

===== src/lib/buildsStore.ts =====

```
1  export type BuildStatus = "draft" | "submitted" | "reviewing" |
"quote_sent" | "approved" | "in_build" | "complete";
2
3  export type BuildCustomer = {
4      name: string;
5      phone: string;
6      email: string;
7      address?: string;
8  };
9
10 export type BuildDims = {
11     lengthIn: number;
12     widthIn: number;
13     heightIn: number;
14     topThicknessIn?: number;
15 };
16
17 export type BuildOptions = {
18     woodSpecies: "Pine" | "Oak" | "Walnut" | "Maple" | "Poplar" |
"Plywood";
19     finish: "Natural" | "Stain" | "Paint" | "Poly";
20     joinery: "Screws" | "Pocket Holes" | "Mortise & Tenon" |
"Dowels";
21 };
22
23 export type RenderStatus = "queued" | "rendering" | "complete" |
"failed";
24
25 export type RenderJob = {
26     renderId: string;
27     view: "iso" | "front" | "top" | "detail";
28     status: RenderStatus;
29     imageDataUrl?: string;
30     startedAt?: string;
31     finishedAt?: string;
32     estimatePublic?: {
33         total: number;
34         rangeLow?: number;
35         rangeHigh?: number;
36         label?: string;
37     };
38 };
39
40 export type NoteAuthor = "customer" | "admin";
41 export type NoteKind = "initial" | "change" | "refinement" |
"note";
42
43 export type NoteItem = {
44     noteId: string;
45     createdAt: string;
46     author: NoteAuthor;
```

```
47   kind: NoteKind;
48   text: string;
49 };
50
51 export type BuildVersion = {
52   versionId: string;
53   createdAt: string;
54   customerChangeRequest?: string;
55   inputsSnapshot: {
56     type: string;
57     dims: BuildDims;
58     options: BuildOptions;
59
60     // Legacy field kept for backwards compatibility ONLY.
61     // Going forward: we keep this EMPTY to prevent duplicate/
phantom notes.
62   notes?: string;
63
64   // Canonical notes source of truth
65   notesLog?: NoteItem[];
66 };
67   renders: RenderJob[];
68
69   estimatePublic?: {
70     total: number;
71     rangeLow?: number;
72     rangeHigh?: number;
73     materials: number;
74     labor: number;
75     overhead: number;
76     finish: number;
77   };
78
79   estimateInternal?: any;
80   generatedPackage?: any;
81 };
82
83 export type BuildSubmission = {
84   id: string;
85   createdAt: string;
86   updatedAt: string;
87
88   status: BuildStatus;
89   accessCode?: string;
90
91   customer: BuildCustomer;
92
93   project: {
94     type: string;
95     dims: BuildDims;
96     options: BuildOptions;
97
98     // Legacy compiled string (we keep it EMPTY going forward)
99     notes?: string;
100 }
```

```
101     // Canonical structured notes
102     notesLog?: NoteItem[];
103
104     refPhotos?: { name: string; type: string; dataUrl: string }[];
105   };
106
107   versions: BuildVersion[];
108 };
109
110 const KEY = "rv_build_submissions";
111
112 function safeParse<T>(raw: string | null, fallback: T): T {
113   try {
114     if (!raw) return fallback;
115     return JSON.parse(raw) as T;
116   } catch {
117     return fallback;
118   }
119 }
120
121 function uid() {
122   return (crypto as any).randomUUID?.() ??
(Math.random().toString(16).slice(2) + Date.now().toString(16));
123 }
124
125 /**
126  * If an old build exists with only legacy `notes` and no
`notesLog`,
127  * migrate it into a single structured "initial" note.
128 */
129 function ensureNotesLog(legacyNotes?: string, existing?: NoteItem[], createdAtHint?: string): NoteItem[] {
130   const log = Array.isArray(existing) ? existing.filter(Boolean) : [];
131   if (log.length) return log;
132
133   const text = String(legacyNotes || "").trim();
134   if (!text) return [];
135
136   return [
137     {
138       noteId: uid(),
139       createdAt: createdAtHint || new Date().toISOString(),
140       author: "customer",
141       kind: "initial",
142       text,
143     },
144   ];
145 }
146
147 function normalizeBuild(b: any): BuildSubmission | null {
148   if (!b || typeof b !== "object") return null;
149
150   const createdAt = String(b.createdAt || new
Date().toISOString());
```

```
151  const project = b.project || {};
152
153  const legacyProjectNotes = String(project.notes || "").trim();
154  const projectNotesLog = ensureNotesLog(legacyProjectNotes,
project.notesLog, createdAt);
155
156  const versions = Array.isArray(b.versions) ? b.versions : [];
157  const nextVersions: BuildVersion[] = versions.map(v: any) => {
158    const inputs = v?.inputsSnapshot || {};
159    const vCreatedAt = String(v?.createdAt || createdAt);
160
161    // Old builds may have inputsSnapshot.notes (compiled blob)
but no notesLog.
162    const legacyVNotes = String(inputs.notes ??
legacyProjectNotes ?? "").trim();
163    const vNotesLog = ensureNotesLog(legacyVNotes,
inputs.notesLog ?? projectNotesLog, vCreatedAt);
164
165    return {
166      ...v,
167      createdAt: vCreatedAt,
168      inputsSnapshot: {
169        type: String(inputs.type || project.type || ""),
170        dims: inputs.dims || project.dims,
171        options: inputs.options || project.options,
172
173        // IMPORTANT: going forward we keep legacy notes empty to
avoid duplication.
174        // We only keep it for old builds that still have it.
175        notes: String(inputs.notes || "").includes("\n---\n") ?
": "",",
176
177        notesLog: vNotesLog,
178      },
179      renders: Array.isArray(v?.renders) ? v.renders : [],
180    } as BuildVersion;
181  });
182
183  const out: BuildSubmission = {
184    ...b,
185    createdAt,
186    updatedAt: String(b.updatedAt || createdAt),
187    status: (b.status || "draft") as BuildStatus,
188    customer: b.customer || { name: "", phone: "", email: "" },
189    project: {
190      ...project,
191      type: String(project.type || ""),
192      dims: project.dims,
193      options: project.options,
194
195      // IMPORTANT: going forward we keep legacy notes empty
(canonical is notesLog)
196      notes: "",
197
198      notesLog: projectNotesLog,
```

```
199     refPhotos: Array.isArray(project.refPhotos) ?
project.refPhotos : [],
200     },
201     versions: nextVersions,
202   };
203
204   return out;
205 }
206
207 /**
208  * Compile notes for display + renderer.
209  * We do NOT prepend legacy notes anymore (we keep legacy notes
empty going forward).
210  * This prevents "remove last note" from appearing to do nothing.
211  */
212 export function compileNotes(notesLog?: NoteItem[], _legacyNotes?: string) {
213   const items = Array.isArray(notesLog) ? notesLog : [];
214   return items
215     .map((n) => String(n?.text || "")).trim())
216     .filter(Boolean)
217     .join("\n\n---\n\n");
218 }
219
220 export function normalizePhone(p: string) {
221   return String(p || "").replace(/\D+/g, "");
222 }
223
224 export function makeAccessCode() {
225   return String(Math.floor(100000 + Math.random() * 900000));
226 }
227
228 export function readBuilds(): BuildSubmission[] {
229   const arr = safeParse<any[]>(localStorage.getItem(KEY), []);
230   const raw = (Array.isArray(arr) ? arr : []).filter(Boolean);
231   return raw.map(normalizeBuild).filter(Boolean) as
BuildSubmission[];
232 }
233
234 export function writeBuilds(items: BuildSubmission[]) {
235   localStorage.setItem(KEY, JSON.stringify(items));
236 }
237
238 export function getBuild(id: string): BuildSubmission | null {
239   const all = readBuilds();
240   const found = all.find((b) => String(b.id) === String(id));
241   return found || null;
242 }
243
244 export function upsertBuild(next: BuildSubmission) {
245   const all = readBuilds();
246   const idx = all.findIndex((b) => String(b.id) ===
String(next.id));
247   if (idx >= 0) all[idx] = next;
248   else all.unshift(next);
```

```
249     writeBuilds(all);
250   }
251
252   export function deleteBuild(id: string) {
253     const all = readBuilds().filter((b) => String(b.id) !==
String(id));
254     writeBuilds(all);
255   }
256
257   export function createDraftBuild(args: {
258     customer: BuildCustomer;
259     type: string;
260     dims: BuildDims;
261     options: BuildOptions;
262     notes?: string;
263   }): BuildSubmission {
264     const now = new Date().toISOString();
265     const id = uid();
266
267     const baseNotes = String(args.notes || "").trim();
268
269     // canonical notesLog (and legacy notes stays empty)
270     const notesLog: NoteItem[] = baseNotes
271       ? [
272         {
273           noteId: uid(),
274           createdAt: now,
275           author: "customer",
276           kind: "initial",
277           text: baseNotes,
278         },
279       ]
280       : [];
281
282     const version: BuildVersion = {
283       versionId: uid(),
284       createdAt: now,
285       inputsSnapshot: {
286         type: args.type,
287         dims: args.dims,
288         options: args.options,
289         notes: "",
290         notesLog,
291       },
292       renders: [
293         { renderId: uid(), view: "iso", status: "queued" },
294         { renderId: uid(), view: "front", status: "queued" },
295         { renderId: uid(), view: "top", status: "queued" },
296       ],
297     };
298
299     const build: BuildSubmission = {
300       id,
301       createdAt: now,
302       updatedAt: now,
```

```

303     status: "draft",
304     customer: args.customer,
305     project: {
306       type: args.type,
307       dims: args.dims,
308       options: args.options,
309       notes: "",
310       notesLog,
311       refPhotos: [],
312     },
313     versions: [version],
314   };
315
316   upsertBuild(build);
317   return build;
318 }
319
320 export function addRevision(
321   id: string,
322   customerChangeRequest: string,
323   patch?: Partial<BuildSubmission["project"]>
324 ) {
325   const b0 = getBuild(id);
326   if (!b0) return null;
327
328   const b = normalizeBuild(b0);
329   if (!b) return null;
330
331   const now = new Date().toISOString();
332   const mergedProject: BuildSubmission["project"] =
{ ...b.project, ...(patch || {}) };
333
334   const mergedNotesLog = ensureNotesLog("", mergedProject.notesLog, b.createdAt);
335
336   const version: BuildVersion = {
337     versionId: uid(),
338     createdAt: now,
339     customerChangeRequest,
340     inputsSnapshot: {
341       type: mergedProject.type,
342       dims: mergedProject.dims,
343       options: mergedProject.options,
344       notes: "",
345       notesLog: mergedNotesLog,
346     },
347     renders: [
348       { renderId: uid(), view: "iso", status: "queued" },
349       { renderId: uid(), view: "front", status: "queued" },
350       { renderId: uid(), view: "top", status: "queued" },
351       { renderId: uid(), view: "detail", status: "queued" },
352     ],
353   };
354
355   const next: BuildSubmission = {

