anime-character-lab/

```
npm install
npm run dev
anime-character-lab/
  package.json
  vite.config.js
  index.html
  src/
    main.jsx
    app.jsx
    styles.css
    data/
      clothingLibrary.js
      exampleSeed.js
      generator.js
      evolution.js
    storage/
      storage.js
      idb.js
    utils/
      ids.js
      textExport.js
      diff.js
    pages/
      GeneratePage.jsx
      LibraryPage.jsx
      CharacterPage.jsx
      SettingsPage.jsx
    components/
      Nav.jsx
      Toast.jsx
      CharacterCard.jsx
      FiltersBar.jsx
      EvolveModal.jsx
```

```
{
  "name": "anime-character-lab",
  "private": true,
  "version": "1.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^18.3.1",
    "react-dom": "^18.3.1"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.3.1",
    "vite": "^5.4.8"
  }
}
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";

export default defineConfig({
  plugins: [react()],
});
<!doctype html>
<html lang="nl">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Anime Character Lab</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
```

```
</html>
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./app.jsx";
import "./styles.css";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
:root {
  --bg: #0b0d12;
  --panel: #121622;
  --panel2: #0f1320;
  --text: #e8ecff;
  --muted: #aab2d6;
  --border: rgba(255,255,255,0.10);
  --accent: #73ffe6;
  --danger: #ff6b8a;
  --ok: #8dff9c;
}

* { box-sizing: border-box; }
body {
  margin: 0;
  font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Arial;
  background: radial-gradient(1200px 600px at 20% 0%, rgba(115,255,230,0.08), transparent
50%),
            radial-gradient(1000px 600px at 80% 10%, rgba(160,140,255,0.08), transparent
55%),
            var(--bg);
  color: var(--text);
}
a { color: inherit; text-decoration: none; }
button, input, select, textarea { font: inherit; }
```

```
.container { max-width: 1100px; margin: 0 auto; padding: 18px; }
.row { display: flex; gap: 12px; flex-wrap: wrap; }
.grid { display: grid; gap: 12px; }
.grid-2 { grid-template-columns: repeat(2, minmax(0, 1fr)); }
@media (max-width: 860px) { .grid-2 { grid-template-columns: 1fr; } }

.card {
  background: linear-gradient(180deg, rgba(255,255,255,0.04), rgba(255,255,255,0.02));
  border: 1px solid var(--border);
  border-radius: 16px;
  padding: 14px;
  backdrop-filter: blur(8px);
}
.card h2, .card h3 { margin: 0 0 10px; }
.subtle { color: var(--muted); font-size: 0.92rem; }
label { display: grid; gap: 6px; min-width: 220px; }
input, select, textarea {
  background: var(--panel2);
  border: 1px solid var(--border);
  color: var(--text);
  padding: 10px 10px;
  border-radius: 12px;
}
textarea { min-height: 110px; resize: vertical; }
.btn {
  padding: 10px 12px;
  border-radius: 12px;
  border: 1px solid var(--border);
  background: rgba(255,255,255,0.06);
  color: var(--text);
  cursor: pointer;
}
.btn:hover { background: rgba(255,255,255,0.10); }
.btn.primary { border-color: rgba(115,255,230,0.35); box-shadow: 0 0 0 1px
rgba(115,255,230,0.15) inset; }
.btn.danger { border-color: rgba(255,107,138,0.35); }
```

```
.pill {
  display: inline-flex;
  gap: 8px;
  align-items: center;
  padding: 6px 10px;
  border-radius: 999px;
  border: 1px solid var(--border);
  background: rgba(255,255,255,0.04);
  color: var(--muted);
  font-size: 0.85rem;
}
hr { border: none; border-top: 1px solid var(--border); margin: 12px 0; }

pre {
  background: rgba(0,0,0,0.25);
  border: 1px solid var(--border);
  padding: 12px;
  border-radius: 14px;
  overflow: auto;
  white-space: pre-wrap;
  line-height: 1.35;
}

.topbar {
  display: flex;
  align-items: center;
  justify-content: space-between;
  gap: 12px;
  margin-bottom: 16px;
}
.brand {
  display: flex;
  gap: 10px;
  align-items: center;
