# excel-delivery — selected files

## .env

# Environment variables declared in this file are automatically made available to Prisma.  
# See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema  
  
# Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB and CockroachDB.  
# See the documentation for all the connection string options: https://pris.ly/d/connection-strings  
  
# The following `prisma+postgres` URL is similar to the URL produced by running a local Prisma Postgres   
# server with the `prisma dev` CLI command, when not choosing any non-default ports or settings. The API key, unlike the   
# one found in a remote Prisma Postgres URL, does not contain any sensitive information.  
# Prisma / Supabase  
# App runtime (PgBouncer pooler, no PREPARE) — same as your Vercel DATABASE\_URL  
DATABASE\_URL=postgresql://postgres.tqlfmswpqbyyhjhjbxpj:FRGKAgQ5y8fRSomq@aws-1-eu-central-1.pooler.supabase.com:6543/postgres?sslmode=require&pgbouncer=true  
INIT\_ADMIN\_TOKEN=some-long-random-string  
ALLOW\_SIGNUPS=true  
DATABASE\_DIRECT\_URL=postgresql://postgres:FRGKAgQ5y8fRSomq@db.tqlfmswpqbyyhjhjbxpj.supabase.co:5432/postgres?sslmode=require  
NEXTAUTH\_URL=http://localhost:3000  
NEXTAUTH\_SECRET=87a235abe2594ab486f300ecfb3265d5a2cebf84746c1713e0642b6c87361b0b  
SUPABASE\_URL=https://tqlfmswpqbyyhjhjbxpj.supabase.co  
SUPABASE\_SERVICE\_ROLE=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InRxbGZtc3dwcWJ5eWhqaGpieHBqIiwicm9sZSI6InNlcnZpY2Vfcm9sZSIsImlhdCI6MTc1ODEzNDQ4NSwiZXhwIjoyMDczNzEwNDg1fQ.saKNirIWE5BPmNsDjfGaVl7-RZN8Bj1NrZ-S0CJyv6M  
SUPABASE\_BUCKET=excel  
PASSWORD\_PEPPER=939850fb87b0b5a51cec0adbc7f1099a912c8e05b18cfc58c08a5caf5f67b39a  
APP\_URL=http://localhost:3000  
ADMIN\_EMAIL=admin@example.com  
ADMIN\_PASSWORD=ChangeMe123!

## middleware.ts

// middleware.ts (must be at repo root, next to package.json)  
import { NextResponse } from "next/server";  
import type { NextRequest } from "next/server";  
import { getToken } from "next-auth/jwt";  
  
/\*\*  
 \* Global middleware with safe matcher:  
 \* - Matches all paths except Next assets and files with extensions.  
 \* - Handles "/" and "/login" routing, plus protected areas.  
 \*/  
export async function middleware(req: NextRequest) {  
 const { pathname, search } = req.nextUrl;  
 const url = (p: string) => new URL(p, req.url);  
 const token = await getToken({ req, secret: process.env.NEXTAUTH\_SECRET });  
  
 // Root gateway  
 if (pathname === "/") {  
 if (!token) return NextResponse.redirect(url("/login"));  
 return NextResponse.redirect(url(token.role === "ADMIN" ? "/dashboard/admin" : "/dashboard"));  
 }  
  
 // Logged-in users should not see /login  
 if (pathname === "/login" && token) {  
 return NextResponse.redirect(url(token.role === "ADMIN" ? "/dashboard/admin" : "/dashboard"));  
 }  
  
 // Paths that need auth  
 const needsAuth =  
 pathname === "/dashboard" ||  
 pathname.startsWith("/dashboard/") ||  
 pathname === "/files" ||  
 pathname.startsWith("/files/") ||  
 pathname === "/assignments" ||  
 pathname.startsWith("/assignments/") ||  
 pathname === "/support" ||  
 pathname.startsWith("/support/") ||  
 pathname === "/admin" ||  
 pathname.startsWith("/admin/");  
  
 if (needsAuth && !token) {  
 const login = url("/login");  
 login.searchParams.set("callbackUrl", pathname + (search || ""));  
 return NextResponse.redirect(login);  
 }  
  
 // Non-admins cannot access /admin  
 if ((pathname === "/admin" || pathname.startsWith("/admin/")) && token?.role !== "ADMIN") {  
 return NextResponse.redirect(url("/dashboard"));  
 }  
  
 // Admins visiting /dashboard go to /dashboard/admin  
 if (pathname === "/dashboard" && token?.role === "ADMIN") {  
 return NextResponse.redirect(url("/dashboard/admin"));  
 }  
  
 return NextResponse.next();  
}  
  
// Match everything except \_next assets and files with an extension (e.g., .png, .css, .ico)  
export const config = {  
 matcher: ["/((?!\_next|.\*\\..\*).\*)"],  
};

## prisma\migrations\20250929175119\_init\migration.sql

-- CreateEnum  
CREATE TYPE "public"."Role" AS ENUM ('USER', 'ADMIN');  
  
-- CreateEnum  
CREATE TYPE "public"."AuditAction" AS ENUM ('USER\_CREATED', 'SUBSCRIPTION\_TOGGLED', 'FILE\_UPLOADED', 'FILE\_ASSIGNED', 'APIKEY\_CREATED', 'APIKEY\_REVOKED', 'DOWNLOAD\_GRANTED');  
  
-- CreateTable  
CREATE TABLE "public"."User" (  
 "id" TEXT NOT NULL,  
 "email" TEXT NOT NULL,  
 "name" TEXT,  
 "role" "public"."Role" NOT NULL DEFAULT 'USER',  
 "passwordHash" TEXT,  
 "subscriptionActive" BOOLEAN NOT NULL DEFAULT false,  
 "subscriptionEndsAt" TIMESTAMP(3),  
 "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 "updatedAt" TIMESTAMP(3) NOT NULL,  
  
 CONSTRAINT "User\_pkey" PRIMARY KEY ("id")  
);  
  
-- CreateTable  
CREATE TABLE "public"."File" (  
 "id" TEXT NOT NULL,  
 "storagePath" TEXT NOT NULL,  
 "originalName" TEXT NOT NULL,  
 "mimeType" TEXT NOT NULL,  
 "sizeBytes" INTEGER NOT NULL,  
 "uploadedById" TEXT NOT NULL,  
 "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 "sha256" TEXT,  
  
 CONSTRAINT "File\_pkey" PRIMARY KEY ("id")  
);  
  
-- CreateTable  
CREATE TABLE "public"."FileAssignment" (  
 "id" TEXT NOT NULL,  
 "userId" TEXT NOT NULL,  
 "fileId" TEXT NOT NULL,  
 "assignedById" TEXT NOT NULL,  
 "note" TEXT,  
 "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
  
 CONSTRAINT "FileAssignment\_pkey" PRIMARY KEY ("id")  
);  
  
-- CreateTable  
CREATE TABLE "public"."ApiKey" (  
 "id" TEXT NOT NULL,  
 "label" TEXT NOT NULL,  
 "keyHash" TEXT NOT NULL,  
 "isActive" BOOLEAN NOT NULL DEFAULT true,  
 "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 "lastUsedAt" TIMESTAMP(3),  
  
 CONSTRAINT "ApiKey\_pkey" PRIMARY KEY ("id")  
);  
  
-- CreateTable  
CREATE TABLE "public"."AuditLog" (  
 "id" TEXT NOT NULL,  
 "actorId" TEXT,  
 "action" "public"."AuditAction" NOT NULL,  
 "targetId" TEXT,  
 "target" TEXT,  
 "meta" JSONB,  
 "createdAt" TIMESTAMP(3) NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
  
 CONSTRAINT "AuditLog\_pkey" PRIMARY KEY ("id")  
);  
  
-- CreateIndex  
CREATE UNIQUE INDEX "User\_email\_key" ON "public"."User"("email");  
  
-- CreateIndex  
CREATE UNIQUE INDEX "FileAssignment\_userId\_fileId\_key" ON "public"."FileAssignment"("userId", "fileId");  
  
-- AddForeignKey  
ALTER TABLE "public"."File" ADD CONSTRAINT "File\_uploadedById\_fkey" FOREIGN KEY ("uploadedById") REFERENCES "public"."User"("id") ON DELETE RESTRICT ON UPDATE CASCADE;  
  
-- AddForeignKey  
ALTER TABLE "public"."FileAssignment" ADD CONSTRAINT "FileAssignment\_userId\_fkey" FOREIGN KEY ("userId") REFERENCES "public"."User"("id") ON DELETE RESTRICT ON UPDATE CASCADE;  
  
-- AddForeignKey  
ALTER TABLE "public"."FileAssignment" ADD CONSTRAINT "FileAssignment\_assignedById\_fkey" FOREIGN KEY ("assignedById") REFERENCES "public"."User"("id") ON DELETE RESTRICT ON UPDATE CASCADE;  
  
-- AddForeignKey  
ALTER TABLE "public"."FileAssignment" ADD CONSTRAINT "FileAssignment\_fileId\_fkey" FOREIGN KEY ("fileId") REFERENCES "public"."File"("id") ON DELETE RESTRICT ON UPDATE CASCADE;  
  
-- AddForeignKey  
ALTER TABLE "public"."AuditLog" ADD CONSTRAINT "AuditLog\_actorId\_fkey" FOREIGN KEY ("actorId") REFERENCES "public"."User"("id") ON DELETE SET NULL ON UPDATE CASCADE;

## prisma\migrations\20251015132023\_user\_account\_status\migration.sql

/\*  
 Warnings:  
  
 - You are about to drop the column `subscriptionEndsAt` on the `User` table. All the data in the column will be lost.  
 - Made the column `name` on table `User` required. This step will fail if there are existing NULL values in that column.  
 - Made the column `passwordHash` on table `User` required. This step will fail if there are existing NULL values in that column.  
  
\*/  
-- CreateEnum  
CREATE TYPE "public"."AccountStatus" AS ENUM ('PENDING', 'ACTIVE', 'SUSPENDED');  
  
-- AlterTable  
ALTER TABLE "public"."User" DROP COLUMN "subscriptionEndsAt",  
ADD COLUMN "status" "public"."AccountStatus" NOT NULL DEFAULT 'PENDING',  
ALTER COLUMN "name" SET NOT NULL,  
ALTER COLUMN "passwordHash" SET NOT NULL;

## prisma\migrations\20251015142120\_add\_auditlog\_backrelation\migration.sql

/\*  
 Warnings:  
  
 - You are about to drop the column `mimeType` on the `File` table. All the data in the column will be lost.  
 - You are about to drop the column `sha256` on the `File` table. All the data in the column will be lost.  
 - You are about to drop the column `sizeBytes` on the `File` table. All the data in the column will be lost.  
 - You are about to drop the column `storagePath` on the `File` table. All the data in the column will be lost.  
 - A unique constraint covering the columns `[fileId,userId]` on the table `FileAssignment` will be added. If there are existing duplicate values, this will fail.  
 - Added the required column `title` to the `File` table without a default value. This is not possible if the table is not empty.  
 - Added the required column `updatedAt` to the `File` table without a default value. This is not possible if the table is not empty.  
  
\*/  
-- DropForeignKey  
ALTER TABLE "public"."File" DROP CONSTRAINT "File\_uploadedById\_fkey";  
  
-- DropForeignKey  
ALTER TABLE "public"."FileAssignment" DROP CONSTRAINT "FileAssignment\_fileId\_fkey";  
  
-- DropForeignKey  
ALTER TABLE "public"."FileAssignment" DROP CONSTRAINT "FileAssignment\_userId\_fkey";  
  
-- DropIndex  
DROP INDEX "public"."FileAssignment\_userId\_fileId\_key";  
  
-- AlterTable  
ALTER TABLE "public"."File" DROP COLUMN "mimeType",  
DROP COLUMN "sha256",  
DROP COLUMN "sizeBytes",  
DROP COLUMN "storagePath",  
ADD COLUMN "mime" TEXT,  
ADD COLUMN "size" INTEGER,  
ADD COLUMN "title" TEXT NOT NULL,  
ADD COLUMN "updatedAt" TIMESTAMP(3) NOT NULL,  
ADD COLUMN "url" TEXT,  
ALTER COLUMN "originalName" DROP NOT NULL,  
ALTER COLUMN "uploadedById" DROP NOT NULL;  
  
-- CreateIndex  
CREATE INDEX "AuditLog\_actorId\_idx" ON "public"."AuditLog"("actorId");  
  
-- CreateIndex  
CREATE INDEX "File\_uploadedById\_idx" ON "public"."File"("uploadedById");  
  
-- CreateIndex  
CREATE INDEX "File\_createdAt\_idx" ON "public"."File"("createdAt");  
  
-- CreateIndex  
CREATE INDEX "FileAssignment\_userId\_createdAt\_idx" ON "public"."FileAssignment"("userId", "createdAt");  
  
-- CreateIndex  
CREATE INDEX "FileAssignment\_assignedById\_idx" ON "public"."FileAssignment"("assignedById");  
  
-- CreateIndex  
CREATE UNIQUE INDEX "FileAssignment\_fileId\_userId\_key" ON "public"."FileAssignment"("fileId", "userId");  
  
-- AddForeignKey  
ALTER TABLE "public"."File" ADD CONSTRAINT "File\_uploadedById\_fkey" FOREIGN KEY ("uploadedById") REFERENCES "public"."User"("id") ON DELETE SET NULL ON UPDATE CASCADE;  
  
-- AddForeignKey  
ALTER TABLE "public"."FileAssignment" ADD CONSTRAINT "FileAssignment\_fileId\_fkey" FOREIGN KEY ("fileId") REFERENCES "public"."File"("id") ON DELETE CASCADE ON UPDATE CASCADE;  
  
-- AddForeignKey  
ALTER TABLE "public"."FileAssignment" ADD CONSTRAINT "FileAssignment\_userId\_fkey" FOREIGN KEY ("userId") REFERENCES "public"."User"("id") ON DELETE CASCADE ON UPDATE CASCADE;

## prisma\migrations\20251019211536\_add\_support\_ticket\migration.sql

-- AlterEnum  
ALTER TYPE "public"."AuditAction" ADD VALUE 'SUPPORT\_TICKET';

## prisma\schema.prisma

generator client {  
 provider = "prisma-client-js"  
}  
  
  
// prisma/schema.prisma  
  
datasource db {  
 provider = "postgresql"  
 // Pooled connection for your running app (goes through PgBouncer on port 6543)  
 url = env("DATABASE\_URL")  
 // Direct connection for migrations & prisma studio (5432)  
 directUrl = env("DATABASE\_DIRECT\_URL")  
}  
  
  
  
  
enum Role {  
 USER  
 ADMIN  
}  
  
enum AuditAction {  
 USER\_CREATED  
 SUBSCRIPTION\_TOGGLED  
 FILE\_UPLOADED  
 FILE\_ASSIGNED  
 APIKEY\_CREATED  
 APIKEY\_REVOKED  
 DOWNLOAD\_GRANTED  
 SUPPORT\_TICKET  
}  
  
enum AccountStatus {  
 PENDING  
 ACTIVE  
 SUSPENDED  
}  
  
model User {  
 id String @id @default(cuid())  
 name String  
 email String @unique  
 passwordHash String  
 role Role @default(USER)  
 subscriptionActive Boolean @default(false)  
 status AccountStatus @default(PENDING)  
 createdAt DateTime @default(now())  
 updatedAt DateTime @updatedAt  
  
 // Relations  
 uploads File[] @relation("UserUploads")  
 assignments FileAssignment[] @relation("UserAssignments")  
 assignmentsGiven FileAssignment[] @relation("AssignmentsAssignedBy")  
 auditLogs AuditLog[] @relation("AuditActor") // 👈 add this  
}  
  
model File {  
 id String @id @default(cuid())  
 title String  
 originalName String?  
 url String? // for external or pre-signed location (we’ll wire actual uploads next day)  
 mime String?  
 size Int?  
 uploadedBy User? @relation("UserUploads", fields: [uploadedById], references: [id])  
 uploadedById String?  
 createdAt DateTime @default(now())  
 updatedAt DateTime @updatedAt  
  
 // Assignments  
 assignments FileAssignment[]  
  
 @@index([uploadedById])  
 @@index([createdAt])  
}  
  
model FileAssignment {  
 id String @id @default(cuid())  
 file File @relation(fields: [fileId], references: [id], onDelete: Cascade)  
 fileId String  
 user User @relation("UserAssignments", fields: [userId], references: [id], onDelete: Cascade)  
 userId String  
 assignedBy User @relation("AssignmentsAssignedBy", fields: [assignedById], references: [id])  
 assignedById String  
 note String?  
 createdAt DateTime @default(now())  
  
 @@unique([fileId, userId]) // prevent double-assigning the same file to same user  
 @@index([userId, createdAt])  
 @@index([assignedById])  
}  
  
model ApiKey {  
 id String @id @default(cuid())  
 label String  
 keyHash String  
 isActive Boolean @default(true)  
 createdAt DateTime @default(now())  
 lastUsedAt DateTime?  
}  
  
model AuditLog {  
 id String @id @default(cuid())  
 actorId String?  
 actor User? @relation("AuditActor", fields: [actorId], references: [id])  
 action AuditAction  
 targetId String?  
 target String?  
 meta Json?  
 createdAt DateTime @default(now())  
  
 @@index([actorId]) // optional  
}

## prisma\seed.ts

import { PrismaClient } from "@prisma/client";  
import bcrypt from "bcrypt";  
  
const prisma = new PrismaClient();  
  
async function main() {  
 const email = "admin@example.com";  
 const password = "Admin123!";  
 const passwordHash = await bcrypt.hash(password, 12);  
  
 await prisma.user.upsert({  
 where: { email },  
 update: { role: "ADMIN", passwordHash, subscriptionActive: true },  
 create: { email, role: "ADMIN", name: "Admin", passwordHash, subscriptionActive: true },  
 });  
  
 console.log("Admin ready:", email, password);  
}  
  
main().finally(() => prisma.$disconnect());

## prisma\sql\check\_enum.sql

﻿SELECT e.enumlabel  
FROM pg\_type t  
JOIN pg\_enum e ON e.enumtypid = t.oid  
WHERE t.typname = 'AuditAction'  
ORDER BY e.enumsortorder;

## prisma\sql\patch\_add\_support\_ticket.sql

-- Add 'SUPPORT\_TICKET' to the enum type used by public."AuditLog".action  
  
DO $$  
DECLARE  
 enum\_typename text;  
BEGIN  
 -- Find the enum type name used by the action column  
 SELECT t.typname  
 INTO enum\_typename  
 FROM pg\_attribute a  
 JOIN pg\_class c ON c.oid = a.attrelid  
 JOIN pg\_namespace n ON n.oid = c.relnamespace  
 JOIN pg\_type t ON t.oid = a.atttypid  
 WHERE n.nspname = 'public'  
 AND c.relname = 'AuditLog' -- <- table name generated by Prisma for model AuditLog  
 AND a.attname = 'action';  
  
 IF enum\_typename IS NULL THEN  
 RAISE EXCEPTION 'Could not find enum type for public."AuditLog".action';  
 END IF;  
  
 -- Add value if it doesn't exist yet  
 IF NOT EXISTS (  
 SELECT 1  
 FROM pg\_enum e  
 JOIN pg\_type t ON t.oid = e.enumtypid  
 WHERE t.typname = enum\_typename  
 AND e.enumlabel = 'SUPPORT\_TICKET'  
 ) THEN  
 EXECUTE format('ALTER TYPE %I ADD VALUE ''SUPPORT\_TICKET''', enum\_typename);  
 END IF;  
END $$;

## src\app\(admin)\admin\audit\page.tsx

// src/app/(admin)/admin/audit/page.tsx  
"use client";  
  
import { useEffect, useState } from "react";  
  
type Row = {  
 id: string;  
 createdAt: string;  
 action: string;  
 target?: string | null;  
 targetId?: string | null;  
 actor?: { id: string; email: string; name?: string | null } | null;  
 meta?: any;  
};  
  
export default function AdminAuditPage() {  
 const [rows, setRows] = useState<Row[]>([]);  
 const [cursor, setCursor] = useState<string | null>(null);  
 const [loading, setLoading] = useState(false);  
 const [hasMore, setHasMore] = useState(true);  
 const [err, setErr] = useState<string | null>(null);  
  
 async function load(next?: string | null) {  
 setLoading(true);  
 setErr(null);  
 try {  
 const q = new URLSearchParams({ limit: "50" });  
 if (next) q.set("cursor", next);  
 const r = await fetch(`/api/admin/audit?${q.toString()}`, { cache: "no-store" });  
 const { items, nextCursor } = await r.json();  
 setRows(prev => (next ? [...prev, ...items] : items));  
 setCursor(nextCursor);  
 setHasMore(!!nextCursor);  
 } catch (e: any) {  
 setErr(e.message || "Σφάλμα φόρτωσης");  
 } finally {  
 setLoading(false);  
 }  
 }  
  
 useEffect(() => { load(null); }, []);  
  
 return (  
 <div className="grid gap-4 text-[inherit]">  
 <h1 className="text-xl font-semibold">Αρχεία Καταγραφής</h1>  
  
 {err && <div className="text-sm text-red-600">{err}</div>}  
  
 {/\* -------- MOBILE LIST (2 σειρές) -------- \*/}  
 <section className="sm:hidden grid gap-3">  
 {rows.map((r) => {  
 const created = new Date(r.createdAt).toLocaleString();  
 const actor = r.actor ? `${r.actor.email}${r.actor.name ? ` (${r.actor.name})` : ""}` : "—";  
 const metaPreview = jsonPreview(r.meta);  
 return (  
 <div key={r.id} className="rounded-2xl border border-[color:var(--border)] bg-[color:var(--card,#fff)] p-3">  
 {/\* Row 1: Action + time, and actor below \*/}  
 <div className="flex items-start justify-between gap-3">  
 <div className="min-w-0">  
 <div className="font-medium break-words">{humanizeAction(r.action)}</div>  
 <div className="text-xs text-gray-600 break-words">{actor}</div>  
 </div>  
 <div className="shrink-0 text-xs text-gray-600 whitespace-nowrap">{created}</div>  
 </div>  
  
 {/\* Row 2: Target + meta preview \*/}  
 <div className="mt-3 grid gap-2">  
 <div className="text-sm break-words">  
 <span className="text-gray-600">Στόχος:</span>{" "}  
 {r.target ?? "—"} {r.targetId ? <span className="text-[color:var(--muted)]">#{r.targetId}</span> : null}  
 </div>  
  
 <pre className="text-xs bg-black/5 rounded p-2 max-h-[12rem] overflow-auto whitespace-pre-wrap break-words">  
 {metaPreview || "—"}  
 </pre>  
 </div>  
 </div>  
 );  
 })}  
  
 <div className="mt-1 flex justify-end">  
 {hasMore && (  
 <button  
 onClick={() => load(cursor)}  
 disabled={loading}  
 className="rounded border border-[color:var(--border)] px-3 py-2 text-sm hover:bg-gray-50 disabled:opacity-50"  
 >  
 {loading ? "Φόρτωση…" : "Φόρτωση περισσότερων"}  
 </button>  
 )}  
 </div>  
 </section>  
  
 {/\* -------- DESKTOP/TABLET TABLE -------- \*/}  
 <section className="hidden sm:block rounded-2xl border border-[color:var(--border)] bg-[color:var(--card,#fff)] p-4">  
 <div className="overflow-hidden">  
 <table className="w-full table-fixed text-sm text-[inherit]">  
 <thead className="bg-gray-50 text-gray-700">  
 <tr className="text-left">  
 <Th className="w-[18%]">Ώρα</Th>  
 <Th className="w-[18%]">Ενέργεια</Th>  
 <Th className="w-[22%]">Χρήστης</Th>  
 <Th className="w-[17%]">Στόχος</Th>  
 <Th className="w-[25%]">Meta</Th>  
 </tr>  
 </thead>  
 <tbody className="divide-y divide-gray-100">  
 {rows.map((r) => (  
 <tr key={r.id} className="align-top">  
 <Td className="whitespace-nowrap">{new Date(r.createdAt).toLocaleString()}</Td>  
 <Td className="font-medium whitespace-normal break-words">{humanizeAction(r.action)}</Td>  
 <Td className="whitespace-normal break-words">  
 {r.actor ? (  
 <>  
 {r.actor.email}  
 {r.actor.name ? <span className="text-gray-500"> ({r.actor.name})</span> : null}  
 </>  
 ) : "—"}  
 </Td>  
 <Td className="whitespace-normal break-words">  
 {r.target ?? "—"}{" "}  
 {r.targetId ? <span className="text-[color:var(--muted)]">#{r.targetId}</span> : null}  
 </Td>  
 <Td className="whitespace-normal break-words">  
 <pre className="text-xs bg-black/5 rounded p-2 max-h-[10rem] overflow-auto">  
 {jsonPreview(r.meta) || "—"}  
 </pre>  
 </Td>  
 </tr>  
 ))}  
 </tbody>  
 </table>  
 </div>  
  
