PROJECT TREE

```
3d-print-frontend/
├── .gitignore
├── README.md
├── codebook.pdf
├── eslint.config.js
├── index.html
├── package.json
├── postcss.config.js
├── public
│   └── vite.svg
├── repo_doc_to_pdf.py
├── src
│   ├── app
│   │   ├── App.tsx
│   │   ├── layout
│   │   │   ├── NavBar.tsx
│   │   │   └── Page.tsx
│   │   ├── main.tsx
│   │   ├── routes.tsx
│   │   └── vite-env.d.ts
│   ├── assets
│   │   └── react.svg
│   ├── features
│   │   ├── auth
│   │   │   └── pages
│   │   │       └── LoginPage.tsx
│   │   ├── cart
│   │   │   └── pages
│   │   │       └── CartPage.tsx
│   │   ├── order
│   │   │   └── pages
│   │   │       ├── CheckoutCancelPage.tsx
│   │   │       ├── CheckoutSuccessPage.tsx
│   │   │       ├── OrderStatusPage.tsx
│   │   │       └── OrdersPage.tsx
│   │   ├── quote
│   │   │   └── pages
│   │   │       └── QuotePage.tsx
│   │   ├── upload
│   │   │   └── pages
│   │   │       └── UploadPage.tsx
│   │   └── workshop
│   │       └── pages
│   │           ├── WorkshopPage.tsx
│   │           └── WorkshopRoute.tsx
│   └── shared
│       ├── api
│       │   ├── client.ts
│       │   ├── endpoints.ts
│       │   └── types.ts
│       ├── auth
│       │   ├── AdminOnly.tsx
│       │   └── role.ts
│       ├── hooks
│       │   └── useAuth.ts
│       ├── i18n
│       │   └── LocaleContext.tsx
│       └── ui
│           ├── AuthImage.tsx
│           ├── Button.tsx
│           ├── ConfirmDialog.tsx
│           ├── Input.tsx
│           ├── LanguageSwitcher.tsx
│           ├── Modal.tsx
```

PROJECT TREE

```
│           ├── ProgressBar.tsx
│           ├── Select.tsx
│           ├── Toast.tsx
│           └── useModal.ts
├── styles
│   └── index.css
├── tailwind.config.js
├── tsconfig.app.json
├── tsconfig.json
├── tsconfig.node.json
├── vite.config.ts
└── Маршрути.MD
```

.gitignore

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*

node_modules
dist
dist-ssr
*.local

# Editor directories and files
.vscode/*
!.vscode/extensions.json
.idea
.DS_Store
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?
/public/
/Маршрути.MD
```

```
stripe listen --forward-to localhost:8000/v1/payments/webhook
```

4242 4242 4242 4242

eslint.config.js

```
import js from '@eslint/js'
import globals from 'globals'
import reactHooks from 'eslint-plugin-react-hooks'
import reactRefresh from 'eslint-plugin-react-refresh'
import tseslint from 'typescript-eslint'
import { globalIgnores } from 'eslint/config'

export default tseslint.config([
  globalIgnores(['dist']),
  {
    files: ['**/*.{ts,tsx}'],
    extends: [
      js.configs.recommended,
      tseslint.configs.recommended,
      reactHooks.configs['recommended-latest'],
      reactRefresh.configs.vite,
    ],
    languageOptions: {
      ecmaVersion: 2020,
      globals: globals.browser,
    },
  },
])
```

index.html

```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React + TS</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/app/main.tsx"></script>
  </body>
</html>
```

package.json

```json
{
  "name": "3d-print-frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^19.1.1",
    "react-dom": "^19.1.1"
  },
  "devDependencies": {
    "@eslint/js": "^9.33.0",
    "@tailwindcss/cli": "^4.1.12",
    "@tailwindcss/postcss": "^4.1.12",
    "@types/node": "^24.2.1",
    "@types/react": "^19.1.10",
    "@types/react-dom": "^19.1.7",
    "@vitejs/plugin-react": "^5.0.0",
    "autoprefixer": "^10.4.21",
    "eslint": "^9.33.0",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.20",
    "globals": "^16.3.0",
    "postcss": "^8.5.6",
    "react-router-dom": "^7.8.0",
    "tailwindcss": "^4.1.12",
    "typescript": "~5.8.3",
    "typescript-eslint": "^8.39.1",
    "vite": "^7.1.2"
  }
}
```

postcss.config.js

```
export default {
    plugins: {
        '@tailwindcss/postcss': {},
        autoprefixer: {},
    },
}
```

repo_doc_to_pdf.py

```
"""
Usage:
  python repo_doc_to_pdf.py --root . --out codebook.pdf --max-bytes 800000 --wrap 100
  --root — корінь проєкту
  --out — шлях до PDF
  --max-bytes — великі файли пропускаються
  --wrap — ширина переносу рядків у символах
"""
import os, sys, argparse, textwrap, pathlib, unicodedata

import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages

try:
    import chardet
except Exception:
    chardet = None

# Безпечно відрубуємо TeX/MathText
mpl.rcParams["text.usetex"] = False
mpl.rcParams["mathtext.default"] = "rm"

EXCLUDE_DIRS = {
    ".git", ".venv", "__pycache__", ".idea", ".mypy_cache", ".pytest_cache",
    "node_modules", "dist", "build", ".DS_Store"
}
EXCLUDE_EXTS = {
    ".png", ".jpg", ".jpeg", ".webp", ".gif", ".ico", ".pdf",
    ".zip", ".tar", ".gz", ".7z", ".mp4", ".mov", ".mp3", ".wav"
}

EXCLUDE_FILES = {".env", "celerybeat-schedule","package-lock.json"}

def is_text_file(path: str) -> bool:
    ext = pathlib.Path(path).suffix.lower()
    if ext in EXCLUDE_EXTS:
        return False
    try:
        with open(path, "rb") as f:
            chunk = f.read(4096)
        if not chunk:
            return True
        # справжній нуль-байт
        if b"\x00" in chunk:
            return False
        return True
    except Exception:
        return False

def detect_encoding(data: bytes) -> str:
    if chardet:
        try:
            enc = chardet.detect(data).get("encoding")
            if enc:
                return enc
        except Exception:
            pass
    return "utf-8"

def read_text(path: str, max_bytes: int) -> str:
    with open(path, "rb") as f:
        data = f.read()
    if len(data) > max_bytes:
        return f"[SKIPPED: file too large ({len(data)} bytes)]"
    enc = detect_encoding(data)
    try:
        return data.decode(enc, errors="replace")
    except Exception:
        try:
            return data.decode("utf-8", errors="replace")
        except Exception:
            return data.decode("latin-1", errors="replace")
```

```
def walk_files(root: str):
    for dirpath, dirnames, filenames in os.walk(root):
        dirnames[:] = [d for d in dirnames if d not in EXCLUDE_DIRS and not d.startswith(".tox")]
        for fn in sorted(filenames):
            if fn in EXCLUDE_FILES:
                continue
            full = os.path.join(dirpath, fn)
            rel = os.path.relpath(full, root)
            if not is_text_file(full):
                continue
            yield rel, full

def build_tree_text(root: str) -> str:
    lines = []
    base = os.path.basename(os.path.abspath(root)) or root
    lines.append(f"{base}/")
    def _print_tree(startpath, prefix=""):
        try:
            items = sorted(
                [n for n in os.listdir(startpath)
                 if n not in EXCLUDE_DIRS and n not in EXCLUDE_FILES]
            )
        except FileNotFoundError:
            return
        for i, name in enumerate(items):
            path = os.path.join(startpath, name)
            connector = "└── " if i == len(items) - 1 else "├── "
            lines.append(prefix + connector + name)
            if os.path.isdir(path):
                extension = "    " if i == len(items) - 1 else "│   "
                _print_tree(path, prefix + extension)
    _print_tree(root)
    return "\n".join(lines)

def sanitize_text(s: str) -> str:
    """Нормалізація для Matplotlib PDF:
    - заміна всіх '□' → '□' (U+FF04), щоб повністю вимкнути mathtext
    - дроп NULL/контрольні (крім \t \n \r), variation selectors, ZWJ/ZWNJ
    - дроп не-BMP (емодзі), щоб бекенд PDF не падав
    - маппінг деяких проблемних гліфів на ASCII
    """
    if not s:
        return s
    # повністю вимкнути mathtext
    s = s.replace('□', '□')

    out = []
    for ch in s:
        code = ord(ch)
        if ch == '\x00':
            continue
        cat = unicodedata.category(ch)
        if cat.startswith('C') and ch not in ('\t', '\n', '\r'):
            continue
        if code in (0xFE0F, 0x200D, 0x200C):  # VS16, ZWJ, ZWNJ
            continue
        if code > 0xFFFF:  # emoji / non-BMP
            continue
        if ch in {'v', 'v'}:
            ch = 'v'
        out.append(ch)
    return ''.join(out)

def add_text_pages(pdf: PdfPages, title: str, text: str, wrap_width=100, font_size=9,
header_size=11):
    # sanitize early
    title = sanitize_text(title)
    text = sanitize_text(text)

    # A4 portrait
    page_w, page_h = 8.27, 11.69
    left_margin, right_margin, top_margin, bottom_margin = 0.5, 0.5, 0.7, 0.7
    usable_height = page_h - top_margin - bottom_margin

    # wrap text
    wrapped_lines = []
    for line in text.splitlines():
        line = sanitize_text(line.expandtabs(4))
        wrapped_lines.extend(textwrap.wrap(line, width=wrap_width, replace_whitespace=False) or
[""])

    # simple line-height calc
    line_height_in = (font_size * 1.2) / 72.0
    lines_per_page = max(1, int(usable_height / line_height_in) - 4)

    # render
    for page_idx in range(0, len(wrapped_lines) or 1, lines_per_page):
        fig = plt.figure(figsize=(page_w, page_h))
        ax = fig.add_axes([0, 0, 1, 1])
        ax.axis("off")
        # header
        ax.text(0.5, 1 - top_margin / page_h + 0.02, title,
                ha="center", va="top", fontsize=header_size, family="monospace")
        # body
        chunk = wrapped_lines[page_idx: page_idx + lines_per_page]
        body_text = "\n".join(chunk) if chunk else ""
        ax.text(left_margin / page_w, 1 - (top_margin + 0.3) / page_h,
                body_text, ha="left", va="top", fontsize=font_size, family="monospace")
        pdf.savefig(fig, bbox_inches="tight")
        plt.close(fig)

def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--root", default=".", help="Project root directory")
    ap.add_argument("--out", default="codebook.pdf", help="Output PDF path")
    ap.add_argument("--max-bytes", type=int, default=800000, help="Skip files larger than this many
bytes")
    ap.add_argument("--wrap", type=int, default=100, help="Characters per line for wrapping")
    args = ap.parse_args()

    tree_text = build_tree_text(args.root)
    with PdfPages(args.out) as pdf:
        add_text_pages(pdf, "PROJECT TREE", tree_text, wrap_width=args.wrap, font_size=9,
header_size=12)
        for rel, full in walk_files(args.root):
            if not is_text_file(full):
                continue
            try:
                content = read_text(full, args.max_bytes)
            except Exception as e:
                content = f"[ERROR reading file: {e}]"
            add_text_pages(pdf, rel, content, wrap_width=args.wrap, font_size=8, header_size=10)

if __name__ == "__main__":
    main()
```

tailwind.config.js

```js
/** @type {import('tailwindcss').Config} */
export default {
    content: ["./index.html", "./src/**/*.{ts,tsx,js,jsx}"],
    theme: { extend: {} },
    plugins: [],
}
```

tsconfig.app.json

```
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.app.tsbuildinfo",
    "target": "ES2022",
    "useDefineForClassFields": true,
    "lib": ["ES2022", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "skipLibCheck": true,

    /* Bundler mode */
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "verbatimModuleSyntax": true,
    "moduleDetection": "force",
    "noEmit": true,
    "jsx": "react-jsx",

    /* Linting */
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "erasableSyntaxOnly": true,
    "noFallthroughCasesInSwitch": true,
    "noUncheckedSideEffectImports": true,

    "baseUrl": ".",
    "paths": {
      "@/*": ["src/*"]
    },
    "types": ["vite/client", "node"]
  },
  "include": ["src"]
}
```

tsconfig.json