  font-weight: 700;
}
```

```
.brand .dot {
  width: 12px; height: 12px; border-radius: 50%;
  background: var(--accent);
  box-shadow: 0 0 18px rgba(115,255,230,0.6);
}
.nav {
  display: flex;
  gap: 10px;
  flex-wrap: wrap;
}
.nav a {
  padding: 8px 10px;
  border-radius: 10px;
  border: 1px solid transparent;
  color: var(--muted);
}
.nav a.active {
  border-color: rgba(115,255,230,0.35);
  color: var(--text);
  background: rgba(115,255,230,0.06);
}

.toast-wrap {
  position: fixed;
  right: 14px;
  bottom: 14px;
  display: grid;
  gap: 10px;
  z-index: 9999;
}
.toast {
  max-width: 360px;
  padding: 10px 12px;
  border-radius: 14px;
  border: 1px solid var(--border);
  background: rgba(18,22,34,0.92);
```

```
}
.toast strong { display: block; margin-bottom: 2px; }
import React, { useEffect, useMemo, useState } from "react";
import Nav from "./components/Nav.jsx";
import ToastHost from "./components/Toast.jsx";
import GeneratePage from "./pages/GeneratePage.jsx";
import LibraryPage from "./pages/LibraryPage.jsx";
import CharacterPage from "./pages/CharacterPage.jsx";
import SettingsPage from "./pages/SettingsPage.jsx";
import { storage } from "./storage/storage.js";
import { ensureExampleSeed } from "./data/exampleSeed.js";

function parseHash() {
  const h = window.location.hash || "#/generate";
  // routes:
  // #/generate
  // #/library
  // #/character/<id>
  // #/settings
  const parts = h.replace("#/", "").split("/");
  return { route: parts[0] || "generate", id: parts[1] || null };
}

export default function App() {
  const [loc, setLoc] = useState(parseHash());
  const [toast, setToast] = useState(null);

  useEffect(() => {
    const onHash = () => setLoc(parseHash());
    window.addEventListener("hashchange", onHash);
    return () => window.removeEventListener("hashchange", onHash);
  }, []);

  const api = useMemo(() => ({
    toast: (title, message) => setToast({ title, message, key: Math.random().toString(36).slice(2)
})
  }), []);
```

```
  useEffect(() => {
    (async () => {
      await storage.init();
      await ensureExampleSeed(storage);
    })();
  }, []);

  let page = null;
  if (loc.route === "generate") page = <GeneratePage api={api} />;
  if (loc.route === "library") page = <LibraryPage api={api} />;
  if (loc.route === "character") page = <CharacterPage api={api} id={loc.id} />;
  if (loc.route === "settings") page = <SettingsPage api={api} />;

  return (
    <>
      <div className="container">
        <div className="topbar">
          <div className="brand">
            <span className="dot" />
            <span>Anime Character Lab</span>
            <span className="pill">Local • Offline • Private</span>
          </div>
          <Nav current={loc.route} />
        </div>

        {page}
      </div>

      <ToastHost toast={toast} onDone={() => setToast(null)} />
    </>
  );
}
import React from "react";

function NavLink({ href, active, children }) {
  return (
```

```
    <a className={active ? "active" : ""} href={href}>
      {children}
    </a>
  );
}

export default function Nav({ current }) {
  return (
    <div className="nav">
      <NavLink href="#/generate" active={current === "generate"}>Generate</NavLink>
      <NavLink href="#/library" active={current === "library"}>Library</NavLink>
      <NavLink href="#/settings" active={current === "settings"}>Settings</NavLink>
    </div>
  );
}
import React, { useEffect } from "react";

export default function ToastHost({ toast, onDone }) {
  useEffect(() => {
    if (!toast) return;
    const t = setTimeout(onDone, 2800);
    return () => clearTimeout(t);
  }, [toast, onDone]);

  if (!toast) return null;
  return (
    <div className="toast-wrap">
      <div className="toast">
        <strong>{toast.title}</strong>
        <div className="subtle">{toast.message}</div>