 <div className="mt-3 flex justify-end">  
 {hasMore && (  
 <button  
 onClick={() => load(cursor)}  
 disabled={loading}  
 className="rounded border border-[color:var(--border)] px-3 py-2 text-sm hover:bg-gray-50 disabled:opacity-50"  
 >  
 {loading ? "Φόρτωση…" : "Φόρτωση περισσότερων"}  
 </button>  
 )}  
 </div>  
 </section>  
 </div>  
 );  
}  
  
/\* helpers \*/  
function Th({ children, className = "" }: { children: React.ReactNode; className?: string }) {  
 return <th className={`px-3 py-3 font-semibold ${className}`}>{children}</th>;  
}  
function Td({ children, className = "" }: { children: React.ReactNode; className?: string }) {  
 return <td className={`px-3 py-3 ${className}`}>{children}</td>;  
}  
  
function humanizeAction(a: string) {  
 switch (a) {  
 case "FILE\_UPLOADED": return "Ανέβασμα αρχείου";  
 case "FILE\_ASSIGNED": return "Ανάθεση αρχείου";  
 case "DOWNLOAD\_GRANTED": return "Παραχώρηση λήψης";  
 case "USER\_CREATED": return "Δημιουργία χρήστη";  
 case "APIKEY\_CREATED": return "Δημιουργία API Key";  
 case "APIKEY\_REVOKED": return "Ανάκληση API Key";  
 case "SUBSCRIPTION\_TOGGLED": return "Εναλλαγή συνδρομής";  
 case "SUPPORT\_TICKET": return "Αίτημα υποστήριξης";  
 default: return a;  
 }  
}  
  
function jsonPreview(meta: any) {  
 if (!meta) return "";  
 try {  
 const s = JSON.stringify(meta, null, 2);  
 // keep it compact for preview  
 return s.length > 1000 ? s.slice(0, 1000) + " …" : s;  
 } catch {  
 return String(meta);  
 }  
}

## src\app\(admin)\admin\files\page.tsx

// src/app/(admin)/admin/files/page.tsx  
"use client";  
  
import { useEffect, useState } from "react";  
  
type FileRow = {  
 id: string;  
 title: string;  
 createdAt: string;  
 url?: string | null;  
 assignments?: { user: { id: string; email: string; name?: string | null } }[];  
};  
  
type UserRow = { id: string; email: string; name: string | null; status?: string };  
  
export default function AdminFilesPage() {  
 const [files, setFiles] = useState<FileRow[]>([]);  
 const [users, setUsers] = useState<UserRow[]>([]);  
 const [loading, setLoading] = useState(true);  
  
 async function load() {  
 setLoading(true);  
 const f = await fetch("/api/files?scope=all", { cache: "no-store" });  
 if (f.ok) setFiles((await f.json()).files);  
 const u = await fetch("/api/admin/users", { cache: "no-store" });  
 if (u.ok) setUsers((await u.json()).users);  
 setLoading(false);  
 }  
 useEffect(() => { load(); }, []);  
  
 async function assign(fileId: string, userId: string) {  
 if (!userId) return;  
 const res = await fetch("/api/assignments", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ fileId, userId }),  
 });  
 if (!res.ok) alert("Αποτυχία ανάθεσης"); else load();  
 }  
  
 return (  
 <div className="grid gap-4 text-[inherit]">  
 <h2 className="text-xl font-semibold">Όλα τα αρχεία</h2>  
  
 {loading ? (  
 <div className="text-sm text-[color:var(--muted)]">Φόρτωση…</div>  
 ) : (  
 <>  
 {/\* --- MOBILE LIST (2 σειρές) --- \*/}  
 <section className="sm:hidden grid gap-3">  
 {files.map((f) => {  
 const assigned = (f.assignments || []).map(a => a.user.email).join(", ");  
 return (  
 <div key={f.id} className="rounded-2xl border border-[color:var(--border)] bg-[color:var(--card,#fff)] p-3">  
 {/\* Row 1: Τίτλος + (προαιρετικά) ημερομηνία \*/}  
 <div className="flex items-start justify-between gap-3">  
 <div className="min-w-0">  
 <div className="font-medium break-words">{f.title}</div>  
 <div className="text-xs text-gray-600">{new Date(f.createdAt).toLocaleString()}</div>  
 </div>  
 {f.url ? (  
 <a  
 href={f.url}  
 target="\_blank"  
 rel="noreferrer"  
 className="shrink-0 rounded border px-3 py-1 hover:bg-black/5"  
 >  
 Λήψη  
 </a>  
 ) : null}  
 </div>  
  
 {/\* Row 2: Assigned + assign form \*/}  
 <div className="mt-3 grid grid-cols-1 gap-2">  
 <div className="text-sm whitespace-normal break-words">  
 <span className="text-gray-600">Ανατεθειμένο σε:</span>{" "}  
 {assigned || <span className="text-[color:var(--muted)]">Δεν έχει ανατεθεί</span>}  
 </div>  
  
 <form  
 onSubmit={e => {  
 e.preventDefault();  
 const userId = (new FormData(e.currentTarget).get("userId") as string) || "";  
 assign(f.id, userId);  
 }}  
 className="flex flex-wrap gap-2"  
 >  
 <select name="userId" className="flex-1 min-w-[50%] border rounded px-2 py-1 text-[inherit] bg-white/90">  
 <option value="">Επιλογή χρήστη…</option>  
 {users  
 .filter(u => u.status === "ACTIVE")  
 .map(u => (  
 <option key={u.id} value={u.id}>  
 {u.email}{u.name ? ` (${u.name})` : ""}  
 </option>  
 ))}  
 </select>  
 <button className="rounded bg-[color:var(--brand)] text-black px-3 py-1 hover:opacity-90">  
 Ανάθεση  
 </button>  
 </form>  
 </div>  
 </div>  
 );  
 })}  
 </section>  
  
 {/\* --- DESKTOP/TABLET TABLE --- \*/}  
 <section className="hidden sm:block rounded-2xl border border-[color:var(--border)] bg-[color:var(--card,#fff)] p-4">  
 <div className="overflow-hidden">  
 <table className="w-full table-fixed text-sm text-[inherit]">  
 <thead className="bg-gray-50 text-gray-700">  
 <tr className="text-left">  
 <Th className="w-[35%]">Τίτλος</Th>  
 <Th className="w-[20%]">Ημερομηνία δημιουργίας</Th>  
 <Th className="w-[25%]">Ανατεθειμένο σε</Th>  
 <Th className="w-[20%]">Ανάθεση</Th>  
 <Th className="w-[15%]">Ενέργειες</Th>  
 </tr>  
 </thead>  
 <tbody className="divide-y divide-gray-100">  
 {files.map(f => {  
 const assigned = (f.assignments || []).map(a => a.user.email).join(", ");  
 return (  
 <tr key={f.id} className="align-top">  
 <Td className="whitespace-normal break-words">{f.title}</Td>  
 <Td className="whitespace-nowrap">{new Date(f.createdAt).toLocaleString()}</Td>  
 <Td className="whitespace-normal break-words">  
 {assigned ? assigned : <span className="text-[color:var(--muted)]">Δεν έχει ανατεθεί</span>}  
 </Td>  
 <Td>  
 <form  
 onSubmit={e => {  
 e.preventDefault();  
 const userId = (new FormData(e.currentTarget).get("userId") as string) || "";  
 assign(f.id, userId);  
 }}  
 className="flex gap-2"  
 >  
 <select name="userId" className="border rounded px-2 py-1 text-[inherit] bg-white/90">  
 <option value="">Επιλογή χρήστη…</option>  
 {users  
 .filter(u => u.status === "ACTIVE")  
 .map(u => (  
 <option key={u.id} value={u.id}>  
 {u.email}{u.name ? ` (${u.name})` : ""}  
 </option>  
 ))}  
 </select>  
 <button className="rounded bg-[color:var(--brand)] text-black px-3 py-1 hover:opacity-90">  
 Ανάθεση  
 </button>  
 </form>  
 </Td>  
 <Td>  
 {f.url ? (  
 <a  
 href={f.url}  
 target="\_blank"  
 rel="noreferrer"  
 className="rounded border px-3 py-1 hover:bg-black/5"  
 >  
 Λήψη  
 </a>  
 ) : (  
 <span className="text-[color:var(--muted)]">Δεν υπάρχει URL αρχείου</span>  
 )}  
 </Td>  
 </tr>  
 );  
 })}  
 </tbody>  
 </table>  
 </div>  
 </section>  
 </>  
 )}  
 </div>  
 );  
}  
  
/\* helpers \*/  
function Th({ children, className = "" }: { children: React.ReactNode; className?: string }) {  
 return <th className={`px-3 py-3 font-semibold ${className}`}>{children}</th>;  
}  
function Td({ children, className = "" }: { children: React.ReactNode; className?: string }) {  
 return <td className={`px-3 py-3 ${className}`}>{children}</td>;  
}

## src\app\(admin)\admin\page.tsx

// src/app/(admin)/admin/page.tsx  
"use client";  
  
import { useEffect, useState } from "react";  
import Link from "next/link";  
  
  
type Stats = { users: number; pending: number; files: number };  
type AuditRow = {  
 id: string;  
 createdAt: string;  
 action: string;  
 target?: string | null;  
 targetId?: string | null;  
 actor?: { id: string; email: string; name?: string | null } | null;  
 meta?: any;  
};  
  
export default function AdminDashboard() {  
 const [stats, setStats] = useState<Stats | null>(null);  
 const [loadingStats, setLoadingStats] = useState(true);  
  
 const [rows, setRows] = useState<AuditRow[]>([]);  
 const [loadingAudit, setLoadingAudit] = useState(true);  
  
 useEffect(() => {  
 // Στατιστικά  
 (async () => {  
 try {  
 const r = await fetch("/api/stats", { cache: "no-store" });  
 if (r.ok) setStats(await r.json());  
 } catch {}  
 setLoadingStats(false);  
 })();  
  
 // Πρόσφατη δραστηριότητα (20 εγγραφές)  
 (async () => {  
 try {  
 const r = await fetch("/api/admin/audit?limit=20", { cache: "no-store" });  
 if (r.ok) {  
 const { items } = await r.json();  
 setRows(items as AuditRow[]);  
 }  
 } catch {}  
 setLoadingAudit(false);  
 })();  
 }, []);  
  
 return (  
 <div className="grid gap-4">  
 {/\* Επάνω κάρτες: 1-στήλη στο κινητό, 3-στήλες σε desktop \*/}  
 <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3 text-sm text-[inherit]">  
 <Card  
 title="Χρήστες"  
 value={loadingStats || !stats ? "—" : stats.users.toLocaleString()}  
 subtitle="Σύνολο εγγεγραμμένων"  
 />  
 <Card  
 title="Εκκρεμείς εγκρίσεις"  
 value={loadingStats || !stats ? "—" : stats.pending.toLocaleString()}  
 subtitle="Αναμονή ενεργοποίησης"  
 />  
 <Card  
 title="Αρχεία"  
 value={loadingStats || !stats ? "—" : stats.files.toLocaleString()}  
 subtitle="Σύνολο ανεβασμένων"  
 />  
 </div>  
  
 {/\* Γρήγορες ενέργειες \*/}  
 <section className="rounded-2xl border border-[var(--border,#E5E7EB)] bg-[var(--card,#fff)] p-4">  
 <div className="text-sm text-[var(--muted,#6B7280)]">Γρήγορες ενέργειες</div>  
 <div className="text-xs text-[var(--muted,#6B7280)] mb-3">Συνηθισμένες εργασίες</div>  
 <div className="flex flex-wrap gap-2">  
 <LinkBtn href="/admin/users">Διαχείριση χρηστών</LinkBtn>  
 <LinkBtn href="/admin/uploads">Ανέβασμα αρχείου</LinkBtn>  
 <LinkBtn href="/admin/audit">Προβολή καταγραφών</LinkBtn>  
 <LinkBtn href="/admin/files">Αρχεία</LinkBtn>  
 </div>  
 </section>  
  
 {/\* Πρόσφατη δραστηριότητα — ΠΙΝΑΚΑΣ ΦΙΛΙΚΟΣ ΣΕ ΚΙΝΗΤΟ \*/}  
 <section className="rounded-2xl border border-[var(--border,#E5E7EB)] bg-[var(--card,#fff)]">  
 <div className="flex items-center justify-between gap-2 px-4 pt-4">  
 <h2 className="text-base font-semibold text-[inherit]">Πρόσφατη δραστηριότητα</h2>  
 <Link href="/admin/audit" className="text-sm underline underline-offset-4">  
 Προβολή όλων  
 </Link>  
 </div>  
  
 {loadingAudit ? (  
 <div className="px-4 py-6 text-sm text-[var(--muted,#6B7280)]">Φόρτωση…</div>  
 ) : rows.length === 0 ? (  
 <div className="px-4 py-6 text-sm text-[var(--muted,#6B7280)]">Δεν υπάρχουν δραστηριότητες ακόμη.</div>  
 ) : (  
 <div className="mt-3 overflow-hidden">  
 {/\* mobile-first table \*/}  
 <table className="w-full table-fixed text-sm text-[inherit]">  
 <thead className="bg-gray-50 text-gray-700">  
 <tr className="text-left">  
 <Th className="w-[40%]">Ενέργεια</Th>  
 <Th className="w-[35%]">Χρήστης</Th>  
 <Th className="w-[25%] hidden sm:table-cell">Ώρα</Th>  
 </tr>  
 </thead>  
 <tbody className="divide-y divide-gray-100">  
 {rows.map((r) => (  
 <tr key={r.id} className="align-top">  
 {/\* Ενέργεια + στόχος/λεπτομέρειες \*/}  
 <Td className="whitespace-normal break-words">  
 <div className="font-medium text-[inherit]">{humanizeAction(r.action)}</div>  
 <div className="text-xs text-gray-500 mt-1">  
 {r.target ? r.target : "—"}  
 {r.targetId ? ` · ${r.targetId}` : ""}  
 </div>  
 {r.meta?.subject && (  
 <div className="text-xs text-gray-600 mt-1">{String(r.meta.subject).slice(0, 100)}</div>  
 )}  
 </Td>  
  
 {/\* Χρήστης \*/}  
 <Td className="whitespace-normal break-words">  
 {r.actor?.name || r.actor?.email || "Σύστημα"}  
 {r.actor?.email && (  
 <div className="text-xs text-gray-500">{r.actor.email}</div>  
 )}  
 </Td>  
  
 {/\* Ώρα (κρυφό στο κινητό) \*/}  
 <Td className="hidden sm:table-cell whitespace-nowrap">  
 {new Date(r.createdAt).toLocaleString()}  
 </Td>  
 </tr>  
 ))}  
 </tbody>  
 </table>  
 </div>  
 )}  
 </section>  
 </div>  
 );  
}  
  
/\* --------------------- Βοηθητικά components --------------------- \*/  
  
function Card({  
 title,  
 value,  
 subtitle,  
 right,  
}: {  
 title: string;  
 value?: string | number;  
 subtitle?: string;  
 right?: React.ReactNode;  
}) {  
 return (  
 <section  
 className="rounded-2xl border border-[var(--border,#E5E7EB)] bg-[var(--card,#fff)] shadow-sm p-4"  
 style={{ color: "var(--app-fg,#0A0F2C)" }} // force dark text inside the card  
 >  
 <div className="flex items-start justify-between gap-2">  
 <div>  
 {/\* muted (NOT white) \*/}  
 <h3 className="text-xs font-medium" style={{ color: "var(--app-muted,#6B7280)" }}>  
 {title}  
 </h3>  
  
 {value !== undefined && (  
 <div className="text-3xl font-semibold mt-2" style={{ color: "var(--app-fg,#0A0F2C)" }}>  
 {value}  
 </div>  
 )}  
  
 {subtitle && (  
 <p className="text-xs mt-1" style={{ color: "var(--app-muted,#6B7280)" }}>  
 {subtitle}  
 </p>  
 )}  
 </div>  
 {right}  
 </div>  
 </section>  
 );  
}  
  
function LinkBtn({ href, children }: { href: string; children: React.ReactNode }) {  
 return (  
 <Link href={href} className="rounded-xl border px-3 py-2 text-sm hover:bg-gray-50 text-[inherit]">  
 {children}  
 </Link>  
 );  
}  
  
function Th({ children, className = "" }: { children: React.ReactNode; className?: string }) {  
 return <th className={`px-3 py-3 font-semibold ${className}`}>{children}</th>;  
}  
function Td({ children, className = "" }: { children: React.ReactNode; className?: string }) {  
 return <td className={`px-3 py-3 ${className}`}>{children}</td>;  
}  
  
/\*\* Μετατροπή κωδικών ενεργειών σε ανθρώπινους τίτλους (ελληνικά) \*/  
function humanizeAction(a: string) {  
 switch (a) {  
 case "FILE\_UPLOADED":  
 return "Ανέβασμα αρχείου";  
 case "FILE\_ASSIGNED":  
 return "Ανάθεση αρχείου";  
 case "DOWNLOAD\_GRANTED":  
 return "Παραχώρηση λήψης";  
 case "USER\_CREATED":  
 return "Δημιουργία χρήστη";  
 case "APIKEY\_CREATED":  
 return "Δημιουργία API Key";  
 case "APIKEY\_REVOKED":  
 return "Ανάκληση API Key";  
 case "SUBSCRIPTION\_TOGGLED":  
 return "Εναλλαγή συνδρομής";  
 case "SUPPORT\_TICKET":  
 return "Αίτημα υποστήριξης";  
 default:  
 return a;  
 }  
}

## src\app\(admin)\admin\support\page.tsx

// src/app/(admin)/admin/support/page.tsx  
import { redirect } from "next/navigation";  
import { currentUser } from "@/lib/auth-helpers";  
import { prisma } from "@/lib/prisma";  
import Link from "next/link";  
  
export const dynamic = "force-dynamic";  
// 👇 Force Node.js runtime so Prisma doesn't run on Edge (prevents 500)  
export const runtime = "nodejs";  
  
type SearchParams = {  
 page?: string;  
 q?: string;  
 priority?: "normal" | "high" | "urgent" | "";  
 sort?: "newest" | "oldest";  
};  
  
const PAGE\_SIZE = 20;  
  
function toInt(v: string | undefined, fallback: number) {  
 const n = Number(v);  
 return Number.isFinite(n) && n > 0 ? Math.floor(n) : fallback;  
}  
  
function buildWhere(sp: SearchParams) {  
 const where: any = { action: "SUPPORT\_TICKET" };  
 if (sp.q) {  
 const q = sp.q.trim();  
 if (q) {  
 where.OR = [  
 { meta: { path: ["subject"], string\_contains: q, mode: "insensitive" } },  
 { meta: { path: ["message"], string\_contains: q, mode: "insensitive" } },  
 { meta: { path: ["userEmail"], string\_contains: q, mode: "insensitive" } },  
 { meta: { path: ["userName"], string\_contains: q, mode: "insensitive" } },  
 ];  
 }  
 }  
 if (sp.priority) {  
 where.AND = [  
 ...(where.AND ?? []),  
 { meta: { path: ["priority"], equals: sp.priority } },  
 ];  
 }  
 return where;  
}  
  
export default async function AdminSupportPage({ searchParams }: { searchParams: SearchParams }) {  
 const me = await currentUser();  
 // 👇 keep paths consistent with your app structure (/dashboard/admin/…)  
 if (!me) redirect("/login?next=/dashboard/admin/support");  
 if (me.role !== "ADMIN") redirect("/dashboard");  
  
 const page = toInt(searchParams.page, 1);  
 const take = PAGE\_SIZE;  
 const skip = (page - 1) \* take;  
 const sort = searchParams.sort === "oldest" ? "asc" : "desc";  
 const where = buildWhere(searchParams);  
  
 let total = 0;  
 let rows:  
 { id: string; createdAt: Date; meta: unknown }[] = [];  
 try {  
 [total, rows] = await Promise.all([  
 prisma.auditLog.count({ where }),  
 prisma.auditLog.findMany({  
 where,  
 orderBy: { createdAt: sort },  
 take,  
 skip,  
 select: { id: true, createdAt: true, meta: true },  
 }),  
 ]);  
 } catch (e) {  
 // If something goes wrong, fail gracefully instead of a 500  
 total = 0;  
 rows = [];  
 }  
  
 const pages = Math.max(1, Math.ceil(total / take));  
 const q = searchParams.q ?? "";  
 const priority = (searchParams.priority ?? "") as "" | "normal" | "high" | "urgent";  
  
 const linkWith = (patch: Partial<SearchParams>) => {  
 const sp = new URLSearchParams();  
 if (q) sp.set("q", q);  
 if (priority) sp.set("priority", priority);  
 sp.set("sort", searchParams.sort === "oldest" ? "oldest" : "newest");  
 sp.set("page", String(page));  
 for (const [k, v] of Object.entries(patch)) {  
 if (v === "" || v == null) continue;  
 sp.set(k, String(v));  
 }  
 // 👇 correct URL base for your routing  
 return `/dashboard/admin/support?${sp.toString()}`;  
 };  
  
 return (  
 <main className="mx-auto max-w-6xl px-3 md:px-6 py-6 text-[inherit]">  
 <div className="flex flex-col gap-3 md:flex-row md:items-center md:justify-between">  
 <div>  
 <h1 className="text-xl md:text-2xl font-semibold text-[inherit]">Αιτήματα Υποστήριξης</h1>  
 <p className="mt-1 text-sm text-gray-600">{total.toLocaleString()} σύνολο</p>  
 </div>  
  
 <form className="flex flex-wrap items-center gap-2" action="/dashboard/admin/support" method="get">  
 <input  
 type="text"  
 name="q"  
 defaultValue={q}  
 placeholder="Αναζήτηση θέμα, μήνυμα ή χρήστη…"  
 className="w-full sm:w-64 rounded-xl border border-gray-300 px-3 py-2 text-sm outline-none focus:ring text-[inherit] bg-white/90"  
 />  
 <select  
 name="priority"  
 defaultValue={priority}  
 className="rounded-xl border border-gray-300 px-3 py-2 text-sm bg-white/90 text-[inherit]"  
 >  
 <option value="">Όλες οι προτεραιότητες</option>  
 <option value="normal">Κανονική</option>  
 <option value="high">Υψηλή</option>  
 <option value="urgent">Επείγον</option>  
 </select>  
 <select  
 name="sort"  
 defaultValue={searchParams.sort === "oldest" ? "oldest" : "newest"}  
 className="rounded-xl border border-gray-300 px-3 py-2 text-sm bg-white/90 text-[inherit]"  
 >  
 <option value="newest">Νεότερα πρώτα</option>  
 <option value="oldest">Παλαιότερα πρώτα</option>  
 </select>  
 <button  
 type="submit"  
 className="rounded-xl bg-black px-4 py-2 text-sm font-medium text-white hover:opacity-90"  
 >  
 Εφαρμογή  
 </button>  
 </form>  
 </div>  
  