```
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": { "@/*": ["src/*"] },
    "types": ["vite/client", "node"],
    "moduleResolution": "bundler"
  },
  "files": [],
  "references": [
    { "path": "./tsconfig.app.json" },
    { "path": "./tsconfig.node.json" }
  ]
}
```

tsconfig.node.json

```
{
  "compilerOptions": {
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.node.tsbuildinfo",
    "target": "ES2023",
    "lib": ["ES2023"],
    "module": "ESNext",
    "skipLibCheck": true,

    /* Bundler mode */
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "verbatimModuleSyntax": true,
    "moduleDetection": "force",
    "noEmit": true,

    /* Linting */
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "erasableSyntaxOnly": true,
    "noFallthroughCasesInSwitch": true,
    "noUncheckedSideEffectImports": true
  },
  "include": ["vite.config.ts"]
}
```

vite.config.ts

```ts
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import { fileURLToPath, URL } from 'node:url'

export default defineConfig({
    plugins: [react()],
    resolve: {
        alias: { '@': fileURLToPath(new URL('./src', import.meta.url)) },
    },
})
```

Маршрути.MD

```
/                       → Welcome (простий екран)
/login                  → LoginPage
/upload                 → UploadPage
/quote                  → QuotePage (?model=ID)
/cart                   → CartPage
/order                  → OrderStatusPage
/checkout/success       → CheckoutSuccessPage (?session_id=...)
/checkout/cancel        → CheckoutCancelPage
/workshop               → WorkshopPage (admin, guard)

API-контракти (під фронт)

Auth
POST /v1/auth/login { email, password } -> { access_token }

Files
POST /v1/files/presign-upload { content_type } -> { url, key, bucket }
PUT <url> (file, header Content-Type)
POST /v1/files/complete { key, content_type } -> { upload, model }

Materials
GET /v1/materials -> [{ id, name, ...}] (для селектора матеріалу)

Quotes
POST /v1/quotes { model_id, material_id, layer_height, infill, qty } -> { id, price_eur, breakdown }
POST /v1/cart/add/{quote_id} { qty? } -> { item_id, qty }

Cart/Orders/Payments
GET  /v1/cart -> {...} (необов’язково, для інспекції)
POST /v1/orders/checkout -> { order_id, total_eur, status }
POST /v1/payments/checkout { order_id } -> { checkout_url, session_id }

Orders/PrintJobs
GET /v1/orders/{id} -> { id, status, total_eur, ... }
GET /v1/orders/{id}/print-jobs -> [{ id, status, progress, ...}]

(admin) GET /v1/print-jobs -> [...]
(admin) POST /v1/print-jobs/{id}/preflight
(admin) POST /v1/print-jobs/{id}/start|pause|resume|cancel
```

public\vite.svg

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" aria-
hidden="true" role="img" class="iconify iconify--logos" width="31.88" height="32"
preserveAspectRatio="xMidYMid meet" viewBox="0 0 256 257"><defs><linearGradient
id="IconifyId1813088fe1fbc01fb466" x1="-.828%" x2="57.636%" y1="7.652%" y2="78.411%"><stop
offset="0%" stop-color="#41D1FF"></stop><stop offset="100%" stop-
color="#BD34FE"></stop></linearGradient><linearGradient id="IconifyId1813088fe1fbc01fb467"
x1="43.376%" x2="50.316%" y1="2.242%" y2="89.03%"><stop offset="0%" stop-
color="#FFEA83"></stop><stop offset="8.333%" stop-color="#FFDD35"></stop><stop offset="100%" stop-
color="#FFA800"></stop></linearGradient></defs><path fill="url(#IconifyId1813088fe1fbc01fb466)"
d="M255.153 37.938L134.897 252.976c-2.483 4.44-8.862 4.466-11.382.048L.875 37.958c-2.746-4.814
1.371-10.646 6.827-9.67l120.385 21.517a6.537 6.537 0 0 0 2.322-.004l117.867-21.483c5.438-.991 9.574
4.796 6.877 9.62Z"></path><path fill="url(#IconifyId1813088fe1fbc01fb467)" d="M185.432.063L96.44
17.501a3.268 3.268 0 0 0-2.634 3.014l-5.474 92.456a3.268 3.268 0 0 0 3.997
3.378l24.777-5.718c2.318-.535 4.413 1.507 3.936 3.838l-7.361 36.047c-.495 2.426 1.782 4.5 4.151
3.78l15.304-4.649c2.372-.72 4.652 1.36 4.15 3.788l-11.698 56.621c-.732 3.542 3.979 5.473 5.943
2.437l1.313-2.028l72.516-144.72c1.215-2.423-.88-5.186-3.54-4.672l-25.505 4.922c-2.396.462-4.435-
1.77-3.759-4.114l16.646-57.705c.677-2.35-1.37-4.583-3.769-4.113Z"></path></svg>
```

src\app\App.tsx

```
import { Routes, Route, Navigate } from "react-router-dom";
import NavBar from "./layout/NavBar";
import Page from "./layout/Page";
import { routes } from "./routes";

export default function App() {
    return (
        <div className="min-h-dvh bg-neutral-50 text-neutral-900">
            <NavBar />
            <main className="max-w-5xl mx-auto px-4 py-8">
                <Routes>
                    {/* Головна-«привітання» */}
                    <Route path="/" element={<Page title="3D-Print Shop MVP">
                        <div className="space-y-2">
                            <p>Ласкаво просимо. Пройдімо шлях: Upload → Quote → Cart/Checkout →
Order Status → Workshop.</p>
                            <p className="text-sm text-neutral-600">Спочатку залогінься.</p>
                        </div>
                    </Page>} />

                    {/* Декларації сторінок із routes.tsx */}
                    {routes.map(r => (
                        <Route key={r.path} path={r.path} element={<r.element />} />
                    ))}

                    {/* запасний редірект */}
                    <Route path="*" element={<Navigate to="/" replace />} />
                </Routes>
            </main>
        </div>
    );
}
```

src\app\main.tsx

```tsx
import React from "react";
import ReactDOM from "react-dom/client";
import {BrowserRouter} from "react-router-dom";
import App from "./App";
import "../../styles/index.css";
import {LocaleProvider} from "@/shared/i18n/LocaleContext";

ReactDOM.createRoot(document.getElementById("root")!).render(
    <React.StrictMode>
        <BrowserRouter>
            <LocaleProvider>
                <App/>
            </LocaleProvider>
        </BrowserRouter>
    </React.StrictMode>
);
```

src\app\routes.tsx

```
import LoginPage from "@/features/auth/pages/LoginPage";
import UploadPage from "@/features/upload/pages/UploadPage";
import QuotePage from "@/features/quote/pages/QuotePage";
import CartPage from "@/features/cart/pages/CartPage";
import CheckoutSuccessPage from "@/features/order/pages/CheckoutSuccessPage";
import CheckoutCancelPage from "@/features/order/pages/CheckoutCancelPage";
import OrdersPage from "@/features/order/pages/OrdersPage";
import OrderStatusPage from "@/features/order/pages/OrderStatusPage";
import WorkshopRoute from "@/features/workshop/pages/WorkshopRoute";

export const routes = [
    { path: "/login",    element: LoginPage },
    { path: "/upload", element: UploadPage },
    { path: "/quote", element: QuotePage },
    { path: "/cart", element: CartPage },
    { path: "/checkout/success", element: CheckoutSuccessPage },
    { path: "/checkout/cancel", element: CheckoutCancelPage },
    { path: "/orders", element: OrdersPage },
    { path: "/orders/:id", element: OrderStatusPage },
    { path: "/workshop", element: WorkshopRoute },
] as const;
```

src\app\vite-env.d.ts

```
/// <reference types="vite/client" />
```

src\app\layout\NavBar.tsx

```
import { Link, NavLink, useNavigate } from "react-router-dom";
import LanguageSwitcher from "@/shared/ui/LanguageSwitcher";
import { isAdmin } from "@/shared/auth/role";

function Item({ to, children }: { to: string; children: React.ReactNode }) {
    return (
        <NavLink
            to={to}
            className={({ isActive }) =>
                "px-3 py-1.5 rounded-md text-sm " +
                (isActive ? "bg-black text-white" : "hover:bg-neutral-200")
            }
        >
            {children}
        </NavLink>
    );
}

export default function NavBar() {
    const nav = useNavigate();
    const token = localStorage.getItem("token");

    const logout = () => {
        localStorage.removeItem("token");
        nav("/login");
    };

    return (
        <header className="border-b bg-white">
            <div className="max-w-5xl mx-auto px-4 h-14 flex items-center justify-between">
                <Link to="/" className="font-semibold">3D-Print Shop</Link>
                <nav className="flex items-center gap-2">
                    <Item to="/upload">Upload</Item>
                    <Item to="/quote">Quote</Item>
                    <Item to="/cart">Cart</Item>
                    <Item to="/orders">Order</Item>
                    {isAdmin() && <Item to="/workshop">Workshop</Item>}
                </nav>
                <div className="ml-auto flex items-center gap-3">
                    <LanguageSwitcher />
                </div>
                <div className="flex items-center gap-2">
                    {!token ? (
                        <Item to="/login">Login</Item>
                    ) : (
                        <button onClick={logout} className="px-3 py-1.5 rounded-md text-sm bg-
neutral-900 text-white">
                            Logout
                        </button>
                    )}
                </div>
            </div>
        </header>
    );
}
```

src\app\layout\Page.tsx

```
export default function Page({ title, children }:{
    title?: string; children: React.ReactNode;
}) {
    return (
        <section className="bg-white rounded-2xl border p-6 shadow-sm">
            {title && <h1 className="text-xl font-semibold mb-4">{title}</h1>}
            {children}
        </section>
    );
}
```