      </div>
    </div>
  );
}
const DB_NAME = "anime_character_lab";
const DB_VERSION = 1;
```

```
function reqToPromise(req) {
  return new Promise((resolve, reject) => {
    req.onsuccess = () => resolve(req.result);
    req.onerror = () => reject(req.error);
  });
}

export async function openDb() {
  if (!("indexedDB" in window)) return null;

  return new Promise((resolve, reject) => {
    const req = indexedDB.open(DB_NAME, DB_VERSION);

    req.onupgradeneeded = () => {
      const db = req.result;
      if (!db.objectStoreNames.contains("characters")) {
        const store = db.createObjectStore("characters", { keyPath: "id" });
        store.createIndex("createdAt", "createdAt", { unique: false });
        store.createIndex("favorite", "meta.favorite", { unique: false });
      }
      if (!db.objectStoreNames.contains("settings")) {
        db.createObjectStore("settings", { keyPath: "key" });
      }
    };

    req.onsuccess = () => resolve(req.result);
    req.onerror = () => reject(req.error);
  });
}

export async function idbGet(db, storeName, key) {
  const tx = db.transaction(storeName, "readonly");
  const store = tx.objectStore(storeName);
  return reqToPromise(store.get(key));
}

export async function idbPut(db, storeName, value) {
```

```
  const tx = db.transaction(storeName, "readwrite");
  const store = tx.objectStore(storeName);
  await reqToPromise(store.put(value));
}

export async function idbDelete(db, storeName, key) {
  const tx = db.transaction(storeName, "readwrite");
  const store = tx.objectStore(storeName);
  await reqToPromise(store.delete(key));
}

export async function idbGetAll(db, storeName) {
  const tx = db.transaction(storeName, "readonly");
  const store = tx.objectStore(storeName);
  return reqToPromise(store.getAll());
}

export async function idbClear(db, storeName) {
  const tx = db.transaction(storeName, "readwrite");
  const store = tx.objectStore(storeName);
  await reqToPromise(store.clear());
}
import { openDb, idbGet, idbPut, idbDelete, idbGetAll, idbClear } from "./idb.js";

const LS_KEY = "acl_store_v1";

function lsLoad() {
  try { return JSON.parse(localStorage.getItem(LS_KEY) || "{}"); }
  catch { return {}; }
}
function lsSave(data) {
  localStorage.setItem(LS_KEY, JSON.stringify(data));
}

class Storage {
  constructor() {
    this.db = null;
```

```
  this.mode = "indexeddb"; // or localstorage
  this.safeMode = false;
}

async init() {
  this.db = await openDb().catch(() => null);

  // settings
  const s = await this.getSetting("app_settings");
  if (s) {
    this.mode = s.mode || this.mode;
    this.safeMode = !!s.safeMode;
  }
  if (this.mode === "indexeddb" && !this.db) this.mode = "localstorage";
}

async getSetting(key) {
  if (this.db) {
    const row = await idbGet(this.db, "settings", key);
    return row ? row.value : null;
  }
  const ls = lsLoad();
  return ls.settings?.[key] ?? null;
}

async setSetting(key, value) {
  if (this.db) {
    await idbPut(this.db, "settings", { key, value });
    return;
  }
  const ls = lsLoad();
  ls.settings = ls.settings || {};
  ls.settings[key] = value;
  lsSave(ls);
}

async saveAppSettings({ mode, safeMode }) {
```

```
  this.mode = mode;
  this.safeMode = !!safeMode;
  await this.setSetting("app_settings", { mode: this.mode, safeMode: this.safeMode });
}

async listCharacters() {
  if (this.mode === "indexeddb" && this.db) {
    return await idbGetAll(this.db, "characters");
  }
  const ls = lsLoad();
  return Object.values(ls.characters || {});
}

async getCharacter(id) {
  if (this.mode === "indexeddb" && this.db) {
    return await idbGet(this.db, "characters", id);
  }
  const ls = lsLoad();
  return ls.characters?.[id] || null;
}

async putCharacter(character) {
  if (this.mode === "indexeddb" && this.db) {