 <div className="mt-4 overflow-hidden rounded-2xl border border-gray-200">  
 <table className="w-full table-fixed text-sm text-[inherit]">  
 <thead className="bg-gray-50 text-gray-700">  
 <tr className="text-left font-semibold">  
 <th className="px-3 md:px-4 py-3 w-[20%]">Ημερ./Ώρα</th>  
 <th className="px-3 md:px-4 py-3 w-[35%]">Θέμα</th>  
 <th className="px-3 md:px-4 py-3 hidden md:table-cell w-[20%]">Από</th>  
 <th className="px-3 md:px-4 py-3 w-[15%]">Προτεραιότητα</th>  
 <th className="px-3 md:px-4 py-3 hidden md:table-cell w-[25%]">Προεπισκόπηση</th>  
 </tr>  
 </thead>  
 <tbody className="divide-y divide-gray-100 bg-white">  
 {rows.map((r) => {  
 const meta: any = (r as any).meta ?? {};  
 const subject = String(meta.subject ?? "");  
 const message = String(meta.message ?? "");  
 const preview = message.length > 120 ? message.slice(0, 120) + "…" : message;  
 const pri = String(meta.priority ?? "normal");  
 const userEmail = String(meta.userEmail ?? "");  
 const userName = String(meta.userName ?? "");  
 const created = new Date((r as any).createdAt).toLocaleString();  
  
 return (  
 <tr key={r.id} className="align-top">  
 <td className="px-3 md:px-4 py-3 text-xs md:text-sm text-gray-600 whitespace-nowrap">{created}</td>  
  
 <td className="px-3 md:px-4 py-3 whitespace-normal break-words">  
 <div className="font-medium text-[inherit]">{subject || <span className="text-gray-400">—</span>}</div>  
 <div className="mt-1 text-xs text-gray-500 md:hidden">  
 {userName || "Άγνωστο"}{userEmail ? ` · ${userEmail}` : ""}  
 </div>  
 <div className="mt-1 text-xs text-gray-600 md:hidden">{preview || "—"}</div>  
 </td>  
  
 <td className="px-3 md:px-4 py-3 hidden md:table-cell text-sm">  
 <div className="flex flex-col">  
 <span className="font-medium text-[inherit]">{userName || "Άγνωστο"}</span>  
 <span className="text-gray-500">{userEmail}</span>  
 </div>  
 </td>  
  
 <td className="px-3 md:px-4 py-3">  
 <span  
 className={[  
 "nowrap inline-flex rounded-full px-2 py-1 text-xs font-medium",  
 pri === "urgent"  
 ? "bg-red-100 text-red-700"  
 : pri === "high"  
 ? "bg-amber-100 text-amber-700"  
 : "bg-gray-100 text-gray-700",  
 ].join(" ")}  
 >  
 {pri === "urgent" ? "Επείγον" : pri === "high" ? "Υψηλή" : "Κανονική"}  
 </span>  
 </td>  
  
 <td className="px-3 md:px-4 py-3 hidden md:table-cell text-gray-600 whitespace-normal break-words">  
 {preview || "—"}  
 </td>  
 </tr>  
 );  
 })}  
 {rows.length === 0 && (  
 <tr>  
 <td className="px-4 py-8 text-center text-sm text-gray-500" colSpan={5}>  
 Δεν βρέθηκαν αιτήματα.  
 </td>  
 </tr>  
 )}  
 </tbody>  
 </table>  
 </div>  
  
 {/\* Σελιδοποίηση \*/}  
 <div className="mt-4 flex items-center justify-between">  
 <div className="text-sm text-gray-600">Σελίδα {page} από {pages}</div>  
 <div className="flex items-center gap-2">  
 <Link  
 aria-disabled={page <= 1}  
 className={`rounded-xl border px-3 py-1.5 text-sm ${page <= 1 ? "pointer-events-none opacity-40" : "hover:bg-gray-50"}`}  
 href={linkWith({ page: String(page - 1) })}  
 >  
 Προηγούμενη  
 </Link>  
 <Link  
 aria-disabled={page >= pages}  
 className={`rounded-xl border px-3 py-1.5 text-sm ${page >= pages ? "pointer-events-none opacity-40" : "hover:bg-gray-50"}`}  
 href={linkWith({ page: String(page + 1) })}  
 >  
 Επόμενη  
 </Link>  
 </div>  
 </div>  
 </main>  
 );  
}

## src\app\(admin)\admin\uploads\page.tsx

// src/app/(admin)/uploads/page.tsx  
import React from "react";  
import { cookies } from "next/headers";  
import UploadForm from "@/components/UploadForm";  
import FilesTable from "@/components/FilesTable";  
  
export const dynamic = "force-dynamic";  
  
async function fetchAdminFiles() {  
 const base = process.env.NEXTAUTH\_URL || "http://localhost:3000";  
 const cookie = cookies().toString();  
  
 const res = await fetch(`${base}/api/files?scope=all`, {  
 cache: "no-store",  
 headers: { cookie },  
 });  
  
 if (!res.ok) {  
 const detail = await res.text().catch(() => "");  
 throw new Error(`Failed to load files (${res.status}): ${detail}`);  
 }  
 const data = await res.json();  
 return data.files as any[];  
}  
  
export default async function AdminUploadsPage() {  
 const files = await fetchAdminFiles();  
  
 return (  
 <div className="mx-auto max-w-5xl space-y-6 p-6">  
 <header className="flex items-end justify-between">  
 <div>  
 <h1 className="text-2xl font-semibold">Admin Uploads</h1>  
 <p className="text-sm text-gray-500">Upload Excel files and manage all users’ files.</p>  
 </div>  
 </header>  
  
 {/\* Client upload form \*/}  
 <UploadForm  
 buttonText="Upload Excel"  
 onUploaded={() => {  
 // No-op here; table refresh is handled by manual reload or you can enhance with SWR.  
 // Keeping this simple and drop-in friendly.  
 }}  
 />  
  
 {/\* Files table (admin mode shows assignees) \*/}  
 <FilesTable initialFiles={files} adminMode />  
 </div>  
 );  
}

## src\app\(admin)\admin\users\page.tsx

// src/app/(admin)/admin/users/page.tsx  
"use client";  
  
import { useEffect, useMemo, useState } from "react";  
  
type User = {  
 id: string;  
 name: string | null;  
 email: string;  
 role: "USER" | "ADMIN";  
 subscriptionActive: boolean;  
 status: "PENDING" | "ACTIVE" | "SUSPENDED";  
 createdAt: string;  
};  
  
export default function AdminUsersPage() {  
 const [users, setUsers] = useState<User[]>([]);  
 const [loading, setLoading] = useState(true);  
 const [err, setErr] = useState<string | null>(null);  
 const [filter, setFilter] = useState<"ALL" | "PENDING" | "ACTIVE" | "SUSPENDED">("ALL");  
  
 const [form, setForm] = useState({  
 name: "",  
 email: "",  
 password: "",  
 role: "USER" as "USER" | "ADMIN",  
 subscriptionActive: false,  
 });  
 const canCreate = useMemo(() => {  
 return form.name.trim().length >= 2 && /\S+@\S+\.\S+/.test(form.email) && form.password.length >= 6;  
 }, [form]);  
  
 async function load() {  
 setLoading(true); setErr(null);  
 try {  
 const q = filter === "ALL" ? "" : `?status=${filter}`;  
 const res = await fetch(`/api/admin/users${q}`, { cache: "no-store" });  
 const json = await res.json();  
 if (!res.ok) throw new Error(json?.error || "Αποτυχία φόρτωσης χρηστών");  
 setUsers(json.users);  
 } catch (e: any) { setErr(e.message); }  
 finally { setLoading(false); }  
 }  
 useEffect(() => { load(); }, [filter]);  
 useEffect(() => { load(); }, []);  
  
 async function toggleSubscription(u: User) {  
 const optimistic = users.map(x => x.id === u.id ? { ...x, subscriptionActive: !u.subscriptionActive } : x);  
 setUsers(optimistic);  
 const res = await fetch(`/api/admin/users/${u.id}`, {  
 method: "PATCH",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ subscriptionActive: !u.subscriptionActive }),  
 });  
 if (!res.ok) load();  
 }  
  
 async function changeRole(u: User, role: "USER" | "ADMIN") {  
 const optimistic = users.map(x => x.id === u.id ? { ...x, role } : x);  
 setUsers(optimistic);  
 const res = await fetch(`/api/admin/users/${u.id}`, {  
 method: "PATCH",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ role }),  
 });  
 if (!res.ok) load();  
 }  
  
 async function remove(u: User) {  
 if (!confirm(`Διαγραφή χρήστη ${u.email}; Η ενέργεια δεν μπορεί να αναιρεθεί.`)) return;  
 const optimistic = users.filter(x => x.id !== u.id);  
 setUsers(optimistic);  
 const res = await fetch(`/api/admin/users/${u.id}`, { method: "DELETE" });  
 if (!res.ok) load();  
 }  
  
 async function createUser(e: React.FormEvent) {  
 e.preventDefault();  
 const res = await fetch("/api/admin/users", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify(form),  
 });  
 const json = await res.json();  
 if (!res.ok) { alert(json?.error || "Αποτυχία δημιουργίας χρήστη"); return; }  
 setForm({ name: "", email: "", password: "", role: "USER", subscriptionActive: false });  
 load();  
 }  
  
 async function setStatus(u: User, status: User["status"]) {  
 const optimistic = users.map(x => x.id === u.id ? { ...x, status } : x);  
 setUsers(optimistic);  
 const res = await fetch(`/api/admin/users/${u.id}`, {  
 method: "PATCH",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ status }),  
 });  
 if (!res.ok) load();  
 }  
  
 return (  
 <div className="grid gap-6 text-[inherit]">  
 {/\* Φίλτρα \*/}  
 <div className="flex flex-wrap gap-2">  
 {(["ALL","PENDING","ACTIVE","SUSPENDED"] as const).map(f => (  
 <button  
 key={f}  
 onClick={() => setFilter(f)}  
 className={[  
 "rounded-md border px-3 py-2 text-sm",  
 filter === f  
 ? "bg-[color:var(--brand)] text-black border-transparent"  
 : "border-[color:var(--border)]"  
 ].join(" ")}  
 >  
 {f === "ALL" ? "Όλοι" : f === "PENDING" ? "ΕΚΚΡΕΜΕΙ" : f === "ACTIVE" ? "ΕΝΕΡΓΟΣ" : "ΑΝΑΣΤΟΛΗ"}  
 </button>  
 ))}  
 <div className="ml-auto">  
 <button onClick={load} className="rounded-md border border-[color:var(--border)] px-3 py-2 text-sm">Ανανέωση</button>  
 </div>  
 </div>  
  
 {/\* Δημιουργία χρήστη \*/}  
 <section className="rounded-2xl border border-[color:var(--border)] bg-[color:var(--card,#fff)] p-4">  
 <h2 className="font-semibold mb-3 text-[inherit]">Δημιουργία Χρήστη</h2>  
 <form onSubmit={createUser} className="grid gap-3 md:grid-cols-5">  
 <input  
 className="rounded-md border border-[color:var(--border)] bg-white/90 px-3 py-2 md:col-span-1 text-[inherit]"  
 placeholder="Όνομα"  
 value={form.name}  
 onChange={e => setForm({ ...form, name: e.target.value })}  
 required  
 />  
 <input  
 type="email"  
 className="rounded-md border border-[color:var(--border)] bg-white/90 px-3 py-2 md:col-span-2 text-[inherit]"  
 placeholder="Ηλεκτρονικό ταχυδρομείο"  
 value={form.email}  
 onChange={e => setForm({ ...form, email: e.target.value })}  
 required  
 />  
 <input  
 type="password"  
 className="rounded-md border border-[color:var(--border)] bg-white/90 px-3 py-2 md:col-span-1 text-[inherit]"  
 placeholder="Κωδικός πρόσβασης"  
 value={form.password}  
 onChange={e => setForm({ ...form, password: e.target.value })}  
 required  
 />  
 <div className="flex items-center gap-2 md:col-span-1">  
 <select  
 className="rounded-md border border-[color:var(--border)] bg-white/90 px-3 py-2 text-[inherit]"  
 value={form.role}  
 onChange={e => setForm({ ...form, role: e.target.value as any })}  
 >  
 <option value="USER">USER</option>  
 <option value="ADMIN">ADMIN</option>  
 </select>  
 <label className="inline-flex items-center gap-2 text-sm">  
 <input  
 type="checkbox"  
 checked={form.subscriptionActive}  
 onChange={e => setForm({ ...form, subscriptionActive: e.target.checked })}  
 />  
 Συνδρομή  
 </label>  
 </div>  
 <div className="md:col-span-5">  
 <button  
 type="submit"  
 disabled={!canCreate}  
 className="rounded-md bg-[color:var(--brand)] hover:bg-[color:var(--brand-600)] text-black font-medium px-4 py-2 disabled:opacity-50"  
 >  
 Δημιουργία  
 </button>  
 </div>  
 </form>  
 </section>  
  
 {/\* --- MOBILE LIST (2 σειρές) --- \*/}  
 <section className="sm:hidden grid gap-3">  
 {err ? (  
 <div className="text-sm text-red-600">{err}</div>  
 ) : loading ? (  
 <div className="text-sm text-[color:var(--muted)]">Φόρτωση…</div>  
 ) : users.length === 0 ? (  
 <div className="text-sm text-[color:var(--muted)]">Δεν βρέθηκαν χρήστες.</div>  
 ) : (  
 users.map(u => (  
 <div key={u.id} className="rounded-2xl border border-[color:var(--border)] bg-[color:var(--card,#fff)] p-3">  
 {/\* Row 1: Όνομα / Email \*/}  
 <div className="flex items-start justify-between gap-3">  
 <div className="min-w-0">  
 <div className="font-medium break-words">{u.name || "—"}</div>  
 <div className="text-xs text-gray-600 break-words">{u.email}</div>  
 </div>  
 {/\* Κατάσταση pill \*/}  
 <span  
 className={[  
 "nowrap inline-flex items-center rounded-full px-2 py-0.5 border text-xs",  
 u.status === "ACTIVE"  
 ? "border-green-300 text-green-700 bg-green-50"  
 : u.status === "PENDING"  
 ? "border-amber-300 text-amber-700 bg-amber-50"  
 : "border-red-300 text-red-700 bg-red-50",  
 ].join(" ")}  
 >  
 {u.status === "ACTIVE" ? "ΕΝΕΡΓΟΣ" : u.status === "PENDING" ? "ΕΚΚΡΕΜΕΙ" : "ΑΝΑΣΤΟΛΗ"}  
 </span>  
 </div>  
  
 {/\* Row 2: Ρόλος / Συνδρομή / Ενέργειες \*/}  
 <div className="mt-3 grid grid-cols-2 gap-2">  
 <div className="flex items-center gap-2">  
 <select  
 value={u.role}  
 onChange={e => changeRole(u, e.target.value as "USER" | "ADMIN")}  
 className="w-full rounded-md border border-[color:var(--border)] bg-white/90 px-2 py-1 text-[inherit]"  
 >  
 <option value="USER">USER</option>  
 <option value="ADMIN">ADMIN</option>  
 </select>  
 </div>  
 <label className="flex items-center justify-start gap-2">  
 <input  
 type="checkbox"  
 checked={u.subscriptionActive}  
 onChange={() => toggleSubscription(u)}  
 disabled={u.status !== "ACTIVE"}  
 />  
 <span className="text-sm">{u.subscriptionActive ? "Ενεργό" : "Ανενεργό"}</span>  
 </label>  
  
 <div className="col-span-2 flex flex-wrap gap-2">  
 {u.status !== "ACTIVE" && (  
 <button  
 onClick={() => setStatus(u, "ACTIVE")}  
 className="rounded-md bg-green-600/90 text-white px-3 py-1 hover:bg-green-700"  
 >  
 Έγκριση  
 </button>  
 )}  
 {u.status === "ACTIVE" && (  
 <button  
 onClick={() => setStatus(u, "SUSPENDED")}  
 className="rounded-md bg-yellow-500/90 text-black px-3 py-1 hover:bg-yellow-500"  
 >  
 Αναστολή  
 </button>  
 )}  
 <button  
 onClick={() => remove(u)}  
 className="rounded-md border border-red-200 text-red-700 px-3 py-1 hover:bg-red-50"  
 >  
 Διαγραφή  
 </button>  
 </div>  
 </div>  
 </div>  
 ))  
 )}  
 </section>  
  
 {/\* --- DESKTOP/TABLET TABLE --- \*/}  
 <section className="hidden sm:block rounded-2xl border border-[color:var(--border)] bg-[color:var(--card,#fff)] p-4">  
 {err ? (  
 <div className="text-sm text-red-600">{err}</div>  
 ) : loading ? (  
 <div className="text-sm text-[color:var(--muted)]">Φόρτωση…</div>  
 ) : users.length === 0 ? (  
 <div className="text-sm text-[color:var(--muted)]">Δεν βρέθηκαν χρήστες.</div>  
 ) : (  
 <div className="overflow-hidden">  
 <table className="w-full table-fixed text-sm text-[inherit]">  
 <thead className="bg-gray-50 text-gray-700">  
 <tr className="text-left">  
 <Th className="w-[20%]">Όνομα</Th>  
 <Th className="w-[25%]">Email</Th>  
 <Th className="w-[15%]">Ρόλος</Th>  
 <Th className="w-[15%]">Κατάσταση</Th>  
 <Th className="w-[15%]">Συνδρομή</Th>  
 <Th className="w-[10%]">Δημιουργήθηκε</Th>  
 <Th className="w-[20%]">Ενέργειες</Th>  
 </tr>  
 </thead>  
 <tbody className="divide-y divide-gray-100">  
 {users.map(u => (  
 <tr key={u.id} className="align-top">  
 <Td className="whitespace-normal break-words">{u.name ?? "—"}</Td>  
 <Td className="whitespace-normal break-words">{u.email}</Td>  
 <Td>  
 <select  
 value={u.role}  
 onChange={e => changeRole(u, e.target.value as "USER" | "ADMIN")}  
 className="rounded-md border border-[color:var(--border)] bg-white/90 px-2 py-1 text-[inherit]"  
 >  
 <option value="USER">USER</option>  
 <option value="ADMIN">ADMIN</option>  
 </select>  
 </Td>  
 <Td>  
 <span  
 className={[  
 "nowrap inline-flex items-center rounded-full px-2 py-0.5 border text-xs",  
 u.status === "ACTIVE"  
 ? "border-green-300 text-green-700 bg-green-50"  
 : u.status === "PENDING"  
 ? "border-amber-300 text-amber-700 bg-amber-50"  
 : "border-red-300 text-red-700 bg-red-50",  
 ].join(" ")}  
 >  
 {u.status === "ACTIVE" ? "ΕΝΕΡΓΟΣ" : u.status === "PENDING" ? "ΕΚΚΡΕΜΕΙ" : "ΑΝΑΣΤΟΛΗ"}  
 </span>  
 </Td>  
 <Td>  
 <label className="inline-flex items-center gap-2">  
 <input  
 type="checkbox"  
 checked={u.subscriptionActive}  
 onChange={() => toggleSubscription(u)}  
 disabled={u.status !== "ACTIVE"}  
 />  
 {u.subscriptionActive ? "Ενεργό" : "Ανενεργό"}  
 </label>  
 </Td>  
 <Td className="whitespace-nowrap">  
 {new Date(u.createdAt).toLocaleDateString()}  
 </Td>  
 <Td>  
 <div className="flex flex-wrap gap-2">  
 {u.status !== "ACTIVE" && (  
 <button  
 onClick={() => setStatus(u, "ACTIVE")}  
 className="rounded-md bg-green-600/90 text-white px-3 py-1 hover:bg-green-700"  
 >  
 Έγκριση  
 </button>  
 )}  
 {u.status === "ACTIVE" && (  
 <button  
 onClick={() => setStatus(u, "SUSPENDED")}  
 className="rounded-md bg-yellow-500/90 text-black px-3 py-1 hover:bg-yellow-500"  
 >  
 Αναστολή  
 </button>  
 )}  
 <button  
 onClick={() => remove(u)}  
 className="rounded-md border border-red-200 text-red-700 px-3 py-1 hover:bg-red-50"  
 >  
 Διαγραφή  
 </button>  
 </div>  
 </Td>  
 </tr>  
 ))}  
 </tbody>  
 </table>  
 </div>  
 )}  
 </section>  
 </div>  
 );  
}  
  
/\* helpers \*/  
function Th({ children, className = "" }: { children: React.ReactNode; className?: string }) {  
 return <th className={`px-3 py-3 font-semibold ${className}`}>{children}</th>;  
}  
function Td({ children, className = "" }: { children: React.ReactNode; className?: string }) {  
 return <td className={`px-3 py-3 ${className}`}>{children}</td>;  
}

## src\app\(admin)\dashboard\admin\page.tsx

// src/app/dashboard/page.tsx  
import { redirect } from "next/navigation";  
import { currentUser } from "@/lib/auth-helpers";  
  
export const dynamic = "force-dynamic";  
export const runtime = "nodejs"; // if currentUser hits DB/session on the server  
  
export default async function DashboardIndex() {  
 const me = await currentUser();  
 if (!me) {  
 redirect("/login?next=/dashboard");  
 }  
  
 if (me.role === "ADMIN") {  
 redirect("/dashboard/admin");  
 }  
  
 // Adjust if your user home is different  
 redirect("/dashboard/user");  
}

## src\app\(admin)\layout.tsx

// src/app/(admin)/layout.tsx  
import { redirect } from "next/navigation";  
import AppShell from "@/components/AppShell";  
import { currentUser } from "@/lib/auth-helpers";  
  
export default async function AdminLayout({  
 children,  
}: {  
 children: React.ReactNode;  
}) {  
 const user = await currentUser();  
 if (!user || user.role !== "ADMIN") {  
 redirect("/login?next=/admin");  
 }  
  
 // AppShell will render the Sidebar and content area  
 return <AppShell>{children}</AppShell>;  
}

## src\app\(user)\dashboard\page.tsx

// src/app/(user)/dashboard/page.tsx  
"use client";  
  
import { useEffect, useMemo, useState } from "react";  
import Link from "next/link";  
import TrendMini from "@/components/TrendMini";  
  
type StatsUser = {  
 assignedTotal: number;  
 myUploadsTotal: number; // kept in API but not rendered as a table anymore  
 series: { x: string; y: number }[];  
};  
  
type FileRow = { id: string; title: string; originalName: string; createdAt: string; url: string };  
  
export default function UserDashboard() {  
 const [stats, setStats] = useState<StatsUser | null>(null);  
 const [assigned, setAssigned] = useState<FileRow[]>([]);  
  
 useEffect(() => {  
 (async () => {  
 try {  
 const r = await fetch("/api/stats-user", { cache: "no-store" });  
 if (r.ok) setStats(await r.json());  
 } catch {}  
  
 try {  
 const f = await fetch("/api/files", { cache: "no-store" });  
 if (f.ok) {  
 const json = await f.json();  
 const rows: FileRow[] = (json.files as any[]).map((x) => ({  
 id: x.id,  
 title: x.title,  
 originalName: x.originalName,  
 createdAt: x.createdAt,  
 url: x.url,  
 }));  
 setAssigned(rows.slice(0, 10));  
 }  
 } catch {}  
 })();  
 }, []);  
  
 const totalUploads30d = useMemo(  
 () => (stats?.series || []).reduce((a, b) => a + b.y, 0),  
 [stats]  
 );  
  
 return (  
 <div className="grid gap-4">  
 {/\* Top cards — 1/3 + 2/3 on large screens (no empty columns) \*/}  
 <div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3">  
 {/\* Assigned count \*/}  
 <div className="rounded-[var(--radius)] border border-[color:var(--border)] bg-[color:var(--card)] shadow-sm p-4">  
 <div className="text-sm text-[color:var(--muted)] mb-3">Αρχεία που μου έχουν ανατεθεί</div>  
 <div className="h-1 w-10 rounded bg-[color:var(--primary,#25C3F4)] mb-3" />  
 <div className="text-sm">Σύνολο αναθέσεων</div>  
 <div className="mt-2 text-2xl font-semibold">  
 {stats ? stats.assignedTotal.toLocaleString() : "—"}  
 </div>  
 </div>  
  
 {/\* Trend card (spans 2 columns on lg) \*/}  
 <div className="rounded-[var(--radius)] border border-[color:var(--border)] bg-[color:var(--card)] shadow-sm p-4 lg:col-span-2">  
 <div className="mb-2 flex items-center justify-between">  
 <span className="text-sm text-[color:var(--muted)]">Τάση (30 ημέρες)</span>  
 <span className="text-xs text-[color:var(--muted)]">{totalUploads30d} σύνολο</span>  
 </div>  
 <TrendMini data={stats?.series || []} />  
 </div>  
 </div>  
  