```

```
356     ...b,
357     updatedAt: now,
358     project: { ...mergedProject, notes: "", notesLog:
mergedNotesLog },
359     versions: [version, ...b.versions],
360   };
361
362   upsertBuild(next);
363   return next;
364 }
365
366 export function addCustomerNote(id: string, changeRequest?: string, extraNotes?: string) {
367   const b0 = getBuild(id);
368   if (!b0) return null;
369
370   const b = normalizeBuild(b0);
371   if (!b) return null;
372
373   const now = new Date().toISOString();
374   const prevLog = ensureNotesLog("", b.project.notesLog,
b.createdAt);
375
376   const nextLog: NoteItem[] = [...prevLog];
377
378   const req = String(changeRequest || "").trim();
379   const add = String(extraNotes || "").trim();
380
381   if (req) {
382     nextLog.push({
383       noteId: uid(),
384       createdAt: now,
385       author: "customer",
386       kind: "change",
387       text: req,
388     });
389   }
390
391   if (add) {
392     nextLog.push({
393       noteId: uid(),
394       createdAt: now,
395       author: "customer",
396       kind: "refinement",
397       text: add,
398     });
399   }
400
401   return addRevision(id, "Customer provided additional details", {
402     notesLog: nextLog,
403     notes: "",
404   });
405 }
406
407 export function removeLastCustomerNote(id: string) {
```

```

408     const b0 = getBuild(id);
409     if (!b0) return null;
410
411     const b = normalizeBuild(b0);
412     if (!b) return null;
413
414     const prevLog = ensureNotesLog("", b.project.notesLog,
b.createdAt);
415     if (!prevLog.length) return b;
416
417     let idx = -1;
418     for (let i = prevLog.length - 1; i >= 0; i--) {
419         if (prevLog[i].author === "customer") {
420             idx = i;
421             break;
422         }
423     }
424     if (idx < 0) return b;
425
426     const nextLog = prevLog.slice(0, idx).concat(prevLog.slice(idx + 1));
427
428     return addRevision(id, "Customer removed last note", {
429         notesLog: nextLog,
430         notes: "",
431     });
432 }
433
434 export function markSubmitted(id: string) {
435     const b = getBuild(id);
436     if (!b) return null;
437
438     const now = new Date().toISOString();
439     const accessCode = b.accessCode &&
String(b.accessCode).trim().length >= 6 ? b.accessCode : makeAccessCode();
440
441     const next: BuildSubmission = {
442         ...b,
443         updatedAt: now,
444         status: b.status === "draft" ? "submitted" : b.status,
445         accessCode,
446     };
447
448     upsertBuild(next);
449     return next;
450 }
451
452 export function findBuildsByPhoneAndCode(phone: string, code: string) {
453     const p = normalizePhone(phone);
454     const c = String(code || "").replace(/\D+/g, "");
455     if (!p || c.length < 6) return [];
456     return readBuilds().filter((b) =>
normalizePhone(b.customer?.phone || "") === p && String(b.accessCode || "") === c);

```

```

457 }
458
459 export function findBuildsByNameAndPhone(name: string, phone:
string) {
460   const n = String(name || "").trim().toLowerCase();
461   const p = normalizePhone(phone);
462   if (!n || p.length < 7) return [];
463   return readBuilds().filter((b) => {
464     const bn = String(b.customer?.name ||
"").trim().toLowerCase();
465     const bp = normalizePhone(b.customer?.phone || "");
466     return bn.includes(n) && bp.endsWith(p.slice(-7));
467   });
468 }