    await idbPut(this.db, "characters", character);
    return;
  }
  const ls = lsLoad();
  ls.characters = ls.characters || {};
  ls.characters[character.id] = character;
  lsSave(ls);
}

async deleteCharacter(id) {
  if (this.mode === "indexeddb" && this.db) {
    await idbDelete(this.db, "characters", id);
    return;
  }
```

```
    const ls = lsLoad();
    if (ls.characters?.[id]) delete ls.characters[id];
    lsSave(ls);
  }

  async clearAllCharacters() {
    if (this.mode === "indexeddb" && this.db) {
      await idbClear(this.db, "characters");
      return;
    }
    const ls = lsLoad();
    ls.characters = {};
    lsSave(ls);
  }
}

export const storage = new Storage();
export function uid(prefix = "char") {
  return `${prefix}_${Math.random().toString(36).slice(2,
10)}${Math.random().toString(36).slice(2, 6)}`;
}

export function nowIso() {
  return new Date().toISOString();
}
export function characterToMarkdown(c) {
  const lines = [];
  lines.push(`# ${c.core.name} (${c.core.alias})`);
  lines.push(`- Age: ${c.core.age} • Pronouns: ${c.core.pronouns || "—"}`);
  lines.push(`- Genre: ${c.core.genre} • Archetype: ${c.core.archetype} • Vibe: ${c.core.vibe}`);
  lines.push(`- Alignment: ${c.core.alignment} • Power: ${c.core.powerLevel}`);
  lines.push(`- Version: ${c.version}${c.parentId ? ` (parent: ${c.parentId})` : ""}`);
  lines.push("");

  lines.push(`## Body`);
  lines.push(`- Height: ${c.body.height}`);
  lines.push(`- Build: ${c.body.build}`);
```

```
if (c.body.chestType) lines.push(`- Chest type: ${c.body.chestType}`);
lines.push("");

lines.push(`## Appearance`);
lines.push(`- Hair: ${c.appearance.hair.color}, ${c.appearance.hair.style}`);
lines.push(`- Eyes: ${c.appearance.eyes}`);
lines.push(`- Skin tone: ${c.appearance.skinTone}`);
lines.push(`- Distinctive feature: ${c.appearance.distinctiveFeature}`);
lines.push("");

lines.push(`## Outfit (Urban Anomaly)`);
lines.push(`- Palette: ${c.outfit.palette.join(", ")}`);
lines.push(`- Anchors: ${c.outfit.anchors.join(", ") || "—"}`);
lines.push(`- Modifiers: ${c.outfit.modifiers.join(", ") || "—"}`);
lines.push(`- Slots:`);
Object.entries(c.outfit.slots).forEach(([k, v]) => {
  lines.push(`  - ${k}: ${Array.isArray(v) ? v.join(", ") : v}`);
});
if (c.outfit.fitNotes) {
  lines.push(`- Fit notes: ${c.outfit.fitNotes}`);
}
lines.push("");

lines.push(`## Personality`);
lines.push(`- Traits: ${c.personality.traits.join(", ")}`);
lines.push(`- Flaw: ${c.personality.flaw}`);
lines.push(`- Motivation: ${c.personality.motivation}`);
lines.push(`- Fear: ${c.personality.fear}`);
lines.push("");

lines.push(`## Ability`);
lines.push(`- ${c.abilities.name}`);
lines.push(`- Description: ${c.abilities.description}`);
lines.push(`- Cost: ${c.abilities.cost}`);
lines.push(`- Counter: ${c.abilities.counter}`);
lines.push("");
```

```
  lines.push(`## Story`);
  lines.push(`- Arc stage: ${c.story.arcStage}`);
  lines.push(`- Backstory: ${c.story.backstory}`);
  lines.push(`- Plot hooks:`);
  c.story.plotHooks.forEach(h => lines.push(`  - ${h}`));
  lines.push("");

  lines.push(`## Meta`);
  lines.push(`- Tags: ${(c.meta.tags || []).join(", ")}`);
  lines.push(`- Favorite: ${c.meta.favorite ? "Yes" : "No"}`);
  if (c.meta.notes) lines.push(`- Notes: ${c.meta.notes}`);

  return lines.join("\n");
}

export function downloadText(filename, text) {
  const blob = new Blob([text], { type: "text/plain;charset=utf-8" });
  const a = document.createElement("a");
  a.href = URL.createObjectURL(blob);
  a.download = filename;
  a.click();
  URL.revokeObjectURL(a.href);
}

export function downloadJson(filename, obj) {
  const text = JSON.stringify(obj, null, 2);
  const blob = new Blob([text], { type: "application/json;charset=utf-8" });
  const a = document.createElement("a");
  a.href = URL.createObjectURL(blob);
  a.download = filename;