 {/\* Recent assigned files (kept) \*/}  
 <section className="rounded-[var(--radius)] border border-[color:var(--border)] bg-[color:var(--card)] shadow-sm p-0">  
 <div className="flex items-center justify-between px-4 pt-3 pb-2">  
 <h2 className="font-semibold">Πρόσφατα ανατεθειμένα αρχεία</h2>  
 <Link href="/files" className="text-sm underline">Προβολή όλων</Link>  
 </div>  
  
 <div className="px-4 pb-4 overflow-x-auto">  
 {!assigned.length ? (  
 <p className="text-sm text-[color:var(--muted)]">Δεν υπάρχουν αρχεία ακόμη.</p>  
 ) : (  
 <table className="min-w-[720px] w-full text-sm">  
 <thead>  
 <tr className="text-left text-[color:var(--muted)] border-b border-[color:var(--border)]">  
 <th className="py-2 pr-3">Τίτλος</th>  
 <th className="py-2 pr-3">Ανέβηκε</th>  
 <th className="py-2 pr-3">Ενέργειες</th>  
 </tr>  
 </thead>  
 <tbody>  
 {assigned.map((f) => (  
 <tr key={f.id} className="border-b last:border-0 border-[color:var(--border)]">  
 <td className="py-2 pr-3">  
 {f.title} · <span className="text-[color:var(--muted)]">{f.originalName}</span>  
 </td>  
 <td className="py-2 pr-3 whitespace-nowrap">  
 {new Date(f.createdAt).toLocaleString()}  
 </td>  
 <td className="py-2 pr-3">  
 <a  
 href={f.url}  
 className="inline-flex rounded-xl bg-black px-3 py-1.5 text-xs font-semibold text-white hover:opacity-90"  
 >  
 Λήψη  
 </a>  
 </td>  
 </tr>  
 ))}  
 </tbody>  
 </table>  
 )}  
 </div>  
 </section>  
 </div>  
 );  
}

## src\app\(user)\files\page.tsx

// src/app/(user)/files/page.tsx  
import React from "react";  
import { cookies } from "next/headers";  
import FilesTable from "@/components/FilesTable";  
  
export const dynamic = "force-dynamic";  
  
async function fetchMyFiles() {  
 const base = process.env.NEXTAUTH\_URL || "https://hygiene-plus.vercel.app";  
 const cookie = cookies().toString();  
  
 const res = await fetch(`${base}/api/files`, {  
 cache: "no-store",  
 headers: { cookie },  
 });  
  
 if (!res.ok) {  
 const detail = await res.text().catch(() => "");  
 throw new Error(`Failed to load files (${res.status}): ${detail}`);  
 }  
 const data = await res.json();  
 return data.files as any[];  
}  
  
export default async function UserFilesPage() {  
 const files = await fetchMyFiles();  
  
 return (  
 <div className="mx-auto max-w-5xl space-y-6 p-6">  
 <header>  
 <h1 className="text-2xl font-semibold">Τα Αρχεία μου</h1>  
 <p className="text-sm text-gray-500">Αρχεία που ανεβάσατε ή σας ανατέθηκαν.</p>  
 </header>  
  
 <FilesTable  
 initialFiles={files}  
 labels={{  
 search: "Αναζήτηση αρχείων…",  
 countSuffix: "αρχείο(α)",  
 title: "Τίτλος",  
 original: "Αρχικό",  
 uploaded: "Ανέβηκε",  
 size: "Μέγεθος",  
 action: "Ενέργεια",  
 download: "Λήψη",  
 empty: "Δεν βρέθηκαν αρχεία.",  
 }}  
 />  
 </div>  
 );  
}

## src\app\(user)\layout.tsx

// src/app/(user)/layout.tsx  
import { redirect } from "next/navigation";  
import AppShell from "@/components/AppShell";  
import { currentUser } from "@/lib/auth-helpers";  
  
export default async function UserLayout({  
 children,  
}: {  
 children: React.ReactNode;  
}) {  
 const user = await currentUser();  
 if (!user) {  
 redirect("/login?next=/dashboard");  
 }  
  
 return <AppShell>{children}</AppShell>;  
}

## src\app\(user)\support\page.tsx

// src/app/(user)/support/page.tsx  
"use client";  
  
import React from "react";  
  
export default function SupportPage() {  
 const [subject, setSubject] = React.useState("");  
 const [message, setMessage] = React.useState("");  
 const [priority, setPriority] = React.useState<"low" | "normal" | "high">("normal");  
 const [status, setStatus] = React.useState<"idle" | "sending" | "ok" | "error">("idle");  
 const [hint, setHint] = React.useState<string | null>(null);  
  
 async function submit(e: React.FormEvent) {  
 e.preventDefault();  
 setStatus("sending");  
 setHint(null);  
 try {  
 const r = await fetch("/api/support", {  
 method: "POST",  
 headers: { "content-type": "application/json" },  
 body: JSON.stringify({ subject, message, priority }),  
 });  
 const j = await r.json();  
 if (!r.ok) throw new Error(j?.detail || j?.error || "Αποτυχία αποστολής");  
 setStatus("ok");  
 setHint("Το αίτημα καταγράφηκε. Αν χρειάζεστε άμεση επικοινωνία, πατήστε εδώ: ");  
 if (j?.mailto) {  
 const a = document.createElement("a");  
 a.href = j.mailto;  
 a.textContent = "Άνοιγμα email";  
 a.className = "underline";  
 const wrap = document.getElementById("mailto-anchor");  
 if (wrap) {  
 wrap.innerHTML = "";  
 wrap.appendChild(a);  
 }  
 }  
 setSubject("");  
 setMessage("");  
 setPriority("normal");  
 } catch (err: any) {  
 setStatus("error");  
 setHint(err?.message || "Κάτι πήγε στραβά");  
 }  
 }  
  
 return (  
 <div className="mx-auto max-w-3xl space-y-6 p-6">  
 <header>  
 <h1 className="text-2xl font-semibold">Υποστήριξη</h1>  
 <p className="text-sm text-gray-500">Στείλτε μας ένα αίτημα ή χρησιμοποιήστε τα στοιχεία επικοινωνίας.</p>  
 </header>  
  
 <section className="rounded-[var(--radius)] border border-[color:var(--border)] bg-[color:var(--card)] shadow-sm p-4">  
 <h2 className="font-semibold mb-3">Νέο αίτημα</h2>  
 <form onSubmit={submit} className="space-y-3">  
 <div>  
 <label className="block text-sm mb-1">Θέμα</label>  
 <input  
 className="w-full rounded-lg border border-gray-300 bg-white p-2 text-sm"  
 value={subject}  
 onChange={(e) => setSubject(e.currentTarget.value)}  
 required  
 />  
 </div>  
  
 <div>  
 <label className="block text-sm mb-1">Περιγραφή</label>  
 <textarea  
 className="w-full rounded-lg border border-gray-300 bg-white p-2 text-sm min-h-[120px]"  
 value={message}  
 onChange={(e) => setMessage(e.currentTarget.value)}  
 required  
 />  
 </div>  
  
 <div className="flex items-center gap-3">  
 <label className="text-sm">Προτεραιότητα:</label>  
 <select  
 className="rounded-lg border border-gray-300 bg-white p-2 text-sm"  
 value={priority}  
 onChange={(e) => setPriority(e.currentTarget.value as any)}  
 >  
 <option value="low">Χαμηλή</option>  
 <option value="normal">Κανονική</option>  
 <option value="high">Υψηλή</option>  
 </select>  
 <div className="ml-auto flex items-center gap-2">  
 <button  
 type="button"  
 onClick={() => {  
 setSubject("");  
 setMessage("");  
 setPriority("normal");  
 }}  
 className="rounded-xl border border-gray-300 px-4 py-2 text-sm"  
 >  
 Καθαρισμός  
 </button>  
 <button  
 type="submit"  
 disabled={status === "sending"}  
 className="rounded-xl bg-black px-4 py-2 text-sm font-semibold text-white disabled:opacity-60"  
 >  
 {status === "sending" ? "Αποστολή…" : "Υποβολή"}  
 </button>  
 </div>  
 </div>  
  
 {status !== "idle" && (  
 <div className={`text-sm ${status === "ok" ? "text-green-700" : "text-red-600"}`}>  
 {hint} <span id="mailto-anchor" />  
 </div>  
 )}  
 </form>  
 </section>  
  
 <section className="rounded-[var(--radius)] border border-[color:var(--border)] bg-[color:var(--card)] shadow-sm p-4">  
 <h2 className="font-semibold mb-3">Επικοινωνία</h2>  
 <p className="text-sm">  
 Τηλέφωνο: <a className="underline" href="tel:+302100000000">+30 210 000 0000</a><br />  
 Email: <a className="underline" href="mailto:support@hygiene-plus.gr">support@hygiene-plus.gr</a>  
 </p>  
 </section>  
 </div>  
 );  
}

## src\app\api\\_health\env\route.ts

import { NextResponse } from "next/server";  
export const runtime = "nodejs";  
export async function GET() {  
 return NextResponse.json({  
 INTEGRATIONS\_API\_KEY: !!process.env.INTEGRATIONS\_API\_KEY,  
 runtime: process.env.VERCEL ? "vercel" : "local",  
 });  
}

## src\app\api\admin\apikeys\[id]\route.ts

import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireRole } from "@/lib/auth-helpers";  
  
type Params = Promise<{ id: string }>;  
  
export async function PATCH(req: Request, ctx: { params: Params }) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const { id } = await ctx.params;  
 const { isActive } = await req.json();  
 const updated = await prisma.apiKey.update({ where: { id }, data: { isActive: !!isActive } });  
 return NextResponse.json({ ok: true, key: { id: updated.id, isActive: updated.isActive } });  
}  
  
export async function DELETE(\_req: Request, ctx: { params: Params }) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const { id } = await ctx.params;  
 await prisma.apiKey.delete({ where: { id } });  
 return NextResponse.json({ ok: true });  
}

## src\app\api\admin\apikeys\route.ts

import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireRole } from "@/lib/auth-helpers";  
import bcrypt from "bcrypt";  
  
export async function GET() {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const keys = await prisma.apiKey.findMany({  
 orderBy: { createdAt: "desc" },  
 select: { id: true, label: true, isActive: true, createdAt: true, lastUsedAt: true }  
 });  
 return NextResponse.json({ keys });  
}  
  
export async function POST(req: Request) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const { label } = await req.json().catch(() => ({ label: "Integration Key" }));  
 const plain = cryptoRandom(48);  
 const keyHash = await bcrypt.hash(plain, 10);  
  
 const key = await prisma.apiKey.create({  
 data: { label: label || "Integration Key", keyHash },  
 select: { id: true, label: true, isActive: true, createdAt: true }  
 });  
  
 // IMPORTANT: return the plain key only once to admin UI  
 return NextResponse.json({ key, plain }, { status: 201 });  
}  
  
function cryptoRandom(len = 48) {  
 const alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";  
 return Array.from({ length: len }, () => alphabet[Math.floor(Math.random() \* alphabet.length)]).join("");  
}

## src\app\api\admin\audit\route.ts

import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireRole } from "@/lib/auth-helpers";  
  
export async function GET(req: Request) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const { searchParams } = new URL(req.url);  
 const limit = Math.min(Number(searchParams.get("limit") ?? 50), 200);  
 const cursor = searchParams.get("cursor") || undefined;  
  
 const items = await prisma.auditLog.findMany({  
 take: limit + 1,  
 ...(cursor ? { cursor: { id: cursor }, skip: 1 } : {}),  
 orderBy: { createdAt: "desc" },  
 select: {  
 id: true,  
 createdAt: true,  
 action: true,  
 target: true,  
 targetId: true,  
 actor: { select: { id: true, email: true, name: true } },  
 meta: true,  
 },  
 });  
  
 let nextCursor: string | null = null;  
 if (items.length > limit) {  
 const next = items.pop();  
 nextCursor = next!.id;  
 }  
  
 return NextResponse.json({ items, nextCursor });  
}

## src\app\api\admin\users\[id]\route.ts

// src/app/api/admin/users/[id]/route.ts  
import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireRole } from "@/lib/auth-helpers";  
import { z } from "zod";  
  
type Params = Promise<{ id: string }>;  
  
export async function PATCH(req: Request, context: { params: Params }) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const { id } = await context.params; // 👈 await params  
  
 const UpdateSchema = z.object({  
 name: z.string().min(2).optional(),  
 role: z.enum(["USER", "ADMIN"]).optional(),  
 subscriptionActive: z.boolean().optional(),  
 status: z.enum(["PENDING", "ACTIVE", "SUSPENDED"]).optional(), // 👈 NEW  
});  
  
 const body = await req.json();  
 const data = UpdateSchema.parse(body);  
  
 const user = await prisma.user.update({  
 where: { id },  
 data,  
 select: { id: true, name: true, email: true, role: true, subscriptionActive: true, createdAt: true },  
 });  
  
 return NextResponse.json({ user });  
}  
  
export async function DELETE(\_req: Request, context: { params: Params }) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const { id } = await context.params; // 👈 await params  
 await prisma.user.delete({ where: { id } });  
  
 return NextResponse.json({ ok: true });  
}

## src\app\api\admin\users\route.ts

import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireRole } from "@/lib/auth-helpers";  
import { z } from "zod";  
import bcrypt from "bcrypt";  
  
export async function GET(req: Request) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const { searchParams } = new URL(req.url);  
 const status = searchParams.get("status") as "PENDING" | "ACTIVE" | "SUSPENDED" | null;  
  
 const users = await prisma.user.findMany({  
 where: status ? { status } : undefined,  
 orderBy: { createdAt: "desc" },  
 select: { id: true, name: true, email: true, role: true, subscriptionActive: true, status: true, createdAt: true },  
 });  
  
 return NextResponse.json({ users });  
}  
  
  
const CreateSchema = z.object({  
 name: z.string().min(2),  
 email: z.string().email(),  
 password: z.string().min(6),  
 role: z.enum(["USER", "ADMIN"]).default("USER"),  
 subscriptionActive: z.boolean().default(false),  
});  
  
export async function POST(req: Request) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const body = await req.json();  
 const data = CreateSchema.parse(body);  
  
 const exists = await prisma.user.findUnique({ where: { email: data.email } });  
 if (exists) return NextResponse.json({ error: "Email already exists" }, { status: 409 });  
  
 const passwordHash = await bcrypt.hash(data.password, 10);  
  
 const user = await prisma.user.create({  
 data: {  
 name: data.name,  
 email: data.email,  
 passwordHash,  
 role: data.role,  
 subscriptionActive: data.subscriptionActive,  
 },  
 select: { id: true, name: true, email: true, role: true, subscriptionActive: true, createdAt: true },  
 });  
  
 return NextResponse.json({ user }, { status: 201 });  
}

## src\app\api\assignments\route.ts

import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { currentUser, requireRole } from "@/lib/auth-helpers";  
import { z } from "zod";  
  
export async function GET() {  
 const me = await currentUser();  
 if (!me) return NextResponse.json({ error: "unauthenticated" }, { status: 401 });  
  
 const items = await prisma.fileAssignment.findMany({  
 where: { userId: (me as any).id },  
 orderBy: { createdAt: "desc" },  
 select: {  
 id: true, note: true, createdAt: true,  
 file: { select: { id: true, title: true, url: true, originalName: true } },  
 assignedBy: { select: { id: true, name: true, email: true } },  
 },  
 });  
  
 return NextResponse.json({ assignments: items });  
}  
  
const CreateSchema = z.object({  
 fileId: z.string().min(1),  
 userId: z.string().min(1),  
 note: z.string().optional(),  
});  
  
export async function POST(req: Request) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const body = await req.json();  
 const data = CreateSchema.parse(body);  
  
 const created = await prisma.fileAssignment.create({  
 data: {  
 fileId: data.fileId,  
 userId: data.userId,  
 note: data.note,  
 assignedById: (guard.user as any).id,  
 },  
 select: { id: true, createdAt: true },  
 });  
  
 return NextResponse.json({ assignment: created }, { status: 201 });  
}

## src\app\api\auth\[...nextauth]\route.ts

import NextAuth from "next-auth";  
import { authOptions } from "@/lib/auth";  
  
const handler = NextAuth(authOptions);  
export { handler as GET, handler as POST };

## src\app\api\auth\register\route.ts

// src/app/api/auth/register/route.ts  
import { NextResponse } from 'next/server';  
import { prisma } from '@/lib/prisma';  
import bcrypt from 'bcrypt';  
import { z } from 'zod';  
  
const SignupSchema = z.object({  
 name: z.string().min(2, 'Name is too short'),  
 email: z.string().email('Invalid email'),  
 password: z.string().min(6, 'Password must be at least 6 characters'),  
});  
  
export async function POST(req: Request) {  
 try {  
 if (process.env.ALLOW\_SIGNUPS !== 'true') {  
 return NextResponse.json({ error: 'Signups are disabled' }, { status: 403 });  
 }  
  
 const body = await req.json();  
 const parsed = SignupSchema.safeParse(body);  
 if (!parsed.success) {  
 return NextResponse.json({ error: parsed.error.flatten() }, { status: 400 });  
 }  
  
 const { name, email, password } = parsed.data;  
  
 // Check if user exists  
 const existing = await prisma.user.findUnique({ where: { email } });  
 if (existing) {  
 return NextResponse.json({ error: 'Email already in use' }, { status: 409 });  
 }  
  
 // Hash password and create user  
 const hash = await bcrypt.hash(password, 10);  
 const user = await prisma.user.create({  
 data: {  
 name,  
 email,  
 passwordHash: hash,  
 role: 'USER', // requires role enum in your schema (from earlier day)  
 subscriptionActive: false,  
 status: 'PENDING', // per Day 4 requirement  
 },  
 select: { id: true, email: true, name: true },  
 });  
  
 return NextResponse.json({ ok: true, user });  
 } catch (err) {  
 console.error(err);  
 return NextResponse.json({ error: 'Server error' }, { status: 500 });  
 }  
}

## src\app\api\files\download\[...key]\route.ts

// src/app/api/files/download/[...key]/route.ts  
import { NextResponse } from "next/server";  
import { createClient } from "@supabase/supabase-js";  
import { logAudit } from "@/lib/audit";  
import { currentUser } from "@/lib/auth-helpers";  
  
const url = process.env.NEXT\_PUBLIC\_SUPABASE\_URL!;  
const serviceKey = process.env.SUPABASE\_SERVICE\_ROLE!; // server-only  
const bucket = process.env.SUPABASE\_BUCKET!;  
  
export async function GET(req: Request, ctx: { params: { key: string[] } }) {  
 try {  
 const keyPath = decodeURIComponent((ctx.params.key || []).join("/"));  
 if (!keyPath) return NextResponse.json({ error: "key\_missing" }, { status: 400 });  
  
 const expiresSec = Number(new URL(req.url).searchParams.get("expires")) || 60 \* 60; // default 1h  
  
 const supa = createClient(url, serviceKey, { auth: { persistSession: false } });  
 const { data, error } = await supa.storage.from(bucket).createSignedUrl(keyPath, expiresSec);  
 if (error) return NextResponse.json({ error: "sign\_failed", detail: error.message }, { status: 500 });  
  
 // optional audit trail with user info if present  
 const me = await currentUser().catch(() => null);  
 await logAudit({  
 action: "DOWNLOAD\_GRANTED",  
 target: "File",  
 targetId: keyPath,  
 actorId: (me as any)?.id ?? null,  
 meta: { key: keyPath, expires: expiresSec },  
 }).catch(() => {});  
  
 return NextResponse.redirect(data!.signedUrl, { status: 302 });  
 } catch (err: any) {  
 return NextResponse.json({ error: "unexpected", detail: err?.message ?? String(err) }, { status: 500 });  
 }  
}

## src\app\api\files\route.ts

// src/app/api/files/route.ts  
import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { currentUser, requireRole } from "@/lib/auth-helpers";  
  
export async function GET(req: Request) {  
 const me = await currentUser();  
 if (!me) return NextResponse.json({ error: "unauthenticated" }, { status: 401 });  
  
 const { searchParams } = new URL(req.url);  
 const scope = searchParams.get("scope"); // "all" for admin view, otherwise user's files  
 const isAdmin = (me as any).role === "ADMIN" && scope === "all";  
  
 const where = isAdmin  
 ? undefined  
 : {  
 // for users: show uploads they made OR assignments to them  
 OR: [  
 { uploadedById: (me as any).id },  
 { assignments: { some: { userId: (me as any).id } } },  
 ],  
 };  
  
 const files = await prisma.file.findMany({  
 where,  
 orderBy: { createdAt: "desc" },  
 select: {  
 id: true,  
 title: true,  
 originalName: true,  
 url: true,  
 mime: true,  
 size: true,  
 createdAt: true,  
 uploadedBy: { select: { id: true, name: true, email: true } },  
 ...(isAdmin  
 ? {  
 assignments: {  
 select: { user: { select: { id: true, email: true, name: true } } },  
 },  
 }  
 : {}),  
 },  
 });  
  
 return NextResponse.json({ files });  
}  
  
// Admin-only manual create (rarely used; your UIs mostly use POST /api/uploads)  
export async function POST(req: Request) {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 const body = await req.json().catch(() => ({}));  
 const file = await prisma.file.create({  
 data: {  
 title: body.title ?? "Untitled",  
 url: body.url,  
 originalName: body.originalName,  
 mime: body.mime,  
 size: body.size,  
 uploadedById: (guard.user as any).id,  
 },  
 select: { id: true, title: true, createdAt: true },  
 });  
  
 return NextResponse.json({ file }, { status: 201 });  
}

## src\app\api\integrations\upload-multipart\route.ts

// src/app/api/integrations/upload-multipart/route.ts  
import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireApiKey } from "@/lib/apiKeyAuth";  
import { supabasePutBuffer } from "@/lib/storage-supabase";  
import { logAudit } from "@/lib/audit";  
import { extractEmailsFromText, resolveAssigneeIdsByEmails } from "@/lib/assignmentRules";  
  
export const runtime = "nodejs";  
  
const ALLOWED\_EXT = new Set([".xlsx", ".xlsm", ".xls"]);  
const MIME\_TO\_EXT: Record<string, string> = {  
 "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet": ".xlsx",  
 "application/vnd.ms-excel.sheet.macroEnabled.12": ".xlsm",  
 "application/vnd.ms-excel": ".xls",  
};  
const MAX\_BYTES = 50 \* 1024 \* 1024;  
  
function pickExtFromName(name?: string | null) {  
 if (!name) return null;  
 const m = /\.[A-Za-z0-9]+$/.exec(name);  
 return m ? m[0].toLowerCase() : null;  
}  
  
async function getAssignerId(uploadedById?: string | null) {  
 if (uploadedById) {  
 const u = await prisma.user.findUnique({ where: { id: uploadedById }, select: { role: true } });  
 if (u?.role === "ADMIN") return uploadedById;  
 }  
 const admin = await prisma.user.findFirst({ where: { role: "ADMIN", status: "ACTIVE" }, select: { id: true } });  
 return admin?.id ?? null;  
}  
  
export async function POST(req: Request) {  
 // API key auth  
 const auth = requireApiKey(req);  
 if (!auth.ok) return auth.res;  
  
 // ✅ Guard: only accept multipart/form-data to avoid “Failed to parse body as FormData”  
 const contentType = req.headers.get("content-type") || "";  
 if (!contentType.includes("multipart/form-data")) {  
 return NextResponse.json(  
 { error: "invalid\_content\_type", expected: "multipart/form-data" },  
 { status: 400 }  
 );  
 }  
  
 const form = await req.formData();  
 const file = form.get("file") as unknown as File | null;  
 const title = String(form.get("title") || "");  
 const uploadedByEmail = form.get("uploadedByEmail") ? String(form.get("uploadedByEmail")) : undefined;  
  
 if (!file) {  
 return NextResponse.json(  
 { error: "No file provided (form-data key 'file' missing or not a File)" },  
 { status: 400 }  
 );  
 }  
  
 const rawName = (file as any).name as string | undefined; // may be "blob"  
 const mime = ((file as any).type as string | undefined) || "application/octet-stream";  
  
 let ext = pickExtFromName(rawName);  
 if (!ext || !ALLOWED\_EXT.has(ext)) {  
 const mimeExt = MIME\_TO\_EXT[mime];  
 if (mimeExt && ALLOWED\_EXT.has(mimeExt)) ext = mimeExt;  
 }  
 if (!ext || !ALLOWED\_EXT.has(ext)) {  
 return NextResponse.json({ error: "Unsupported file type", details: { rawName, mime } }, { status: 415 });  
 }  
  