```

===== src/pages/BuildDesigner.tsx =====

```

1 import { useMemo, useState } from "react";
2 import { createDraftBuild, type BuildOptions, type BuildDims } from
"../lib/buildsStore";
3
4 function num(v: any, fallback: number) {
5   const n = Number(v);
6   return Number.isFinite(n) ? n : fallback;
7 }
8
9 export default function BuildDesigner() {
10   const [customerName, setCustomerName] = useState("");
11   const [customerPhone, setCustomerPhone] = useState("");
12   const [customerEmail, setCustomerEmail] = useState("");
13   const [customerAddress, setCustomerAddress] = useState("");
14
15   const [type, setType] = useState("Table");
16   const [lengthIn, setLengthIn] = useState(60);
17   const [widthIn,setWidthIn] = useState(30);
18   const [heightIn, setHeightIn] = useState(30);
19   const [topThicknessIn, setTopThicknessIn] = useState(1.5);
20
21   const [woodSpecies, setWoodSpecies] =
useState<BuildOptions["woodSpecies"]>("Pine");
22   const [finish, setFinish] =
useState<BuildOptions["finish"]>("Natural");
23   const [joinery, setJoinery] =
useState<BuildOptions["joinery"]>("Pocket Holes");
24
25   const [notes, setNotes] = useState("");
26
27   const dims: BuildDims = useMemo(
28     () => ({
29       lengthIn: Math.max(12, num(lengthIn, 60)),
30       widthIn: Math.max(10, num(widthIn, 30)),
31       heightIn: Math.max(10, num(heightIn, 30)),
32       topThicknessIn: Math.max(0.5, num(topThicknessIn, 1.5)),
33     }),

```

```
34     [lengthIn, widthIn, heightIn, topThicknessIn]
35   );
36
37   const options: BuildOptions = useMemo(
38     () => ({ woodSpecies, finish, joinery }),
39     [woodSpecies, finish, joinery]
40   );
41
42   function onStart() {
43     if (!customerName.trim() || !customerPhone.trim() || !
customerEmail.trim()) {
44       alert("Please enter name, phone, and email.");
45       return;
46     }
47
48   const draft = createDraftBuild({
49     customer: {
50       name: customerName.trim(),
51       phone: customerPhone.trim(),
52       email: customerEmail.trim(),
53       address: customerAddress.trim() || "",
54     },
55     type,
56     dims,
57     options,
58     notes,
59   });
60
61   window.location.href = `/builds/${draft.id}`;
62 }
63
64   return (
65     <div className="stack page" style={{ gap: 16 }}>
66       <section className="panel card card-center"
style={{ maxWidth: 980, margin: "0 auto", padding: 18 }}>
67         <h1 className="h2" style={{ margin: 0, fontWeight: 950 }}>
>Start a Custom Build</h1>
68         <p className="lead" style={{ maxWidth: 820 }}>
69           Fill in your project details. On the next screen you'll
see 3D render previews + estimate boxes update as each view completes.
70         </p>
71
72         <div className="panel" style={{ padding: 14, borderRadius:
14, width: "100%", maxWidth: 860 }}>
73           <div style={{ fontWeight: 950, color: "#0f172a" }}>
>Contact Info (required)</div>
74           <div style={{ display: "grid", gap: 10, marginTop: 10 }}>
>
75             <input className="field" placeholder="Full Name"
value={customerName} onChange={(e) => setCustomerName(e.target.value)} />
76             <input className="field" placeholder="Phone Number"
value={customerPhone} onChange={(e) => setCustomerPhone(e.target.value)} />
77             <input className="field" placeholder="Email"
value={customerEmail} onChange={(e) => setCustomerEmail(e.target.value)} />
```

```

>
    78          <input className="field" placeholder="Address
(optional)" value={customerAddress} onChange={(e) =>
setCustomerAddress(e.target.value)} />
    79          </div>
    80      </div>
    81
    82          <div className="panel" style={{ padding: 14, borderRadius:
14, width: "100%", maxWidth: 860 }}>
    83              <div style={{ fontWeight: 950, color: "#0f172a" }}>
>Project Basics</div>
    84
    85          <div style={{ display: "grid", gap: 10, marginTop: 10 }}>
>
    86          <label style={{ display: "grid", gap: 6 }}>
    87              <span className="label">Project Type</span>
    88              <select className="field" value={type} onChange={(e)
=> setType(e.target.value)}>
    89                  <option>Table</option>
    90                  <option>Bench</option>
    91                  <option>Shelf</option>
    92                  <option>Cabinet</option>
    93                  <option>Planter Box</option>
    94                  <option>Workbench</option>
    95              </select>
    96          </label>
    97
    98          <div className="row" style={{ gap: 10, flexWrap:
"wrap" }}>
    99              <label style={{ display: "grid", gap: 6, flex: 1,
minWidth: 160 }}>
    100                  <span className="label">Length (in)</span>
    101                  <input className="field" inputMode="numeric"
value={lengthIn} onChange={(e) => setLengthIn(Number(e.target.value))} />
    102              </label>
    103              <label style={{ display: "grid", gap: 6, flex: 1,
minWidth: 160 }}>
    104                  <span className="label">Width (in)</span>
    105                  <input className="field" inputMode="numeric"
value={widthIn} onChange={(e) => setWidthIn(Number(e.target.value))} />
    106              </label>
    107              <label style={{ display: "grid", gap: 6, flex: 1,
minWidth: 160 }}>
    108                  <span className="label">Height (in)</span>
    109                  <input className="field" inputMode="numeric"
value={heightIn} onChange={(e) => setHeightIn(Number(e.target.value))} />
    110              </label>
    111              <label style={{ display: "grid", gap: 6, flex: 1,
minWidth: 160 }}>
    112                  <span className="label">Top thickness (in)</span>
    113                  <input className="field" inputMode="decimal"
value={topThicknessIn} onChange={(e) =>
setTopThicknessIn(Number(e.target.value))} />
    114              </label>
    115          </div>

```

```
116
117      <div className="row" style={{ gap: 10, flexWrap:
118 "wrap" }}>
119          <label style={{ display: "grid", gap: 6, flex: 1,
120 minWidth: 220 }}>
121              <span className="label">Wood Species</span>
122              <select className="field" value={woodSpecies}
123 onChange={(e) => setWoodSpecies(e.target.value as any)}>
124                  <option value="Pine">Pine</option>
125                  <option value="Poplar">Poplar</option>
126                  <option value="Plywood">Plywood</option>
127                  <option value="Oak">Oak</option>
128                  <option value="Maple">Maple</option>
129                  <option value="Walnut">Walnut</option>
130          </select>
131      </label>
132
133      <label style={{ display: "grid", gap: 6, flex: 1,
134 minWidth: 220 }}>
135          <span className="label">Finish</span>
136          <select className="field" value={finish}
137 onChange={(e) => setFinish(e.target.value as any)}>
138              <option value="Natural">Natural</option>
139              <option value="Stain">Stain</option>
140              <option value="Paint">Paint</option>
141              <option value="Poly">Poly</option>
142          </select>
143      </label>
144
145      <label style={{ display: "grid", gap: 6, flex: 1,
146 minWidth: 220 }}>
147          <span className="label">Joinery Preference</span>
148          <select className="field" value={joinery}
149 onChange={(e) => setJoinery(e.target.value as any)}>
150              <option value="Screws">Screws</option>
151              <option value="Pocket Holes">Pocket Holes</
152 option>
153              <option value="Dowels">Dowels</option>
154              <option value="Mortise & Tenon">Mortise &
155 Tenon</option>
156          </select>
157      </label>
158
159      <div>
160          <label style={{ display: "grid", gap: 6 }}>
161              <span className="label">Notes / Special Requests</
162 span>
163              <textarea className="field" rows={4} value={notes}
164 onChange={(e) => setNotes(e.target.value)} placeholder="Example: tapered
165 legs, lower shelf, rounded corners, hidden fasteners, etc." />
166          </label>
167
168          <button className="btn btn-primary" onClick={onStart}
169 style={{ fontWeight: 950 }}>
170              Generate Render Previews →
171      </div>
172
```

```
158         </button>
159
160         <div className="muted" style={{ fontWeight: 850 }}>
161             You'll receive an Access Code after submission to
view your build in the Build Portal.
162         </div>
163     </div>
164     </div>
165     </section>
166   </div>
167 );
168 }
```