  a.click();
  URL.revokeObjectURL(a.href);
}
export function diffCharacters(prev, next) {
  const changes = [];

  function changed(path, a, b) {
```

```
    const same = JSON.stringify(a) === JSON.stringify(b);
    if (!same) changes.push({ path, from: a, to: b });
  }

  changed("core", prev.core, next.core);
  changed("body", prev.body, next.body);
  changed("appearance", prev.appearance, next.appearance);
  changed("outfit.palette", prev.outfit.palette, next.outfit.palette);
  changed("outfit.modifiers", prev.outfit.modifiers, next.outfit.modifiers);
  changed("outfit.anchors", prev.outfit.anchors, next.outfit.anchors);
  changed("outfit.slots", prev.outfit.slots, next.outfit.slots);
  changed("personality", prev.personality, next.personality);
  changed("abilities", prev.abilities, next.abilities);
  changed("story", prev.story, next.story);

  return changes;
}
export const MODIFIERS = {
  glowSeams: "Subtle glowing lines along seams",
  sigilEmbroidery: "Geometric sigil embroidery patterns",
  neonTrim: "Thin neon accent trim",
  wearState: ["clean", "worn", "battle-damaged"],
  corruption: "Darker palette with unstable glow"
};

export const URBAN_ANOMALY = {
  stylePack: "Urban Anomaly",
  description: "Modern anime streetwear with subtle supernatural anomaly details.",
  items: [
    // TOPS (12)
    { id:"top_hoodie_01", slot:"top", name:"Oversized high-collar hoodie",
tags:["modern","streetwear","casual"], silhouette:"oversized",
materials:["cotton","tech-fabric"], paletteHints:["black","gray"],
modifiersAllowed:["glowSeams","sigilEmbroidery","wearState"] },
    { id:"top_hoodie_02", slot:"top", name:"Zip hoodie with paneling", tags:["techwear","urban"],
silhouette:"layered", materials:["tech-fabric"], paletteHints:["black","accent"],
```

```
modifiersAllowed:["neonTrim","wearState"] },
  { id:"top_jacket_01", slot:"top", name:"Cropped tech jacket", tags:["techwear","urban"],
silhouette:"short", materials:["synthetic","reinforced"], paletteHints:["black","off-white"],
modifiersAllowed:["neonTrim","wearState"] },
  { id:"top_shirt_01", slot:"top", name:"Longline tee (asymmetric hem)", tags:["streetwear"],
silhouette:"longline", materials:["cotton"], paletteHints:["off-white","gray"],
modifiersAllowed:["sigilEmbroidery"] },
  { id:"top_vest_01", slot:"top", name:"Light tactical vest (minimal)",
tags:["techwear","combat"], silhouette:"layered", materials:["synthetic"],
paletteHints:["black"], modifiersAllowed:["wearState"] },
  { id:"top_sweater_01", slot:"top", name:"Mock-neck sweater", tags:["modern","clean"],
silhouette:"slim", materials:["knit"], paletteHints:["black","gray"],
modifiersAllowed:["sigilEmbroidery"] },
  { id:"top_blouse_01", slot:"top", name:"Clean blouse with hidden zipper",
tags:["modern","smart"], silhouette:"slim", materials:["tech-fabric"], paletteHints:["off-white"],
modifiersAllowed:["neonTrim"] },
  { id:"top_wrap_01", slot:"top", name:"Wrap top with belt strap", tags:["urban","anomaly"],
silhouette:"layered", materials:["tech-fabric"], paletteHints:["black"],
modifiersAllowed:["sigilEmbroidery","glowSeams"] },
  { id:"top_tank_01", slot:"top", name:"Base layer tank", tags:["combat","layer"],
silhouette:"slim", materials:["synthetic"], paletteHints:["black"],
modifiersAllowed:["wearState"] },
  { id:"top_tee_02", slot:"top", name:"Graphic tee (abstract noise)", tags:["streetwear"],
silhouette:"regular", materials:["cotton"], paletteHints:["off-white","accent"],
modifiersAllowed:["wearState"] },
  { id:"top_jacket_02", slot:"top", name:"Short bomber with ribbed collar",
tags:["modern","streetwear"], silhouette:"short", materials:["synthetic"], paletteHints:["black"],
modifiersAllowed:["neonTrim","wearState"] },
  { id:"top_uniform_01", slot:"top", name:"Urban academy jacket", tags:["uniform","modern"],