src\assets\react.svg

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" aria-
hidden="true" role="img" class="iconify iconify--logos" width="35.93" height="32"
preserveAspectRatio="xMidYMid meet" viewBox="0 0 256 228"><path fill="#00D8FF" d="M210.483
73.824a171.49 171.49 0 0 0-8.24-2.597c.465-1.9.893-3.777 1.273-5.621c6.238-30.281
2.16-54.676-11.769-62.708c-13.355-7.7-35.196.329-57.254 19.526a171.23 171.23 0 0 0-6.375
5.848a155.866 155.866 0 0 0-4.241-3.917C100.759 3.829 77.587-4.822 63.673 3.233C50.33 10.957 46.379
33.89 51.995 62.588a170.974 170.974 0 0 0 1.892 8.48c-3.28.932-6.445 1.924-9.474 2.98C17.309 83.498
0 98.307 0 113.668c0 15.865 18.582 31.778 46.812 41.427a145.52 145.52 0 0 0 6.921 2.165a167.467
167.467 0 0 0-2.01 9.138c-5.354 28.2-1.173 50.591 12.134 58.266c13.744 7.926 36.812-.22
59.273-19.855a145.567 145.567 0 0 0 5.342-4.923a168.064 168.064 0 0 0 6.92 6.314c21.758 18.722
43.246 26.282 56.54 18.586c13.731-7.949 18.194-32.003 12.4-61.268a145.016 145.016 0 0
0-1.535-6.842c1.62-.48 3.21-.974 4.76-1.488c29.348-9.723 48.443-25.443
48.443-41.52c0-15.417-17.868-30.326-45.517-39.844Zm-6.365 70.984c-1.4.463-2.836.91-4.3
1.345c-3.24-10.257-7.612-21.163-12.963-32.432c5.106-11 9.31-21.767 12.459-31.957c2.619.758 5.16
1.557 7.61 2.4c23.69 8.156 38.14 20.213 38.14 29.504c0 9.896-15.606 22.743-40.946 31.14Zm-10.514
20.834c2.562 12.94 2.927 24.64 1.23 33.787c-1.524 8.219-4.59 13.698-8.382 15.893c-8.067
4.67-25.32-1.4-43.927-17.412a156.726 156.726 0 0 1-6.437-5.87c7.214-7.889 14.423-17.06
21.459-27.246c12.376-1.098 24.068-2.894 34.671-5.345a134.17 134.17 0 0 1 1.386 6.193ZM87.276
214.515c-7.882 2.783-14.16 2.863-17.955.675c-8.075-4.657-11.432-22.636-6.853-46.752a156.923 156.923
0 0 1 1.869-8.499c10.486 2.32 22.093 3.988 34.498 4.994c7.084 9.967 14.501 19.128 21.976
27.15a134.668 134.668 0 0 1-4.877 4.492c-9.933 8.682-19.886 14.842-28.658 17.94ZM50.35
144.747c-12.483-4.267-22.792-9.812-29.858-15.863c-6.35-5.437-9.555-10.836-9.555-15.216c0-9.322
13.897-21.212 37.076-29.293c2.813-.98 5.757-1.905 8.812-2.773c3.204 10.42 7.406 21.315 12.477
32.332c-5.137 11.18-9.399 22.249-12.634 32.792a134.718 134.718 0 0
1-6.318-1.979Zm12.378-84.26c-4.811-24.587-1.616-43.134 6.425-47.789c8.564-4.958 27.502 2.111 47.463
19.835a144.318 144.318 0 0 1 3.841 3.545c-7.438 7.987-14.787 17.08-21.808 26.988c-12.04 1.116-23.565
2.908-34.161 5.309a160.342 160.342 0 0 1-1.76-7.887Zm110.427 27.268a347.8 347.8 0 0
0-7.785-12.803c8.168 1.033 15.994 2.404 23.343 4.08c-2.206 7.072-4.956 14.465-8.193 22.045a381.151
381.151 0 0 0-7.365-13.322Zm-45.032-43.861c5.044 5.465 10.096 11.566 15.065 18.186a322.04 322.04 0 0
0-30.257-.006c4.974-6.559 10.069-12.652 15.192-18.18ZM82.802 87.83a323.167 323.167 0 0 0-7.227
13.238c-3.184-7.553-5.909-14.98-8.134-22.152c7.304-1.634 15.093-2.97 23.209-3.984a321.524 321.524 0
0 0-7.848 12.897Zm8.081 65.352c-8.385-.936-16.291-2.203-23.593-3.793c2.26-7.3 5.045-14.885
8.298-22.6a321.187 321.187 0 0 0 7.257 13.246c2.594 4.48 5.28 8.868 8.038 13.147Zm37.542
31.03c-5.184-5.592-10.354-11.779-15.403-18.433c4.902.192 9.899.29 14.978.29c5.218 0 10.376-.117
15.453-.343c-4.985 6.774-10.018 12.97-15.028 18.486Zm52.198-57.817c3.422 7.8 6.306 15.345 8.596
22.52c-7.422 1.694-15.436 3.058-23.88 4.071a382.417 382.417 0 0 0 7.859-13.026a347.403 347.403 0 0 0
7.425-13.565Zm-16.898 8.101a358.557 358.557 0 0 1-12.281 19.815a329.4 329.4 0 0 1-23.444.823c-7.967
0-15.716-.248-23.178-.732a310.202 310.202 0 0 1-12.513-19.846h.001a307.41 307.41 0 0
1-10.923-20.627a310.278 310.278 0 0 1 10.89-20.637l-.001.001a307.318 307.318 0 0 1
12.413-19.761c7.613-.576 15.42-.876 23.31-.876H128c7.926 0 15.743.303 23.354.883a329.357 329.357 0 0
1 12.335 19.695a358.489 358.489 0 0 1 11.036 20.54a329.472 329.472 0 0 1-11
20.722Zm22.56-122.124c8.572 4.944 11.906 24.881 6.52 51.026c-.344 1.668-.73 3.367-1.15
5.09c-10.622-2.452-22.155-4.275-34.23-5.408c-7.034-10.017-14.323-19.124-21.64-27.008a160.789 160.789
0 0 1 5.888-5.4c18.9-16.447 36.564-22.941 44.612-18.3ZM128 90.808c12.625 0 22.86 10.235 22.86
22.86s-10.235 22.86-22.86 22.86s-22.86-10.235-22.86-22.86s10.235-22.86 22.86-22.86Z"></path></svg>
```

src\features\auth\pages\LoginPage.tsx

```
import Page from "@/app/layout/Page";
import { API } from "@/shared/api/endpoints";
import { api } from "@/shared/api/client";
import { useNavigate } from "react-router-dom";
import { useState } from "react";
import type { LoginReq, LoginRes } from "@/shared/api/types";

export default function LoginPage() {
    const nav = useNavigate();
    const [email, setEmail] = useState("admin@example.com");
    const [password, setPassword] = useState("admin");
    const [loading, setLoading] = useState(false);
    const [err, setErr] = useState<string | null>(null);

    const submit = async (e: React.FormEvent) => {
        e.preventDefault();
        setErr(null); setLoading(true);
        try {
            const body: LoginReq = { email, password };
            const res = await api<LoginRes>(API.login, {
                method: "POST",
                body: JSON.stringify(body),
            });
            localStorage.setItem("token", res.access_token);
            nav("/upload"); // після логіна одразу на Upload
        } catch (e: any) {
            setErr(e?.message || "Login failed");
        } finally {
            setLoading(false);
        }
    };

    return (
        <Page title="Login">
            <form onSubmit={submit} className="grid gap-3 max-w-sm">
                <label className="grid gap-1">
                    <span className="text-sm">Email</span>
                    <input
                        type="email"
                        className="rounded-md border px-3 py-2"
                        value={email}
                        onChange={e => setEmail(e.target.value)}
                        autoComplete="username"
                    />
                </label>
                <label className="grid gap-1">
                    <span className="text-sm">Password</span>
                    <input
                        type="password"
                        className="rounded-md border px-3 py-2"
                        value={password}
                        onChange={e => setPassword(e.target.value)}
                        autoComplete="current-password"
                    />
                </label>

                {err && <div className="text-sm text-red-700">{err}</div>}

                <button
                    type="submit"
                    disabled={loading}
                    className="rounded-md bg-black text-white px-4 py-2 disabled:opacity-50"
                >
                    {loading ? "Signing in..." : "Sign in"}
                </button>
            </form>

            <div className="mt-4 text-sm text-neutral-600">
                Demo: <code>admin@example.com / admin</code> або <code>user@example.com /
user</code>
            </div>
        </Page>
    );
}
```

src\features\cart\pages\CartPage.tsx

```tsx
import Page from "@/app/layout/Page";
import { api } from "@/shared/api/client";
import { API } from "@/shared/api/endpoints";
import Modal from "@/shared/ui/Modal";
import { useModal } from "@/shared/ui/useModal";
import { useEffect, useMemo, useState } from "react";
import { useLocale } from "@/shared/i18n/LocaleContext";

type CartItem = {
    id: number;
    quote_id: number;
    qty: number;
    unit_price: number;      // € за шт
    subtotal: number;        // € за позицію
    quote: {
        id: number;
        price_eur: string;
        settings_json: any;
        material: { id: number; name?: string | null; color?: string | null };
    }
};

type CartResp = {
    id: number | null;
    items: CartItem[];
    total_eur: string;       // сума за кошик
};

export default function CartPage() {
    const { locale } = useLocale();
    const [data, setData] = useState<CartResp | null>(null);
    const [loading, setLoading] = useState(false);
    const [busy, setBusy] = useState(false);
    const [error, setError] = useState<string | null>(null);

    const redirecting = useModal(false);
    const failed = useModal(false);

    async function load() {
        setLoading(true);
        try {
            const cart = await api<CartResp>(API.cart.me + `?locale=□{locale}`);
            setData(cart);
        } catch (e: any) {
            setError(e?.message || "Не вдалося завантажити кошик");
            failed.show();
        } finally {
            setLoading(false);
        }
    }

    useEffect(() => { load(); }, [locale]);

    async function removeItem(id: number) {
        setBusy(true);
        try {
            await api(API.cart.remove(id), { method: "DELETE" });
            await load();
        } catch (e: any) {
            setError(e?.message || "Не вдалося видалити позицію");
            failed.show();
        } finally {
            setBusy(false);
        }
    }

    async function setQty(itemId: number, next: number) {
        setBusy(true);
        try {
            await api(API.cart.update(itemId), {
                method: "PATCH",
                body: JSON.stringify({ qty: Math.max(0, Number(next || 0)) }),
            });
            await load();
        } catch (e: any) {
            setError(e?.message || "Не вдалося оновити кількість");
            failed.show();
        } finally {
            setBusy(false);
        }
    }

    async function checkout() {
        if (!data || !data.items?.length) return;
        setBusy(true);
        redirecting.show();
        try {
            // 1) створюємо Order з кошика
            const order = await api<{ order_id: number; total_eur: string; status: string }>(
                API.orders.checkout,
                { method: "POST" }
            );

            // збережемо для екрана успіху
            sessionStorage.setItem("last_order_id", String(order.order_id));

            // 2) отримуємо Stripe checkout session URL
            const pay = await api<{ checkout_url: string; session_id: string }>(
                API.payments.checkout,
                { method: "POST", body: JSON.stringify({ order_id: order.order_id }) }
            );

            // 3) редірект на Stripe
            window.location.href = pay.checkout_url;
        } catch (e: any) {
            redirecting.hide();
            setError(e?.message || "Оплата наразі недоступна");
            failed.show();
        } finally {
            setBusy(false);
        }
    }

    const total = useMemo(() => data?.total_eur ?? "0.00", [data]);

    return (
        <Page title="Cart">
            <div className="rounded-xl border p-4">
                <div className="overflow-x-auto">
                    <table className="min-w-full text-sm">
                        <thead className="text-left">
                        <tr className="border-b">
                            <th className="py-2 pr-4">Quote #</th>
                            <th className="py-2 pr-4">Матеріал</th>
                            <th className="py-2 pr-4 text-center">К-сть</th>
                            <th className="py-2 pr-4 text-right">Ціна/шт, €</th>
                            <th className="py-2 pr-4 text-right">Сума, €</th>
                            <th className="py-2"></th>
                        </tr>
                        </thead>
                        <tbody>
                        {data?.items?.map((it) => (
                            <tr key={it.id} className="border-b">
                                <td className="py-2 pr-4">#{it.quote_id}</td>
                                <td className="py-2 pr-4">
                                    {[it.quote.material?.name,
it.quote.material?.color].filter(Boolean).join(" — ")}
                                </td>
                                <td className="py-2 pr-4 text-center">
                                    <div className="inline-flex items-center gap-1">
                                        <button
                                            className="px-2 py-1 rounded border text-xs"
                                            disabled={busy}
                                            onClick={() => setQty(it.id, it.qty - 1)}
                                            title="−1"
                                        >−</button>
                                        <input
                                            type="number"
                                            min={0}
                                            value={it.qty}
                                            onChange={(e) => setQty(it.id, Number(e.target.value))}
                                            className="w-16 rounded border px-2 py-1 text-center"
                                            disabled={busy}
                                        />
                                        <button
                                            className="px-2 py-1 rounded border text-xs"
                                            disabled={busy}
                                            onClick={() => setQty(it.id, it.qty + 1)}
                                            title="+1"
                                        >+</button>
                                    </div>
                                </td>
                                <td className="py-2 pr-4 text-right">{it.unit_price.toFixed(2)}</td>
                                <td className="py-2 pr-4 text-right">{it.subtotal.toFixed(2)}</td>
                                <td className="py-2">
                                    <button
                                        className="px-2 py-1 rounded border text-xs"
                                        disabled={busy}
                                        onClick={() => removeItem(it.id)}
                                    >
                                        Видалити
                                    </button>
                                </td>
                            </tr>
                        ))}
                        <tr>
                            <td className="py-2 pr-4" colSpan={4}><b>Всього</b></td>
                            <td className="py-2 pr-4 text-right"><b>{total}</b></td>
                            <td></td>
                        </tr>
                        </tbody>
                    </table>
                </div>

                <div className="mt-4 flex gap-3">
                    <button
                        onClick={checkout}
                        disabled={busy || loading || !data?.items?.length}
                        className="px-4 py-2 rounded-md bg-black text-white disabled:opacity-50"
                    >
                        Перейти до оплати
                    </button>
                    <button onClick={load} disabled={busy || loading} className="px-4 py-2 rounded-
md border">
                        Оновити
                    </button>
                </div>
            </div>