 // Size guard  
 // @ts-ignore  
 const declaredSize = (file as any).size as number | undefined;  
 const ab = await file.arrayBuffer();  
 if ((declaredSize ?? ab.byteLength) > MAX\_BYTES) {  
 return NextResponse.json({ error: `File too large (> ${Math.floor(MAX\_BYTES / (1024 \* 1024))} MB)` }, { status: 413 });  
 }  
  
 const buf = Buffer.from(ab);  
 const safeBase = (rawName && rawName !== "blob" ? rawName : `upload${ext}`).replace(/[^\w.\-@]+/g, "\_");  
  
 // Store in Supabase Storage  
 const now = new Date();  
 const keyPath = `uploads/${now.getUTCFullYear()}/${String(now.getUTCMonth() + 1).padStart(2, "0")}/${Date.now()}\_${safeBase}`;  
 const put = await supabasePutBuffer(keyPath, buf, mime);  
  
 // Attribution (who uploaded & who assigns)  
 let uploadedById: string | undefined;  
 if (uploadedByEmail) {  
 const u = await prisma.user.findUnique({ where: { email: uploadedByEmail }, select: { id: true } });  
 uploadedById = u?.id;  
 }  
 const assignerId = await getAssignerId(uploadedById);  
  
 // Create DB record (dashboard reads this)  
 const publicUrl = `/api/files/download/${keyPath}`; // signed on demand  
 const record = await prisma.file.create({  
 data: {  
 title: title || safeBase,  
 originalName: safeBase,  
 url: publicUrl,  
 mime,  
 size: buf.byteLength,  
 uploadedById,  
 },  
 select: { id: true, title: true, originalName: true, url: true, createdAt: true },  
 });  
  
 // Auto-assign by emails in filename/title  
 const candidates = extractEmailsFromText(`${record.originalName} ${record.title}`);  
 const assigneeIds = candidates.length ? await resolveAssigneeIdsByEmails(candidates) : [];  
 if (assigneeIds.length && assignerId) {  
 await prisma.fileAssignment.createMany({  
 data: assigneeIds.map((userId) => ({  
 fileId: record.id,  
 userId,  
 assignedById: assignerId!,  
 })),  
 skipDuplicates: true,  
 });  
 }  
  
 // Audit  
 await logAudit({  
 action: "FILE\_UPLOADED",  
 target: "File",  
 targetId: record.id,  
 meta: {  
 via: "integration\_multipart",  
 storageKey: keyPath,  
 mime,  
 size: buf.byteLength,  
 assigned: assigneeIds.length,  
 emails: candidates,  
 },  
 });  
  
 return NextResponse.json(  
 { ok: true, file: record, assignedCount: assigneeIds.length, key: keyPath, signedUrl: put.signedUrl },  
 { status: 201 }  
 );  
}

## src\app\api\integrations\uploads\route.ts

// src/app/api/integrations/uploads/route.ts  
import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireApiKey } from "@/lib/apiKeyAuth";  
import { supabasePutBuffer } from "@/lib/storage-supabase";  
import { logAudit } from "@/lib/audit";  
import { extractEmailsFromText, resolveAssigneeIdsByEmails } from "@/lib/assignmentRules";  
import { z } from "zod";  
  
export const runtime = "nodejs";  
  
const BodySchema = z.object({  
 title: z.string().optional(),  
 url: z.string().url(), // public URL to fetch from  
 originalName: z.string().optional(), // e.g. orders\_john@doe.com.xlsx  
 uploadedByEmail: z.string().email().optional(),  
});  
  
async function getAssignerId(uploadedById?: string | null) {  
 if (uploadedById) {  
 const u = await prisma.user.findUnique({ where: { id: uploadedById }, select: { role: true } });  
 if (u?.role === "ADMIN") return uploadedById;  
 }  
 const admin = await prisma.user.findFirst({ where: { role: "ADMIN", status: "ACTIVE" }, select: { id: true } });  
 return admin?.id ?? null;  
}  
  
export async function POST(req: Request) {  
 const auth = requireApiKey(req);  
 if (!auth.ok) return auth.res;  
  
 let data: z.infer<typeof BodySchema>;  
 try { data = BodySchema.parse(await req.json()); }  
 catch { return NextResponse.json({ error: "invalid\_payload" }, { status: 400 }); }  
  
 // 1) Download the file bytes  
 const r = await fetch(data.url, { redirect: "follow" });  
 if (!r.ok) return NextResponse.json({ error: `Fetch failed (${r.status})` }, { status: 400 });  
 const contentType = r.headers.get("content-type") || "application/octet-stream";  
 const ab = await r.arrayBuffer();  
 const buf = Buffer.from(ab);  
  
 // 2) Store in Supabase Storage  
 const now = new Date();  
 const safeName = (data.originalName || data.title || "upload.xlsx").replace(/[^\w.\-@]+/g, "\_");  
 const keyPath = `uploads/${now.getUTCFullYear()}/${String(now.getUTCMonth() + 1).padStart(2, "0")}/${Date.now()}\_${safeName}`;  
 const put = await supabasePutBuffer(keyPath, buf, contentType);  
  
 // 3) Work out attribution (who uploaded & who assigns)  
 let uploadedById: string | undefined;  
 if (data.uploadedByEmail) {  
 const u = await prisma.user.findUnique({ where: { email: data.uploadedByEmail }, select: { id: true } });  
 uploadedById = u?.id;  
 }  
 const assignerId = await getAssignerId(uploadedById);  
  
 // 4) Create DB record (THIS is what the dashboard reads)  
 const publicUrl = `/api/files/download/${keyPath}`; // signed on demand  
 const record = await prisma.file.create({  
 data: {  
 title: data.title || safeName,  
 originalName: data.originalName || safeName,  
 url: publicUrl, // <- dashboard uses this  
 mime: contentType,  
 size: buf.byteLength,  
 uploadedById,  
 },  
 select: { id: true, title: true, originalName: true, url: true, createdAt: true },  
 });  
  
 // 5) Auto-assign by emails in filename/title (optional)  
 const candidates = extractEmailsFromText(`${record.originalName} ${record.title}`);  
 const assigneeIds = candidates.length ? await resolveAssigneeIdsByEmails(candidates) : [];  
 if (assigneeIds.length && assignerId) {  
 await prisma.fileAssignment.createMany({  
 data: assigneeIds.map((userId) => ({  
 fileId: record.id,  
 userId,  
 assignedById: assignerId,  
 })),  
 skipDuplicates: true,  
 });  
 }  
  
 // 6) Audit  
 await logAudit({  
 action: "FILE\_UPLOADED",  
 target: "File",  
 targetId: record.id,  
 meta: {  
 via: "integration\_url\_ingest",  
 sourceUrl: data.url,  
 storageKey: keyPath,  
 mime: contentType,  
 size: buf.byteLength,  
 assigned: assigneeIds.length,  
 emails: candidates,  
 },  
 });  
  
 return NextResponse.json({  
 ok: true,  
 file: record,  
 assignedCount: assigneeIds.length,  
 key: keyPath,  
 signedUrl: put.signedUrl,  
 }, { status: 201 });  
}

## src\app\api\me\route.ts

import { NextResponse } from "next/server";  
import { getServerSession } from "next-auth";  
import { authOptions } from "@/lib/auth";  
  
export async function GET() {  
 const session = await getServerSession(authOptions);  
 if (!session) return NextResponse.json({ ok: false }, { status: 401 });  
 return NextResponse.json({ ok: true, user: session.user });  
}

## src\app\api\static\uploads\[...path]\route.ts

// src/app/api/static/uploads/[...path]/route.ts  
import { NextResponse } from "next/server";  
  
export function GET() {  
 return NextResponse.json(  
 {  
 error: "deprecated",  
 message: "Local /uploads is not used on Vercel. Use /api/files/download/<key> instead (Supabase Storage).",  
 },  
 { status: 410 }  
 );  
}

## src\app\api\stats-user\route.ts

// src/app/api/stats-user/route.ts  
export const runtime = "nodejs";  
  
import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { currentUser } from "@/lib/auth-helpers";  
  
function startOfDay(d = new Date()) {  
 const x = new Date(d);  
 x.setHours(0, 0, 0, 0);  
 return x;  
}  
  
export async function GET() {  
 const me = await currentUser();  
 if (!me) return NextResponse.json({ error: "unauthenticated" }, { status: 401 });  
  
 try {  
 const userId = (me as any).id as string;  
  
 // counts (same connection)  
 const [assignedTotal, myUploadsTotal] = await prisma.$transaction([  
 prisma.fileAssignment.count({ where: { userId } }),  
 prisma.file.count({ where: { uploadedById: userId } }),  
 ]);  
  
 // 30-day uploads series (by current user)  
 const today = startOfDay(new Date());  
 const since = new Date(today);  
 since.setDate(today.getDate() - 29);  
  
 const uploads = await prisma.file.findMany({  
 where: {  
 uploadedById: userId,  
 createdAt: { gte: since, lte: new Date(today.getTime() + 24 \* 60 \* 60 \* 1000 - 1) },  
 },  
 select: { createdAt: true },  
 });  
  
 // bucket by day  
 const byDay = new Map<string, number>();  
 for (let i = 29; i >= 0; i--) {  
 const d = new Date(today);  
 d.setDate(today.getDate() - i);  
 const k = d.toISOString().slice(0, 10);  
 byDay.set(k, 0);  
 }  
 for (const f of uploads) {  
 const k = startOfDay(new Date(f.createdAt)).toISOString().slice(0, 10);  
 if (byDay.has(k)) byDay.set(k, (byDay.get(k) || 0) + 1);  
 }  
 const series = Array.from(byDay.entries()).map(([x, y]) => ({ x, y }));  
  
 return NextResponse.json({  
 assignedTotal,  
 myUploadsTotal,  
 series,  
 });  
 } catch (err: any) {  
 return NextResponse.json(  
 { error: "stats\_user\_failed", detail: err?.message || String(err) },  
 { status: 500 }  
 );  
 }  
}

## src\app\api\stats\route.ts

// src/app/api/stats/route.ts  
export const runtime = "nodejs";  
  
import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireRole } from "@/lib/auth-helpers";  
  
/\*\*  
 \* Returns a flat shape to match the AdminDashboard expectations:  
 \* { users: number, pending: number, files: number }  
 \*/  
export async function GET() {  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) {  
 return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
 }  
  
 try {  
 const [totalUsers, activeUsers, totalFiles] = await prisma.$transaction([  
 prisma.user.count(),  
 prisma.user.count({ where: { status: "ACTIVE" } }),  
 prisma.file.count(),  
 ]);  
  
 return NextResponse.json({  
 users: totalUsers,  
 pending: Math.max(0, totalUsers - activeUsers),  
 files: totalFiles,  
 });  
 } catch (err: any) {  
 const message = err?.message || String(err);  
 const hint =  
 /26000|42P05/.test(message)  
 ? "Make sure DATABASE\_URL uses the Supabase pooler (6543) and includes '?sslmode=require&pgbouncer=true'."  
 : undefined;  
  
 console.error("api/stats error:", { message, hint });  
 return NextResponse.json({ error: "stats\_failed", detail: message, hint }, { status: 500 });  
 }  
}

## src\app\api\support\route.ts

// src/app/api/support/route.ts  
export const runtime = "nodejs";  
  
import { NextResponse } from "next/server";  
import { currentUser } from "@/lib/auth-helpers";  
import { prisma } from "@/lib/prisma";  
import type { Prisma } from "@prisma/client"; // <-- type-only import  
  
export async function POST(req: Request) {  
 const me = await currentUser();  
 if (!me) return NextResponse.json({ error: "unauthenticated" }, { status: 401 });  
  
 try {  
 const body = await req.json().catch(() => ({}));  
 const subject = String(body.subject || "").slice(0, 140);  
 const message = String(body.message || "").slice(0, 5000);  
 const priority = String(body.priority || "normal");  
  
 if (!subject || !message) {  
 return NextResponse.json(  
 { error: "validation", detail: "subject and message are required" },  
 { status: 400 }  
 );  
 }  
  
 await prisma.auditLog.create({  
 data: {  
 actorId: (me as any).id,  
 // Avoid runtime enum import; pass string and type it as the DB enum  
 action: "SUPPORT\_TICKET" as Prisma.AuditAction,  
 target: "Ticket",  
 targetId: null,  
 meta: {  
 subject,  
 message,  
 priority,  
 userEmail: (me as any).email,  
 userName: (me as any).name,  
 userAgent: req.headers.get("user-agent") || "",  
 },  
 },  
 });  
  
 const supportEmail = process.env.SUPPORT\_EMAIL || "support@hygiene-plus.gr";  
 const mailto = `mailto:${encodeURIComponent(supportEmail)}?subject=${encodeURIComponent(  
 `[HygienePlus] ${subject} (${priority})`  
 )}&body=${encodeURIComponent(message + `\n\n— ${((me as any).email || "")}`)}`;  
  
 return NextResponse.json({ ok: true, mailto });  
 } catch (err: any) {  
 return NextResponse.json(  
 { error: "support\_failed", detail: err?.message || String(err) },  
 { status: 500 }  
 );  
 }  
}

## src\app\api\uploads\route.ts

// src/app/api/uploads/route.ts  
import { NextResponse } from "next/server";  
import { prisma } from "@/lib/prisma";  
import { requireRole } from "@/lib/auth-helpers";  
import { logAudit } from "@/lib/audit";  
import { extractEmailsFromText, resolveAssigneeIdsByEmails } from "@/lib/assignmentRules";  
import { supabasePutBuffer } from "@/lib/storage-supabase";  
  
// No filesystem usage → works on Vercel  
export const runtime = "nodejs";  
  
export async function POST(req: Request) {  
 // Admin-only  
 const guard = await requireRole("ADMIN");  
 if (!guard.ok) return NextResponse.json({ error: "unauthorized" }, { status: guard.status });  
  
 // ✅ Guard: multipart only  
 const contentType = req.headers.get("content-type") || "";  
 if (!contentType.includes("multipart/form-data")) {  
 return NextResponse.json({ error: "invalid\_content\_type", expected: "multipart/form-data" }, { status: 400 });  
 }  
  
 const form = await req.formData();  
 const file = form.get("file") as unknown as File | null;  
 const title = (form.get("title") as string | null) ?? undefined;  
  
 if (!file) return NextResponse.json({ error: "file\_missing" }, { status: 400 });  
  
 const rawName = (file as any).name as string | undefined;  
 const mime = ((file as any).type as string | undefined) || "application/octet-stream";  
 const ab = await file.arrayBuffer();  
 const buf = Buffer.from(ab);  
  
 const safeBase = (title || rawName || "upload.xlsx").replace(/[^\w.\-@]+/g, "\_").slice(0, 100);  
 const now = new Date();  
 const keyPath = `uploads/${now.getUTCFullYear()}/${String(now.getUTCMonth() + 1).padStart(2, "0")}/${Date.now()}\_${safeBase}`;  
  
 // Store in Supabase Storage  
 const put = await supabasePutBuffer(keyPath, buf, mime);  
  
 // Create DB record  
 const publicUrl = `/api/files/download/${keyPath}`; // signed on demand  
 const record = await prisma.file.create({  
 data: {  
 title: title || safeBase,  
 originalName: rawName || safeBase,  
 url: publicUrl,  
 mime,  
 size: buf.byteLength,  
 uploadedById: (guard.user as any).id,  
 },  
 select: { id: true, title: true, originalName: true, url: true, createdAt: true },  
 });  
  
 // Optional: auto-assign based on emails in the filename/title  
 const candidates = extractEmailsFromText(`${record.originalName} ${record.title}`);  
 const assigneeIds = candidates.length ? await resolveAssigneeIdsByEmails(candidates) : [];  
 if (assigneeIds.length) {  
 await prisma.fileAssignment.createMany({  
 data: assigneeIds.map((userId) => ({  
 fileId: record.id,  
 userId,  
 assignedById: (guard.user as any).id,  
 })),  
 skipDuplicates: true,  
 });  
 }  
  
 await logAudit({  
 action: "FILE\_UPLOADED",  
 target: "File",  
 targetId: record.id,  
 actorId: (guard.user as any).id,  
 meta: { via: "admin", storageKey: keyPath, mime, size: buf.byteLength, assigned: assigneeIds.length, emails: candidates },  
 });  
  
 return NextResponse.json({ ok: true, file: record, key: keyPath, signedUrl: put.signedUrl, assignedCount: assigneeIds.length }, { status: 201 });  
}

## src\app\globals.css

@import "tailwindcss";  
@import "tw-animate-css";  
  
@custom-variant dark (&:is(.dark \*));  
  
:root{  
 /\* existing tokens … \*/  
 --brand:#25C3F4;  
 --text:#0A0F2C;  
 --muted:oklch(0.97 0 0);  
 --bg:#F9FAFB;  
 --card:oklch(1 0 0);  
 --border:oklch(0.922 0 0);  
 --radius:0.625rem;  
  
 --sidebar-bg:#061630; /\* <- requested color \*/  
 --sidebar-text:#ECF5F8;  
 --sidebar-muted:#A7BECC;  
 --sidebar-active-bg:rgba(37,195,244,.15);  
 --sidebar-active-text:#FFFFFF;  
 --sidebar-hover-bg:rgba(255,255,255,.06); /\* <- used on hover \*/  
 --sidebar-border:oklch(0.922 0 0);  
 --background:oklch(1 0 0);  
 --foreground:oklch(0.145 0 0);  
 --card-foreground:oklch(0.145 0 0);  
 --popover:oklch(1 0 0);  
 --popover-foreground:oklch(0.145 0 0);  
 --primary:oklch(0.205 0 0);  
 --primary-foreground:oklch(0.985 0 0);  
 --secondary:oklch(0.97 0 0);  
 --secondary-foreground:oklch(0.205 0 0);  
 --muted-foreground:oklch(0.556 0 0);  
 --accent:oklch(0.97 0 0);  
 --accent-foreground:oklch(0.205 0 0);  
 --destructive:oklch(0.577 0.245 27.325);  
 --input:oklch(0.922 0 0);  
 --ring:oklch(0.708 0 0);  
 --chart-1:oklch(0.646 0.222 41.116);  
 --chart-2:oklch(0.6 0.118 184.704);  
 --chart-3:oklch(0.398 0.07 227.392);  
 --chart-4:oklch(0.828 0.189 84.429);  
 --chart-5:oklch(0.769 0.188 70.08);  
 --sidebar:oklch(0.985 0 0);  
 --sidebar-foreground:oklch(0.145 0 0);  
 --sidebar-primary:oklch(0.205 0 0);  
 --sidebar-primary-foreground:oklch(0.985 0 0);  
 --sidebar-accent:oklch(0.97 0 0);  
 --sidebar-accent-foreground:oklch(0.205 0 0);  
 --sidebar-ring:oklch(0.708 0 0);  
}  
  
aside a:hover{ background-color: rgba(255,255,255,.06); }  
  
html,body{background:var(--bg);color:var(--text)}  
  
/\* Gentle defaults for card-like blocks (border + rounded) \*/  
.card, .panel, .box, .surface {  
 padding: 1rem; /\* 16px \*/  
}  
/\* ===== Mobile Tables: fit within the screen (no horizontal scroll) ===== \*/  
/\* ===== Mobile Tables: fit within the screen (no horizontal scroll) ===== \*/  
@media (max-width: 639px) {  
 /\* Scope to main content so we don't affect other layouts \*/  
 .app-main table {  
 width: 100%;  
 table-layout: fixed; /\* columns shrink to fit screen \*/  
 border-collapse: collapse;  
 }  
  
 .app-main table thead th,  
 .app-main table tbody td {  
 white-space: normal; /\* allow wrapping for text \*/  
 overflow-wrap: anywhere; /\* wrap long emails/filenames \*/  
 word-break: break-word;  
 padding: 0.5rem 0.75rem; /\* compact \*/  
 vertical-align: top;  
 }  
  
 .app-main table {   
 font-size: 0.9375rem;   
 line-height: 1.35;   
 }  
  
 /\* Keep badges/pills in one line so they don't stack vertically \*/  
 .app-main table .inline-flex.rounded-full,  
 .app-main table .badge,  
 .app-main table .tag,  
 .app-main table .btn,  
 .app-main table .nowrap {  
 white-space: nowrap !important;  
 overflow-wrap: normal !important;  
 word-break: normal !important;  
 }  
  
 /\* If any header cells were marked nowrap, keep them too \*/  
 .app-main table thead .whitespace-nowrap {  
 white-space: nowrap !important;  
 }  
}  
  
/\* Utility for explicit wrapping anywhere (safe outside tables too) \*/  
.break-words, .truncate-soft {  
 overflow-wrap: anywhere;  
 word-break: break-word;  
}  
  
/\* Make charts/media never push layout wider than screen \*/  
.app-main canvas,  
.app-main img,  
.app-main svg {  
 max-width: 100%;  
 height: auto;  
}  
  
@theme inline {  
 --radius-sm:calc(var(--radius) - 4px);  
 --radius-md:calc(var(--radius) - 2px);  
 --radius-lg:var(--radius);  
 --radius-xl:calc(var(--radius) + 4px);  
 --color-background:var(--background);  
 --color-foreground:var(--foreground);  
 --color-card:var(--card);  
 --color-card-foreground:var(--card-foreground);  
 --color-popover:var(--popover);  
 --color-popover-foreground:var(--popover-foreground);  
 --color-primary:var(--primary);  
 --color-primary-foreground:var(--primary-foreground);  
 --color-secondary:var(--secondary);  
 --color-secondary-foreground:var(--secondary-foreground);  
 --color-muted:var(--muted);  
 --color-muted-foreground:var(--muted-foreground);  
 --color-accent:var(--accent);  
 --color-accent-foreground:var(--accent-foreground);  
 --color-destructive:var(--destructive);  
 --color-border:var(--border);  
 --color-input:var(--input);  
 --color-ring:var(--ring);  
 --color-chart-1:var(--chart-1);  
 --color-chart-2:var(--chart-2);  
 --color-chart-3:var(--chart-3);  
 --color-chart-4:var(--chart-4);  
 --color-chart-5:var(--chart-5);  
 --color-sidebar:var(--sidebar);  
 --color-sidebar-foreground:var(--sidebar-foreground);  
 --color-sidebar-primary:var(--sidebar-primary);  
 --color-sidebar-primary-foreground:var(--sidebar-primary-foreground);  
 --color-sidebar-accent:var(--sidebar-accent);  
 --color-sidebar-accent-foreground:var(--sidebar-accent-foreground);  
 --color-sidebar-border:var(--sidebar-border);  
 --color-sidebar-ring:var(--sidebar-ring);  
}  
  
.dark {  
 --background:oklch(0.145 0 0);  
 --foreground:oklch(0.985 0 0);  
 --card:oklch(0.205 0 0);  
 --card-foreground:oklch(0.985 0 0);  
 --popover:oklch(0.205 0 0);  
 --popover-foreground:oklch(0.985 0 0);  
 --primary:oklch(0.922 0 0);  
 --primary-foreground:oklch(0.205 0 0);  
 --secondary:oklch(0.269 0 0);  
 --secondary-foreground:oklch(0.985 0 0);  
 --muted:oklch(0.269 0 0);  
 --muted-foreground:oklch(0.708 0 0);  
 --accent:oklch(0.269 0 0);  
 --accent-foreground:oklch(0.985 0 0);  
 --destructive:oklch(0.704 0.191 22.216);  
 --border:oklch(1 0 0 / 10%);  
 --input:oklch(1 0 0 / 15%);  
 --ring:oklch(0.556 0 0);  
 --chart-1:oklch(0.488 0.243 264.376);  
 --chart-2:oklch(0.696 0.17 162.48);  
 --chart-3:oklch(0.769 0.188 70.08);  
 --chart-4:oklch(0.627 0.265 303.9);  
 --chart-5:oklch(0.645 0.246 16.439);  
 --sidebar:oklch(0.205 0 0);  
 --sidebar-foreground:oklch(0.985 0 0);  
 --sidebar-primary:oklch(0.488 0.243 264.376);  
 --sidebar-primary-foreground:oklch(0.985 0 0);  
 --sidebar-accent:oklch(0.269 0 0);  
 --sidebar-accent-foreground:oklch(0.985 0 0);  
 --sidebar-border:oklch(1 0 0 / 10%);  
 --sidebar-ring:oklch(0.556 0 0);  
}  
  