===== src/pages/BuildPreview.tsx =====

```
1 import { useEffect, useMemo, useState } from "react";
2 import { useParams, Link } from "react-router-dom";
3 import { estimateBuild } from "../lib/buildPricing";
4 import { renderBuildPreviewPng } from "../lib/render3d";
5 import {
6   addCustomerNote,
7   compileNotes,
8   getBuild,
9   markSubmitted,
10  removeLastCustomerNote,
11  type BuildSubmission,
12  type RenderJob,
13  upsertBuild,
14 } from "../lib/buildsStore";
15
16 function fmt(iso: string) {
17   const d = new Date(iso);
18   if (Number.isNaN(d.getTime())) return iso;
19   return d.toLocaleString();
20 }
21
22 function money(n: number) {
23   return new Intl.NumberFormat("en-US", { style: "currency",
currency: "USD" }).format(n);
24 }
25
26 export default function BuildPreview() {
27   const { id } = useParams();
28   const [build, setBuild] = useState<BuildSubmission | null>(null);
29
30   // Customer refinement fields
31   const [changeRequest, setChangeRequest] = useState("");
32   const [extraNotes, setExtraNotes] = useState("");
33
34   useEffect(() => {
35     if (!id) return;
36     setBuild(getBuild(id));
37   }, [id]);
```

```
38      const version = useMemo(() => build?.versions?.[0] ?? null,
[build]);
39
40      // Render queue: ONE image at a time
41      useEffect(() => {
42          if (!build || !version) return;
43
44          const latest0 = getBuild(build.id) ?? build;
45          const lv0 = latest0.versions?.[0];
46          if (!lv0) return;
47
48          const renders0 = lv0.renders || [];
49
50
51          const currentlyRendering = renders0.find((r) => r.status ===
"rendering") ?? null;
52          let target: RenderJob | null = currentlyRendering;
53
54          if (!target) {
55              const firstQueued = renders0.find((r) => r.status ===
"queued") ?? null;
56
57              if (firstQueued) {
58                  const startedAt = new Date().toISOString();
59                  const updatedRenders = renders0.map((r) =>
60                      r.renderId === firstQueued.renderId ? ({ ...r, status:
"rendering" } as const, startedAt } as RenderJob) : r
61                  );
62
63                  const nextV = { ...lv0, renders: updatedRenders };
64                  const nextBuild: BuildSubmission = {
65                      ...latest0,
66                      updatedAt: new Date().toISOString(),
67                      versions: [nextV, ...latest0.versions.slice(1)],
68                  };
69
70                  upsertBuild(nextBuild);
71                  setBuild(nextBuild);
72
73                  target = updatedRenders.find((r) => r.renderId ===
firstQueued.renderId) as RenderJob;
74              }
75          }
76
77          if (!target) return;
78
79          let cancelled = false;
80
81          const run = async () => {
82              try {
83                  await new Promise((res) => setTimeout(res, 450));
84                  if (cancelled) return;
85
86                  const latest = getBuild(build.id);
87                  if (!latest) return;
```

```
88      const lv = latest.versions[0];
89      const est = estimateBuild(lv.inputsSnapshot.dims,
90 lv.inputsSnapshot.options);
91      const title = `${lv.inputsSnapshot.type} • ${
92 {lv.inputsSnapshot.dims.lengthIn}"×${lv.inputsSnapshot.dims.widthIn}"×$(
93 {lv.inputsSnapshot.dims.heightIn})`;
94      // Notes used for rendering (structured log + base notes)
95      const notesCompiled = compileNotes((lv.inputsSnapshot as
any).notesLog, lv.inputsSnapshot.notes);
96
97      const png = await renderBuildPreviewPng({
98          projectType: lv.inputsSnapshot.type,
99          view: target!.view,
100         title,
101         notes: notesCompiled,
102         dims: lv.inputsSnapshot.dims,
103         options: lv.inputsSnapshot.options,
104         width: 1200,
105         height: 800,
106     });
107
108     if (cancelled) return;
109
110     const updatedRenders = (lv.renders || []).map((x) => {
111         if (x.renderId !== target!.renderId) return x;
112         return {
113             ...x,
114             status: "complete" as const,
115             finishedAt: new Date().toISOString(),
116             imageDataUrl: png,
117             estimatePublic: {
118                 total: est.total,
119                 rangeLow: est.rangeLow,
120                 rangeHigh: est.rangeHigh,
121                 label: "Est. total (updates per view)",
122             },
123         };
124     });
125
126     const nextV = {
127         ...lv,
128         renders: updatedRenders,
129         estimatePublic: {
130             total: est.total,
131             rangeLow: est.rangeLow,
132             rangeHigh: est.rangeHigh,
133             materials: est.materials,
134             labor: est.labor,
135             overhead: est.overhead,
136             finish: est.finish,
137         },
138     };