            {/* індикатор редіректу */}
            <Modal open={redirecting.open} onClose={() => {}} title="Переходимо до оплати">
                <div className="text-sm text-neutral-700">Зараз відкриється Stripe Checkout…</div>
            </Modal>

            {/* помилка */}
            <Modal
                open={failed.open}
                onClose={failed.hide}
                title="Помилка"
                footer={<button onClick={failed.hide} className="px-3 py-1.5 rounded bg-black text-
white">Гаразд</button>}
            >
                <div className="text-sm text-red-700">{error}</div>
            </Modal>
        </Page>
    );
}
```

src\features\order\pages\CheckoutCancelPage.tsx

```
import Page from "@/app/layout/Page";
import { Link } from "react-router-dom";

export default function CheckoutCancelPage() {
    return (
        <Page title="Оплату скасовано">
            <div className="space-y-2">
                <div className="text-sm text-neutral-700">Платіж не завершено. Можеш повторити, коли
будеш готовий.</div>
                <Link className="underline" to="/cart">Повернутись у кошик</Link>
            </div>
        </Page>
    );
}
```

src\features\order\pages\CheckoutSuccessPage.tsx

```
import Page from "@/app/layout/Page";
import { Link, useSearchParams } from "react-router-dom";

export default function CheckoutSuccessPage() {
    const [sp] = useSearchParams();
    const sessionId = sp.get("session_id") || "";
    const lastOrderId = sessionStorage.getItem("last_order_id");

    return (
        <Page title="Оплата пройшла успішно ">
            <div className="space-y-2">
                <div className="text-sm text-neutral-700">
                    Дякуємо! Stripe підтвердив платіж. Webhook на бекенді виставить
<code>orders.status=paid</code>.
                </div>
                <div className="text-sm text-neutral-600">Session ID: <code>{sessionId}</code></div>
                {lastOrderId && (
                    <Link className="inline-block mt-3 underline"
to={`/order?order=□{lastOrderId}`}>
                        Перейти до статусу замовлення #{lastOrderId}
                    </Link>
                )}
                <div><Link className="underline" to="/workshop">В майстерню</Link></div>
            </div>
        </Page>
    );
}
```

src\features\order\pages\OrderStatusPage.tsx

```
import Page from "@/app/layout/Page";
import { api } from "@/shared/api/client";
import { API } from "@/shared/api/endpoints";
import Modal from "@/shared/ui/Modal";
import { useModal } from "@/shared/ui/useModal";
import { useEffect, useRef, useState } from "react";
import { Link, useParams } from "react-router-dom";
import AuthImage from "@/shared/ui/AuthImage";

type OrderDTO = {
    id: number;
    status: "pending_payment" | "paid" | "fulfilled" | string;
    total_eur: string;
    created_at: string;
};

type JobDTO = {
    id: number;
    status:
        | "awaiting_preflight"
        | "ready"
        | "queued"
        | "printing"
        | "paused"
        | "needs_attention"
        | "done"
        | "canceled"
        | string;
    printer_id: number | null;
    progress: number;
    est_time_min: number;
    started_at?: string | null;
    finished_at?: string | null;
    model: { id: number; preview_url?: string | null }; // preview_url можна ігнорити
};

// Локальний SVG-плейсхолдер (без мережевих запитів)
function PlaceholderBox({ className = "" }: { className?: string }) {
    return (
        <svg viewBox="0 0 64 64" className={className}>
            <rect x="1" y="1" width="62" height="62" rx="8" fill="#f3f4f6" stroke="#e5e7eb" />
            <path d="M10 44l12-14 10 12 8-8 14 16H10z" fill="#e5e7eb" />
            <circle cx="24" cy="22" r="5" fill="#e5e7eb" />
        </svg>
    );
}

export default function OrderStatusPage() {
    const { id } = useParams();
    const orderId = Number(id || 0);

    const [order, setOrder] = useState<OrderDTO | null>(null);
    const [jobs, setJobs] = useState<JobDTO[]>([]);
    const [loading, setLoading] = useState(true);
    const [err, setErr] = useState<string | null>(null);

    // anti-flicker/anti-spam: кеш останньої відповіді та «активності»
    const snapshot = useRef<string>("");
    const activeRef = useRef<boolean>(false);

    // перегляд у повний розмір
    const viewer = useModal(false);
    const [activeModelId, setActiveModelId] = useState<number | null>(null);

    const fmtTime = (s?: string | null) => (s ? new Date(s).toLocaleTimeString() : "—");

    const load = async () => {
        try {
            setErr(null);
            const [o, j] = await Promise.all([
                api<OrderDTO>(API.orders.get(orderId)),
                api<JobDTO[]>(API.orders.jobs(orderId)),
            ]);
            // оновлюємо стан лише якщо є реальні зміни
            const next = JSON.stringify({ o, j });
            if (next !== snapshot.current) {
                setOrder(o);
                setJobs(j.slice().sort((a, b) => a.id - b.id));
                snapshot.current = next;
            }

            // чи ще є активні задачі
            activeRef.current = j.some((x) =>
                ["ready", "printing", "paused", "awaiting_preflight", "queued",
"needs_attention"].includes(x.status)
            );
        } catch (e: any) {
            setErr(e?.message || "Не вдалося завантажити замовлення");
        } finally {
            setLoading(false);
        }
    };

    // один цикл оновлення без залежностей від isActive
    useEffect(() => {
        if (!orderId) return;

        let stopped = false;
        let timer: any;

        const loop = async () => {
            if (stopped) return;
            if (document.visibilityState === "visible") {
                await load();
            }
            // 4с коли є активні задачі, 10с — коли все спокійно
            const interval = activeRef.current ? 4000 : 10000;
            timer = setTimeout(loop, interval);
        };

        setLoading(true);
        loop();

        return () => {
            stopped = true;
            clearTimeout(timer);
        };
    }, [orderId]);

    if (!orderId) {
        return (
            <Page title="Замовлення">
                <div>Невірний ідентифікатор.</div>
            </Page>
        );
    }
    if (loading) {
        return (
            <Page title={`Замовлення № {orderId}`}>
                <div>Завантаження…</div>
            </Page>
        );
    }
    if (err) {
        return (
            <Page title={`Замовлення № {orderId}`}>
                <div className="text-red-700">{err}</div>
            </Page>
        );
    }
    if (!order) {
        return (
            <Page title={`Замовлення № {orderId}`}>
                <div>Не знайдено.</div>
            </Page>
        );
    }

    const badge =
        order.status === "paid"
            ? "bg-blue-100 text-blue-800"
            : order.status === "fulfilled"
                ? "bg-green-100 text-green-800"
                : order.status === "pending_payment"
                    ? "bg-amber-100 text-amber-800"
                    : "bg-neutral-100 text-neutral-800";

    return (
        <Page title={`Замовлення №□{order.id}`}>
            <div className="mb-4 rounded-lg border bg-neutral-50 p-4 flex items-start justify-
between">
                <div>
                    <div className="flex items-center gap-3">
                        <span className={`text-xs px-2 py-0.5 rounded
□{badge}`}>{order.status}</span>
                        <div className="text-lg font-medium">
                            {Number(order.total_eur).toLocaleString("uk-UA")} €
                        </div>
                    </div>
                    <div className="text-sm text-neutral-600 mt-1">
                        Створено: {new Date(order.created_at).toLocaleString()}
                    </div>
                </div>
                <Link to="/orders" className="text-sm underline">
                    ← до списку
                </Link>
            </div>

            <div className="rounded-lg border overflow-hidden">
                <div className="px-4 py-2 border-b font-medium bg-neutral-50">Друкарські
задачі</div>
                <table className="w-full text-sm">
                    <thead className="bg-neutral-50">
                    <tr className="[&>th]:px-4 [&>th]:py-2 text-left">
                        <th>Job #</th>
                        <th>Прев’ю</th>
                        <th>Статус</th>
                        <th>Принтер</th>
                        <th>Прогрес</th>
                        <th>Оцінка, хв</th>
                        <th>Старт</th>
                        <th>Фініш</th>
                    </tr>
                    </thead>
                    <tbody>
                    {jobs.map((j) => {
                        const previewPath = API.files.preview(j.model.id);
                        return (
                            <tr key={j.id} className="border-t">
                                <td className="px-4 py-2">#{j.id}</td>
                                <td className="px-4 py-2">
                                    <button
                                        type="button"
                                        onClick={() => {
                                            setActiveModelId(j.model.id);
                                            viewer.show();
                                        }}
                                        className="h-14 w-14 border rounded overflow-hidden bg-white
cursor-zoom-in"
                                        title="Відкрити у повний розмір"
                                    >
                                        <AuthImage
                                            path={previewPath}
                                            alt={`Model □{j.model.id}`}
                                            className="h-full w-full object-contain"
                                        />
                                    </button>
                                </td>
                                <td className="px-4 py-2">{j.status}</td>
                                <td className="px-4 py-2">{j.printer_id ?? "—"}</td>
                                <td className="px-4 py-2">
                                    <div className="w-40 h-2 rounded bg-neutral-200">
                                        <div className="h-2 rounded bg-neutral-800" style={{ width:
`{j.progress ?? 0}%` }} />
                                    </div>
                                    <div className="text-xs text-neutral-500 mt-1">{j.progress ??
0}%</div>
                                </td>
                                <td className="px-4 py-2">{j.est_time_min ?? "—"}</td>
                                <td className="px-4 py-2">{fmtTime(j.started_at)}</td>
                                <td className="px-4 py-2">{fmtTime(j.finished_at)}</td>
                            </tr>
                        );
                    })}
                    {!jobs.length && (
                        <tr>
                            <td colSpan={8} className="px-4 py-8 text-center text-neutral-500">
                                Немає задач
                            </td>
                        </tr>
                    )}
                    </tbody>
                </table>
            </div>

            {/* Модалка повнорозмірного прев’ю */}
            <Modal
                open={viewer.open}
                onClose={viewer.hide}
                title={activeModelId ? `Модель #{activeModelId}` : "Прев’ю"}
                footer={<button onClick={viewer.hide} className="px-3 py-1.5 rounded bg-black text-
white">Закрити</button>}
            >
                <div className="flex items-center justify-center">
                    {activeModelId ? (
                        <AuthImage
                            path={API.files.preview(activeModelId)}
                            alt={`Model {activeModelId}`}
                            className="max-h-[80vh] max-w-[90vw] object-contain"
                        />
                    ) : (
                        <PlaceholderBox className="h-[60vh] w-[70vw]" />
                    )}
                </div>
            </Modal>
        </Page>
    );
}
```

src\features\order\pages\OrdersPage.tsx

```
import Page from "@/app/layout/Page";
import { api } from "@/shared/api/client";
import { API } from "@/shared/api/endpoints";
import ProgressBar from "@/shared/ui/ProgressBar";
import { useEffect,  useState } from "react";
import { Link } from "react-router-dom";

type OrderRow = {
    id: number;
    status: string;
    total_eur: string;
    created_at: string;
    jobs_total: number;
    jobs_done: number;
};

type PrintJob = {
    id: number;
    status: string;
    printer_id: number | null;
    progress: number;
    est_time_min: number;
    started_at: string | null;
    finished_at: string | null;
};

export default function OrdersPage() {
    const [orders, setOrders] = useState<OrderRow[]>([]);
    const [loading, setLoading] = useState(true);
    const [expanded, setExpanded] = useState<Record<number, { loading: boolean; jobs: PrintJob[] |
null }>>({});
    const [err, setErr] = useState<string | null>(null);

    useEffect(() => {
        (async () => {
            try {
                setLoading(true);
                const list = await api<OrderRow[]>(API.orders.list);
                setOrders(list);
            } catch (e: any) {
                setErr(e?.message || "Не вдалося завантажити замовлення");
            } finally {
                setLoading(false);
            }
        })();
    }, []);

    const toggle = async (oid: number) => {
        const isOpen = !!expanded[oid]?.jobs; // чи вже розкрито

        // 1) якщо відкрито — просто згортаємо і ВИХОДИМО
        if (isOpen) {
            setExpanded(prev => {
                const copy = { ...prev };
                delete copy[oid];            // прибрати ключ = згорнути
                return copy;
            });
            return;                        // ← важливо: не фетчимо
        }

