = str + "===".slice((str.length % 4) || 4);
    const json = decodeURIComponent(escape(atob(pad)));
    const obj = JSON.parse(json);
    if (!obj || !obj.text) return null;
    return obj;
  } catch { return null; }
}

function updateTags(){
  tagCat.textContent = category.value;
  diffLabel.textContent = difficulty.value;
  tagDiff.textContent = "difficulty: " + difficulty.value;
}

function setChallenge(ch){
  currentChallenge = ch;
  challengeText.textContent = ch.text;
  const share = new URL(location.href);
  share.searchParams.set("c", encodeChallenge(ch));
  history.replaceState(null, "", share.toString());
}

function startTimer(){
  stopTimer();
  const minutes = parseInt(timebox.value, 10);
  if (!minutes){ timerText.textContent = ""; return; }
  timerEndsAt = Date.now() + minutes * 60 * 1000;
  const tick = () => {
    const ms = timerEndsAt - Date.now();
    if (ms <= 0){
      timerText.textContent = "⏱️ Timebox complete. Go do it now.";
```

```
      stopTimer();
      return;
    }
    const m = Math.floor(ms / 60000);
    const s = Math.floor((ms % 60000) / 1000);
    timerText.textContent = `⏱️ Time left: ${m}:${String(s).padStart(2,"0")}`;
  };
  tick();
  timer = setInterval(tick, 250);
}
function stopTimer(){
  if (timer) clearInterval(timer);
  timer = null;
  timerEndsAt = null;
}

function updateStats(){
  const logs = loadLogs();
  totalEl.textContent = logs.length;

  const rejected = logs.filter(x => x.result === "rejected").length;
  const rate = logs.length ? Math.round((rejected / logs.length) * 100) : 0;
  rateEl.textContent = rate + "%";

  streakEl.textContent = loadStreak();

  renderLogs(logs);
}

function renderLogs(logs){
  if (!logs.length){
    logBody.innerHTML = `<tr><td colspan="3" class="muted">No logs yet. Go get rejected (respectfully).</td></tr>`;
    return;
  }
  const rows = logs.slice().reverse().slice(0, 12).map(l => {
    const dt = new Date(l.time);
    const when = dt.toLocaleString([], {year:"numeric", month:"short", day:"2-digit", hour:"2-digit", minute:"2-digit"});
    const res = l.result === "rejected" ? "Rejected ✅" : "Accepted 😳";
    const extra = l.note ? `<div class="muted" style="margin-top:6px">${escapeHtml(l.note)}</div>` : "";
    return `
      <tr>
        <td>${escapeHtml(when)}</td>
        <td>
          <div style="font-weight:700">${escapeHtml(l.challenge)}</div>
```

```
          <div class="muted" style="margin-top:4px">
            <span class="tag">${escapeHtml(l.cat)}</span>
            <span class="tag">d${escapeHtml(String(l.diff))}</span>
          </div>
          ${extra}
        </td>
        <td>${res}</td>
      </tr>
    `;
  }).join("");
  logBody.innerHTML = rows;
}

function escapeHtml(s){
  return
String(s).replaceAll("&","&amp;").replaceAll("<","&lt;").replaceAll(">","&gt;")
    .replaceAll('"',"&quot;").replaceAll("'","&#039;");
}

function updateStreakOnLog(){
  const today = dayKey();
  const last = localStorage.getItem(LAST_LOG_DAY_KEY);
  let streak = loadStreak();

  if (!last){
    streak = 1;
  } else {
    const lastDate = new Date(last + "T00:00:00");
    const todayDate = new Date(today + "T00:00:00");
    const diffDays = Math.round((todayDate - lastDate) / (24*60*60*1000));

    if (diffDays === 0){
      // same day: no streak change
    } else if (diffDays === 1){
      streak += 1;
    } else {
      streak = 1; // broke streak
    }
  }
  localStorage.setItem(LAST_LOG_DAY_KEY, today);
  saveStreak(streak);
}

function logResult(result){
  if (!currentChallenge){
    toastMsg("Generate a challenge first.");
    return;
```

```
    }
    const logs = loadLogs();
    logs.push({
      time: nowISO(),
      result,
      cat: currentChallenge.cat,
      diff: currentChallenge.diff,
      challenge: currentChallenge.text,
      note: note.value.trim()
    });
    saveLogs(logs);
    updateStreakOnLog();
    note.value = "";
    toastMsg(result === "rejected" ? "Logged: Rejected ✅" : "Logged:
Accepted 😳");
    updateStats();
  }

  function exportCSV(){
    const logs = loadLogs();
    if (!logs.length){ toastMsg("Nothing to export yet."); return; }
    const header = ["time_iso","result","category","difficulty","challenge","note"];
    const rows = logs.map(l => [
      l.time, l.result, l.cat, String(l.diff), l.challenge.replaceAll('"','""'),
(l.note||"").replaceAll('"','""')
    ]);
    const csv = [header.join(","), ...rows.map(r => r.map(v => `"$
{v}"`).join(","))].join("\n");
    const blob = new Blob([csv], {type:"text/csv;charset=utf-8"});
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url;
    a.download = "rejection-roulette-logs.csv";
    document.body.appendChild(a);
    a.click();
    a.remove();
    URL.revokeObjectURL(url);
  }

  async function copyCurrentLink(){
    const url = location.href;
    try{
      await navigator.clipboard.writeText(url);
      toastMsg("Link copied.");
    } catch {
      prompt("Copy this link:", url);
    }
```

```
  }

  async function shareApp(){
    const url = location.origin + location.pathname;
    if (navigator.share){
      try{ await navigator.share({title:"Rejection Roulette", text:"Try this challenge
generator + tracker:", url}); }
      catch {}
    } else {
      try{
        await navigator.clipboard.writeText(url);
        toastMsg("App link copied.");
      } catch {
        prompt("Copy this app link:", url);
      }
    }
  }

  function resetAll(){
    if (!confirm("Reset logs and streak?")) return;
    localStorage.removeItem(STORAGE_KEY);
    localStorage.removeItem(STREAK_KEY);
    localStorage.removeItem(LAST_LOG_DAY_KEY);
    toastMsg("Reset complete.");
    updateStats();
  }

  function generate(){
    updateTags();
    const ch = pickChallenge(category.value, parseInt(difficulty.value,10));
    setChallenge(ch);
    startTimer();
  }

  // Wire up UI
  difficulty.addEventListener("input", updateTags);
  category.addEventListener("change", updateTags);
  timebox.addEventListener("change", () => startTimer());

  newBtn.addEventListener("click", generate);
  acceptedBtn.addEventListener("click", () => logResult("accepted"));
  rejectedBtn.addEventListener("click", () => logResult("rejected"));
  exportBtn.addEventListener("click", exportCSV);
  resetBtn.addEventListener("click", resetAll);
  copyLinkBtn.addEventListener("click", copyCurrentLink);
  shareAppBtn.addEventListener("click", shareApp);
```

```
  document.addEventListener("keydown", (e) => {
    if (e.key.toLowerCase() === "g") generate();
  });

  // Load from share link if present
  const params = new URLSearchParams(location.search);
  const c = params.get("c");
  if (c){
    const decoded = decodeChallenge(c);
    if (decoded){
      currentChallenge = decoded;
      challengeText.textContent = decoded.text;
      category.value =
["food","social","help","negotiation","weird","career"].includes(decoded.cat) ?
decoded.cat : "any";
      difficulty.value = String(Math.min(5, Math.max(1, parseInt(decoded.diff ||
3,10) || 3)));
    }
  }
  updateTags();
  updateStats();
})();
</script>
</body>
</html>
```