```

```
139
140     const nextBuild: BuildSubmission = {
141         ...latest,
142         updatedAt: new Date().toISOString(),
143         versions: [nextV, ...latest.versions.slice(1)],
144     };
145
146     upsertBuild(nextBuild);
147     setBuild(nextBuild);
148 } catch (e) {
149     console.error(e);
150     if (cancelled) return;
151
152     const latest = getBuild(build.id);
153     if (!latest) return;
154
155     const lv = latest.versions[0];
156     const updatedRenders = (lv.renders || []).map((x) => {
157         if (x.renderId !== target!.renderId) return x;
158         return { ...x, status: "failed" as const, finishedAt:
new Date().toISOString() };
159     });
160
161     const nextV = { ...lv, renders: updatedRenders };
162     const nextBuild: BuildSubmission = {
163         ...latest,
164         updatedAt: new Date().toISOString(),
165         versions: [nextV, ...latest.versions.slice(1)],
166     };
167
168     upsertBuild(nextBuild);
169     setBuild(nextBuild);
170 }
171 };
172
173 run();
174
175 return () => {
176     cancelled = true;
177 },
178 [build?.id, build?.updatedAt, version?.versionId]);
179
180 if (!build || !version) {
181     return (
182         <div className="panel card card-center" style={{ maxWidth:
900, margin: "0 auto" }}>
183             <h3 className="h3">Build not found</h3>
184             <Link className="btn btn-primary" to="/builds/new">
185                 Start a Build
186             </Link>
187         </div>
188     );
189 }
190
191 const v = version;
```

```
192     const b = build;
193
194     const notesLog = ((v.inputsSnapshot as any).notesLog || []) as
any[];
195     const compiledNotes = compileNotes(notesLog,
v.inputsSnapshot.notes);
196
197     const canRemoveCustomerNote =
198         Array.isArray(notesLog) &&
199         notesLog.some((n) => String(n?.author || "").toLowerCase() ===
"customer");
200
201     function submit() {
202         const next = markSubmitted(b.id);
203         if (!next) return alert("Could not submit. Try again.");
204         setBuild(next);
205         alert(`Submitted! Your Build Access Code: $
{String(next.accessCode || "-")}`);
206     }
207
208     function submitRefinement() {
209         const req = changeRequest.trim();
210         const add = extraNotes.trim();
211
212         if (!req && !add) {
213             alert("Please add a change request and/or extra notes.");
214             return;
215         }
216
217         const next = addCustomerNote(b.id, req, add);
218         if (!next) {
219             alert("Could not save changes. Please refresh and try
again.");
220             return;
221         }
222
223         setChangeRequest("");
224         setExtraNotes("");
225         setBuild(next);
226         alert("Saved! We're generating updated previews now.");
227     }
228
229     function removeLastCustomerNoteClick() {
230         if (!confirm("Remove your most recent note and regenerate
previews?")) return;
231
232         const next = removeLastCustomerNote(b.id);
233         if (!next) {
234             alert("Could not remove the last note. Please refresh and
try again.");
235             return;
236         }
237
238         setBuild(next);
239         alert("Removed! We're generating updated previews now.");
```

```
240     }
241
242     const est = v.estimatePublic;
243
244     return (
245         <div className="stack page" style={{ gap: 16 }}>
246             <section className="panel card card-center"
style={{ maxWidth: 1100, margin: "0 auto", padding: 18 }}>
247                 <div style={{ display: "grid", gap: 8 }}>
248                     <h1 className="h2" style={{ margin: 0, fontWeight:
950 }}>
249                         Build Preview
250                     </h1>
251                     <div className="muted" style={{ fontWeight: 850 }}>
252                         Created: {fmt(b.createdAt)} • Status: <span
className="badge">{String(b.status).toUpperCase()}</span>
253                     </div>
254                     <div className="muted" style={{ fontWeight: 850 }}>
255                         Customer: <strong>{b.customer?.name}</strong> •
{b.customer?.phone} • {b.customer?.email}
256                     </div>
257                 </div>
258
259                 <div className="panel" style={{ padding: 14, borderRadius:
14, marginTop: 12, width: "100%", maxWidth: 1000 }}>
260                     <div style={{ fontWeight: 950, color: "#0f172a" }}>
261                         {v.inputsSnapshot.type} -
{v.inputsSnapshot.dims.lengthIn}" x {v.inputsSnapshot.dims.widthIn}" x{"
"}>
262                         {v.inputsSnapshot.dims.heightIn}"
263                     </div>
264                     <div className="muted" style={{ fontWeight: 850,
marginTop: 6 }}>
265                         Wood: {v.inputsSnapshot.options.woodSpecies} • Finish:
{v.inputsSnapshot.options.finish} • Joinery:{" "}
266                         {v.inputsSnapshot.options.joinery}
267                     </div>
268
269                         {compiledNotes ? (
270                             <div className="panel" style={{ padding: 12,
borderRadius: 12, marginTop: 10 }}>
271                                 <div className="label">Notes on file (used to
improve the model)</div>
272                                 <div className="muted" style={{ fontWeight: 850,
whiteSpace: "pre-wrap", marginTop: 6 }}>
273                                     {compiledNotes}
274                                 </div>
275                                 </div>
276                         ) : null}
277
278                         <div className="row" style={{ gap: 10, flexWrap: "wrap",
marginTop: 12 }}>
279                             <Link className="btn btn-ghost" to="/builds/new">
280                                 Start Another
281                             </Link>
```

```
282             <Link className="btn btn-ghost" to="/builds/portal">
283                 Build Portal
284             </Link>
285             <button className="btn btn-primary" onClick={submit}
style={{ fontWeight: 950 }}>
286                 Submit for Review
287             </button>
288         </div>
289
290         <div className="muted" style={{ fontWeight: 850,
marginTop: 10 }}>
291             After submitting you'll get a 6-digit Access Code to
view your build in the Build Portal.
292         </div>
293     </div>
294
295     <div className="panel" style={{ padding: 14, borderRadius:
14, marginTop: 12, width: "100%", maxWidth: 1000 }}>
296         <div style={{ fontWeight: 950, color: "#0f172a" }}>
>Refine this build (add details)</div>
297         <div className="muted" style={{ fontWeight: 850,
marginTop: 6 }}>
298             Add specific details and we'll regenerate previews.
Examples: "tapered legs", "lower shelf", "drawer", "apron", "feet", "2
shelves".
299         </div>
300
301         <div style={{ display: "grid", gap: 10, marginTop: 10 }}>
>
302             <label style={{ display: "grid", gap: 6 }}>
303                 <span className="label">What would you like changed?
</span>
304                 <input
305                     className="field"
306                     value={changeRequest}
307                     onChange={(e) => setChangeRequest(e.target.value)}
308                     placeholder="Example: Add a lower shelf and
tapered legs"
309                     />
310             </label>
311
312             <label style={{ display: "grid", gap: 6 }}>
313                 <span className="label">Extra notes (used by the
renderer)</span>
314                 <textarea
315                     className="field"
316                     rows={4}
317                     value={extraNotes}
318                     onChange={(e) => setExtraNotes(e.target.value)}
319                     placeholder="Example: drawer centered on front,
apron on all sides, rounded corners, etc."
320                     />
321             </label>
322
323         <div className="row" style={{ gap: 10, flexWrap:
```

```
"wrap", justifyContent: "center" }}>
  324           <button type="button" className="btn btn-primary"
onClick={(e) => { e.preventDefault(); e.stopPropagation();
submitRefinement(); }} style={{ fontWeight: 950 }}>
  325             Save refinement + regenerate previews →
  326           </button>
  327
  328           {canRemoveCustomerNote ? (
  329             <button type="button" className="btn btn-ghost"
onClick={(e) => { e.preventDefault(); e.stopPropagation();
removeLastCustomerNoteClick(); }} style={{ fontWeight: 950 }}>
  330               Remove my last note
  331             </button>
  332           ) : null}
  333         </div>
  334       </div>
  335
  336       {Array.isArray(notesLog) && notesLog.length ? (
  337         <div className="panel" style={{ padding: 12,
borderRadius: 12, marginTop: 12 }}>
  338           <div className="label">Notes timeline</div>
  339           <div className="muted" style={{ fontWeight: 850,
marginTop: 6 }}>
  340             Notes are saved in separate chunks so Admin can
remove any later if needed.
  341           </div>
  342
  343           <div style={{ display: "grid", gap: 10, marginTop:
10 }}>
  344             {notesLog.map((n) => (
  345               <div key={String(n.noteId)} className="panel"
style={{ padding: 10, borderRadius: 12 }}>
  346                 <div className="row" style={{ justifyContent:
"space-between", gap: 10, flexWrap: "wrap" }}>
  347                   <div style={{ fontWeight: 950, color:
"#0f172a" }}>
  348                     {String(n.kind || "NOTE").toUpperCase() +
String(n.author || "UNKNOWN").toUpperCase()}
  349                     <span className="badge"
style={{ marginLeft: 8 }}>
  350                       {fmt(String(n.createdAt))}
  351                     </span>
  352                   </div>
  353                   <div className="muted" style={{ fontWeight:
850 }}>
  354                     Note ID: <span
className="badge">{String(n.noteId).slice(-8).toUpperCase()}</span>
  355                     </div>
  356                   </div>
  357                   <div className="muted" style={{ fontWeight:
850, whiteSpace: "pre-wrap", marginTop: 8 }}>
  358                     {String(n.text || "")}
  359                   </div>
  360                 </div>
  361               ))}
  362             </div>
  363           </div>
  364         </div>
  365       </div>
  366     </div>
  367   </div>
  368 </div>
```

```
362          </div>
363      </div>
364  ) : null}
365      </div>
366
367      <div className="panel" style={{ padding: 14, borderRadius:
14, marginTop: 12, width: "100%", maxWidth: 1000 }}>
368          <div style={{ fontWeight: 950, color: "#0f172a" }}>
>Estimate (public)</div>
369          {!est ? (
370              <div className="muted" style={{ fontWeight: 850,
marginTop: 6 }}>Estimating... (will populate as render previews complete)</
div>
371          ) : (
372              <div style={{ display: "grid", gap: 10, marginTop:
10 }}>
373                  <div className="row" style={{ gap: 10, flexWrap:
"wrap" }}>
374                      <span className="badge rate-bright">Estimated
Total: {money(est.total)}</span>
375                      {typeof est.rangeLow === "number" && typeof
est.rangeHigh === "number" ? (
376                          <span className="badge">Range:
{money(est.rangeLow)} - {money(est.rangeHigh)}</span>
377                          ) : null}
378                      </div>
379
380                  <div className="muted" style={{ fontWeight: 850 }}>
381                      Breakdown (customer-safe): Materials
{money(est.materials)} • Labor {money(est.labor)} • Finish
{money(est.finish)} • Overhead {money(est.overhead)}
382                      </div>
383                      </div>
384                  )
385                  </div>
386          </section>
387
388          <section className="stack" style={{ maxWidth: 1100, margin:
"0 auto", width: "100%" }}>
389              <div className="h3" style={{ margin: 0 }}>Render
Previews</div>
390              <div className="muted" style={{ fontWeight: 850 }}>
391                  One render runs at a time (queued → rendering →
complete). Each render has its own estimate box.
392              </div>
393
394              <div style={{ display: "grid", gridTemplateColumns:
"repeat(auto-fit, minmax(280px, 1fr))", gap: 12, marginTop: 10 }}>
395                  {(v.renders || []).map((r) => (
396                      <article key={r.renderId} className="panel card"
style={{ padding: 12, display: "grid", gap: 10 }}>
397                          <div style={{ fontWeight: 950, color: "#0f172a" }}>
398                              View: {String(r.view).toUpperCase()}
399                              <span className="badge" style={{ marginLeft: 8 }}>
{String(r.status).toUpperCase()}</span>
```

```
400             </div>
401
402         <div
403             style={{
404                 width: "100%",
405                 height: 180,
406                 borderRadius: 14,
407                 overflow: "hidden",
408                 border: "1px solid rgba(15,23,42,0.18)",
409                 background: "rgba(2,6,23,0.25)",
410             }}
411         >
412             {r.imageUrl ? (
413                 <img
414                     src={r.imageUrl}
415                     alt={`${r.view} render`}
416                     style={{ width: "100%", height: "100%",
objectFit: "cover", display: "block" }}
417                         />
418                     ) : (
419                         <div className="card-center" style={{ width:
"100%", height: "100%", padding: 12 }}>
420                             <div className="muted" style={{ fontWeight:
900 }}>
421                                 {r.status === "queued"
422                                     ? "Queued..."
423                                     : r.status === "rendering"
424                                     ? "Rendering..."
425                                     : r.status === "failed"
426                                     ? "Render failed (will re-run on refresh
later)"
427                                         : "No image"}
428                         </div>
429                     </div>
430                 )
431             </div>
432
433             <div className="panel" style={{ padding: 10,
borderRadius: 12 }}>
434                 <div className="label">Estimate for this render</
div>
435                 {!r.estimatePublic ? (
436                     <div className="muted" style={{ fontWeight: 850,
marginTop: 6 }}>
437                         {r.status === "complete" ? "Finalizing
estimate..." : "Waiting for render to complete..."}
438                     </div>
439                 ) : (
440                     <div style={{ display: "grid", gap: 8,
marginTop: 8 }}>
441                         <div className="badge rate-bright"
style={{ justifyContent: "center" }}>
442                             {r.estimatePublic.label || "Estimated
Total": {money(r.estimatePublic.total)}}
443                     </div>
```

```

444             {typeof r.estimatePublic.rangeLow === "number"
&& typeof r.estimatePublic.rangeHigh === "number" ? (
445                 <div className="muted" style={{ fontWeight:
850 }}>
446                     Range: {money(r.estimatePublic.rangeLow)}
- {money(r.estimatePublic.rangeHigh)}
447                         </div>
448                         ) : null}
449                         </div>
450                         )
451                     </div>
452
453                     <div className="muted" style={{ fontWeight: 850 }}>
454                         {r.startedAt ? `Started: ${fmt(r.startedAt)}` :
"Not started yet"}
455                         {r.finishedAt ? ` • Finished: ${fmt(r.finishedAt)}`
` : ""}
456                         </div>
457                     </article>
458                     ))
459                 </div>
460             </section>
461         </div>
462     );
463 }