        // 2) якщо закрито — показуємо "loading" і фетчимо джоби
        setExpanded(prev => ({ ...prev, [oid]: { loading: true, jobs: null } }));
        try {
            const jobs = await api<PrintJob[]>(API.orders.jobs(oid));
            setExpanded(prev => ({ ...prev, [oid]: { loading: false, jobs } }));
        } catch (e: any) {
            setExpanded(prev => ({ ...prev, [oid]: { loading: false, jobs: [] } }));
            setErr(e?.message || "Не вдалося завантажити задачі друку");
        }
    };

    return (
        <Page title="Мої замовлення">
```

```
            {err && <div className="mb-3 rounded border border-red-300 bg-red-50 text-red-700 px-3
py-2 text-sm">{err}</div>}

            {loading ? (
                <div className="text-sm text-neutral-600">Завантаження…</div>
            ) : orders.length === 0 ? (
                <div className="text-sm text-neutral-600">Замовлень ще немає. Перейди до <Link
className="underline" to="/upload">Upload</Link>.</div>
            ) : (
                <div className="overflow-x-auto rounded-lg border">
                    <table className="w-full text-sm">
                        <thead className="bg-neutral-50 text-neutral-600">
                        <tr>
                            <th className="px-3 py-2 text-left">#</th>
                            <th className="px-3 py-2 text-left">Статус</th>
                            <th className="px-3 py-2 text-right">Сума, €</th>
                            <th className="px-3 py-2 text-left">Створено</th>
                            <th className="px-3 py-2 text-left">Джоби</th>
                            <th className="px-3 py-2"></th>
                        </tr>
                        </thead>
                        <tbody>
                        {orders.map(o => {
                            const ex = expanded[o.id];
                            const open = !!ex?.jobs;
                            return (
                                <Fragment key={o.id}>
                                    <tr className="border-t">
                                        <td className="px-3 py-2">#{o.id}</td>
                                        <td className="px-3 py-2">
                        <span className="inline-flex items-center rounded-full border px-2 py-0.5
text-xs">
                          {o.status}
                        </span>
                                        </td>
                                        <td className="px-3 py-2 text-right">{o.total_eur}</td>
                                        <td className="px-3 py-2">{new
Date(o.created_at).toLocaleString()}</td>
                                        <td className="px-3 py-2">
                                            {o.jobs_done}/{o.jobs_total}
                                        </td>
                                        <td className="px-3 py-2 text-right">
                                            <button
                                                onClick={() => toggle(o.id)}
                                                className="text-sm underline inline-flex items-
center gap-1"

                                                disabled={ex?.loading}
                                            >
                                                <span className={`transition □{open ? "rotate-90" :
""}`}>▶</span>

                                                {open ? "Згорнути" : ex?.loading ? "Завантаження…" :
"Показати задачі"}
                                            </button>
                                            <Link to={`/orders/□{o.id}`} className="ml-3 text-sm
underline">Деталі</Link>
                                        </td>
                                    </tr>

                                    {open && (
                                        <tr className="border-t bg-neutral-50/50">
                                            <td colSpan={6} className="px-3 py-3">
                                                {ex?.jobs && ex.jobs.length === 0 ? (
                                                    <div className="text-sm text-neutral-600">Для
цього замовлення задач поки немає.</div>
                                                ) : (
                                                    <table className="w-full text-sm">
                                                        <thead>
                                                        <tr className="text-neutral-600">
                                                            <th className="text-left py-1">Job
#</th>

                                                            <th className="text-left
py-1">Статус</th>

                                                            <th className="text-left
py-1">Принтер</th>
                                                            <th className="text-left

py-1">Прогрес</th>
                                                            <th className="text-left py-1">Оцінка,

хв</th>
                                                            <th className="text-left

py-1">Старт</th>
                                                            <th className="text-left

py-1">Фініш</th>
                                                        </tr>
                                                        </thead>
                                                        <tbody>
                                                        {ex?.jobs?.map(j => (
                                                            <tr key={j.id} className="border-t">
                                                                <td className="py-1">#{j.id}</td>
                                                                <td className="py-1">{j.status}</td>
                                                                <td className="py-1">{j.printer_id
?? "—"}</td>
                                                                <td className="py-1">
                                                                    <div
className="w-56"><ProgressBar value={j.progress ?? 0} /></div>
                                                                </td>
                                                                <td className="py-1">{j.est_time_min
?? "—"}</td>
                                                                <td className="py-1">{j.started_at ?
new Date(j.started_at).toLocaleString() : "—"}</td>
                                                                <td className="py-1">{j.finished_at
? new Date(j.finished_at).toLocaleString() : "—"}</td>
                                                            </tr>
                                                        ))}
                                                        </tbody>
                                                    </table>
                                                )}
                                            </td>
                                        </tr>
                                    )}
                                </Fragment>
                            );
                        })}
                        </tbody>
                    </table>
                </div>
            )}
        </Page>
    );
}

import { Fragment } from "react";
```

src\features\quote\pages\QuotePage.tsx

```
import Page from "@/app/layout/Page";
import { api, API_BASE } from "@/shared/api/client";
import { API } from "@/shared/api/endpoints";
import Modal from "@/shared/ui/Modal";
import { useModal } from "@/shared/ui/useModal";
import { Link, useNavigate, useSearchParams } from "react-router-dom";
import { useEffect, useState } from "react";
import { useLocale } from "@/shared/i18n/LocaleContext";

/* ────────── helpers ────────── */

type Material = { id: number; name?: string; color?: string };
type QuoteResp = {
    id: number;
    price_eur: string;
    settings: any;
    breakdown: {
        qty: number;
        unit_price_eur: number;
        total_eur: number;
        est_time_min: number;
        est_filament_g: number;
        costs: Record<string, number>;
    };
    breakdown_labels?: Record<string, string>;
};

const INFILL_PRESETS = ["15% grid", "20% gyroid", "25% cubic", "30% cubic", "40% gyroid"];

/** простий placeholder-бокс (коли прев’ю немає) */
function PlaceholderBox({ className = "" }: { className?: string }) {
    return (
        <svg viewBox="0 0 64 64" className={className}>
            <rect x="1" y="1" width="62" height="62" rx="8" fill="#f3f4f6" stroke="#e5e7eb" />
            <path d="M10 44l12-14 10 12 8-8 14 16H10z" fill="#e5e7eb" />
            <circle cx="24" cy="22" r="5" fill="#e5e7eb" />
        </svg>
    );
}

/**
 * Картинка з авторизацією: тягне blob через fetch із Bearer токеном,
 * віддає тимчасовий object URL. Якщо впало — показує Placeholder.
 */
function AuthImage({
                       path,
                       alt,
                       className = "",
                   }: {
    path: string | null;
    alt: string;
    className?: string;
}) {
    const [src, setSrc] = useState<string | null>(null);

    useEffect(() => {
        let canceled = false;
        let revoke: string | null = null;

        async function run() {
            if (!path) {
                setSrc(null);
                return;
            }
            try {
                const token = localStorage.getItem("token") || "";
                const res = await fetch(`□{API_BASE}□{path}`, {
                    headers: token ? { Authorization: `Bearer □{token}` } : {},
                });
                if (!res.ok) throw new Error(String(res.status));
                const blob = await res.blob();
                const url = URL.createObjectURL(blob);
                revoke = url;
                if (!canceled) setSrc(url);
            } catch {
                if (!canceled) setSrc(null);
            }
        }

        run();
        return () => {
            canceled = true;
            if (revoke) URL.revokeObjectURL(revoke);
        };
    }, [path]);

    if (!src) return <PlaceholderBox className={className} />;
    return <img src={src} alt={alt} className={className} loading="lazy" />;
}

/** компактна кнопка, щоб не «роздувалась» */
function Btn({
                 children,
                 className = "",
                 ...rest
             }: React.ButtonHTMLAttributes<HTMLButtonElement> & { className?: string }) {
    return (
        <button
            {...rest}
            className={
                "inline-flex h-10 items-center justify-center gap-2 rounded-md px-4 text-sm font-
medium " +
                className
            }
        >
            {children}
        </button>
    );
}

/* ────────── page ────────── */

export default function QuotePage() {
    const nav = useNavigate();
    const [sp] = useSearchParams();
    const modelId = Number(sp.get("model") || "0");
    const { locale } = useLocale();

    const [materials, setMaterials] = useState<Material[]>([]);
    const [loadingMats, setLoadingMats] = useState(true);
    const [labels, setLabels] = useState<Record<string, string>>({});
    const [form, setForm] = useState({
        material_id: 0,
        layer_height: 0.2,
        infill: "20% gyroid",
        qty: 1,
    });

    const [busy, setBusy] = useState(false);
    const [quote, setQuote] = useState<QuoteResp | null>(null);
    const [error, setError] = useState<string | null>(null);

    const added = useModal(false); // «додано в кошик»
    const failed = useModal(false); // помилка
    const preview = useModal(false); // прев’ю повний розмір

    // 1) матеріали
    useEffect(() => {
        (async () => {
            try {
                setLoadingMats(true);
                const url = API.catalog.materials(locale);
                const mats = await api<Material[]>(url);
                const sorted = (mats || [])
                    .slice()
                    .sort(
                        (a, b) =>
                            (a.name || "").localeCompare(b.name || "", locale) ||
                            (a.color || "").localeCompare(b.color || "", locale),
                    );
                setMaterials(sorted);
                if (sorted.length && !form.material_id)
                    setForm((f) => ({ ...f, material_id: sorted[0].id }));
            } catch (e: any) {
                setError(e?.message || "Не вдалося завантажити матеріали");
                failed.show();
            } finally {
                setLoadingMats(false);
            }
        })();
        // eslint-disable-next-line react-hooks/exhaustive-deps
    }, [locale]);

    const onChange = (k: keyof typeof form, v: any) =>
        setForm((s) => ({ ...s, [k]: v }));

    // 2) розрахунок
    const makeQuote = async () => {
        if (!modelId) {
            setError("Нема ідентифікатора моделі. Перейди з Upload або вкажи ?model=ID.");
            failed.show();
            return;
        }
        setBusy(true);
        setError(null);
        setQuote(null);
        try {
            const payload = {
                model_id: modelId,
                material_id: Number(form.material_id),
                layer_height: Number(form.layer_height),
                infill: form.infill,
                qty: Number(form.qty),
            };
            const res = await api<QuoteResp>(
                API.quotes + `?locale=□{encodeURIComponent(locale)}`,
                { method: "POST", body: JSON.stringify(payload) },
            );
            setQuote(res);
            setLabels(res.breakdown_labels || {});
        } catch (e: any) {
            setError(e?.message || "Помилка розрахунку ціни");
            failed.show();
        } finally {
            setBusy(false);
        }
    };

    // 3) додати в кошик
    const addToCart = async () => {
        if (!quote?.id) return;
        setBusy(true);
        try {
            await api(API.cart.add(quote.id), {
                method: "POST",
                body: JSON.stringify({ qty: form.qty }),
            });
            added.show();
        } catch (e: any) {
            setError(e?.message || "Не вдалося додати в кошик");
            failed.show();
        } finally {
            setBusy(false);
        }
    };

    // const breakdown = useMemo(() => {
    //     if (!quote) return null;
    //     const b = quote.breakdown;
```

```
//     const list = Object.entries(b.costs || {});
//     return { ...b, list };
// }, [quote]);

const previewPath = modelId ? `/v1/files/preview/□{modelId}.png` : null;

return (
    <Page title="Quote">
        {!modelId && (
            <div className="mb-4 p-3 rounded border border-amber-300 bg-amber-50 text-sm">
                Немає параметра <code>?model=ID</code>.{" "}
                <Link className="underline" to="/upload">
                    Завантаж модель
                </Link>{" "}
                і повертайся.
            </div>
        )}