@layer base {  
 \* {  
 @apply border-border outline-ring/50;  
 }  
 body {  
 @apply bg-background text-foreground;  
 }  
}  
  
/\* ===== Responsive table helper ===== \*/  
.table-fluid {  
 width: 100%;  
 table-layout: fixed; /\* columns shrink to fit \*/  
 border-collapse: collapse;  
}  
.table-fluid th,  
.table-fluid td {  
 white-space: normal;  
 overflow-wrap: anywhere;  
 word-break: break-word;  
 vertical-align: top;  
}  
@media (max-width: 639px) {  
 /\* hide overflowy headers that push layout \*/  
 .table-fluid thead .hide-sm { display: none; }  
 .table-fluid tbody .hide-sm { display: none; }  
 /\* keep pills/badges on one line \*/  
 .table-fluid .nowrap,  
 .table-fluid .inline-flex.rounded-full,  
 .table-fluid .badge,  
 .table-fluid .tag {  
 white-space: nowrap !important;  
 word-break: normal !important;  
 overflow-wrap: normal !important;  
 }  
}  
  
/\* ==== Content area (light theme) ==== \*/  
/\* Scope to the main content wrapper so it won't affect the dark sidebar or the mobile drawer \*/  
.app-main {  
 --app-fg: #0A0F2C; /\* primary text \*/  
 --app-muted: #25c3f4; /\* secondary text \*/  
 --app-card: #FFFFFF; /\* card bg \*/  
 color: var(--app-fg);  
}  
  
/\* Ensure common text elements inherit the light color even after hydration \*/  
.app-main :where(p, span, div, li, a, button, label, input, select, textarea) {  
 color: inherit;  
}  
  
/\* Muted helpers you already use in pages \*/  
.app-main .text-muted,   
.app-main .muted {  
 color: var(--app-muted) !important;  
}  
  
/\* Tables: make sure headers and cells use content color unless overridden explicitly \*/  
.app-main table thead th,  
.app-main table tbody td {  
 color: inherit;  
 background-color: transparent;  
}  
  
/\* Cards that rely on CSS vars \*/  
.app-main .card,  
.app-main section {  
 background-color: var(--app-card);  
}  
  
/\* If any utility injected 'text-white' into cells, neutralize it ONLY inside tables in the content area \*/  
.app-main table :where(th, td).text-white {  
 color: var(--app-fg) !important;  
}

## src\app\layout.tsx

// src/app/layout.tsx  
import type { Metadata } from "next";  
import "./globals.css";  
import Providers from "./providers";  
  
export const metadata: Metadata = {  
 title: "Excel Delivery",  
 description: "Πλατφόρμα διαχείρισης αρχείων Excel",  
};  
  
export default function RootLayout({ children }: { children: React.ReactNode }) {  
 return (  
 <html lang="el">  
 <body>  
 <Providers>{children}</Providers>  
 </body>  
 </html>  
 );  
}

## src\app\login\page.tsx

// src/app/login/page.tsx  
"use client";  
  
import { Suspense, useEffect, useState } from "react";  
import { signIn } from "next-auth/react";  
import { useRouter, useSearchParams } from "next/navigation";  
  
export const dynamic = "force-dynamic";  
  
function QueryEffect({ setError }: { setError: (msg: string) => void }) {  
 const sp = useSearchParams();  
 useEffect(() => {  
 const authError = sp.get("error");  
 const notice = sp.get("notice");  
 if (authError === "AccountPending") setError("Ο λογαριασμός σας αναμένει έγκριση από διαχειριστή.");  
 else if (authError === "AccountSuspended") setError("Ο λογαριασμός σας έχει ανασταλεί. Επικοινωνήστε με την υποστήριξη.");  
 else if (notice === "pending") setError("Η εγγραφή ολοκληρώθηκε. Περιμένετε έγκριση από διαχειριστή.");  
 }, [sp, setError]);  
 return null;  
}  
  
export default function LoginPage() {  
 const [email, setEmail] = useState("");  
 const [password, setPassword] = useState("");  
 const [loading, setLoading] = useState(false);  
 const [error, setError] = useState("");  
 const router = useRouter();  
  
 async function onSubmit(e: React.FormEvent) {  
 e.preventDefault();  
 setError("");  
 setLoading(true);  
 const res = await signIn("credentials", { redirect: false, email, password });  
 setLoading(false);  
 if (res?.error) {  
 if (res.error === "AccountPending") return setError("Ο λογαριασμός σας αναμένει έγκριση από διαχειριστή.");  
 if (res.error === "AccountSuspended") return setError("Ο λογαριασμός σας έχει ανασταλεί. Επικοινωνήστε με την υποστήριξη.");  
 return setError("Λάθος email ή κωδικός");  
 }  
 router.push("/dashboard");  
 }  
  
 return (  
 <main className="min-h-screen flex items-center justify-center p-6">  
 <Suspense fallback={null}>  
 <QueryEffect setError={setError} />  
 </Suspense>  
  
 <div className="w-full max-w-sm">  
 <form onSubmit={onSubmit} className="space-y-3 border rounded-2xl p-6 bg-white">  
 <h1 className="text-xl font-semibold">Σύνδεση</h1>  
  
 <label className="block">  
 <span className="text-sm">Ηλεκτρονικό ταχυδρομείο</span>  
 <input  
 className="w-full border rounded p-2"  
 value={email}  
 onChange={e => setEmail(e.target.value)}  
 required  
 type="email"  
 autoComplete="email"  
 />  
 </label>  
  
 <label className="block">  
 <span className="text-sm">Κωδικός πρόσβασης</span>  
 <input  
 className="w-full border rounded p-2"  
 value={password}  
 onChange={e => setPassword(e.target.value)}  
 required  
 type="password"  
 autoComplete="current-password"  
 />  
 </label>  
  
 {error && <p className="text-red-600 text-sm">{error}</p>}  
  
 <button  
 disabled={loading}  
 className="w-full rounded bg-black text-white py-2 disabled:opacity-60"  
 >  
 {loading ? "Σύνδεση..." : "Σύνδεση"}  
 </button>  
  
 <p className="text-sm mt-2 text-center">  
 Νέος χρήστης; <a className="underline" href="/register">Δημιουργία λογαριασμού</a>  
 </p>  
 </form>  
 </div>  
 </main>  
 );  
}

## src\app\page.tsx

// src/app/page.tsx  
export const runtime = "nodejs"; // ensure NextAuth runs on Node  
  
import { redirect } from "next/navigation";  
import { currentUser } from "@/lib/auth-helpers";  
  
export const dynamic = "force-dynamic";  
  
export default async function RootPage() {  
 const me = await currentUser(); // null if not logged in  
  
 if (!me) {  
 redirect("/login");  
 }  
  
 redirect(me.role === "ADMIN" ? "/dashboard/admin" : "/dashboard");  
}

## src\app\providers.tsx

"use client";  
import { SessionProvider } from "next-auth/react";  
  
export default function Providers({ children }: { children: React.ReactNode }) {  
 return <SessionProvider>{children}</SessionProvider>;  
}

## src\app\register\page.tsx

// src/app/register/page.tsx  
"use client";  
  
import { FormEvent, useState } from "react";  
import { useRouter } from "next/navigation";  
  
export default function RegisterPage() {  
 const router = useRouter();  
 const [loading, setLoading] = useState(false);  
 const [error, setError] = useState<string | null>(null);  
  
 async function onSubmit(e: FormEvent<HTMLFormElement>) {  
 e.preventDefault();  
 setError(null);  
 setLoading(true);  
  
 const form = new FormData(e.currentTarget);  
 const name = String(form.get("name") || "");  
 const email = String(form.get("email") || "");  
 const password = String(form.get("password") || "");  
  
 try {  
 const res = await fetch("/api/auth/register", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ name, email, password }),  
 });  
  
 const data = await res.json();  
 if (!res.ok) {  
 const zerr = (data?.error?.fieldErrors && Object.values(data.error.fieldErrors)[0]?.[0]) as string | undefined;  
 setError(zerr || data?.error || "Η εγγραφή απέτυχε");  
 setLoading(false);  
 return;  
 }  
  
 setLoading(false);  
 router.replace("/login?notice=pending");  
 } catch {  
 setError("Κάτι πήγε στραβά");  
 setLoading(false);  
 }  
 }  
  
 return (  
 <main className="min-h-[80vh] flex items-center justify-center p-6">  
 <div className="w-full max-w-md rounded-2xl border p-6 shadow-sm">  
 <h1 className="text-2xl font-semibold mb-1">Δημιουργία λογαριασμού</h1>  
 <p className="text-sm text-gray-500 mb-6">Αποκτήστε πρόσβαση στον πίνακα ελέγχου μετά την εγγραφή.</p>  
  
 {error && (  
 <div className="mb-4 text-sm border rounded-md p-3 bg-red-50 border-red-200 text-red-700">  
 {error}  
 </div>  
 )}  
  
 <form onSubmit={onSubmit} className="space-y-4">  
 <div>  
 <label className="block text-sm mb-1" htmlFor="name">Όνομα</label>  
 <input id="name" name="name" required className="w-full rounded-md border p-2" />  
 </div>  
  
 <div>  
 <label className="block text-sm mb-1" htmlFor="email">Ηλεκτρονικό ταχυδρομείο</label>  
 <input id="email" name="email" type="email" required className="w-full rounded-md border p-2" />  
 </div>  
  
 <div>  
 <label className="block text-sm mb-1" htmlFor="password">Κωδικός πρόσβασης</label>  
 <input id="password" name="password" type="password" required className="w-full rounded-md border p-2" />  
 </div>  
  
 <button  
 type="submit"  
 disabled={loading}  
 className="w-full rounded-md bg-black text-white py-2 disabled:opacity-60"  
 >  
 {loading ? "Δημιουργία..." : "Δημιουργία λογαριασμού"}  
 </button>  
 </form>  
  
 <p className="text-sm mt-4">  
 Έχετε ήδη λογαριασμό; <a className="underline" href="/login">Σύνδεση</a>  
 </p>  
 </div>  
 </main>  
 );  
}

## src\components\AppShell.tsx

"use client";  
  
import React from "react";  
import { useSession } from "next-auth/react";  
import Sidebar from "@/components/Sidebar";  
import MobileHeader from "@/components/MobileHeader";  
  
export default function AppShell({ children }: { children: React.ReactNode }) {  
 const { data } = useSession();  
 const role = ((data?.user as any)?.role ?? "USER") as "ADMIN" | "USER";  
 const name = data?.user?.name ?? null;  
  
 return (  
 <div className="min-h-screen w-full bg-[var(--app-bg,#F6FAFC)] text-[var(--app-fg,#0A0F2C)]">  
 {/\* Mobile header \*/}  
 <MobileHeader role={role} name={name} />  
  
 <div className="flex min-h-screen">  
 {/\* Desktop sidebar \*/}  
 <div className="hidden md:block md:sticky md:top-0 md:h-screen">  
 <Sidebar role={role} name={name} />  
 </div>  
  
 {/\* Content \*/}  
 <main className="app-main flex-1 min-w-0 px-3 md:px-4 py-4 md:py-5">  
 {/\* One guard wrapper so children never overflow horizontally \*/}  
 <div className="max-w-full">{children}</div>  
 </main>  
 </div>  
 </div>  
 );  
}

## src\components\DashboardCard.tsx

"use client";  
import { ReactNode } from "react";  
  
export default function DashboardCard({  
 title, value, subtitle, right, children  
}:{  
 title: string;  
 value?: string | number;  
 subtitle?: string;  
 right?: ReactNode;  
 children?: ReactNode;  
}) {  
 return (  
 <section className="rounded-[var(--radius)] border border-[color:var(--border)] bg-[color:var(--card)] shadow-sm p-4">  
 <div className="flex items-start justify-between gap-2">  
 <div>  
 <h3 className="text-xs font-medium text-[color:var(--muted)]">{title}</h3>  
 {value !== undefined && (  
 <div className="text-3xl font-semibold mt-2">{value}</div>  
 )}  
 {subtitle && <p className="text-xs text-[color:var(--muted)] mt-1">{subtitle}</p>}  
 </div>  
 {right}  
 </div>  
 {children}  
 </section>  
 );  
}

## src\components\FilesTable.tsx

"use client";  
  
import React from "react";  
  
type FileRow = {  
 id: string;  
 title: string;  
 originalName: string;  
 url: string; // /api/files/download/<key>  
 mime?: string;  
 size?: number;  
 createdAt: string;  
 uploadedBy?: { id: string; name: string | null; email: string | null } | null;  
 assignments?: { user: { id: string; email: string | null; name: string | null } }[];  
};  
  
type Labels = {  
 search: string;  
 countSuffix: string; // e.g., "αρχείο(α)"  
 title: string;  
 original: string;  
 uploaded: string;  
 size: string;  
 assignees: string;  
 action: string;  
 download: string;  
 empty: string;  
};  
  
type Props = {  
 initialFiles: FileRow[];  
 adminMode?: boolean; // when true, show Assignees column  
 labels?: Partial<Labels>;  
};  
  
const DEFAULT\_LABELS: Labels = {  
 search: "Search files…",  
 countSuffix: "file(s)",  
 title: "Title",  
 original: "Original",  
 uploaded: "Uploaded",  
 size: "Size",  
 assignees: "Assignees",  
 action: "Action",  
 download: "Download",  
 empty: "No files found.",  
};  
  
function formatBytes(n?: number) {  
 if (!n && n !== 0) return "";  
 const units = ["B", "KB", "MB", "GB", "TB"];  
 let i = 0;  
 let v = n;  
 while (v >= 1024 && i < units.length - 1) {  
 v /= 1024;  
 i++;  
 }  
 return `${v.toFixed(v < 10 && i > 0 ? 1 : 0)} ${units[i]}`;  
}  
  
function formatDate(iso: string) {  
 const d = new Date(iso);  
 if (Number.isNaN(d.getTime())) return iso;  
 return d.toLocaleString();  
}  
  
export default function FilesTable({ initialFiles, adminMode = false, labels: lbl }: Props) {  
 const labels = { ...DEFAULT\_LABELS, ...(lbl || {}) };  
 const [query, setQuery] = React.useState("");  
 const [rows, setRows] = React.useState<FileRow[]>(initialFiles);  
  
 React.useEffect(() => setRows(initialFiles), [initialFiles]);  
  
 const filtered = React.useMemo(() => {  
 const q = query.trim().toLowerCase();  
 if (!q) return rows;  
 return rows.filter((r) =>  
 [r.title, r.originalName, r.uploadedBy?.email, r.uploadedBy?.name]  
 .filter(Boolean)  
 .some((s) => String(s).toLowerCase().includes(q))  
 );  
 }, [rows, query]);  
  
 return (  
 <div className="w-full">  
 <div className="mb-3 flex items-center justify-between gap-3">  
 <input  
 type="text"  
 placeholder={labels.search}  
 value={query}  
 onChange={(e) => setQuery(e.currentTarget.value)}  
 className="w-full max-w-md rounded-xl border border-gray-300 bg-white p-2 text-sm"  
 />  
 <span className="text-xs text-gray-500">  
 {filtered.length} {labels.countSuffix}  
 </span>  
 </div>  
  
 <div className="overflow-x-auto rounded-2xl border border-gray-200 shadow-sm">  
 <table className="min-w-full text-left text-sm">  
 <thead className="bg-gray-50 text-gray-600">  
 <tr>  
 <th className="px-4 py-3 font-semibold">{labels.title}</th>  
 <th className="px-4 py-3 font-semibold">{labels.original}</th>  
 <th className="px-4 py-3 font-semibold">{labels.uploaded}</th>  
 <th className="px-4 py-3 font-semibold">{labels.size}</th>  
 {adminMode && <th className="px-4 py-3 font-semibold">{labels.assignees}</th>}  
 <th className="px-4 py-3 font-semibold">{labels.action}</th>  
 </tr>  
 </thead>  
 <tbody className="divide-y divide-gray-100">  
 {filtered.map((f) => (  
 <tr key={f.id} className="hover:bg-gray-50/50">  
 <td className="px-4 py-3">  
 <div className="flex flex-col">  
 <span className="font-medium text-gray-900">{f.title}</span>  
 {f.uploadedBy?.email && (  
 <span className="text-xs text-gray-500">  
 {f.uploadedBy.name || f.uploadedBy.email}  
 </span>  
 )}  
 </div>  
 </td>  
 <td className="px-4 py-3 text-gray-700">{f.originalName}</td>  
 <td className="px-4 py-3 text-gray-700">{formatDate(f.createdAt)}</td>  
 <td className="px-4 py-3 text-gray-700">{formatBytes(f.size)}</td>  
 {adminMode && (  
 <td className="px-4 py-3 text-gray-700">  
 {f.assignments?.length  
 ? f.assignments  
 .map((a) => a.user.name || a.user.email || a.user.id)  
 .join(", ")  
 : <span className="text-gray-400">—</span>}  
 </td>  
 )}  
 <td className="px-4 py-3">  
 <a  
 href={f.url}  
 className="inline-flex rounded-xl bg-black px-3 py-1.5 text-xs font-semibold text-white hover:opacity-90"  
 >  
 {labels.download}  
 </a>  
 </td>  
 </tr>  
 ))}  
 {!filtered.length && (  
 <tr>  
 <td colSpan={adminMode ? 6 : 5} className="px-4 py-8 text-center text-sm text-gray-500">  
 {labels.empty}  
 </td>  
 </tr>  
 )}  
 </tbody>  
 </table>  
 </div>  
 </div>  
 );  
}

## src\components\MobileHeader.tsx

// src/components/MobileHeader.tsx  
"use client";  
  
import { useEffect, useState } from "react";  
import Image from "next/image";  
import { usePathname } from "next/navigation";  
import Sidebar from "@/components/Sidebar";  
  
type Role = "ADMIN" | "USER";  
  
export default function MobileHeader({  
 role,  
 name,  
}: {  
 role: Role;  
 name?: string | null;  
}) {  
 const [open, setOpen] = useState(false);  
 const pathname = usePathname();  
  
 // Close when route changes  
 useEffect(() => { setOpen(false); }, [pathname]);  
  
 // Close with ESC  
 useEffect(() => {  
 const esc = (e: KeyboardEvent) => e.key === "Escape" && setOpen(false);  
 window.addEventListener("keydown", esc);  
 return () => window.removeEventListener("keydown", esc);  
 }, []);  
  
 return (  
 <>  
 {/\* Top bar (mobile only) \*/}  
 <div  
 className="md:hidden sticky top-0 z-40 border-b"  
 style={{ background: "#061630", borderColor: "rgba(255,255,255,.08)" }}  
 >  
 <div className="flex items-center gap-3 px-3 py-2">  
 {/\* Logo full-width on the left \*/}  
 <div className="relative h-9 flex-1 overflow-hidden">  
 <Image src="/logo.png" alt="HygienePlus" fill className="object-contain" priority />  
 </div>  
  
 {/\* Hamburger on the right \*/}  
 <button  
 aria-label="Άνοιγμα μενού"  
 onClick={() => setOpen(true)}  
 className="inline-flex h-9 w-9 items-center justify-center rounded-xl border active:scale-[0.98]"  
 style={{ background: "rgba(255,255,255,.06)", borderColor: "rgba(255,255,255,.15)", color: "#ECF5F8" }}  
 >  
 <svg width="20" height="20" viewBox="0 0 24 24" fill="none">  
 <path d="M4 6h16M4 12h16M4 18h16" stroke="currentColor" strokeWidth="2" strokeLinecap="round" />  
 </svg>  
 </button>  
 </div>  
 </div>  
  
 {/\* Backdrop \*/}  
 <div  
 className={`md:hidden fixed inset-0 z-50 bg-black/40 transition-opacity ${open ? "opacity-100" : "pointer-events-none opacity-0"}`}  
 onClick={() => setOpen(false)}  
 />  
  
 {/\* Drawer with your existing Sidebar \*/}  
 <aside  
 className={`md:hidden fixed inset-y-0 left-0 z-50 w-72 max-w-[85vw] transform transition-transform ${open ? "translate-x-0" : "-translate-x-full"}`}  
 style={{ background: "#061630" }}  
 role="dialog"  
 aria-modal="true"  
 >  
 <div className="h-screen overflow-hidden flex flex-col">  
 <div className="flex items-center justify-between px-3 py-3 border-b" style={{ borderColor: "rgba(255,255,255,.08)" }}>  
 <div className="text-sm font-semibold" style={{ color: "#ECF5F8" }}>  
 {name || (role === "ADMIN" ? "Διαχειριστής" : "Χρήστης")}  
 </div>  
 <button  
 aria-label="Κλείσιμο μενού"  
 onClick={() => setOpen(false)}  
 className="inline-flex h-9 w-9 items-center justify-center rounded-xl border active:scale-[0.98]"  
 style={{ background: "rgba(255,255,255,.06)", borderColor: "rgba(255,255,255,.15)", color: "#ECF5F8" }}  
 >  
 <svg width="18" height="18" viewBox="0 0 24 24" fill="none">  
 <path d="M6 6l12 12M18 6l-12 12" stroke="currentColor" strokeWidth="2" strokeLinecap="round" />  
 </svg>  
 </button>  
 </div>  
 <div className="min-h-0 flex-1 overflow-y-auto">  
 <Sidebar role={role} name={name ?? undefined} onNavigate={() => setOpen(false)} />  
 </div>  
 </div>  
 </aside>  
 </>  
 );  
}

## src\components\ResponsiveTable.tsx

import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from "@/components/ui/table";  
import { ReactNode } from "react";  
  
/\*\*  
 \* A thin wrapper around shadcn <Table/> that guarantees mobile fitness.  
 \* Use className="table-fluid" on the <Table> to enforce wrapping and fixed layout.  
 \*/  
export function ResponsiveTable({  
 header,  
 rows,  
 className = "",  
}: {  
 header: ReactNode;  
 rows: ReactNode;  
 className?: string;  
}) {  
 return (  
 <div className="rounded-2xl border border-gray-200 overflow-hidden">  
 <Table className={`table-fluid ${className}`}>  
 {header}  
 {rows}  
 </Table>  
 </div>  
 );  
}  
  
/\* usage example:  
  
<ResponsiveTable  
 header={  
 <TableHeader>  
 <TableRow>  
 <TableHead>Όνομα</TableHead>  
 <TableHead className="hide-sm">Email</TableHead>  
 <TableHead>Ρόλος</TableHead>  
 <TableHead className="hide-sm">Προεπισκόπηση</TableHead>  
 </TableRow>  
 </TableHeader>  
 }  
 rows={  
 <TableBody>  
 {users.map(u => (  
 <TableRow key={u.id}>  
 <TableCell>{u.name}</TableCell>  
 <TableCell className="hide-sm">{u.email}</TableCell>  
 <TableCell><span className="nowrap inline-flex rounded-full px-2 py-1 text-xs bg-gray-100">USER</span></TableCell>  
 <TableCell className="hide-sm">{u.preview}</TableCell>  
 </TableRow>  
 ))}  
 </TableBody>  
 }  
/>  
  
\*/

## src\components\Sidebar.tsx

// src/components/Sidebar.tsx  
"use client";  
  
import Image from "next/image";  
import { usePathname, useRouter } from "next/navigation";  
import { signOut } from "next-auth/react";  
import { getMenu } from "@/lib/menu";  
  
type Role = "ADMIN" | "USER";  
  
function isActive(pathname: string, href: string) {  
 if (href === "/") return pathname === "/";  
 if (pathname === href) return true;  
 return pathname.startsWith(href + "/");  
}  
  
export default function Sidebar({  
 role,  
 name,  
 onNavigate, // <-- NEW  
}: {  
 role: Role;  
 name?: string | null;  
 onNavigate?: () => void; // <-- NEW  
}) {  
 const pathname = usePathname() || "/";  
 const router = useRouter();  
 const items = getMenu(role);  
  
 return (  
 <aside  
 className="flex h-screen w-60 shrink-0 flex-col border-r"  
 style={{  
 backgroundColor: "var(--sidebar-bg,#061630)",  
 color: "var(--sidebar-text,#ECF5F8)",  
 borderColor: "var(--sidebar-border,rgba(255,255,255,.08))",  
 }}  
 >  
 {/\* Top: full-width logo ABOVE the status \*/}  
 <div  
 className="px-4 pt-4 pb-3 border-b"  
 style={{ borderColor: "var(--sidebar-border,rgba(255,255,255,.08))" }}  
 >  
 <div className="relative w-full h-12 overflow-hidden rounded-md bg-white/5 mb-2">  
 <Image src="/logo.png" alt="Company logo" fill className="object-contain" priority />  
 </div>  
 <div className="leading-tight">  
 <div className="text-[11px]" style={{ color: "var(--sidebar-muted,#A7BECC)" }}>  
 Συνδεδεμένος — {name || (role === "ADMIN" ? "Διαχειριστής" : "Χρήστης")}  
 </div>  
 </div>  
 </div>  
  