```

===== src/lib/render3d.ts =====

```

1 import * as THREE from "three";
2 import type { BuildDims, BuildOptions } from "./buildsStore";
3
4 type View = "iso" | "front" | "top" | "detail";
5
6 function clamp(n: number, min: number, max: number) {
7     return Math.max(min, Math.min(max, n));
8 }
9
10 function safeIn(n: number, fallback: number) {
11     const v = Number(n);
12     return Number.isFinite(v) && v > 0 ? v : fallback;
13 }
14
15 function woodColor(species: BuildOptions["woodSpecies"]) {
16     // Simple readable tones (non-photoreal). Upgrade later with
textures.
17     switch (species) {
18         case "Pine": return 0xE6D2A6;
19         case "Poplar": return 0xD8E0A8;
20         case "Plywood": return 0xD9C7A3;
21         case "Oak": return 0xC8A06C;
22         case "Maple": return 0xEAD9B6;
23         case "Walnut": return 0x6B4A2E;
24         default: return 0xC8A06C;
25     }

```

```
26 }
27
28 function finishSheen(finish: BuildOptions["finish"]) {
29   if (finish === "Natural") return { roughness: 0.65, metalness:
0.02 };
30   if (finish === "Stain") return { roughness: 0.55, metalness:
0.03 };
31   if (finish === "Paint") return { roughness: 0.35, metalness:
0.01 };
32   if (finish === "Poly") return { roughness: 0.25, metalness:
0.04 };
33   return { roughness: 0.55, metalness: 0.03 };
34 }
35
36 function fitToView(length: number, depth: number, height: number)
{
37   const maxDim = Math.max(length, depth, height);
38   return clamp(maxDim * 0.95, 40, 320);
39 }
40
41 function makeCamera(view: View, frustumSize: number) {
42   const cam = new THREE.OrthographicCamera(
43     -frustumSize, frustumSize, frustumSize, -frustumSize,
44     0.1, 4000
45   );
46
47   if (view === "top") {
48     cam.position.set(0, 600, 0.001);
49   } else if (view === "front") {
50     cam.position.set(0, 220, 600);
51   } else if (view === "detail") {
52     cam.position.set(420, 280, 420);
53   } else {
54     cam.position.set(520, 360, 520); // iso
55   }
56
57   cam.lookAt(0, 0, 0);
58   cam.updateProjectionMatrix();
59   return cam;
60 }
61
62 function normType(t: string) {
63   return String(t || "").trim().toLowerCase();
64 }
65
66 function normNotes(n: string | undefined) {
67   return String(n || "").trim().toLowerCase();
68 }
69
70 function has(notes: string, ...phrases: string[]) {
71   return phrases.some((p) => notes.includes(p));
72 }
73
74 function addBox(
75   group: THREE.Group,
```

```

76   w: number,
77   h: number,
78   d: number,
79   x: number,
80   y: number,
81   z: number,
82   mat: THREE.Material
83 ) {
84   const geom = new THREE.BoxGeometry(w, h, d);
85   const mesh = new THREE.Mesh(geom, mat);
86   mesh.position.set(x, y, z);
87   group.add(mesh);
88   return geom;
89 }
90
91 /**
92  * Notes-driven features (simple heuristics, safe defaults):
93  * - "lower shelf", "bottom shelf", "shelf" -> adds a shelf panel
for table/bench/workbench
94  * - "apron" -> adds apron rails (unless notes include "no apron")
95  * - "drawer" -> adds a simple drawer box under top (front-facing)
96  * - "taper" / "tapered legs" -> visually tapers legs using mesh
scaling
97  * - "feet" -> adds small feet blocks for planter/cabinet
98  *
99  * This is NOT photoreal; it's an improving proxy model based on
customer intent.
100 */
101 function buildModel(args: {
102   projectType: string;
103   dims: BuildDims;
104   options: BuildOptions;
105   notes?: string;
106 }) {
107   const group = new THREE.Group();
108   const notes = normNotes(args.notes);
109
110   // Interpret dims consistently:
111   // lengthIn => X (long)
112   // widthIn  => Z (depth)
113   // heightIn => Y (overall height)
114   const length = clamp(safeIn(args.dims.lengthIn, 60), 12, 240);
115   const depth  = clamp(safeIn(args.dims.widthIn, 30), 10, 240);
116   const height = clamp(safeIn(args.dims.heightIn, 30), 10, 240);
117
118   const topThickness = clamp(safeIn(args.dims.topThicknessIn ??
1.5, 1.5), 0.5, 6);
119
120   const sheen = finishSheen(args.options.finish);
121   const woodMat = new THREE.MeshStandardMaterial({
122     color: woodColor(args.options.woodSpecies),
123     roughness: sheen.roughness,
124     metalness: sheen.metalness,
125   });
126