silhouette:"regular", materials:["tech-fabric"], paletteHints:["black","off-white"],
modifiersAllowed:["sigilEmbroidery","neonTrim"] },

  // BOTTOMS (10)
  { id:"bottom_pants_01", slot:"bottom", name:"Slim tech pants with straps",
tags:["techwear","combat"], silhouette:"slim", materials:["synthetic"], paletteHints:["black"],
modifiersAllowed:["wearState"] },
```

```
  { id:"bottom_pants_02", slot:"bottom", name:"Wide urban pants (clean seams)", tags:["streetwear"], silhouette:"wide", materials:["cotton"], paletteHints:["gray","off-white"], modifiersAllowed:[] },
  { id:"bottom_cargo_01", slot:"bottom", name:"Minimal cargo pants", tags:["urban","utility"], silhouette:"regular", materials:["tech-fabric"], paletteHints:["black","gray"], modifiersAllowed:["wearState"] },
  { id:"bottom_skirt_01", slot:"bottom", name:"Layered skirt over leggings", tags:["anime-classic","urban"], silhouette:"layered", materials:["synthetic","cotton"], paletteHints:["black","accent"], modifiersAllowed:["neonTrim"] },
  { id:"bottom_leggings_01", slot:"bottom", name:"Compression leggings", tags:["combat","layer"], silhouette:"slim", materials:["synthetic"], paletteHints:["black"], modifiersAllowed:["wearState"] },
  { id:"bottom_shorts_01", slot:"bottom", name:"Shorts over tights", tags:["streetwear","anime"], silhouette:"layered", materials:["cotton"], paletteHints:["black","off-white"], modifiersAllowed:["wearState"] },
  { id:"bottom_uniform_01", slot:"bottom", name:"Academy trousers (tapered)", tags:["uniform","modern"], silhouette:"slim", materials:["tech-fabric"], paletteHints:["black"], modifiersAllowed:["sigilEmbroidery"] },
  { id:"bottom_denim_01", slot:"bottom", name:"Distressed dark denim", tags:["streetwear"], silhouette:"regular", materials:["denim"], paletteHints:["black"], modifiersAllowed:["wearState"] },
  { id:"bottom_hakama_01", slot:"bottom", name:"Urban hakama-inspired pants", tags:["anomaly","dramatic"], silhouette:"wide", materials:["tech-fabric"], paletteHints:["black"], modifiersAllowed:["sigilEmbroidery","glowSeams"] },
  { id:"bottom_jogger_01", slot:"bottom", name:"Tech joggers", tags:["modern","mobility"], silhouette:"regular", materials:["synthetic"], paletteHints:["gray","accent"], modifiersAllowed:["neonTrim"] },

  // OUTERS (8)
  { id:"outer_coat_01", slot:"outer", name:"Long urban coat", tags:["urban","dramatic"], silhouette:"longline", materials:["tech-fabric"], paletteHints:["black"], modifiersAllowed:["glowSeams","corruption","wearState"] },
  { id:"outer_coat_02", slot:"outer", name:"Hooded coat with inner lining", tags:["mystery","urban"], silhouette:"longline", materials:["tech-fabric"], paletteHints:["black","accent"], modifiersAllowed:["glowSeams","wearState"] },
  { id:"outer_jacket_01", slot:"outer", name:"Detachable-sleeve jacket", tags:["techwear"],
```

```
silhouette:"layered", materials:["synthetic"], paletteHints:["black"],
modifiersAllowed:["neonTrim","wearState"] },
  { id:"outer_cape_01", slot:"outer", name:"Half-cape drape", tags:["anomaly","dramatic"],
silhouette:"layered", materials:["tech-fabric"], paletteHints:["black"],
modifiersAllowed:["sigilEmbroidery","glowSeams"] },
  { id:"outer_blos_01", slot:"outer", name:"Minimal blazer (urban)", tags:["smart","modern"],
silhouette:"regular", materials:["tech-fabric"], paletteHints:["off-white","gray"],
modifiersAllowed:["sigilEmbroidery"] },
  { id:"outer_poncho_01", slot:"outer", name:"Utility poncho", tags:["urban","weather"],
silhouette:"oversized", materials:["waterproof"], paletteHints:["black"],
modifiersAllowed:["wearState"] },
  { id:"outer_coat_03", slot:"outer", name:"Short coat with high shoulders",
tags:["modern","stylish"], silhouette:"short", materials:["synthetic"],
paletteHints:["black","off-white"], modifiersAllowed:["neonTrim"] },
  { id:"outer_bomber_02", slot:"outer", name:"Long bomber (quilted)",