        {/* компактна двоколонка: форма + вузький aside 20rem */}
        <div className="mx-auto max-w-6xl grid grid-cols-1 lg:grid-cols-[1fr_20rem] gap-6">
            {/* форма */}
            <div className="grid gap-3">
                <label className="block">
                    <span className="text-sm">Матеріал</span>
                    <select
                        disabled={loadingMats || busy}
                        value={form.material_id}
                        onChange={(e) => onChange("material_id", Number(e.target.value))}
                        className="mt-1 block w-full rounded-md border px-3 py-2"
                    >
                        {materials.map((m) => (
                            <option key={m.id} value={m.id}>
                                {[m.name, m.color].filter(Boolean).join(" — ")}
                            </option>
                        ))}
                    </select>
                </label>

                <label className="block">
                    <span className="text-sm">Висота шару (мм)</span>
                    <input
                        type="number"
                        step="0.01"
                        min="0.06"
                        max="0.4"
                        value={form.layer_height}
                        onChange={(e) => onChange("layer_height", Number(e.target.value))}
                        className="mt-1 block w-full rounded-md border px-3 py-2"
                    />
                </label>

                <label className="block">
                    <span className="text-sm">Заповнення</span>
                    <select
                        value={form.infill}
                        onChange={(e) => onChange("infill", e.target.value)}
                        className="mt-1 block w-full rounded-md border px-3 py-2"
                    >
                        {INFILL_PRESETS.map((x) => (
                            <option key={x} value={x}>
                                {x}
                            </option>
                        ))}
                    </select>
                </label>

                <label className="block">
                    <span className="text-sm">Кількість</span>
                    <input
                        type="number"
                        min={1}
                        value={form.qty}
                        onChange={(e) => onChange("qty", Number(e.target.value || 1))}
```

```
                            className="mt-1 block w-full rounded-md border px-3 py-2"
                        />
                    </label>

                    <div className="flex gap-2">
                        <Btn
                            onClick={makeQuote}
                            disabled={busy || !modelId}
                            className="bg-black text-white disabled:opacity-50"
                        >
                            {busy ? "Рахуємо…" : "Розрахувати"}
                        </Btn>

                        {quote?.id && (
                            <Btn onClick={addToCart} disabled={busy} className="border">
                                Додати в кошик
                            </Btn>
                        )}
                    </div>
                </div>

                {/* aside 20rem — компактні картки з фіксованими min-height */}
                <div className="space-y-3">
                    {/* ціна */}
                    <section className="rounded-lg border p-3 min-h-[136px]">
                        {!quote ? (
                            <div className="animate-pulse space-y-2">
                                <div className="h-5 w-32 bg-neutral-200 rounded" />
                                <div className="h-4 w-40 bg-neutral-200 rounded" />
                                <div className="h-3 w-full bg-neutral-200 rounded" />
                            </div>
                        ) : (
                            <>
                                <div className="text-base font-medium">Ціна: {quote.price_eur}
€</div>
                                <div className="text-xs text-neutral-600">
                                    за {quote.breakdown.qty} шт · {quote.breakdown.unit_price_eur} €
/ шт
                                </div>
                                <div className="mt-2 text-xs">
                                    <div>Час: {Math.round(quote.breakdown.est_time_min)} хв</div>
                                    <div>Філамент: {quote.breakdown.est_filament_g} г</div>
                                </div>

                                {quote.breakdown.costs && (
                                    <details className="mt-2 text-xs">
                                        <summary className="cursor-pointer select-none text-
neutral-700">

                                            Деталі собівартості
                                        </summary>
                                        <ul className="mt-1 list-disc pl-4 space-y-0.5">
                                            {Object.entries(quote.breakdown.costs).map(([k, v]) => (
                                                <li key={k}>
                                                    {(labels[k] ?? k)}: {k === "margin_mult" ?
`×□{v}` : (v as any)}

                                                </li>
                                            ))}
                                        </ul>
                                    </details>
                                )}
                            </>
                        )}
                    </section>

                    {/* прев’ю */}
                    <section className="rounded-lg border p-3 min-h-[220px]">
                        <div className="mb-1 text-sm font-medium">Прев’ю моделі</div>
                        <button
                            type="button"
                            onClick={preview.show}
                            className="group w-full h-[170px] rounded border bg-white overflow-
hidden flex items-center justify-center cursor-zoom-in"
                            title="Збільшити"
                        >
                            {quote ? (
                                <AuthImage
                                    path={previewPath}
                                    alt={`Model □{modelId}`}
                                    className="h-full w-full object-contain group-hover:opacity-90"
                                />
                            ) : (
                                <div className="h-full w-full bg-neutral-100" />
                            )}
                        </button>
                        <div className="mt-1 text-[11px] text-neutral-500">Клік, щоб збільшити</div>
                    </section>
                </div>
            </div>

            {/* успіх додавання в кошик */}
            <Modal
                open={added.open}
                onClose={added.hide}
                title="Додано в кошик"
                footer={
                    <>
                        <Btn onClick={added.hide} className="border">
                            Продовжити
                        </Btn>
                        <Btn onClick={() => nav("/cart")} className="bg-black text-white">
                            Перейти в кошик
                        </Btn>
                    </>
                }
            >
                <div className="text-sm text-neutral-700">
                    Пропозицію додано. Можеш оформити замовлення в кошику.
                </div>
            </Modal>

            {/* модалка прев’ю */}
            <Modal
                open={preview.open}
                onClose={preview.hide}
                title="Прев’ю моделі"
                footer={
                    <Btn onClick={preview.hide} className="bg-black text-white">
                        Закрити
                    </Btn>
                }
            >
                <div className="aspect-[4/3] w-full max-h-[70vh] overflow-hidden rounded border bg-
white">
                    <AuthImage
                        path={previewPath}
                        alt={`Model □{modelId}`}
                        className="w-full h-full object-contain"
                    />
                </div>
            </Modal>

            {/* помилка */}
            <Modal
                open={failed.open}
                onClose={failed.hide}
                title="Помилка"
                footer={
                    <Btn onClick={failed.hide} className="bg-black text-white">
                        Гаразд
                    </Btn>
                }
            >
                <div className="text-sm text-red-700">{error}</div>
            </Modal>
        </Page>
    );
```

```
}
```

src\features\upload\pages\UploadPage.tsx

```
import Page from "@/app/layout/Page";
import { api } from "@/shared/api/client";
import { API } from "@/shared/api/endpoints";
import Modal from "@/shared/ui/Modal";
import { useModal } from "@/shared/ui/useModal";
import ProgressBar from "@/shared/ui/ProgressBar";
import { Link } from "react-router-dom";
import { useRef, useState } from "react";

// просте визначення типу вмісту по розширенню (коли file.type порожній)
function sniffContentType(file: File): string {
    if (file.type) return file.type;
    const ext = file.name.toLowerCase().split(".").pop();
    switch (ext) {
        case "stl":  return "model/stl";
        case "obj":  return "model/obj";
        case "gltf": return "model/gltf+json";
        case "glb":  return "model/gltf-binary";
        default:     return "application/octet-stream";
    }
}

// PUT із прогресом через XHR (fetch не дає прогрес завантаження)
function putWithProgress(url: string, file: File, contentType: string, onProgress: (pct: number) =>
void) {
    return new Promise<void>((resolve, reject) => {
        const xhr = new XMLHttpRequest();
        xhr.upload.onprogress = (e) => {
            if (e.lengthComputable) {
                const pct = (e.loaded / e.total) * 100;
                onProgress(pct);
            }
        };
        xhr.onload = () => {
            (xhr.status >= 200 && xhr.status < 300)
                ? resolve()
                : reject(new Error(`Upload failed: □{xhr.status} □{xhr.statusText}`));
        };
        xhr.onerror = () => reject(new Error("Network error during upload"));
        xhr.open("PUT", url, true);
        xhr.setRequestHeader("Content-Type", contentType);
        xhr.send(file);
    });
}

export default function UploadPage() {
    // const nav = useNavigate();
    const fileInput = useRef<HTMLInputElement | null>(null);
    const [file, setFile] = useState<File | null>(null);
    const [progress, setProgress] = useState(0);
    const [busy, setBusy] = useState(false);
    const [error, setError] = useState<string | null>(null);

    // що повернемо після complete
    const [uploadId, setUploadId] = useState<number | null>(null);
    const [modelId, setModelId] = useState<number | null>(null);

    const success = useModal(false);
    const fail    = useModal(false);

    const onPick = (e: React.ChangeEvent<HTMLInputElement>) => {
        setError(null); setUploadId(null); setModelId(null); setProgress(0);
        const f = e.target.files?.[0] || null;
        setFile(f);
    };

    const reset = () => {
        setFile(null); setProgress(0); setUploadId(null); setModelId(null);
        if (fileInput.current) fileInput.current.value = "";
    };

    const startUpload = async () => {
        if (!file) { setError("Спочатку обери файл (.stl/.obj/.gltf/.glb)"); fail.show(); return; }
        setBusy(true); setError(null);

        try {
            const contentType = sniffContentType(file);

            // 1) presign
            const presign = await api<{ url: string; key: string; bucket: string }>(API.presign, {
                method: "POST",
                body: JSON.stringify({ content_type: contentType }),
            });

            // 2) PUT з прогресом у MinIO
            await putWithProgress(presign.url, file, contentType, (pct) => setProgress(pct));

            // 3) complete (бек сам перевірить розмір/etag і створить Model3D)
            const complete = await api<{
                upload: { id: number; key: string; bucket: string; status: string };
                model:  { id: number; status: string };
            }>(API.complete, {
                method: "POST",
                body: JSON.stringify({ key: presign.key, content_type: contentType }),
            });

            setUploadId(complete.upload?.id ?? null);
            setModelId(complete.model?.id ?? null);
            success.show();
        } catch (e: any) {
            setError(e?.message || "Помилка завантаження");
            fail.show();
        } finally {
            setBusy(false);
        }
    };

    return (
        <Page title="Upload">
            <div className="grid gap-4 max-w-xl">
                <label className="block">
                    <span className="text-sm">Обери 3D-файл (.stl / .obj / .gltf / .glb)</span>
                    <input
                        ref={fileInput}
                        type="file"
                        accept=".stl,.obj,.gltf,.glb,model/stl,model/obj,model/gltf-
binary,model/gltf+json"
                        onChange={onPick}
                        className="mt-1 block w-full rounded-md border px-3 py-2"
                    />
                </label>

                {file && (
                    <div className="text-sm text-neutral-700">
                        Файл: <b>{file.name}</b> ({(file.size/1024).toFixed(1)} KB)
                    </div>
                )}

                {busy && (
                    <div className="grid gap-2">
                        <ProgressBar value={progress} />
                        <div className="text-xs text-neutral-500">Завантаження:
{Math.round(progress)}%</div>
                    </div>
                )}

                <div className="flex gap-2">
                    <button
                        onClick={startUpload}
                        disabled={!file || busy}
                        className="px-4 py-2 rounded-md bg-black text-white disabled:opacity-50"
                    >
                        {busy ? "Uploading…" : "Upload"}
                    </button>
                    <button
                        onClick={reset}
                        disabled={busy}
                        className="px-4 py-2 rounded-md border"
                    >
                        Скинути
                    </button>
                </div>
            </div>

            {/* успіх */}
            <Modal
                open={success.open}
                onClose={success.hide}
                title="Модель завантажено"
                footer={
                    <>
                        <button onClick={success.hide} className="px-3 py-1.5 rounded
border">Закрити</button>
                        {modelId && (
                            <Link to={`/quote?model=${modelId}`} className="px-3 py-1.5 rounded bg-
black text-white">
                                Перейти до Quote
                            </Link>
                        )}
                    </>
                }
            >
                <div className="text-sm text-neutral-700 space-y-1">
                    <div>Upload ID: <b>{uploadId ?? "—"}</b></div>
                    <div>Model ID: <b>{modelId ?? "—"}</b></div>
                    <div>Статус аналізу: <i>processing → analyzed</i> (можеш перейти до Quote вже
зараз)</div>
                </div>
            </Modal>