 {/\* Nav \*/}  
 <nav className="p-2 space-y-1 overflow-auto">  
 {items.map((i) => {  
 const active = isActive(pathname, i.href);  
 return (  
 <button  
 key={i.href}  
 type="button"  
 onClick={() => {  
 router.push(i.href);  
 onNavigate?.(); // <-- CLOSE DRAWER IF PROVIDED  
 }}  
 className="w-full text-left flex items-center gap-2 rounded-md px-3 py-2 text-sm transition-colors"  
 style={{  
 backgroundColor: active ? "var(--sidebar-active-bg,rgba(37,195,244,.15))" : "transparent",  
 color: active ? "var(--sidebar-active-text,#FFFFFF)" : "var(--sidebar-text,#ECF5F8)",  
 }}  
 onMouseEnter={(e) => {  
 if (!active) (e.currentTarget.style.backgroundColor = "var(--sidebar-hover-bg,rgba(255,255,255,.06))");  
 }}  
 onMouseLeave={(e) => {  
 if (!active) (e.currentTarget.style.backgroundColor = "transparent");  
 }}  
 >  
 <span>{i.label}</span>  
 </button>  
 );  
 })}  
 </nav>  
  
 {/\* Bottom: sign out + contact block \*/}  
 <div className="mt-auto p-4 border-t" style={{ borderColor: "var(--sidebar-border,rgba(255,255,255,.08))" }}>  
 <button  
 type="button"  
 onClick={() => signOut({ callbackUrl: "/login" })}  
 className="w-full text-left rounded-md px-3 py-2 text-sm transition-colors"  
 style={{ backgroundColor: "transparent" }}  
 onMouseEnter={(e) => (e.currentTarget.style.backgroundColor = "var(--sidebar-hover-bg,rgba(255,255,255,.06))")}  
 onMouseLeave={(e) => (e.currentTarget.style.backgroundColor = "transparent")}  
 >  
 Αποσύνδεση  
 </button>  
  
 <div className="mt-3 text-[11px] leading-5" style={{ color: "var(--sidebar-muted,#A7BECC)" }}>  
 <div className="font-semibold mb-1" style={{ color: "var(--sidebar-text,#ECF5F8)" }}>Επικοινωνία</div>  
 <div>  
 Τηλ: <a className="underline underline-offset-2" href="tel:+302100000000">+30 210 000 0000</a>  
 </div>  
 <div>  
 Email: <a className="underline underline-offset-2" href="mailto:support@hygiene-plus.gr">support@hygiene-plus.gr</a>  
 </div>  
 <div className="mt-2">v0.1</div>  
 </div>  
 </div>  
 </aside>  
 );  
}

## src\components\StatCards.tsx

// src/components/StatCards.tsx  
"use client";  
  
import \* as React from "react";  
  
type Stats = {  
 users: { total: number; active: number };  
 files: { total: number };  
 assignments: { total: number };  
};  
  
export default function StatCards({ initial }: { initial?: Stats | null }) {  
 const [stats, setStats] = React.useState<Stats | null>(initial ?? null);  
 const [loading, setLoading] = React.useState(!initial);  
 const [error, setError] = React.useState<string | null>(null);  
  
 React.useEffect(() => {  
 if (initial) return;  
 let alive = true;  
 (async () => {  
 try {  
 setLoading(true);  
 const res = await fetch("/api/stats", { cache: "no-store" });  
 const json = await res.json();  
 if (!res.ok || !json?.ok) throw new Error(json?.error || "Failed to load stats");  
 if (alive) setStats(json.stats);  
 } catch (e: any) {  
 if (alive) setError(e?.message || "Failed to load stats");  
 } finally {  
 if (alive) setLoading(false);  
 }  
 })();  
 return () => { alive = false; };  
 }, [initial]);  
  
 const muted = "text-[color:var(--muted)]";  
 const cardClass =  
 "rounded-[var(--radius)] border border-[color:var(--border)] bg-[color:var(--card)] shadow-sm p-4";  
  
 const fmt = (n?: number) => (typeof n === "number" ? n.toLocaleString() : "—");  
  
 return (  
 <div className="grid grid-cols-1 gap-4 md:grid-cols-3">  
 {/\* Users \*/}  
 <div className={cardClass}>  
 <div className={`${muted} text-sm mb-3`}>Χρήστες</div>  
 <div className="h-1 w-10 rounded bg-[color:var(--primary,#25C3F4)] mb-3" />  
 <div className="text-sm">{loading ? "—" : `Σύνολο εγγεγραμμένων`}</div>  
 <div className="mt-2 text-2xl font-semibold">  
 {loading ? "—" : fmt(stats?.users?.total)}  
 </div>  
 </div>  
  
 {/\* Pending approvals (active vs total as a proxy) \*/}  
 <div className={cardClass}>  
 <div className={`${muted} text-sm mb-3`}>Εκκρεμείς εγκρίσεις</div>  
 <div className="h-1 w-10 rounded bg-[color:var(--primary,#25C3F4)] mb-3" />  
 <div className="text-sm">{loading ? "—" : `Αναμονή ενεργοποίησης`}</div>  
 <div className="mt-2 text-2xl font-semibold">  
 {loading || !stats ? "—" : fmt((stats.users.total ?? 0) - (stats.users.active ?? 0))}  
 </div>  
 </div>  
  
 {/\* Files \*/}  
 <div className={cardClass}>  
 <div className={`${muted} text-sm mb-3`}>Αρχεία</div>  
 <div className="h-1 w-10 rounded bg-[color:var(--primary,#25C3F4)] mb-3" />  
 <div className="text-sm">{loading ? "—" : `Σύνολο ανεβασμένων`}</div>  
 <div className="mt-2 text-2xl font-semibold">  
 {loading ? "—" : fmt(stats?.files?.total)}  
 </div>  
 </div>  
  
 {/\* Error helper (non-blocking) \*/}  
 {error && (  
 <div className="md:col-span-3 text-xs text-red-600">  
 Δεν ήταν δυνατή η φόρτωση των στατιστικών: {error}  
 </div>  
 )}  
 </div>  
 );  
}

## src\components\TrendMini.tsx

"use client";  
import { ResponsiveContainer, LineChart, Line, Tooltip } from "recharts";  
  
export default function TrendMini({ data }:{ data:{x:string|number;y:number}[] }) {  
 return (  
 <div className="mt-2 h-10">  
 <ResponsiveContainer width="100%" height="100%">  
 <LineChart data={data}>  
 <Line type="monotone" dataKey="y" stroke="var(--brand)" strokeWidth={2} dot={false} />  
 <Tooltip contentStyle={{ fontSize: 12 }} />  
 </LineChart>  
 </ResponsiveContainer>  
 </div>  
 );  
}

## src\components\UploadForm.tsx

"use client";  
  
import React from "react";  
  
type UploadedFile = {  
 id: string;  
 title: string;  
 originalName: string;  
 url: string; // /api/files/download/<key> (redirects to signed URL)  
 createdAt: string;  
};  
  
type Props = {  
 onUploaded?: (f: UploadedFile) => void;  
 buttonText?: string;  
 accept?: string;  
};  
  
export default function UploadForm({ onUploaded, buttonText = "Upload", accept = ".xlsx,.xlsm,.xls" }: Props) {  
 const [file, setFile] = React.useState<File | null>(null);  
 const [title, setTitle] = React.useState("");  
 const [busy, setBusy] = React.useState(false);  
 const [error, setError] = React.useState<string | null>(null);  
 const inputRef = React.useRef<HTMLInputElement | null>(null);  
  
 async function handleSubmit(e: React.FormEvent) {  
 e.preventDefault();  
 setError(null);  
  
 if (!file) {  
 setError("Please choose a file first.");  
 return;  
 }  
  
 if (file.size > 50 \* 1024 \* 1024) {  
 setError("File too large (max 50MB).");  
 return;  
 }  
  
 try {  
 setBusy(true);  
 const fd = new FormData();  
 fd.append("file", file);  
 if (title?.trim()) fd.append("title", title.trim());  
  
 const res = await fetch("/api/uploads", { method: "POST", body: fd });  
 const json = await res.json();  
 if (!res.ok) {  
 throw new Error(json?.detail || json?.error || "Upload failed");  
 }  
  
 // json.file is the DB record we shaped in the API  
 onUploaded?.(json.file);  
  
 // reset  
 setFile(null);  
 setTitle("");  
 if (inputRef.current) inputRef.current.value = "";  
 } catch (err: any) {  
 setError(err?.message || "Unexpected error");  
 } finally {  
 setBusy(false);  
 }  
 }  
  
 return (  
 <form onSubmit={handleSubmit} className="flex w-full flex-col gap-3 rounded-2xl border border-gray-200 p-4 shadow-sm">  
 <div className="flex flex-col gap-1">  
 <label className="text-sm font-medium text-gray-700">Excel file (.xlsx / .xlsm / .xls)</label>  
 <input  
 ref={inputRef}  
 type="file"  
 accept={accept}  
 onChange={(e) => {  
 const f = e.currentTarget.files?.[0] ?? null;  
 setFile(f || null);  
 if (f && !title) setTitle(f.name.replace(/\.[A-Za-z0-9]+$/, ""));  
 }}  
 className="block w-full rounded-lg border border-gray-300 bg-white p-2 text-sm file:mr-3 file:rounded-lg file:border-0 file:bg-gray-100 file:px-3 file:py-2"  
 />  
 </div>  
  
 <div className="flex flex-col gap-1">  
 <label className="text-sm font-medium text-gray-700">Title (optional)</label>  
 <input  
 type="text"  
 value={title}  
 onChange={(e) => setTitle(e.currentTarget.value)}  
 placeholder="e.g. October Deliveries"  
 className="w-full rounded-lg border border-gray-300 bg-white p-2 text-sm"  
 />  
 </div>  
  
 {error && <p className="text-sm text-red-600">{error}</p>}  
  
 <div className="flex items-center justify-end gap-2">  
 <button  
 type="button"  
 onClick={() => {  
 setError(null);  
 setFile(null);  
 setTitle("");  
 if (inputRef.current) inputRef.current.value = "";  
 }}  
 disabled={busy}  
 className="rounded-xl border border-gray-300 px-4 py-2 text-sm"  
 >  
 Clear  
 </button>  
 <button  
 type="submit"  
 disabled={busy || !file}  
 className="rounded-xl bg-black px-4 py-2 text-sm font-semibold text-white disabled:cursor-not-allowed disabled:opacity-60"  
 >  
 {busy ? "Uploading…" : buttonText}  
 </button>  
 </div>  
 </form>  
 );  
}

## src\components\ui\Button.tsx

import \* as React from "react"  
import { Slot } from "@radix-ui/react-slot"  
import { cva, type VariantProps } from "class-variance-authority"  
  
import { cn } from "@/lib/utils"  
  
const buttonVariants = cva(  
 "inline-flex items-center justify-center gap-2 whitespace-nowrap rounded-md text-sm font-medium transition-all disabled:pointer-events-none disabled:opacity-50 [&\_svg]:pointer-events-none [&\_svg:not([class\*='size-'])]:size-4 shrink-0 [&\_svg]:shrink-0 outline-none focus-visible:border-ring focus-visible:ring-ring/50 focus-visible:ring-[3px] aria-invalid:ring-destructive/20 dark:aria-invalid:ring-destructive/40 aria-invalid:border-destructive",  
 {  
 variants: {  
 variant: {  
 default: "bg-primary text-primary-foreground hover:bg-primary/90",  
 destructive:  
 "bg-destructive text-white hover:bg-destructive/90 focus-visible:ring-destructive/20 dark:focus-visible:ring-destructive/40 dark:bg-destructive/60",  
 outline:  
 "border bg-background shadow-xs hover:bg-accent hover:text-accent-foreground dark:bg-input/30 dark:border-input dark:hover:bg-input/50",  
 secondary:  
 "bg-secondary text-secondary-foreground hover:bg-secondary/80",  
 ghost:  
 "hover:bg-accent hover:text-accent-foreground dark:hover:bg-accent/50",  
 link: "text-primary underline-offset-4 hover:underline",  
 },  
 size: {  
 default: "h-9 px-4 py-2 has-[>svg]:px-3",  
 sm: "h-8 rounded-md gap-1.5 px-3 has-[>svg]:px-2.5",  
 lg: "h-10 rounded-md px-6 has-[>svg]:px-4",  
 icon: "size-9",  
 "icon-sm": "size-8",  
 "icon-lg": "size-10",  
 },  
 },  
 defaultVariants: {  
 variant: "default",  
 size: "default",  
 },  
 }  
)  
  
function Button({  
 className,  
 variant,  
 size,  
 asChild = false,  
 ...props  
}: React.ComponentProps<"button"> &  
 VariantProps<typeof buttonVariants> & {  
 asChild?: boolean  
 }) {  
 const Comp = asChild ? Slot : "button"  
  
 return (  
 <Comp  
 data-slot="button"  
 className={cn(buttonVariants({ variant, size, className }))}  
 {...props}  
 />  
 )  
}  
  
export { Button, buttonVariants }

## src\components\ui\badge.tsx

import \* as React from "react"  
import { Slot } from "@radix-ui/react-slot"  
import { cva, type VariantProps } from "class-variance-authority"  
  
import { cn } from "@/lib/utils"  
  
const badgeVariants = cva(  
 "inline-flex items-center justify-center rounded-md border px-2 py-0.5 text-xs font-medium w-fit whitespace-nowrap shrink-0 [&>svg]:size-3 gap-1 [&>svg]:pointer-events-none focus-visible:border-ring focus-visible:ring-ring/50 focus-visible:ring-[3px] aria-invalid:ring-destructive/20 dark:aria-invalid:ring-destructive/40 aria-invalid:border-destructive transition-[color,box-shadow] overflow-hidden",  
 {  
 variants: {  
 variant: {  
 default:  
 "border-transparent bg-primary text-primary-foreground [a&]:hover:bg-primary/90",  
 secondary:  
 "border-transparent bg-secondary text-secondary-foreground [a&]:hover:bg-secondary/90",  
 destructive:  
 "border-transparent bg-destructive text-white [a&]:hover:bg-destructive/90 focus-visible:ring-destructive/20 dark:focus-visible:ring-destructive/40 dark:bg-destructive/60",  
 outline:  
 "text-foreground [a&]:hover:bg-accent [a&]:hover:text-accent-foreground",  
 },  
 },  
 defaultVariants: {  
 variant: "default",  
 },  
 }  
)  
  
function Badge({  
 className,  
 variant,  
 asChild = false,  
 ...props  
}: React.ComponentProps<"span"> &  
 VariantProps<typeof badgeVariants> & { asChild?: boolean }) {  
 const Comp = asChild ? Slot : "span"  
  
 return (  
 <Comp  
 data-slot="badge"  
 className={cn(badgeVariants({ variant }), className)}  
 {...props}  
 />  
 )  
}  
  
export { Badge, badgeVariants }

## src\components\ui\card.tsx

import \* as React from "react"  
  
import { cn } from "@/lib/utils"  
  
function Card({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="card"  
 className={cn(  
 "bg-card text-card-foreground flex flex-col gap-6 rounded-xl border py-6 shadow-sm",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function CardHeader({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="card-header"  
 className={cn(  
 "@container/card-header grid auto-rows-min grid-rows-[auto\_auto] items-start gap-2 px-6 has-data-[slot=card-action]:grid-cols-[1fr\_auto] [.border-b]:pb-6",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function CardTitle({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="card-title"  
 className={cn("leading-none font-semibold", className)}  
 {...props}  
 />  
 )  
}  
  
function CardDescription({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="card-description"  
 className={cn("text-muted-foreground text-sm", className)}  
 {...props}  
 />  
 )  
}  
  
function CardAction({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="card-action"  
 className={cn(  
 "col-start-2 row-span-2 row-start-1 self-start justify-self-end",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function CardContent({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="card-content"  
 className={cn("px-6", className)}  
 {...props}  
 />  
 )  
}  
  
function CardFooter({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="card-footer"  
 className={cn("flex items-center px-6 [.border-t]:pt-6", className)}  
 {...props}  
 />  
 )  
}  
  
export {  
 Card,  
 CardHeader,  
 CardFooter,  
 CardTitle,  
 CardAction,  
 CardDescription,  
 CardContent,  
}

## src\components\ui\input.tsx

import \* as React from "react"  
  
import { cn } from "@/lib/utils"  
  
function Input({ className, type, ...props }: React.ComponentProps<"input">) {  
 return (  
 <input  
 type={type}  
 data-slot="input"  
 className={cn(  
 "file:text-foreground placeholder:text-muted-foreground selection:bg-primary selection:text-primary-foreground dark:bg-input/30 border-input h-9 w-full min-w-0 rounded-md border bg-transparent px-3 py-1 text-base shadow-xs transition-[color,box-shadow] outline-none file:inline-flex file:h-7 file:border-0 file:bg-transparent file:text-sm file:font-medium disabled:pointer-events-none disabled:cursor-not-allowed disabled:opacity-50 md:text-sm",  
 "focus-visible:border-ring focus-visible:ring-ring/50 focus-visible:ring-[3px]",  
 "aria-invalid:ring-destructive/20 dark:aria-invalid:ring-destructive/40 aria-invalid:border-destructive",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
export { Input }

## src\components\ui\label.tsx

"use client"  
  
import \* as React from "react"  
import \* as LabelPrimitive from "@radix-ui/react-label"  
  
import { cn } from "@/lib/utils"  
  
function Label({  
 className,  
 ...props  
}: React.ComponentProps<typeof LabelPrimitive.Root>) {  
 return (  
 <LabelPrimitive.Root  
 data-slot="label"  
 className={cn(  
 "flex items-center gap-2 text-sm leading-none font-medium select-none group-data-[disabled=true]:pointer-events-none group-data-[disabled=true]:opacity-50 peer-disabled:cursor-not-allowed peer-disabled:opacity-50",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
export { Label }

## src\components\ui\select.tsx

"use client"  
  
import \* as React from "react"  
import \* as SelectPrimitive from "@radix-ui/react-select"  
import { CheckIcon, ChevronDownIcon, ChevronUpIcon } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
function Select({  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.Root>) {  
 return <SelectPrimitive.Root data-slot="select" {...props} />  
}  
  
function SelectGroup({  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.Group>) {  
 return <SelectPrimitive.Group data-slot="select-group" {...props} />  
}  
  
function SelectValue({  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.Value>) {  
 return <SelectPrimitive.Value data-slot="select-value" {...props} />  
}  
  
function SelectTrigger({  
 className,  
 size = "default",  
 children,  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.Trigger> & {  
 size?: "sm" | "default"  
}) {  
 return (  
 <SelectPrimitive.Trigger  
 data-slot="select-trigger"  
 data-size={size}  
 className={cn(  
 "border-input data-[placeholder]:text-muted-foreground [&\_svg:not([class\*='text-'])]:text-muted-foreground focus-visible:border-ring focus-visible:ring-ring/50 aria-invalid:ring-destructive/20 dark:aria-invalid:ring-destructive/40 aria-invalid:border-destructive dark:bg-input/30 dark:hover:bg-input/50 flex w-fit items-center justify-between gap-2 rounded-md border bg-transparent px-3 py-2 text-sm whitespace-nowrap shadow-xs transition-[color,box-shadow] outline-none focus-visible:ring-[3px] disabled:cursor-not-allowed disabled:opacity-50 data-[size=default]:h-9 data-[size=sm]:h-8 \*:data-[slot=select-value]:line-clamp-1 \*:data-[slot=select-value]:flex \*:data-[slot=select-value]:items-center \*:data-[slot=select-value]:gap-2 [&\_svg]:pointer-events-none [&\_svg]:shrink-0 [&\_svg:not([class\*='size-'])]:size-4",  
 className  
 )}  
 {...props}  
 >  
 {children}  
 <SelectPrimitive.Icon asChild>  
 <ChevronDownIcon className="size-4 opacity-50" />  
 </SelectPrimitive.Icon>  
 </SelectPrimitive.Trigger>  
 )  
}  
  
function SelectContent({  
 className,  
 children,  
 position = "popper",  
 align = "center",  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.Content>) {  
 return (  
 <SelectPrimitive.Portal>  
 <SelectPrimitive.Content  
 data-slot="select-content"  
 className={cn(  
 "bg-popover text-popover-foreground data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2 relative z-50 max-h-(--radix-select-content-available-height) min-w-[8rem] origin-(--radix-select-content-transform-origin) overflow-x-hidden overflow-y-auto rounded-md border shadow-md",  
 position === "popper" &&  
 "data-[side=bottom]:translate-y-1 data-[side=left]:-translate-x-1 data-[side=right]:translate-x-1 data-[side=top]:-translate-y-1",  
 className  
 )}  
 position={position}  
 align={align}  
 {...props}  
 >  
 <SelectScrollUpButton />  
 <SelectPrimitive.Viewport  
 className={cn(  
 "p-1",  
 position === "popper" &&  
 "h-[var(--radix-select-trigger-height)] w-full min-w-[var(--radix-select-trigger-width)] scroll-my-1"  
 )}  
 >  
 {children}  
 </SelectPrimitive.Viewport>  
 <SelectScrollDownButton />  
 </SelectPrimitive.Content>  
 </SelectPrimitive.Portal>  
 )  
}  
  
function SelectLabel({  
 className,  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.Label>) {  
 return (  
 <SelectPrimitive.Label  
 data-slot="select-label"  
 className={cn("text-muted-foreground px-2 py-1.5 text-xs", className)}  
 {...props}  
 />  
 )  
}  
  
function SelectItem({  
 className,  
 children,  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.Item>) {  
 return (  
 <SelectPrimitive.Item  
 data-slot="select-item"  
 className={cn(  
 "focus:bg-accent focus:text-accent-foreground [&\_svg:not([class\*='text-'])]:text-muted-foreground relative flex w-full cursor-default items-center gap-2 rounded-sm py-1.5 pr-8 pl-2 text-sm outline-hidden select-none data-[disabled]:pointer-events-none data-[disabled]:opacity-50 [&\_svg]:pointer-events-none [&\_svg]:shrink-0 [&\_svg:not([class\*='size-'])]:size-4 \*:[span]:last:flex \*:[span]:last:items-center \*:[span]:last:gap-2",  
 className  
 )}  
 {...props}  
 >  
 <span className="absolute right-2 flex size-3.5 items-center justify-center">  
 <SelectPrimitive.ItemIndicator>  
 <CheckIcon className="size-4" />  
 </SelectPrimitive.ItemIndicator>  
 </span>  
 <SelectPrimitive.ItemText>{children}</SelectPrimitive.ItemText>  
 </SelectPrimitive.Item>  
 )  
}  
  
function SelectSeparator({  
 className,  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.Separator>) {  
 return (  
 <SelectPrimitive.Separator  
 data-slot="select-separator"  
 className={cn("bg-border pointer-events-none -mx-1 my-1 h-px", className)}  
 {...props}  
 />  
 )  
}  
  
function SelectScrollUpButton({  
 className,  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.ScrollUpButton>) {  
 return (  
 <SelectPrimitive.ScrollUpButton  
 data-slot="select-scroll-up-button"  
 className={cn(  
 "flex cursor-default items-center justify-center py-1",  
 className  
 )}  
 {...props}  
 >  
 <ChevronUpIcon className="size-4" />  
 </SelectPrimitive.ScrollUpButton>  
 )  
}  
  
function SelectScrollDownButton({  
 className,  
 ...props  
}: React.ComponentProps<typeof SelectPrimitive.ScrollDownButton>) {  
 return (  
 <SelectPrimitive.ScrollDownButton  
 data-slot="select-scroll-down-button"  
 className={cn(  
 "flex cursor-default items-center justify-center py-1",  
 className  
 )}  
 {...props}  
 >  
 <ChevronDownIcon className="size-4" />  
 </SelectPrimitive.ScrollDownButton>  
 )  
}  
  
export {  
 Select,  
 SelectContent,  
 SelectGroup,  
 SelectItem,  
 SelectLabel,  
 SelectScrollDownButton,  
 SelectScrollUpButton,  
 SelectSeparator,  
 SelectTrigger,  
 SelectValue,  
}