```

```

127     const darkMat = new THREE.MeshStandardMaterial({
128         color: 0x0f172a,
129         roughness: 0.85,
130         metalness: 0.05,
131     });
132
133     const t = normType(args.projectType);
134
135     // Shared thickness defaults (inches-as-units)
136     const boardT = clamp(Math.min(depth, length) * 0.035, 0.6,
1.25); // panel/board thickness
137     const legSize = clamp(Math.min(depth, length) * 0.06, 1.5, 4.0);
138
139     // Helpers
140     const xMin = -length / 2;
141     const xMax = length / 2;
142     const zMin = -depth / 2;
143     const zMax = depth / 2;
144
145     const geoms: THREE.BufferGeometry[] = [];
146
147     const isTable = t.includes("table");
148     const isBench = t.includes("bench");
149     const isWorkbench = t.includes("workbench");
150
151     // --- TABLE / BENCH / WORKBENCH (top + 4 legs) ---
152     if (isTable || isBench || isWorkbench) {
153         const topY = height - topThickness / 2;
154
155         // Top
156         geoms.push(addBox(group, length, topThickness, depth, 0, topY,
0, woodMat));
157
158         // Legs
159         const legH = Math.max(2, height - topThickness);
160         const inset = clamp(legSize * 0.65, 1.25, 4.5);
161
162         const lx1 = xMin + inset + legSize / 2;
163         const lx2 = xMax - inset - legSize / 2;
164         const lz1 = zMin + inset + legSize / 2;
165         const lz2 = zMax - inset - legSize / 2;
166
167         const legY = legH / 2;
168
169         const legGeom = new THREE.BoxGeometry(legSize, legH, legSize);
170
171         function addLeg(x: number, z: number) {
172             const mesh = new THREE.Mesh(legGeom, darkMat);
173             mesh.position.set(x, legY, z);
174
175             // Notes: tapered legs (visual taper using non-uniform
scale)
176             if (has(notes, "taper", "tapered leg", "tapered legs")) {
177                 // slightly narrower at the top by scaling X/Z a bit
(simple proxy)

```

```

178         // Using scale affects the whole mesh uniformly; we fake
taper by scaling and adding a small "cap"
179         mesh.scale.set(0.88, 1, 0.88);
180         const cap = new THREE.Mesh(new THREE.BoxGeometry(legSize *
0.92, legSize * 0.18, legSize * 0.92), darkMat);
181         cap.position.set(x, legH - (legSize * 0.09), z);
182         group.add(cap);
183         geoms.push(cap.geometry as THREE.BufferGeometry);
184     }
185
186     group.add(mesh);
187 }
188
189     addLeg(lx1, lz1);
190     addLeg(lx2, lz1);
191     addLeg(lx1, lz2);
192     addLeg(lx2, lz2);
193
194     geoms.push(legGeom);
195
196     // Notes: apron rails (default on workbench, optional on
table/bench)
197     const wantsApron = isWorkbench || (has(notes, "apron") && !
has(notes, "no apron", "noapron"));
198     if (wantsApron) {
199         const apronH = clamp(height * 0.12, 2, 6);
200         const apronY = height - topThickness - apronH / 2;
201
202         // Front/back rails (along X)
203         geoms.push(addBox(group, length - inset * 2, apronH, boardT,
0, apronY, zMax - inset - boardT / 2, darkMat));
204         geoms.push(addBox(group, length - inset * 2, apronH, boardT,
0, apronY, zMin + inset + boardT / 2, darkMat));
205
206         // Left/right rails (along Z)
207         geoms.push(addBox(group, boardT, apronH, depth - inset * 2,
xMin + inset + boardT / 2, apronY, 0, darkMat));
208         geoms.push(addBox(group, boardT, apronH, depth - inset * 2,
xMax - inset - boardT / 2, apronY, 0, darkMat));
209     }
210
211     // Notes: lower shelf / bottom shelf
212     const wantsShelf = isWorkbench || has(notes, "lower shelf",
"bottom shelf") || (has(notes, "shelf") && !has(notes, "no shelf"));
213     if (wantsShelf) {
214         const shelfY = clamp(legH * 0.28, 6, legH - 8);
215         const shelfT = clamp(boardT, 0.6, 1.5);
216         geoms.push(addBox(group, length - inset * 2 - legSize * 0.2,
shelfT, depth - inset * 2 - legSize * 0.2, 0, shelfY, 0, woodMat));
217     }
218
219     // Notes: drawer (simple centered drawer under top, front-
facing)
220     if (has(notes, "drawer", "drawers")) {
221         const drawerH = clamp(height * 0.18, 3, 8);

```

```

222     const drawerW = clamp(length * 0.35, 10, length - 10);
223     const drawerD = clamp(depth * 0.45, 8, depth - 6);
224     const drawerY = height - topThickness - drawerH / 2 - 1.2;
225     const drawerZ = zMax - drawerD / 2 - 1.2;
226
227     geoms.push(addBox(group, drawerW, drawerH, drawerD, 0,
drawerY, drawerZ, darkMat));
228
229     // drawer face
230     const faceT = clamp(boardT * 0.8, 0.4, 1.1);
231     geoms.push(addBox(group, drawerW * 0.96, drawerH * 0.92,
faceT, 0, drawerY, zMax - 0.6, woodMat));
232 }
233
234     // Simple stretcher for workbench feel
235     if (isWorkbench) {
236         const stretcherH = clamp(legSize * 0.45, 0.8, 2.0);
237         const stretcherY = clamp(legH * 0.35, 6, legH - 4);
238         geoms.push(addBox(group, length - inset * 2 - legSize,
stretcherH, legSize * 0.6, 0, stretcherY, 0, darkMat));
239     }
240 }
241
242 // --- SHELF (two uprights + shelves) ---
243 else if (t.includes("shelf")) {
244     const sideT = boardT;
245     const shelfT = boardT;
246
247     // Uprights (left/right)
248     const upH = height;
249     const upY = upH / 2;
250     const upX = length / 2 - sideT / 2;
251
252     geoms.push(addBox(group, sideT, upH, depth, -upX, upY, 0,
woodMat));
253     geoms.push(addBox(group, sideT, upH, depth, upX, upY, 0,
woodMat));
254
255     // Shelves: bottom, mid, top
256     const insideL = Math.max(8, length - sideT * 2);
257     const shelfCount = has(notes, "5 shelf", "five shelf") ? 5 :
has(notes, "4 shelf", "four shelf") ? 4 : 3;
258
259     for (let i = 0; i < shelfCount; i++) {
260         const frac = (i + 1) / (shelfCount + 1);
261         const y = clamp(frac * (height - shelfT) + shelfT / 2,
shelfT / 2, height - shelfT / 2);
262         geoms.push(addBox(group, insideL, shelfT, depth, 0, y, 0,
woodMat));
263     }
264 }
265
266 // --- CABINET (box with shelf + back) ---
267 else if (t.includes("cabinet")) {
268     const wallT = clamp(boardT, 0.6, 1.25);