            {/* помилка */}
            <Modal
                open={fail.open}
                onClose={fail.hide}
                title="Помилка"
                footer={<button onClick={fail.hide} className="px-3 py-1.5 rounded bg-black text-
white">Гаразд</button>}
            >
                <div className="text-sm text-red-700">{error}</div>
                <div className="text-xs text-neutral-500 mt-2 space-y-1">
                    <p>Швидкі перевірки, якщо не залетіло:</p>
                    <ul className="list-disc pl-5 space-y-0.5">
                        <li>Ти залогінений (Bearer токен є в localStorage)?</li>
                        <li>У бекенді CORS дозволяє <code>http://localhost:5173</code>?</li>
                        <li>MinIO CORS:
<code>MINIO_API_CORS_ALLOW_ORIGIN=http://localhost:5173</code> (і <code>PUT</code> у methods)?</li>
                        <li>Бакет існує (<code>models</code>)?</li>
                    </ul>
                </div>
            </Modal>
        </Page>
    );
}
```

```
// src/features/workshop/pages/WorkshopPage.tsx
import Page from "@/app/layout/Page";
import { api } from "@/shared/api/client";
import { API } from "@/shared/api/endpoints";
import ProgressBar from "@/shared/ui/ProgressBar";
import Modal from "@/shared/ui/Modal";
import { useEffect, useMemo, useState } from "react";

/** ---------------- Types ---------------- */
type JobRow = {
    id: number;
    order_id?: number | null;
    status:
        | "awaiting_preflight"
        | "ready"
        | "queued"
        | "printing"
        | "paused"
        | "needs_attention"
        | "done"
        | "canceled"
        | string;
    printer_id: number | null;
    progress: number;
    est_time_min: number;
    started_at?: string | null;
    finished_at?: string | null;
};

type JobAction = "start" | "pause" | "resume" | "cancel";

/** ---------------- Helpers ---------------- */
function toArray<T = any>(x: any): T[] {
    if (Array.isArray(x)) return x;
    if (x && Array.isArray(x.items)) return x.items;
    if (x && Array.isArray(x.data)) return x.data;
    return [];
}

export default function WorkshopPage() {
    const [raw, setRaw] = useState<JobRow[]>([]);
    const [loading, setLoading] = useState(true);
    const [err, setErr] = useState<string | null>(null);
    const [busy, setBusy] = useState<Record<number, boolean>>({});

    // фільтри
    const [status, setStatus] = useState<string>("all");
    const [query, setQuery] = useState("");

    // пагінація
    const [page, setPage] = useState(1);
    const pageSize = 25;

    // ---- Preflight modal state ----
    const [preflightFor, setPreflightFor] = useState<JobRow | null>(null);
    const [preflight, setPreflight] = useState({
        bed_cleared: true,
        filament_ok: true,
        nozzle_ok: true,
    });

    const JOBS_LIST = (API.jobs as any).list ?? API.jobs.list;

    const load = async () => {
        try {
            setErr(null);
            setLoading(true);
            const res = await api<any>(JOBS_LIST);
            setRaw(toArray<JobRow>(res));
        } catch (e: any) {
            setErr(e?.message || "Не вдалося завантажити задачі");
            setRaw([]);
        } finally {
```

```
        setLoading(false);
    }
};

useEffect(() => {
    load();
}, []);

// застосовуємо фільтри
const filtered = useMemo(() => {
    const q = query.trim().toLowerCase();
    return raw.filter((j) => {
        const statusOk = status === "all" ? true : j.status === status;
        if (!statusOk) return false;
        if (!q) return true;
        const txt = [`#{j.id}`, j.status, j.printer_id ?? "", j.order_id ?? ""]
            .join(" ")
            .toLowerCase();
        return txt.includes(q);
    });
}, [raw, status, query]);

// сторінка
const totalPages = Math.max(1, Math.ceil(filtered.length / pageSize));
const safePage = Math.min(Math.max(1, page), totalPages);
const pageRows = useMemo(() => {
    const start = (safePage - 1) * pageSize;
    return filtered.slice(start, start + pageSize);
}, [filtered, safePage]);

const statuses: { value: string; label: string }[] = [
    { value: "all", label: "Усі" },
    { value: "awaiting_preflight", label: "awaiting_preflight" },
    { value: "ready", label: "ready" },
    { value: "queued", label: "queued" },
    { value: "printing", label: "printing" },
    { value: "paused", label: "paused" },
    { value: "needs_attention", label: "needs_attention" },
    { value: "done", label: "done" },
    { value: "canceled", label: "canceled" },
];

/** -------- Actions (start/pause/resume/cancel) -------- */
async function doAction(job: JobRow, action: JobAction) {
    if (busy[job.id]) return;
    if (action === "cancel" && !confirm(`Скасувати job #{job.id}?`)) return;

    setBusy((s) => ({ ...s, [job.id]: true }));
    try {
        const path =
            action === "start"
                ? API.jobs.start(job.id)
                : action === "pause"
                    ? API.jobs.pause(job.id)
                    : action === "resume"
                        ? API.jobs.resume(job.id)
                        : API.jobs.cancel(job.id);

        await api(path, { method: "POST" });
        await load();
    } catch (e: any) {
        alert(e?.message || "Не вдалося виконати дію");
    } finally {
        setBusy((s) => ({ ...s, [job.id]: false }));
    }
}

/** -------- Preflight flow -------- */
function openPreflight(job: JobRow) {
    setPreflight({ bed_cleared: true, filament_ok: true, nozzle_ok: true });
    setPreflightFor(job);
}
```

```
async function submitPreflight() {
    const job = preflightFor;
    if (!job) return;
    setBusy((s) => ({ ...s, [job.id]: true }));
    try {
        await api(API.jobs.preflight(job.id), {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify(preflight),
        });
        setPreflightFor(null);
        await load();
    } catch (e: any) {
        alert(e?.message || "Не вдалося виконати preflight");
    } finally {
        setBusy((s) => ({ ...s, [job.id]: false }));
    }
}

function renderActions(j: JobRow) {
    const disabled = !!busy[j.id];

    switch (j.status) {
        case "awaiting_preflight":
            return (
                <button
                    disabled={disabled}
                    onClick={() => openPreflight(j)}
                    className="rounded border px-2 py-1 text-xs"
                >
                    Preflight
                </button>
            );
        case "ready":
        case "queued":
            return (
                <button
                    disabled={disabled}
                    onClick={() => doAction(j, "start")}
                    className="rounded border px-2 py-1 text-xs"
                >
                    Start
                </button>
            );
        case "printing":
            return (
                <div className="flex gap-1">
                    <button
                        disabled={disabled}
                        onClick={() => doAction(j, "pause")}
                        className="rounded border px-2 py-1 text-xs"
                    >
                        Pause
                    </button>
                    <button
                        disabled={disabled}
                        onClick={() => doAction(j, "cancel")}
                        className="rounded border px-2 py-1 text-xs"
                    >
                        Cancel
                    </button>
                </div>
            );
        case "paused":
        case "needs_attention":
            return (
                <div className="flex gap-1">
                    <button
                        disabled={disabled}
                        onClick={() => doAction(j, "resume")}
                        className="rounded border px-2 py-1 text-xs"
                    >
                        Resume
                        </button>
                        <button
                            disabled={disabled}
                            onClick={() => doAction(j, "cancel")}
                            className="rounded border px-2 py-1 text-xs"
                        >
                            Cancel
                        </button>
                    </div>
                );
            default:
                return <span className="text-neutral-400 text-xs">—</span>;
        }
    }

    return (
        <Page title="Workshop (admin)">
            {err && (
                <div className="mb-3 rounded border border-red-300 bg-red-50 text-red-700 px-3 py-2
text-sm">
                    {err}
                </div>
            )}

            <div className="mb-3 flex flex-wrap items-center gap-2">
                <label className="text-sm">Статус</label>
                <select
                    value={status}
                    onChange={(e) => {
                        setStatus(e.target.value);
                        setPage(1);
                    }}
                    className="rounded border px-2 py-1"
                >
                    {statuses.map((s) => (
                        <option key={s.value} value={s.value}>
                            {s.label}
                        </option>
                    ))}
                </select>

                <input
                    value={query}
                    onChange={(e) => {
                        setQuery(e.target.value);
                        setPage(1);
                    }}
                    placeholder="Пошук: job/order/printer/status…"
                    className="ml-2 w-80 rounded border px-3 py-1"
                />

                <button
                    onClick={load}
                    className="ml-auto rounded bg-black px-3 py-1.5 text-white"
                    disabled={loading}
                >
                    Оновити
                </button>
            </div>

            <div className="rounded-lg border overflow-x-auto">
                <table className="w-full text-sm">
                    <thead className="bg-neutral-50">
                    <tr className="[&>th]:px-3 [&>th]:py-2 text-left">
                        <th>Job #</th>
                        <th>Order #</th>
                        <th>Статус</th>
                        <th>Принтер</th>
                        <th className="w-64">Прогрес</th>
                        <th>Оцінка, хв</th>
                        <th>Старт</th>
                        <th>Фініш</th>
                        <th>Дії</th>
        </tr>
        </thead>
        <tbody>
        {loading ? (
            <tr>
                <td colSpan={9} className="px-3 py-8 text-center text-neutral-600">
                    Завантаження…
                </td>
            </tr>
        ) : pageRows.length === 0 ? (
            <tr>
                <td colSpan={9} className="px-3 py-8 text-center text-neutral-600">
                    Нічого не знайдено.
                </td>
            </tr>
        ) : (
            pageRows.map((j) => (
                <tr key={j.id} className="border-t">
                    <td className="px-3 py-2">#{j.id}</td>
                    <td className="px-3 py-2">{j.order_id ?? "—"}</td>
                    <td className="px-3 py-2">{j.status}</td>
                    <td className="px-3 py-2">{j.printer_id ?? "—"}</td>
                    <td className="px-3 py-2">
                        <div className="w-56">
                            <ProgressBar value={j.progress ?? 0} />
                        </div>
                    </td>
                    <td className="px-3 py-2">{j.est_time_min ?? "—"}</td>
                    <td className="px-3 py-2">
                        {j.started_at ? new Date(j.started_at).toLocaleString() : "—"}
                    </td>
                    <td className="px-3 py-2">
                        {j.finished_at ? new Date(j.finished_at).toLocaleString() : "—"}
                    </td>
                    <td className="px-3 py-2">{renderActions(j)}</td>
                </tr>
            ))
        )}
        </tbody>
    </table>
</div>

{/* пагінація */}
<div className="mt-3 flex items-center gap-2">
    <button
        className="rounded border px-2 py-1 disabled:opacity-50"
        onClick={() => setPage((p) => Math.max(1, p - 1))}
        disabled={safePage <= 1}
    >
        Попередня
    </button>
    <div className="text-sm">
        Сторінка {safePage} з {totalPages} (всього {filtered.length})
    </div>
    <button
        className="rounded border px-2 py-1 disabled:opacity-50"
        onClick={() => setPage((p) => Math.min(totalPages, p + 1))}
        disabled={safePage >= totalPages}
    >
        Наступна
    </button>
</div>

{/* -------- Preflight modal -------- */}
<Modal
    open={!!preflightFor}
    onClose={() => setPreflightFor(null)}
    title={`Preflight для job #□{preflightFor?.id ?? ""}`}
    footer={
        <>
            <button
                onClick={() => setPreflightFor(null)}
                className="px-3 py-1.5 rounded border"
```

```
                        >
                            Скасувати
                        </button>
                        <button
                            onClick={submitPreflight}
                            className="px-3 py-1.5 rounded bg-black text-white"
                        >
                            Підтвердити
                        </button>
                    </>
                }
            >
                <div className="space-y-2 text-sm">
                    <label className="flex items-center gap-2">
                        <input
                            type="checkbox"
                            checked={preflight.bed_cleared}
                            onChange={(e) =>
                                setPreflight((s) => ({ ...s, bed_cleared: e.target.checked }))
                            }
                        />
                        Робочий стіл очищено
                    </label>
                    <label className="flex items-center gap-2">
                        <input
                            type="checkbox"
                            checked={preflight.filament_ok}
                            onChange={(e) =>
                                setPreflight((s) => ({ ...s, filament_ok: e.target.checked }))
                            }
                        />
                        Філамент заправлено / достатньо
                    </label>
                    <label className="flex items-center gap-2">
                        <input
                            type="checkbox"
                            checked={preflight.nozzle_ok}
                            onChange={(e) =>
                                setPreflight((s) => ({ ...s, nozzle_ok: e.target.checked }))
                            }
                        />
                        Сопло чисте / готове
                    </label>
                </div>
            </Modal>
        </Page>
    );
}
```

src\features\workshop\pages\WorkshopRoute.tsx