tags:["streetwear","dramatic"], silhouette:"longline", materials:["synthetic"],
paletteHints:["black"], modifiersAllowed:["wearState","neonTrim"] },

  // FOOTWEAR (8)
  { id:"shoes_01", slot:"footwear", name:"Futuristic high-top sneakers",
tags:["modern","mobility"], silhouette:"high-top", materials:["synthetic","rubber"],
paletteHints:["black","accent"], modifiersAllowed:["neonTrim"] },
  { id:"shoes_02", slot:"footwear", name:"Clean combat boots", tags:["combat","urban"],
silhouette:"boots", materials:["leather","rubber"], paletteHints:["black"],
modifiersAllowed:["wearState"] },
  { id:"shoes_03", slot:"footwear", name:"Minimal runners", tags:["modern"],
silhouette:"sneakers", materials:["synthetic"], paletteHints:["gray","accent"],
modifiersAllowed:["neonTrim"] },
  { id:"shoes_04", slot:"footwear", name:"Strapped ankle boots", tags:["urban","stylish"],
silhouette:"boots", materials:["leather"], paletteHints:["black"], modifiersAllowed:["wearState"]
},
  { id:"shoes_05", slot:"footwear", name:"Academy shoes (sleek)", tags:["uniform","modern"],
silhouette:"shoes", materials:["synthetic"], paletteHints:["black"],
modifiersAllowed:["sigilEmbroidery"] },
  { id:"shoes_06", slot:"footwear", name:"Chunky tech sneakers",
tags:["streetwear","techwear"], silhouette:"sneakers", materials:["synthetic"],
```

```
paletteHints:["off-white","accent"], modifiersAllowed:["neonTrim"] },
   { id:"shoes_07", slot:"footwear", name:"Urban sandals with wraps",
tags:["anomaly","casual"], silhouette:"sandals", materials:["synthetic"], paletteHints:["black"],
modifiersAllowed:["sigilEmbroidery"] },
   { idid:"", slot:"footwear", name:"Reinforced toe boots", tags:["combat","techwear"],
silhouette:"boots", materials:["reinforced"], paletteHints:["black"],
modifiersAllowed:["wearState","neonTrim"] }
  ].map(x => x),

  // ACCESSORIES (16) + SIGNATURE (6) appended below
};

const accessory = (id, name, tags, modifiersAllowed=[]) => ({ id, slot:"accessory", name, tags,
silhouette:"—", materials:["mixed"], paletteHints:["black","accent"], modifiersAllowed });
const signature = (id, name, tags, modifiersAllowed=[]) => ({ id, slot:"signature", name, tags,
silhouette:"—", materials:["mixed"], paletteHints:["black","accent"], modifiersAllowed });

export const ACCESSORIES_AND_SIGNATURE = [
  accessory("acc_harness_01","Utility harness",["techwear","combat"]),
  accessory("acc_gloves_01","Fingerless gloves",["combat","streetwear"],["wearState"]),
  accessory("acc_wrap_01","Neck wrap",["mystery","urban"]),
  accessory("acc_earpiece_01","Tech earpiece",["techwear"],["neonTrim"]),
  accessory("acc_belt_01","Utility belt",["urban","utility"]),
  accessory("acc_pouches_01","Minimal pouches set",["utility","combat"],["wearState"]),
  accessory("acc_bracer_01","Wrist device",["techwear"],["neonTrim"]),
  accessory("acc_ring_01","Sigil ring",["anomaly"],["sigilEmbroidery"]),
  accessory("acc_chain_01","Chain accessory",["streetwear"]),
  accessory("acc_cap_01","Low-profile cap",["modern"]),
  accessory("acc_hood_01","Detachable hood",["mystery"],["corruption"]),
  accessory("acc_mask_01","Half-face anomaly mask",["mystery"],["corruption","wearState"]),
  accessory("acc_patch_01","Faction patch set",["urban"],["sigilEmbroidery"]),
  accessory("acc_sleeve_01","Arm sleeve with
markings",["anomaly"],["sigilEmbroidery","glowSeams"]),
  accessory("acc_bandage_01","Wrap bandages",["combat"],["wearState"]),
  accessory("acc_glasses_01","Tinted glasses",["modern","smart"]),
```

```
  signature("sig_scarf_01","Anomaly-pattern
scarf",["anchor"],["sigilEmbroidery","glowSeams"]),
  signature("sig_coatpin_01","Stabilizer pin",["anchor"],["glowSeams"]),
  signature("sig_talisman_01","Talisman strip set",["anchor"],["sigilEmbroidery"]),
  signature("sig_mask_01","Cracked translucent
```