## src\components\ui\sheet.tsx

"use client"  
  
import \* as React from "react"  
import \* as SheetPrimitive from "@radix-ui/react-dialog"  
import { XIcon } from "lucide-react"  
  
import { cn } from "@/lib/utils"  
  
function Sheet({ ...props }: React.ComponentProps<typeof SheetPrimitive.Root>) {  
 return <SheetPrimitive.Root data-slot="sheet" {...props} />  
}  
  
function SheetTrigger({  
 ...props  
}: React.ComponentProps<typeof SheetPrimitive.Trigger>) {  
 return <SheetPrimitive.Trigger data-slot="sheet-trigger" {...props} />  
}  
  
function SheetClose({  
 ...props  
}: React.ComponentProps<typeof SheetPrimitive.Close>) {  
 return <SheetPrimitive.Close data-slot="sheet-close" {...props} />  
}  
  
function SheetPortal({  
 ...props  
}: React.ComponentProps<typeof SheetPrimitive.Portal>) {  
 return <SheetPrimitive.Portal data-slot="sheet-portal" {...props} />  
}  
  
function SheetOverlay({  
 className,  
 ...props  
}: React.ComponentProps<typeof SheetPrimitive.Overlay>) {  
 return (  
 <SheetPrimitive.Overlay  
 data-slot="sheet-overlay"  
 className={cn(  
 "data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 fixed inset-0 z-50 bg-black/50",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function SheetContent({  
 className,  
 children,  
 side = "right",  
 ...props  
}: React.ComponentProps<typeof SheetPrimitive.Content> & {  
 side?: "top" | "right" | "bottom" | "left"  
}) {  
 return (  
 <SheetPortal>  
 <SheetOverlay />  
 <SheetPrimitive.Content  
 data-slot="sheet-content"  
 className={cn(  
 "bg-background data-[state=open]:animate-in data-[state=closed]:animate-out fixed z-50 flex flex-col gap-4 shadow-lg transition ease-in-out data-[state=closed]:duration-300 data-[state=open]:duration-500",  
 side === "right" &&  
 "data-[state=closed]:slide-out-to-right data-[state=open]:slide-in-from-right inset-y-0 right-0 h-full w-3/4 border-l sm:max-w-sm",  
 side === "left" &&  
 "data-[state=closed]:slide-out-to-left data-[state=open]:slide-in-from-left inset-y-0 left-0 h-full w-3/4 border-r sm:max-w-sm",  
 side === "top" &&  
 "data-[state=closed]:slide-out-to-top data-[state=open]:slide-in-from-top inset-x-0 top-0 h-auto border-b",  
 side === "bottom" &&  
 "data-[state=closed]:slide-out-to-bottom data-[state=open]:slide-in-from-bottom inset-x-0 bottom-0 h-auto border-t",  
 className  
 )}  
 {...props}  
 >  
 {children}  
 <SheetPrimitive.Close className="ring-offset-background focus:ring-ring data-[state=open]:bg-secondary absolute top-4 right-4 rounded-xs opacity-70 transition-opacity hover:opacity-100 focus:ring-2 focus:ring-offset-2 focus:outline-hidden disabled:pointer-events-none">  
 <XIcon className="size-4" />  
 <span className="sr-only">Close</span>  
 </SheetPrimitive.Close>  
 </SheetPrimitive.Content>  
 </SheetPortal>  
 )  
}  
  
function SheetHeader({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="sheet-header"  
 className={cn("flex flex-col gap-1.5 p-4", className)}  
 {...props}  
 />  
 )  
}  
  
function SheetFooter({ className, ...props }: React.ComponentProps<"div">) {  
 return (  
 <div  
 data-slot="sheet-footer"  
 className={cn("mt-auto flex flex-col gap-2 p-4", className)}  
 {...props}  
 />  
 )  
}  
  
function SheetTitle({  
 className,  
 ...props  
}: React.ComponentProps<typeof SheetPrimitive.Title>) {  
 return (  
 <SheetPrimitive.Title  
 data-slot="sheet-title"  
 className={cn("text-foreground font-semibold", className)}  
 {...props}  
 />  
 )  
}  
  
function SheetDescription({  
 className,  
 ...props  
}: React.ComponentProps<typeof SheetPrimitive.Description>) {  
 return (  
 <SheetPrimitive.Description  
 data-slot="sheet-description"  
 className={cn("text-muted-foreground text-sm", className)}  
 {...props}  
 />  
 )  
}  
  
export {  
 Sheet,  
 SheetTrigger,  
 SheetClose,  
 SheetContent,  
 SheetHeader,  
 SheetFooter,  
 SheetTitle,  
 SheetDescription,  
}

## src\components\ui\table.tsx

"use client"  
  
import \* as React from "react"  
  
import { cn } from "@/lib/utils"  
  
function Table({ className, ...props }: React.ComponentProps<"table">) {  
 return (  
 <div  
 data-slot="table-container"  
 className="relative w-full overflow-x-auto"  
 >  
 <table  
 data-slot="table"  
 className={cn("w-full caption-bottom text-sm", className)}  
 {...props}  
 />  
 </div>  
 )  
}  
  
function TableHeader({ className, ...props }: React.ComponentProps<"thead">) {  
 return (  
 <thead  
 data-slot="table-header"  
 className={cn("[&\_tr]:border-b", className)}  
 {...props}  
 />  
 )  
}  
  
function TableBody({ className, ...props }: React.ComponentProps<"tbody">) {  
 return (  
 <tbody  
 data-slot="table-body"  
 className={cn("[&\_tr:last-child]:border-0", className)}  
 {...props}  
 />  
 )  
}  
  
function TableFooter({ className, ...props }: React.ComponentProps<"tfoot">) {  
 return (  
 <tfoot  
 data-slot="table-footer"  
 className={cn(  
 "bg-muted/50 border-t font-medium [&>tr]:last:border-b-0",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function TableRow({ className, ...props }: React.ComponentProps<"tr">) {  
 return (  
 <tr  
 data-slot="table-row"  
 className={cn(  
 "hover:bg-muted/50 data-[state=selected]:bg-muted border-b transition-colors",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function TableHead({ className, ...props }: React.ComponentProps<"th">) {  
 return (  
 <th  
 data-slot="table-head"  
 className={cn(  
 "text-foreground h-10 px-2 text-left align-middle font-medium whitespace-nowrap [&:has([role=checkbox])]:pr-0 [&>[role=checkbox]]:translate-y-[2px]",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function TableCell({ className, ...props }: React.ComponentProps<"td">) {  
 return (  
 <td  
 data-slot="table-cell"  
 className={cn(  
 "p-2 align-middle whitespace-nowrap [&:has([role=checkbox])]:pr-0 [&>[role=checkbox]]:translate-y-[2px]",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function TableCaption({  
 className,  
 ...props  
}: React.ComponentProps<"caption">) {  
 return (  
 <caption  
 data-slot="table-caption"  
 className={cn("text-muted-foreground mt-4 text-sm", className)}  
 {...props}  
 />  
 )  
}  
  
export {  
 Table,  
 TableHeader,  
 TableBody,  
 TableFooter,  
 TableHead,  
 TableRow,  
 TableCell,  
 TableCaption,  
}

## src\components\ui\tabs.tsx

"use client"  
  
import \* as React from "react"  
import \* as TabsPrimitive from "@radix-ui/react-tabs"  
  
import { cn } from "@/lib/utils"  
  
function Tabs({  
 className,  
 ...props  
}: React.ComponentProps<typeof TabsPrimitive.Root>) {  
 return (  
 <TabsPrimitive.Root  
 data-slot="tabs"  
 className={cn("flex flex-col gap-2", className)}  
 {...props}  
 />  
 )  
}  
  
function TabsList({  
 className,  
 ...props  
}: React.ComponentProps<typeof TabsPrimitive.List>) {  
 return (  
 <TabsPrimitive.List  
 data-slot="tabs-list"  
 className={cn(  
 "bg-muted text-muted-foreground inline-flex h-9 w-fit items-center justify-center rounded-lg p-[3px]",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function TabsTrigger({  
 className,  
 ...props  
}: React.ComponentProps<typeof TabsPrimitive.Trigger>) {  
 return (  
 <TabsPrimitive.Trigger  
 data-slot="tabs-trigger"  
 className={cn(  
 "data-[state=active]:bg-background dark:data-[state=active]:text-foreground focus-visible:border-ring focus-visible:ring-ring/50 focus-visible:outline-ring dark:data-[state=active]:border-input dark:data-[state=active]:bg-input/30 text-foreground dark:text-muted-foreground inline-flex h-[calc(100%-1px)] flex-1 items-center justify-center gap-1.5 rounded-md border border-transparent px-2 py-1 text-sm font-medium whitespace-nowrap transition-[color,box-shadow] focus-visible:ring-[3px] focus-visible:outline-1 disabled:pointer-events-none disabled:opacity-50 data-[state=active]:shadow-sm [&\_svg]:pointer-events-none [&\_svg]:shrink-0 [&\_svg:not([class\*='size-'])]:size-4",  
 className  
 )}  
 {...props}  
 />  
 )  
}  
  
function TabsContent({  
 className,  
 ...props  
}: React.ComponentProps<typeof TabsPrimitive.Content>) {  
 return (  
 <TabsPrimitive.Content  
 data-slot="tabs-content"  
 className={cn("flex-1 outline-none", className)}  
 {...props}  
 />  
 )  
}  
  
export { Tabs, TabsList, TabsTrigger, TabsContent }

## src\lib\apiKeyAuth.ts

// src/lib/apiKeyAuth.ts  
export function requireApiKey(req: Request) {  
 const header1 = req.headers.get("x-api-key")?.trim();  
 const auth = req.headers.get("authorization") || "";  
 const header2 = auth.toLowerCase().startsWith("bearer ") ? auth.slice(7).trim() : undefined;  
 const provided = header1 || header2;  
  
 const expected = (process.env.INTEGRATIONS\_API\_KEY || "").trim();  
 if (!expected || !provided || provided !== expected) {  
 return {  
 ok: false as const,  
 res: new Response(JSON.stringify({ error: "Unauthorized" }), {  
 status: 401,  
 headers: { "Content-Type": "application/json" },  
 }),  
 };  
 }  
 return { ok: true as const };  
}

## src\lib\assignmentRules.ts

import { prisma } from "@/lib/prisma";  
  
/\*\* Pull emails from any text (filename, title, URL, etc.) \*/  
export function extractEmailsFromText(text?: string | null): string[] {  
 if (!text) return [];  
 try {  
 if (/^https?:\/\//i.test(text)) {  
 const u = new URL(text);  
 text = decodeURIComponent(u.pathname.split("/").pop() || "");  
 }  
 } catch { /\* ignore URL parse errors \*/ }  
  
 const emailRe = /[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}/gi;  
 const found = text.match(emailRe) || [];  
 return Array.from(new Set(found.map((e) => e.trim().toLowerCase())));  
}  
  
/\*\* Resolve ACTIVE non-admin users whose emails match \*/  
export async function resolveAssigneeIdsByEmails(emails: string[]) {  
 if (emails.length === 0) return [];  
 const users = await prisma.user.findMany({  
 where: { email: { in: emails }, role: "USER", status: "ACTIVE" },  
 select: { id: true },  
 });  
 return users.map((u) => u.id);  
}

## src\lib\audit.ts

import { prisma } from "@/lib/prisma";  
  
type LogParams = {  
 actorId?: string | null;  
 action:  
 | "USER\_CREATED"  
 | "SUBSCRIPTION\_TOGGLED"  
 | "FILE\_UPLOADED"  
 | "FILE\_ASSIGNED"  
 | "APIKEY\_CREATED"  
 | "APIKEY\_REVOKED"  
 | "DOWNLOAD\_GRANTED"  
 | "SUPPORT\_TICKET"; // <-- add this  
 targetId?: string | null;  
 target?: string | null;  
 meta?: Record<string, any> | null;  
};  
  
  
export async function logAudit(p: LogParams) {  
 await prisma.auditLog.create({  
 data: {  
 actorId: p.actorId ?? null,  
 action: p.action,  
 targetId: p.targetId ?? null,  
 target: p.target ?? null,  
 meta: p.meta ?? null,  
 },  
 });  
}

## src\lib\auth-helpers.ts

import { getServerSession } from "next-auth";  
import { authOptions } from "@/lib/auth";  
  
export async function currentUser() {  
 const session = await getServerSession(authOptions);  
 return session?.user ?? null; // has .role and .id  
}  
  
export async function requireRole(role: "ADMIN" | "USER") {  
 const user = await currentUser();  
 if (!user) return { ok: false as const, status: 401 as const };  
 if (role === "ADMIN" && (user as any).role !== "ADMIN")  
 return { ok: false as const, status: 403 as const };  
 return { ok: true as const, user };  
}

## src\lib\auth.ts

// src/lib/auth.ts  
import type { NextAuthOptions } from "next-auth";  
import Credentials from "next-auth/providers/credentials";  
import { prisma } from "@/lib/prisma";  
import bcrypt from "bcrypt";  
import { z } from "zod";  
  
export const authOptions: NextAuthOptions = {  
 session: { strategy: "jwt" },  
 providers: [  
 Credentials({  
 name: "Email & Password",  
 credentials: { email: {}, password: {} },  
 async authorize(raw) {  
 const creds = z.object({  
 email: z.string().email(),  
 password: z.string().min(6),  
 }).parse(raw);  
  
 //const user = await prisma.user.findUnique({ where: { email: creds.email } });  
 //if (!user || !user.passwordHash) return null;  
  
 //const ok = await bcrypt.compare(creds.password, user.passwordHash);  
 //if (!ok) return null;  
  
 //return { id: user.id, email: user.email, name: user.name ?? "", role: user.role };  
  
 // ...  
 const user = await prisma.user.findUnique({ where: { email: creds.email } });  
 if (!user || !user.passwordHash) return null;  
  
 const ok = await bcrypt.compare(creds.password, user.passwordHash);  
 if (!ok) return null;  
  
 // 👇 block if not approved  
 if (user.status !== 'ACTIVE') {  
 // Option A: return null (generic error)  
 // return null;  
  
 // Option B: throw a custom error to show message on /login  
 throw new Error(user.status === 'PENDING' ? 'AccountPending' : 'AccountSuspended');  
 }  
  
 return { id: user.id, email: user.email, name: user.name ?? "", role: user.role };  
 },  
 }),  
 ],  
 callbacks: {  
 async jwt({ token, user }) {  
 if (user) {  
 (token as any).id = (user as any).id;  
 (token as any).role = (user as any).role;  
 }  
 return token;  
 },  
 async session({ session, token }) {  
 if (session.user) {  
 (session.user as any).id = (token as any).id;  
 (session.user as any).role = (token as any).role;  
 }  
 return session;  
 },  
 },  
 pages: { signIn: "/login" },  
};

## src\lib\files-validate.ts

// src/lib/files-validate.ts  
import { extname } from "node:path";  
  
export const ALLOWED\_EXT = new Set([".xlsx", ".xlsm", ".xls"]);  
export const MAX\_BYTES = 50 \* 1024 \* 1024; // 50MB  
  
export function checkExt(name: string) {  
 const ext = extname(name || "").toLowerCase();  
 if (!ALLOWED\_EXT.has(ext)) throw new Error("Unsupported file type");  
 return ext;  
}

## src\lib\i18n-el.ts

// src/lib/i18n-el.ts  
export const el = {  
 // Generic  
 "Sign out": "Αποσύνδεση",  
 "Signed in": "Συνδεδεμένος",  
 "Admin": "Διαχειριστής",  
 "User": "Χρήστης",  
 "Dashboard": "Πίνακας Ελέγχου",  
 "Users": "Χρήστες",  
 "Files": "Αρχεία",  
 "Settings": "Ρυθμίσεις",  
 "Audit": "Αρχεία Καταγραφής",  
 "My Files": "Τα Αρχεία μου",  
 "Assignments": "Αναθέσεις",  
 "Support": "Υποστήριξη",  
  
 // Login / Register  
 "Log in": "Σύνδεση",  
 "Logging in...": "Σύνδεση...",  
 "Email": "Ηλεκτρονικό ταχυδρομείο",  
 "Password": "Κωδικός πρόσβασης",  
 "Wrong email or password": "Λάθος email ή κωδικός",  
 "Create account": "Δημιουργία λογαριασμού",  
 "Creating...": "Δημιουργία...",  
 "Already have an account?": "Έχετε ήδη λογαριασμό;",  
 "Sign in": "Σύνδεση",  
 "New here?": "Νέος χρήστης;",  
 "Access your dashboard after signing up.": "Αποκτήστε πρόσβαση μετά την εγγραφή σας.",  
 "Your account is awaiting admin approval.": "Ο λογαριασμός σας αναμένει έγκριση από διαχειριστή.",  
 "Your account is suspended. Contact support.": "Ο λογαριασμός σας έχει ανασταλεί. Επικοινωνήστε με την υποστήριξη.",  
 "Signup successful. Wait for admin approval before logging in.": "Η εγγραφή ολοκληρώθηκε. Περιμένετε έγκριση από διαχειριστή.",  
  
 // Admin Dashboard  
 "Users": "Χρήστες",  
 "Pending approvals": "Εκκρεμείς εγκρίσεις",  
 "Files": "Αρχεία",  
 "Quick actions": "Γρήγορες ενέργειες",  
 "Manage users": "Διαχείριση χρηστών",  
 "Upload file": "Ανέβασμα αρχείου",  
 "View logs": "Προβολή αρχείων καταγραφής",  
 "Uploads — last 30 days": "Ανεβάσματα — τελευταίες 30 ημέρες",  
 "Daily count": "Καθημερινός αριθμός",  
 "Recent activity": "Πρόσφατη δραστηριότητα",  
 "View all": "Προβολή όλων",  
 "No activity yet.": "Δεν υπάρχουν δραστηριότητες ακόμη.",  
  
 // Admin Users  
 "Create User": "Δημιουργία Χρήστη",  
 "Name": "Όνομα",  
 "Email": "Ηλεκτρονικό ταχυδρομείο",  
 "Password": "Κωδικός πρόσβασης",  
 "Role": "Ρόλος",  
 "Status": "Κατάσταση",  
 "Subscription": "Συνδρομή",  
 "Created": "Δημιουργήθηκε",  
 "Actions": "Ενέργειες",  
 "Active": "Ενεργό",  
 "Inactive": "Ανενεργό",  
 "Approve": "Έγκριση",  
 "Suspend": "Αναστολή",  
 "Delete": "Διαγραφή",  
 "No users found.": "Δεν βρέθηκαν χρήστες.",  
 "Refresh": "Ανανέωση",  
 "Loading…": "Φόρτωση…",  
  
 // Admin Files  
 "All Files": "Όλα τα αρχεία",  
 "Created": "Ημερομηνία δημιουργίας",  
 "Assigned to": "Ανατεθειμένο σε",  
 "Assign": "Ανάθεση",  
 "Actions": "Ενέργειες",  
 "No user assigned": "Δεν έχει ανατεθεί σε χρήστη",  
 "Select user…": "Επιλογή χρήστη…",  
 "Download": "Λήψη",  
 "No file URL": "Δεν υπάρχει σύνδεσμος αρχείου",  
  
 // Admin Upload  
 "Upload Excel File": "Ανέβασμα αρχείου Excel",  
 "Title (optional)": "Τίτλος (προαιρετικά)",  
 "Upload": "Ανέβασμα",  
 "No file selected.": "Δεν επιλέχθηκε αρχείο.",  
 "Uploaded": "Ανέβηκε",  
  
 // User Dashboard  
 "Files assigned to me": "Αρχεία που μου έχουν ανατεθεί",  
 "My uploads": "Τα δικά μου ανεβάσματα",  
 "Total assignments": "Σύνολο αναθέσεων",  
 "Files I uploaded": "Αρχεία που ανέβασα",  
 "Shortcuts": "Συντομεύσεις",  
 "Common actions": "Συνηθισμένες ενέργειες",  
 "My files": "Τα αρχεία μου",  
 "Support": "Υποστήριξη",  
 "Recent assigned files": "Πρόσφατα ανατεθειμένα αρχεία",  
 "No files yet.": "Δεν υπάρχουν αρχεία ακόμη.",  
 "View all": "Προβολή όλων",  
};

## src\lib\menu.ts

// src/lib/menu.ts  
export type Role = "ADMIN" | "USER";  
export type MenuItem = { label: string; href: string };  
  
export function getMenu(role: Role): MenuItem[] {  
 if (role === "ADMIN") {  
 return [  
 { label: "Πίνακας Ελέγχου", href: "/dashboard/admin" },  
 { label: "Χρήστες", href: "/admin/users" },  
 { label: "Αρχεία", href: "/admin/files" },  
 { label: "Υποστήριξη", href: "/admin/support" }, // ← add this  
 { label: "Αρχεία Καταγραφής", href: "/admin/audit" },  
 ];  
 }  
  
 return [  
 { label: "Πίνακας Ελέγχου", href: "/dashboard" },  
 { label: "Τα Αρχεία μου", href: "/files" },  
 { label: "Υποστήριξη", href: "/support" },  
 ];  
}

## src\lib\prisma.ts

// src/lib/prisma.ts  
import { PrismaClient } from "@prisma/client";  
  
declare global {  
 // eslint-disable-next-line no-var  
 var \_\_PRISMA\_\_: PrismaClient | undefined;  
}  
  
/\*\*  
 \* Use a single PrismaClient across hot reloads in dev  
 \* and a new one per serverless worker in production.  
 \*/  
export const prisma =  
 global.\_\_PRISMA\_\_ ||  
 new PrismaClient({  
 log: process.env.NODE\_ENV === "development"  
 ? ["query", "error", "warn"]  
 : ["error"],  
 });  
  
if (process.env.NODE\_ENV !== "production") {  
 global.\_\_PRISMA\_\_ = prisma;  
}

## src\lib\storage-supabase.ts

// src/lib/storage-supabase.ts  
import { createClient } from "@supabase/supabase-js";  
  
const url = process.env.NEXT\_PUBLIC\_SUPABASE\_URL!;  
const key = process.env.SUPABASE\_SERVICE\_ROLE!; // server-only  
const bucket = process.env.SUPABASE\_BUCKET!;  
  
const supa = createClient(url, key, { auth: { persistSession: false } });  
  
export async function supabasePutBuffer(  
 keyPath: string,  
 buf: Buffer,  
 contentType?: string  
) {  
 const { error } = await supa.storage.from(bucket).upload(keyPath, buf, {  
 contentType: contentType || "application/octet-stream",  
 upsert: true, // overwrite if same key  
 });  
 if (error) throw error;  
  
 // Create a short-lived signed URL for downloads (e.g., 24h)  
 const { data: signed, error: signErr } = await supa.storage  
 .from(bucket)  
 .createSignedUrl(keyPath, 60 \* 60 \* 24);  
  
 if (signErr) throw signErr;  
 return { key: keyPath, signedUrl: signed?.signedUrl ?? null };  
}

## src\lib\t.ts

// src/lib/t.ts  
import { el } from "./i18n-el";  
  
export function t(key: string): string {  
 return el[key as keyof typeof el] ?? key;  
}

## src\lib\utils.ts

import { clsx, type ClassValue } from "clsx"  
import { twMerge } from "tailwind-merge"  
  
export function cn(...inputs: ClassValue[]) {  
 return twMerge(clsx(inputs))  
}

## src\types\next-auth.d.ts

// src/types/next-auth.d.ts  
import NextAuth, { DefaultSession } from "next-auth";  
  
declare module "next-auth" {  
 interface Session {  
 user: DefaultSession["user"] & {  
 id?: string;  
 role?: "ADMIN" | "USER";  
 };  
 }  
}  
  
declare module "next-auth/jwt" {  
 interface JWT {  
 id?: string;  
 role?: "ADMIN" | "USER";  
 }  
}