```

```

269     const shelfT = wallT;
270     const insideL = Math.max(10, length - wallT * 2);
271     const insideD = Math.max(8, depth - wallT * 2);
272
273     geoms.push(addBox(group, length, wallT, depth, 0, wallT / 2,
274                         0, woodMat)); // bottom
274     geoms.push(addBox(group, length, wallT, depth, 0, height -
275                         wallT / 2, 0, woodMat)); // top
275     geoms.push(addBox(group, wallT, height, depth, -length / 2 +
276                         wallT / 2, height / 2, 0, woodMat)); // left
276     geoms.push(addBox(group, wallT, height, depth, length / 2 -
277                         wallT / 2, height / 2, 0, woodMat)); // right
277     geoms.push(addBox(group, length, height, wallT, 0, height / 2,
278                         -depth / 2 + wallT / 2, woodMat)); // back
278
279     // Shelf count from notes
280     const shelves = has(notes, "2 shelf", "two shelf") ? 2 :
281     has(notes, "3 shelf", "three shelf") ? 3 : 1;
281     for (let i = 0; i < shelves; i++) {
282         const frac = (i + 1) / (shelves + 1);
283         const y = clamp(frac * (height - shelfT) + shelfT / 2,
284                         shelfT / 2, height - shelfT / 2);
284         geoms.push(addBox(group, insideL, shelfT, insideD, 0, y,
285                         wallT / 2, woodMat));
285     }
286
287     // Notes: feet
288     if (has(notes, "feet", "legs")) {
289         const foot = clamp(wallT * 1.2, 0.8, 2.2);
290         const fy = foot / 2;
291         geoms.push(addBox(group, foot, foot, foot, xMin + foot, fy,
292                         zMin + foot, darkMat));
292         geoms.push(addBox(group, foot, foot, foot, xMax - foot, fy,
293                         zMin + foot, darkMat));
293         geoms.push(addBox(group, foot, foot, foot, xMin + foot, fy,
294                         zMax - foot, darkMat));
294         geoms.push(addBox(group, foot, foot, foot, xMax - foot, fy,
295                         zMax - foot, darkMat));
295     }
296 }
297
298 // --- PLANTER BOX (open top, 4 walls + bottom) ---
299 else if (t.includes("planter")) {
300     const wallT = clamp(boardT, 0.6, 1.5);
301     const bottomT = wallT;
302
303     geoms.push(addBox(group, length, bottomT, depth, 0, bottomT /
304                         2, 0, woodMat)); // bottom
304
305     const wallH = Math.max(8, height - bottomT);
306     const wallY = bottomT + wallH / 2;
307
308     // front/back
309     geoms.push(addBox(group, length, wallH, wallT, 0, wallY,
depth / 2 - wallT / 2, woodMat));

```

```

310     geoms.push(addBox(group, length, wallH, wallT, 0, wallY,
-depth / 2 + wallT / 2, woodMat));
311
312     // left/right
313     geoms.push(addBox(group, wallT, wallH, depth - wallT * 2,
-length / 2 + wallT / 2, wallY, 0, woodMat));
314     geoms.push(addBox(group, wallT, wallH, depth - wallT * 2,
length / 2 - wallT / 2, wallY, 0, woodMat));
315
316     // Notes: feet
317     if (has(notes, "feet", "legs", "stand")) {
318         const foot = clamp(wallT * 1.25, 0.8, 2.4);
319         const fy = foot / 2;
320         geoms.push(addBox(group, foot, foot, foot, xMin + foot, fy,
zMin + foot, darkMat));
321         geoms.push(addBox(group, foot, foot, foot, xMax - foot, fy,
zMin + foot, darkMat));
322         geoms.push(addBox(group, foot, foot, foot, xMin + foot, fy,
zMax - foot, darkMat));
323         geoms.push(addBox(group, foot, foot, foot, xMax - foot, fy,
zMax - foot, darkMat));
324     }
325 }
326
327     // --- DEFAULT (fallback block) ---
328     else {
329         geoms.push(addBox(group, length, height, depth, 0, height / 2,
0, woodMat));
330     }
331
332     return { group, length, depth, height, geoms };
333 }
334
335 export async function renderBuildPreviewPng(args: {
336     view: View;
337     projectType: string;
338     title?: string;
339     notes?: string;
340     dims: BuildDims;
341     options: BuildOptions;
342     width?: number;
343     height?: number;
344 }) {
345     const W = args.width ?? 1200;
346     const H = args.height ?? 800;
347
348     // Scene
349     const scene = new THREE.Scene();
350     scene.background = new THREE.Color(0x0b1220);
351
352     // Lights
353     const hemi = new THREE.HemisphereLight(0xBBD7FF, 0x101827,
0.95);
354     scene.add(hemi);
355 }
```

```

356     const key = new THREE.DirectionalLight(0xFFFFFFF, 1.05);
357     key.position.set(240, 280, 200);
358     scene.add(key);
359
360     const rim = new THREE.DirectionalLight(0xA78BFA, 0.55);
361     rim.position.set(-260, 160, -200);
362     scene.add(rim);
363
364     // Model (now notes-aware)
365     const model = buildModel({
366         projectType: args.projectType,
367         dims: args.dims,
368         options: args.options,
369         notes: args.notes,
370     });
371     scene.add(model.group);
372
373     // Ground
374     const ground = new THREE.Mesh(
375         new THREE.PlaneGeometry(1200, 1200),
376         new THREE.MeshStandardMaterial({ color: 0x0f1a2f, roughness:
1, metalness: 0 })
377     );
378     ground.rotation.x = -Math.PI / 2;
379     ground.position.y = 0;
380     scene.add(ground);
381
382     const grid = new THREE.GridHelper(1000, 32, 0x334155, 0x1f2a44);
383     (grid.material as THREE.Material).transparent = true;
384     (grid.material as THREE.Material).opacity = 0.25;
385     scene.add(grid);
386
387     // Camera
388     const frustumSize = fitToView(model.length, model.depth,
model.height);
389     const camera = makeCamera(args.view, frustumSize);
390
391     // Renderer (offscreen)
392     const canvas = document.createElement("canvas");
393     const renderer = new THREE.WebGLRenderer({
394         canvas,
395         antialias: true,
396         alpha: false,
397         preserveDrawingBuffer: true,
398     });
399
400     renderer.setPixelRatio(Math.min(window.devicePixelRatio || 1,
2));
401     renderer.setSize(W, H, false);
402
403     // Ortho aspect correction
404     const aspect = W / H;
405     const ortho = camera as THREE.OrthographicCamera;
406     const s = frustumSize;
407     ortho.left = -s * aspect;

```

```
408     ortho.right = s * aspect;
409     ortho.top = s;
410     ortho.bottom = -s;
411     ortho.updateProjectionMatrix();
412
413     renderer.render(scene, camera);
414
415     // Cleanup
416     model.geoms.forEach((g) => g.dispose());
417     (ground.geometry as THREE.BufferGeometry).dispose();
418     (ground.material as THREE.Material).dispose();
419     (grid.geometry as THREE.BufferGeometry).dispose();
420     (grid.material as THREE.Material).dispose();
421     renderer.dispose();
422
423     return canvas.toDataURL("image/png");
424 }
```