```
import AdminOnly from "@/shared/auth/AdminOnly";
import WorkshopPage from "./WorkshopPage";

export default function WorkshopRoute() {
    return (
        <AdminOnly>
            <WorkshopPage />
        </AdminOnly>
    );
}
```

src\shared\api\client.ts

```
const BASE = import.meta.env.VITE_API_BASE ?? "http://localhost:8000";
export const API_BASE = import.meta.env.VITE_API_BASE ?? "http://localhost:8000";

export async function api<T=any>(path: string, init: RequestInit = {}): Promise<T> {
    const token = localStorage.getItem("token");
    const headers: Record<string,string> = { "Content-Type": "application/json", ...(init.headers as
any || {}) };
    if (token) headers["Authorization"] = `Bearer □{token}`;
    const res = await fetch(`□{BASE}□{path}`, { ...init, headers });
    const ct = res.headers.get("content-type") || "";
    const data = ct.includes("application/json") ? await res.json() : await res.text();
    if (!res.ok) {
        if (res.status === 401) {
            localStorage.removeItem("token");
            if (location.pathname !== "/login") location.href = "/login";
        }
        const ct = res.headers.get("content-type") || "";
        const data = ct.includes("application/json") ? await res.json() : await res.text();
        const reqId = res.headers.get("X-Request-ID") || undefined;
        const msg = (data && (data.message || data.detail)) || res.statusText;
        const e: any = new Error(msg); if (reqId) e.requestId = reqId; throw e;
    }
    return data as T;
}
```

src\shared\api\endpoints.ts

```
export const API = {
    login: "/v1/auth/login",
    presign: "/v1/files/presign-upload",
    complete: "/v1/files/complete",

    catalog: {
        materials: (locale: string) =>
            `/v1/catalog/materials?locale=□{encodeURIComponent(locale)}`,
    },

    quotes: "/v1/quotes",

    cart: {
        me: "/v1/cart",
        add: (quoteId: number) => `/v1/cart/add/□{quoteId}`,
        remove: (itemId: number) => `/v1/cart/item/□{itemId}`,
        update: (itemId: number) => `/v1/cart/item/□{itemId}`,
    },

    orders: {
        list: "/v1/orders",
        checkout: "/v1/orders/checkout",
        get: (id: number) => `/v1/orders/□{id}`,
        jobs: (id: number) => `/v1/orders/□{id}/print-jobs`,
    },

    payments: {
        checkout: "/v1/payments/checkout",
    },

    files: {
        preview: (modelId: number) => `/v1/files/preview/□{modelId}.png`,
    },

    jobs: {
        list: "/v1/print-jobs",
        preflight: (id: number) => `/v1/print-jobs/□{id}/preflight`,
        start:    (id: number) => `/v1/print-jobs/□{id}/start`,
        pause:    (id: number) => `/v1/print-jobs/□{id}/pause`,
        resume:   (id: number) => `/v1/print-jobs/□{id}/resume`,
        cancel:   (id: number) => `/v1/print-jobs/□{id}/cancel`,
    },
} as const;
```

src\shared\api\types.ts

```
export type LoginReq = { email: string; password: string };
export type LoginRes = { access_token: string };
```

src\shared\auth\AdminOnly.tsx

```
import { Navigate } from "react-router-dom";
import { isAdmin } from "./role";

export default function AdminOnly({ children }: { children: React.ReactNode }) {
    if (!isAdmin()) return <Navigate to="/" replace />;
    return <>{children}</>;
}
```

src\shared\auth\role.ts

```
function b64urlToUtf8(b64: string) {
    try {
        const s = atob(b64.replace(/-/g, "+").replace(/_/g, "/"));
        return decodeURIComponent(escape(s));
    } catch { return "{}"; }
}

export function getUserRole(): string | null {
    const t = localStorage.getItem("token");
    if (!t) return null;
    const parts = t.split(".");
    if (parts.length < 2) return null;
    try {
        const payload = JSON.parse(b64urlToUtf8(parts[1]));
        return payload.role ?? null;
    } catch { return null; }
}

export const isAdmin = () => getUserRole() === "admin";
```

src\shared\hooks\useAuth.ts

```
import { useState } from "react";
export function useAuthState(){
    const [token,setToken] = useState<string|null>(localStorage.getItem("token"));
    const login = (t:string)=>{ localStorage.setItem("token",t); setToken(t); };
    const logout = ()=>{ localStorage.removeItem("token"); setToken(null); };
    return { token, login, logout };
}
```

src\shared\i18n\LocaleContext.tsx

```tsx
import { createContext, useContext, useMemo, useState } from "react";

const SUPPORTED = ["en", "uk", "pt"] as const;
type Locale = typeof SUPPORTED[number];

function detect(): Locale {
    const saved = localStorage.getItem("locale") as Locale | null;
    if (saved && (SUPPORTED as readonly string[]).includes(saved)) return saved as Locale;
    const nav = (navigator.language || "en").slice(0, 2).toLowerCase();
    return (SUPPORTED as readonly string[]).includes(nav) ? (nav as Locale) : "en";
}

type Ctx = { locale: Locale; setLocale: (l: Locale) => void };
const LocaleContext = createContext<Ctx>({ locale: "en", setLocale: () => {} });

export function LocaleProvider({ children }: { children: React.ReactNode }) {
    const [locale, set] = useState<Locale>(detect());
    const setLocale = (l: Locale) => { localStorage.setItem("locale", l); set(l); };
    const value = useMemo(() => ({ locale, setLocale }), [locale]);
    return <LocaleContext.Provider value={value}>{children}</LocaleContext.Provider>;
}

export const useLocale = () => useContext(LocaleContext);
export const LOCALES = SUPPORTED;
```

src\shared\ui\AuthImage.tsx

```tsx
import { useEffect, useState } from "react";
import { API_BASE } from "@/shared/api/client";

/**
 * Картинка з авторизацією: тягне blob через fetch із Bearer токеном,
 * віддає тимчасовий object URL. Якщо впало — показує простий placeholder.
 * Опціонально можна відкласти завантаження через prop `load`.
 */
export default function AuthImage({
                                      path,         // бековий шлях, напр. /v1/files/preview/11.png
                                      alt,
                                      className = "",
                                      load = true,  // якщо false — не фетчимо, поки не стане true
                                  }: {
    path: string | null;
    alt: string;
    className?: string;
    load?: boolean;
}) {
    const [src, setSrc] = useState<string | null>(null);

    useEffect(() => {
        let canceled = false;
        let revoke: string | null = null;

        async function run() {
            if (!load || !path) {
                setSrc(null);
                return;
            }
            try {
                const token = localStorage.getItem("token") || "";
                const res = await fetch(`${API_BASE}${path}`, {
                    headers: token ? { Authorization: `Bearer ${token}` } : {},
                });
                if (!res.ok) throw new Error(`HTTP ${res.status}`);
                const blob = await res.blob();
                const url = URL.createObjectURL(blob);
                revoke = url;
                if (!canceled) setSrc(url);
            } catch {
                if (!canceled) setSrc(null);
            }
        }

        run();
        return () => {
            canceled = true;
            if (revoke) URL.revokeObjectURL(revoke);
        };
    }, [path, load]);

    if (!src) {
        // легкий SVG-плейсхолдер, щоб не стукатись мережею
        return (
            <svg viewBox="0 0 64 64" className={className}>
                <rect x="1" y="1" width="62" height="62" rx="8" fill="#f3f4f6" stroke="#e5e7eb" />
                <path d="M10 44l12-14 10 12 8-8 14 16H10z" fill="#e5e7eb" />
                <circle cx="24" cy="22" r="5" fill="#e5e7eb" />
            </svg>
        );
    }
    return <img src={src} alt={alt} className={className} loading="lazy" />;
}
```

src\shared\ui\Button.tsx

src\shared\ui\ConfirmDialog.tsx

```
import Modal from "./Modal";

type ConfirmDialogProps = {
    open: boolean;
    title?: string;
    message?: string;
    confirmText?: string;
    cancelText?: string;
    onConfirm: () => void;
    onCancel: () => void;
};

export default function ConfirmDialog({
                                          open,
                                          title = "Підтвердження",
                                          message = "Ви впевнені?",
                                          confirmText = "Підтвердити",
                                          cancelText = "Скасувати",
                                          onConfirm,
                                          onCancel,
                                      }: ConfirmDialogProps) {
    return (
        <Modal
            open={open}
            onClose={onCancel}
            title={title}
            footer={
                <>
                    <button onClick={onCancel} className="px-3 py-1.5 rounded border border-
neutral-300">
                        {cancelText}
                    </button>
                    <button onClick={onConfirm} className="px-3 py-1.5 rounded bg-black text-white">
                        {confirmText}
                    </button>
                </>
            }
        >
            <p className="text-sm text-neutral-700">{message}</p>
        </Modal>
    );
}
```

src\shared\ui\Input.tsx

src\shared\ui\LanguageSwitcher.tsx

```
import { LOCALES, useLocale } from "@/shared/i18n/LocaleContext";

export default function LanguageSwitcher() {
    const { locale, setLocale } = useLocale();
    return (
        <select
            value={locale}
            onChange={(e) => setLocale(e.target.value as any)}
            className="border rounded px-2 py-1 text-sm bg-white"
            title="Language"
        >
            {LOCALES.map((l) => (
                <option key={l} value={l}>
                    {l.toUpperCase()}
                </option>
            ))}
        </select>
    );
}
```

src\shared\ui\Modal.tsx

```
import React, { useEffect } from "react";

type ModalProps = {
    open: boolean;
    onClose: () => void;
    title?: string;
    children: React.ReactNode;
    footer?: React.ReactNode;
    widthClass?: string;
    closeOnBackdrop?: boolean;
};

export default function Modal({
                                  open, onClose, title, children, footer,
                                  widthClass = "max-w-md", closeOnBackdrop = true,
                              }: ModalProps) {
    useEffect(() => {
        if (!open) return;
        const onKey = (e: KeyboardEvent) => e.key === "Escape" && onClose();
        document.addEventListener("keydown", onKey);
        document.body.style.overflow = "hidden";
        return () => {
            document.removeEventListener("keydown", onKey);
            document.body.style.overflow = "";
        };
    }, [open, onClose]);

    if (!open) return null;

    return (
        <div className="fixed inset-0 z-[1000]">
            <div className="absolute inset-0 bg-black/40 backdrop-blur-[1px]" onClick={() =>
closeOnBackdrop && onClose()} />
            <div className="absolute inset-0 flex items-center justify-center p-4">
                <div role="dialog" aria-modal="true"
                     className={`w-full □{widthClass} bg-white rounded-2xl shadow-xl border border-
neutral-200`}>
                    {title && <div className="px-5 py-4 border-b"><h3 className="text-lg font-
semibold">{title}</h3></div>}
                    <div className="px-5 py-4">{children}</div>
                    {footer && <div className="px-5 py-3 border-t flex justify-end gap-2 bg-
neutral-50 rounded-b-2xl">{footer}</div>}
                </div>
            </div>
        </div>
    );
}
```

src\shared\ui\ProgressBar.tsx

```tsx
export default function ProgressBar({ value }: { value: number }) {
    const clamped = Math.max(0, Math.min(100, Math.round(value || 0)));
    return (
        <div className="w-full h-2 bg-neutral-200 rounded-full overflow-hidden">
            <div
                className="h-full bg-emerald-600 transition-[width] duration-200"
                style={{ width: `□{clamped}%` }}
                aria-valuemin={0}
                aria-valuemax={100}
                aria-valuenow={clamped}
                role="progressbar"
            />
        </div>
    );
}
```

src\shared\ui\Select.tsx

src\shared\ui\Toast.tsx

src\shared\ui\useModal.ts

```
import { useCallback, useState } from "react";
export function useModal(initial = false) {
    const [open, setOpen] = useState<boolean>(initial);
    const show = useCallback(() => setOpen(true), []);
    const hide = useCallback(() => setOpen(false), []);
    const toggle = useCallback(() => setOpen(v => !v), []);
    return { open, show, hide, toggle };
}
```

styles\index.css

```
@import "tailwindcss";
```